IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Caelen Lin
06 March 2022

Outline

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

Executive Summary

- Methodologies
  - Data was collected via API and Web Scraping.
  - The collected data was then cleaned/wrangled such that the final output could provide an accurate analysis, and consistent for prediction.
  - Exploratory Data Analysis using Data Visualization such as Bar Graphs and Scatter Plots help to identify if there were any strong correlation between the different attributes.
  - SQL was also used to extract and manipulate data.
  - Interactive maps and dashboard were created using Folium and Plotly Dash respectively, to allow dynamic visualizations.
  - 4 Machine Learning Algorithms were used, with Grid Search to determine the best parameters and best accuracy scores. Confusion Matrices were created to display the accuracies.

Executive Summary (cont'd)

- Results
  - Success rates get better over the years.
  - One of the launch sites yield better success rates.
  - The Machine Learning algorithms separately yield close accuracy scores.

Introduction

- In the commercial space age today, SpaceX has been the most successful in bringing down average cost of 165 million per launch, to 62 million, via Falcon 9 rocket launches.
- These savings are due to Falcon 9's first stage reusability.
- To help SpaceY to compete with SpaceX, public information is gathered to predict, using various machine learning models.
- Analysis will identify factors SpaceX based on, to reuse the first stages.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected either via SpaceX's own API via JSON objects, or
  - Web scraping Wiki pages using Python BeautifulSoup package.
- Perform data wrangling
  - There are several landing test types, be it a ground pad, drone ship, or ocean landing.
  - Categorising these various outcome into either a "Success" or "Failure" landing.

Methodology (cont'd)

Executive Summary

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Train-test split was utilized for out-of-sample accuracy.
  - Grid Search was performed to find the best hyperparameters that provides the best accuracy
  - Logistic Regression, Support Vector Machines, Decision Tree Classifier, and K-Nearest Neighbours (KNN) algorithm were used, generating the individual confusion matrix.

Data Collection

- SpaceX API
  - Get request called to access the API.
  - For the purpose of this project, a static json is utilized as an alternative.
  - The json response was then normalized using pandas.
  - Customized functions are created to capture specific data from the response.
  - These data are then stored into a pre-defined dictionary, then processed into a dataframe for further use.
- Web Scraping
  - Same steps as the API, but accessing a public website, here, Wikipedia page (archived).
  - BeautifulSoup package is used, to search for html tags.
    - “th”, “col” for the columns/attributes.
    - “tr”, “td” for the data of each attributes per observation, allocated to the right columns.
  - Customized functions, pre-defined dictionary stored and processed into another dataframe for further use.

Data Collection – SpaceX API

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

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

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

```
# Use json_normalize method to convert the json result into a dataframe

data = pd.json_normalize(response.json())
```

GitHub API URL

Data Collection – SpaceX API (cont'd)

```
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
BoosterVersion[0:5]
```

```
7]: ['Falcon 1', 'Falcon 1
```

we can apply the rest of the fu

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

```
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}
```

```
# Create a data from launch_dict

launch_df = pd.DataFrame.from_dict(launch_dict)
```

GitHub API URL

Data Collection - Scraping

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

```
response = requests.get(static_url)
html_data = response.text
```

```
beautiful_soup = BeautifulSoup(html_data, "html5lib")
```

```
html_tables = beautiful_soup.find_all("tbody")
```

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each
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']=[]
```

GitHub Scrape URL

Data Collection – Scraping (cont'd)

```
df = pd.DataFrame(launch_dict)
```

```
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

for col in first_launch_table.find_all('th', scope="col"):
    column_names.append(extract_column_from_header(col)) # mutable list, thus can append straight away

# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
    column_names[:] = [item for item in column_names if item != '']
```

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

GitHub Scrape URL

Data Wrangling

- After the initial storing of data extracted from the API, the various attributes and observations were looked in detail to check for data wrangling for the machine learning prediction later.
- Using “isnull”, “dypes” allow us to calculate the number/percentage of null values, and the type of data stored respectively.
- “value_counts()” function provided a breakdown of the total counts per attributes we are interested in.
  - In the lab, we grouped out the different launch sites, as well as the different orbits each rocket launch was supposed to be sent to.
- Lastly, based on the outcomes, we understood that due to the different types of sites in collecting the first stage, the data was required to be wrangled to be determined as either a “success” or “failure” landing.

Data Wrangling

```
df.isnull().sum()/df.count()*100
```

```
df.dtypes
```

```
# 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
```

GitHub Wrangle URL

Data Wrangling (cont'd)

```
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
```

```
good_outcomes = set(landing_outcomes.keys()[[0,2,4]])
# landing_class = 0 if bad_outcome
landing_class = df['Outcome']
landing_class.replace(good_outcomes, 1, inplace = True)
# landing_class = 1 otherwise
#s.replace(13, 42, inplace=True)
landing_class.replace(bad_outcomes, 0, inplace = True)
```

```
df['Class'] = landing_class
```

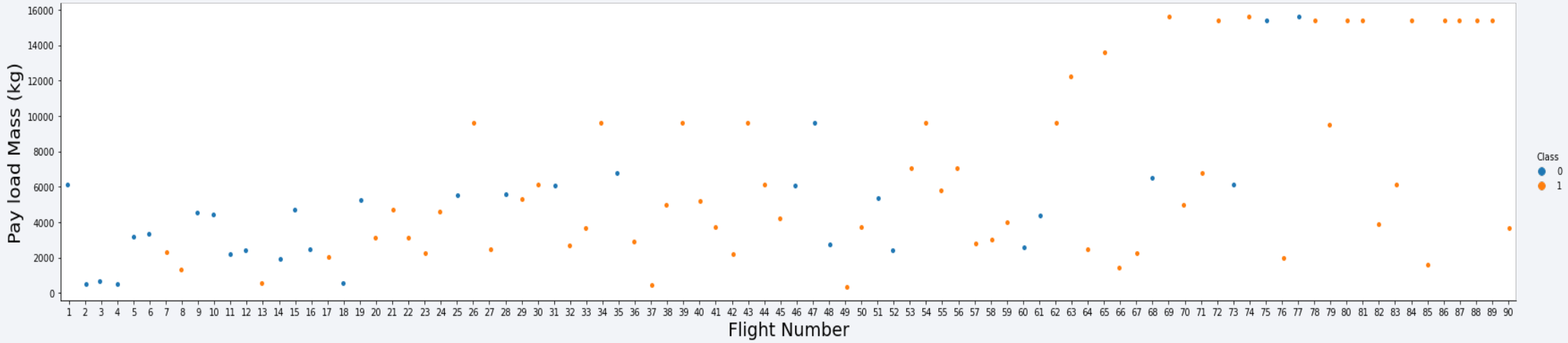
GitHub Wrangle URL

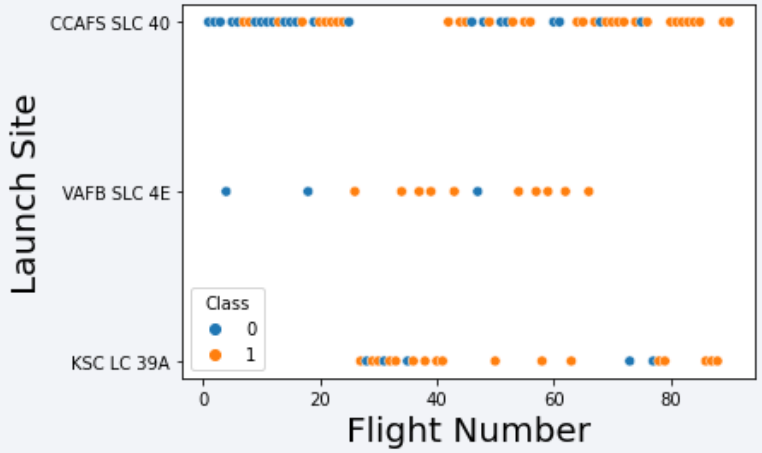
EDA with Data Visualization

- Scatter Plots
  - Such plots are usually used to determine if there is any correlation between 2 variables.
    - E.g., Pay Load Mass, Flight Number, Launch Site, Orbit correlate with each other
- Line Graph
  - This kind of graphs display information that changes continuously over time.
  - Displayed how success rate changes over the years since 2010.
- Bar Chart
  - Bar charts usually are used for presenting categorical data with heights/lengths proportional to values each category represents.
  - Used for representing the landing success rate of various orbits the rockets were launched to.

GitHub Viz URL

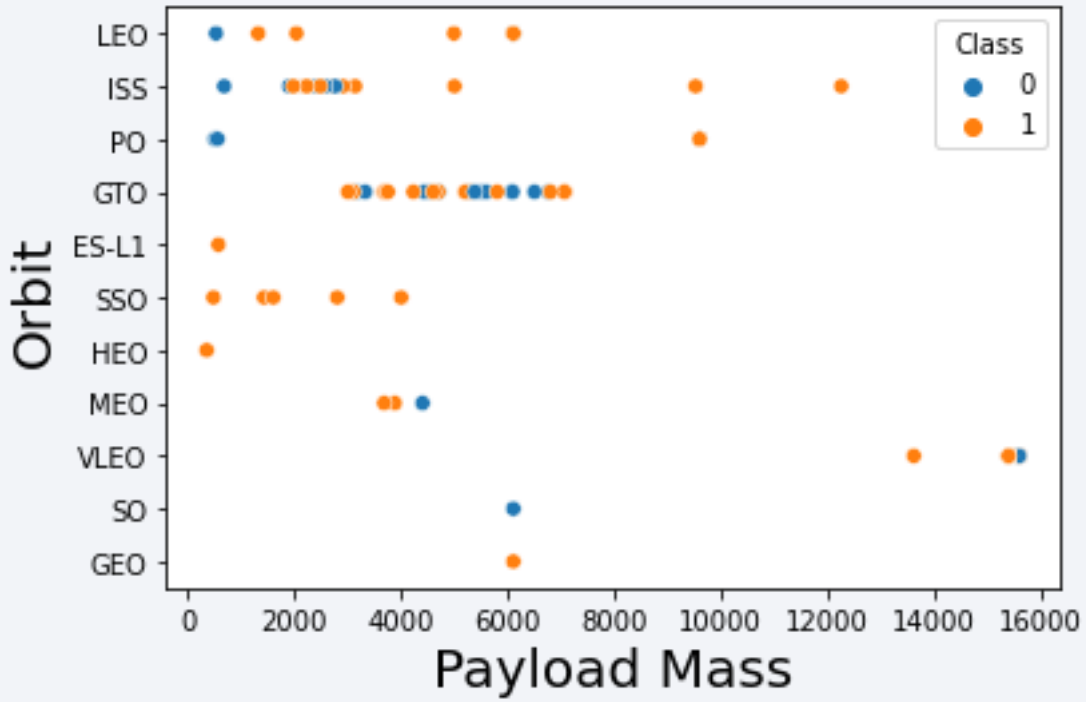
EDA with Data Visualization – Scatter Plots

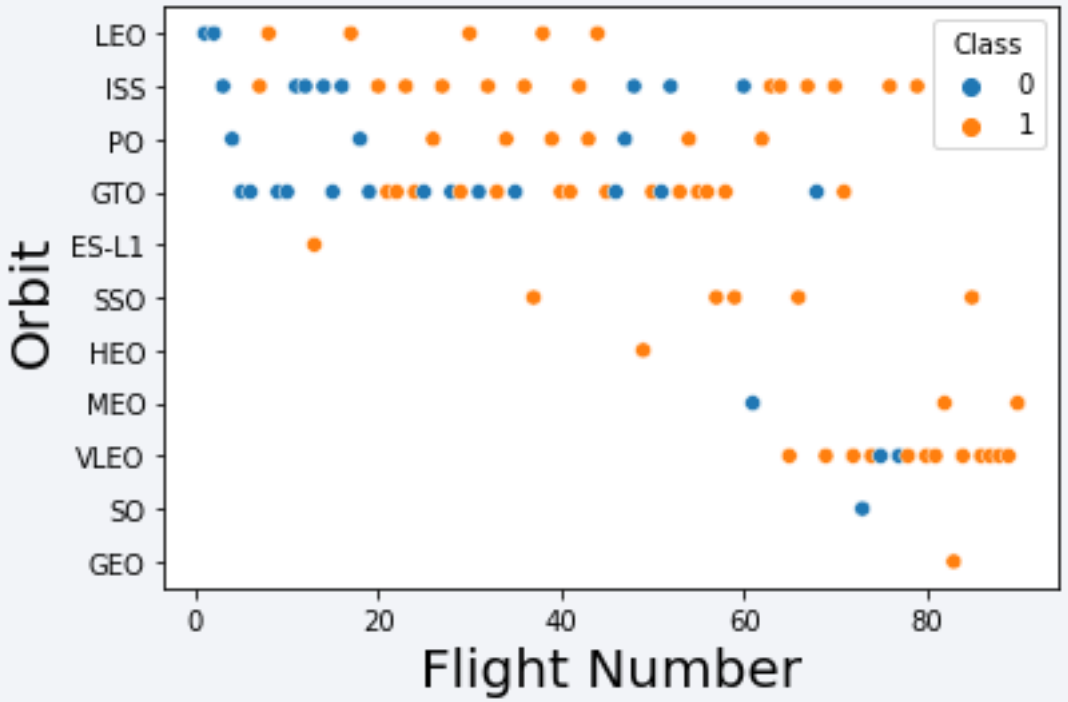




[GitHub Viz URL]

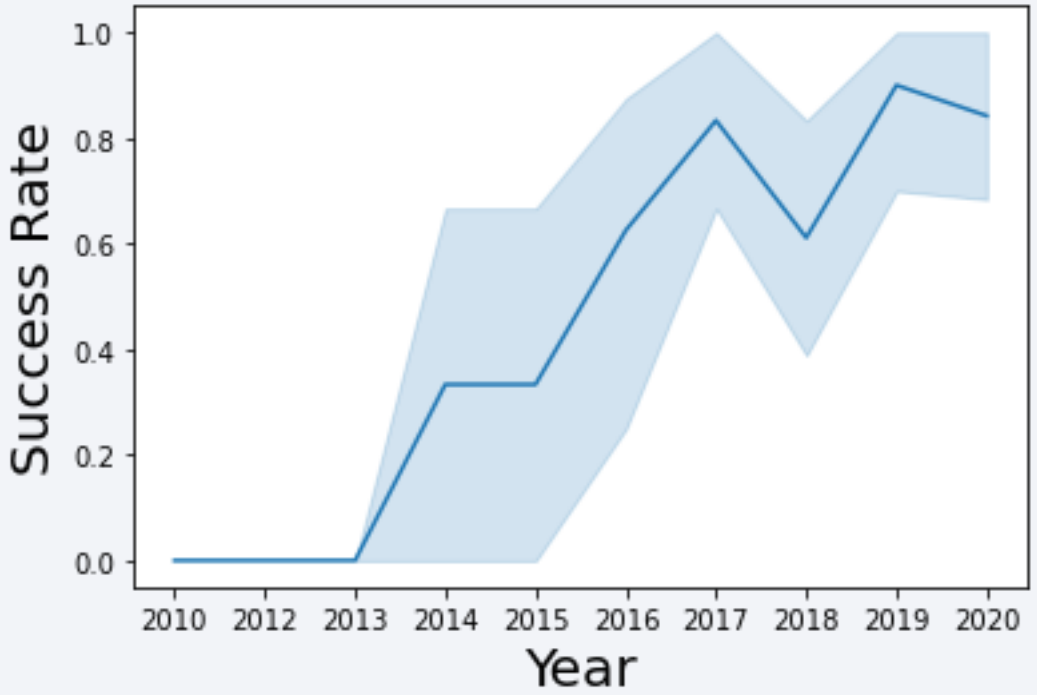
EDA with Data Visualization – Scatter Plots (cont'd)

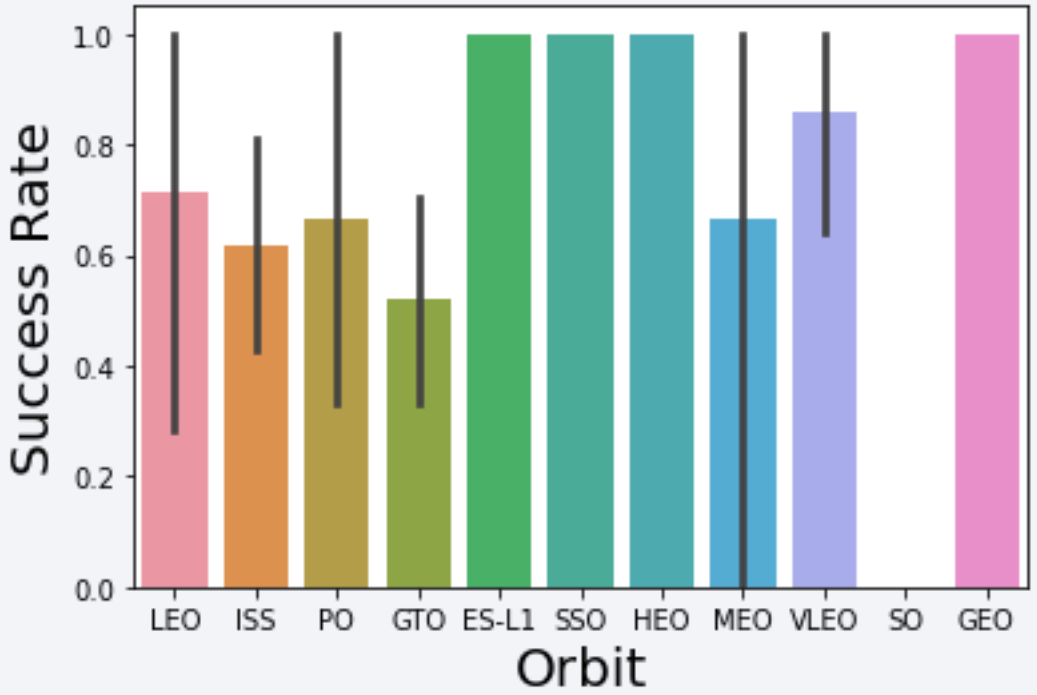




[GitHub Viz URL]

EDA with Data Visualization – Line, Bar Charts





GitHub Viz URL

EDA with SQL

- SQLs performed:
  - Connect to database.
  - Display names of unique launch sites in the space mission.
  - Display 5 records where launch sites begin with string “CCA”.
  - Display average payload mass carried by booster version “F9 v1.1”.
  - List the date when the first successful landing outcome in ground pad was achieved.
  - List names of boosters which have success in drone ship and having payload mass > 4000, < 6000.
  - List the total number of successful and failure mission outcomes.
  - List names of booster versions which have carried maximum payload mass, via a subquery.
  - List the failed landing outcomes in year 2015, on drone ship with their booster versions.
  - Rank the count of landing outcomes (all variations), between date “4th June 2010” and “20th March 2017”, in descending order.

GitHub SQL URL

Build an Interactive Map with Folium

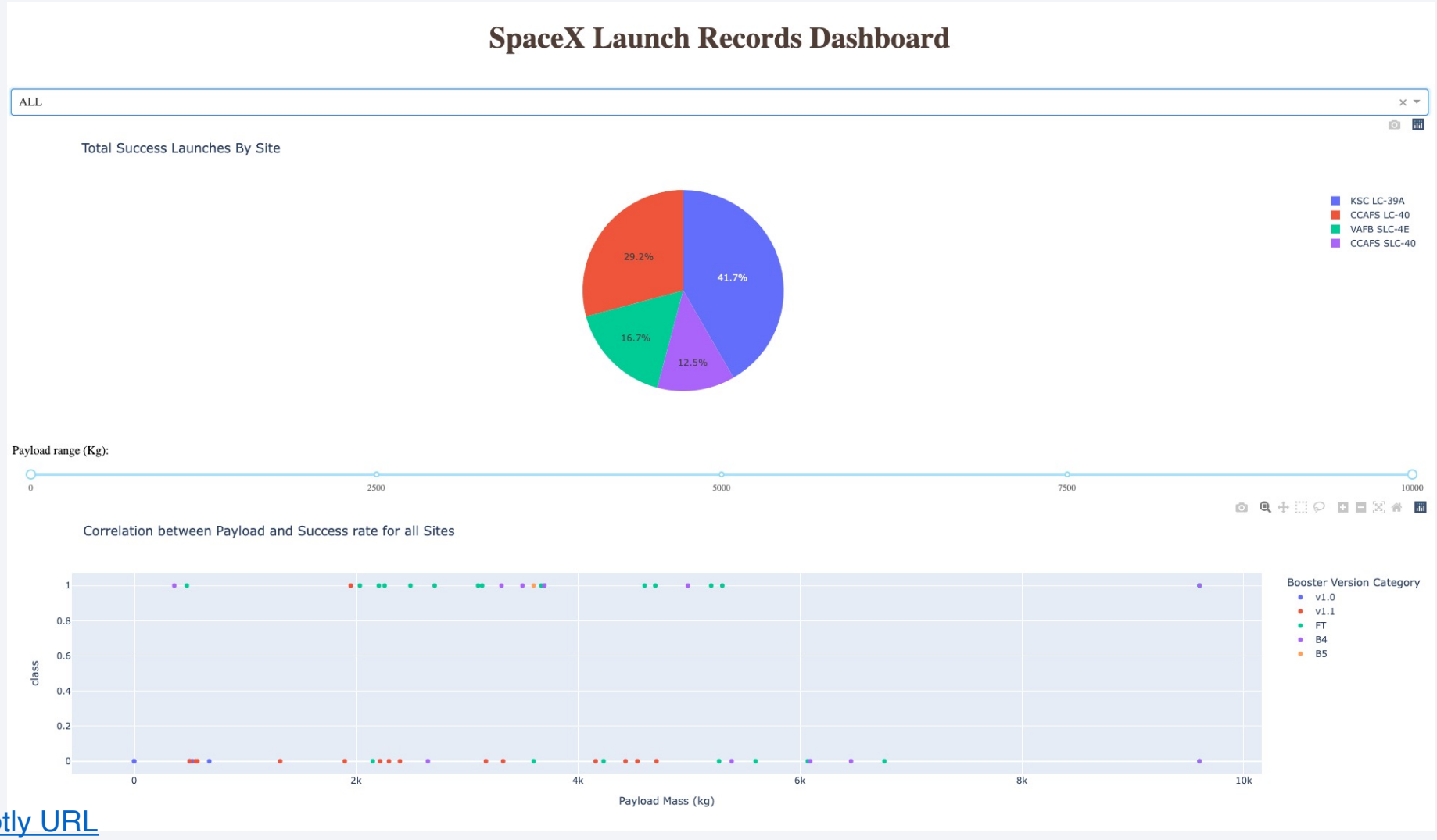
- Circles
  - Used for marking each launch sites via their coordinates, with the ID labels of each site.
- Cluster
  - This object was used as to consolidate launches with exact coordinates (because of same launch sites).
  - Also make the map less cluttery with numerous information per zoom view.
  - Marker colors (green and red) were used to display a success or failure launch outcome.
- Mouse-over
  - For easy identification of coordinates when user interacts with the folium map.
- Lines and Label Marker
  - To connect two coordinates-pairs, showing user the “length” or distance between them. The distances in total were also marked on the map to provide a quantitative information.
  - Distance is calculated using the “Haversine formula”.

GitHub Folium URL

Build a Dashboard with Plotly Dash

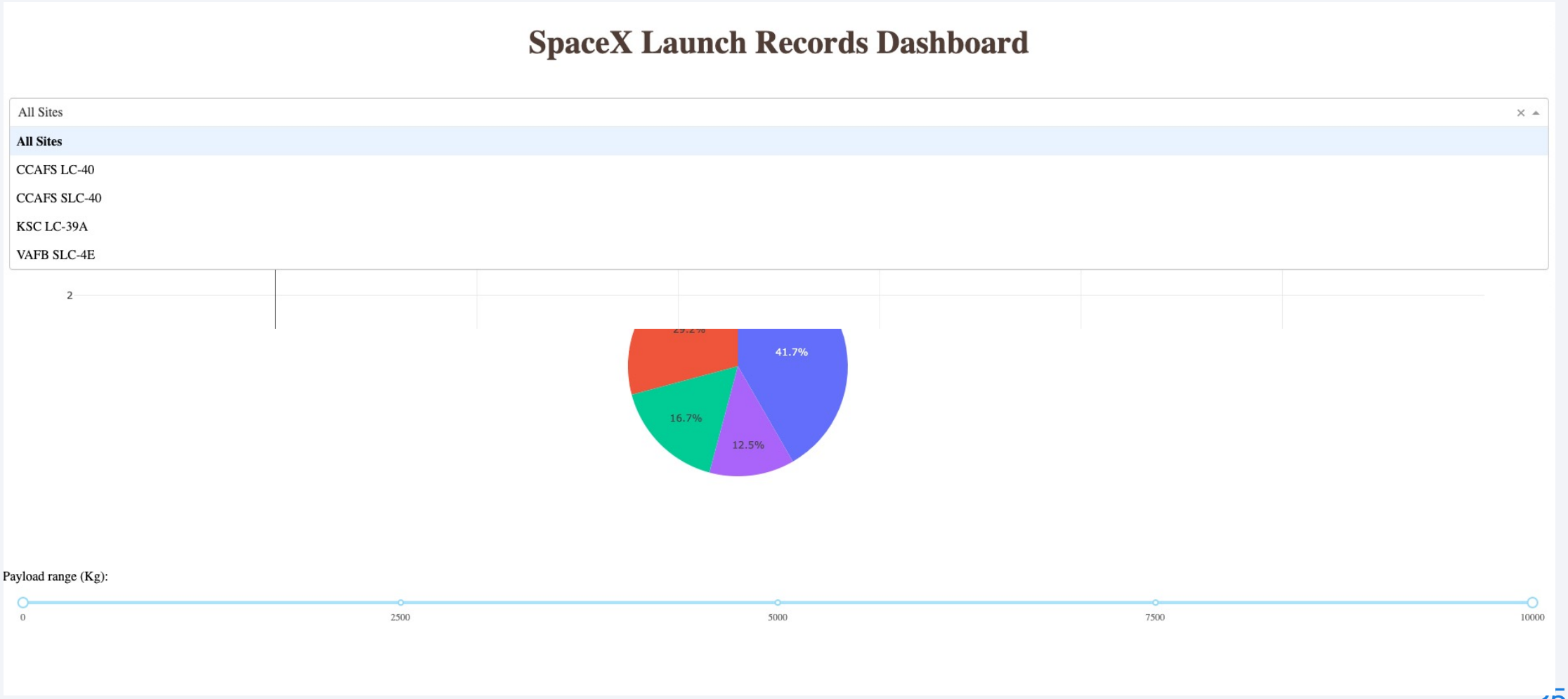
- Pie Chart
  - Show percentages of a whole.
  - Used pie charts to display:
    - The percentages of launch sites in the dataset.
    - The percentage of each outcome (i.e., success / failure) per launch site.
- Scatter Plot
  - Show correlation of two variables/attributes.
    - Here, the success of a launch affected by the payload mass.
- Dropdown
  - To allow user to look at an overview (“ALL”), or the individual launch sites.

Build a Dashboard with Plotly Dash - Default



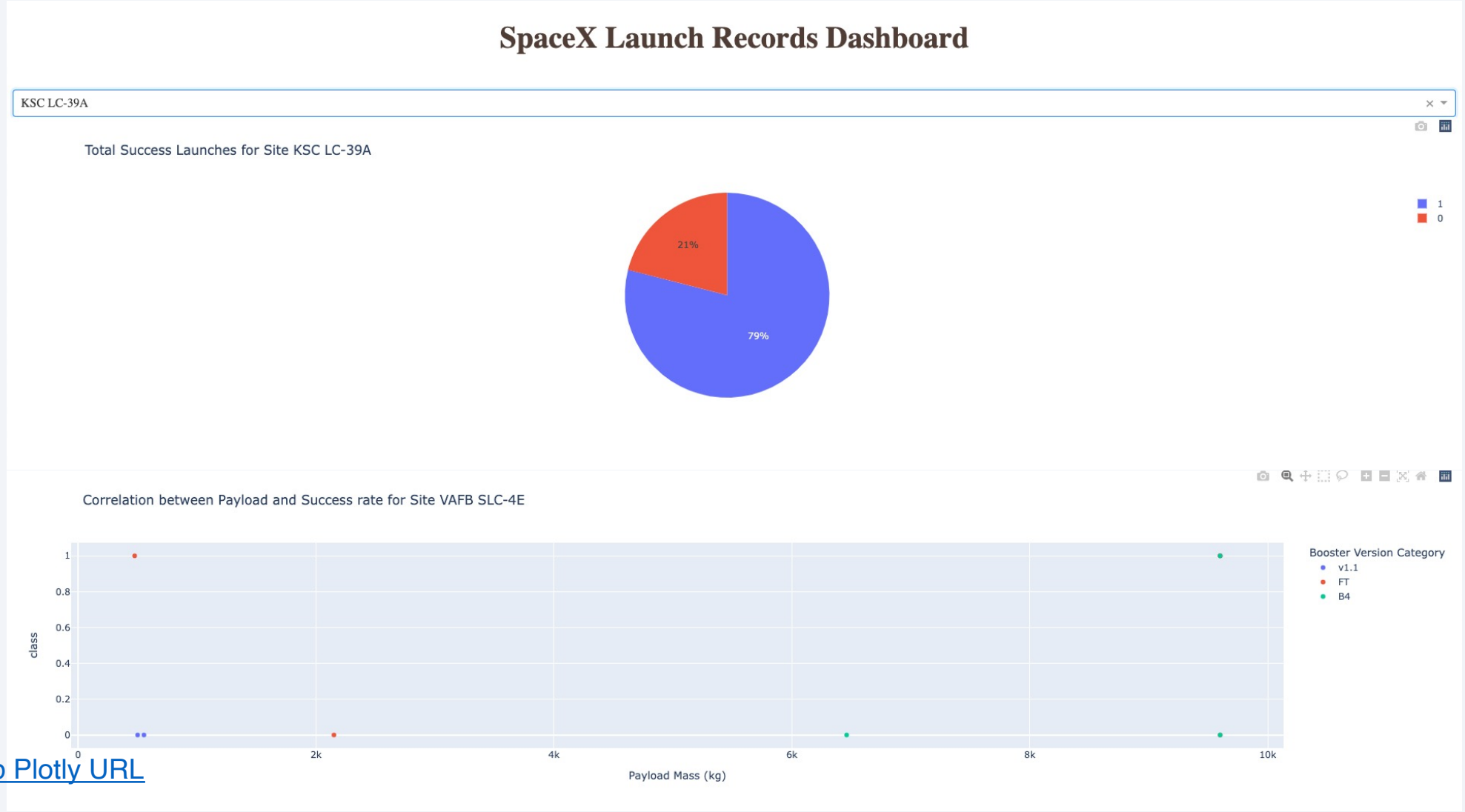
GitHub Plotly URL

Build a Dashboard with Plotly Dash – Dropdown / Slider



[GitHub Plotly URL]

Build a Dashboard with Plotly Dash – Interactivity



GitHub Plotly URL

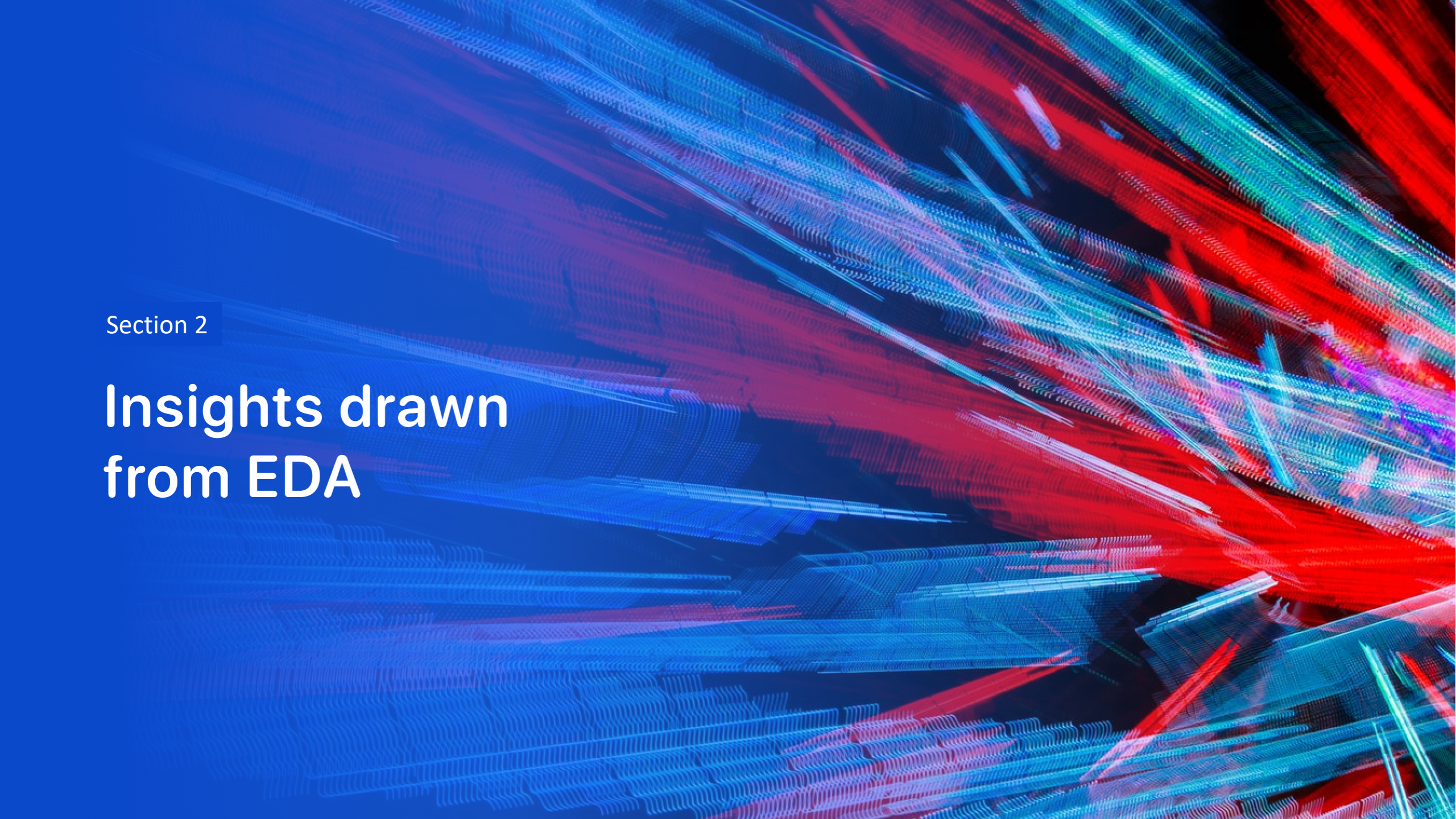
Predictive Analysis (Classification)

- The dataset was separated with an attributes dataframe (stored as “X”), with the target (“Y”).
- Both undergone a “train_test_split” function, to split them into training and testing data, where 20% was set aside for testing to improve out-of-sample accuracy (i.e., 18 observations).
- Random state seed is set at 2.
- Logistic Regression, Decision Tree classifier, K-nearest neighbours (KNN) and Support Vector Machines were used.
- Confusion matrix was used on all, to display how well they performed visually.
- Grid Search was then used on all, to identify the best score, and the best parameters. The best estimator feature is then used to identify each method’s score.

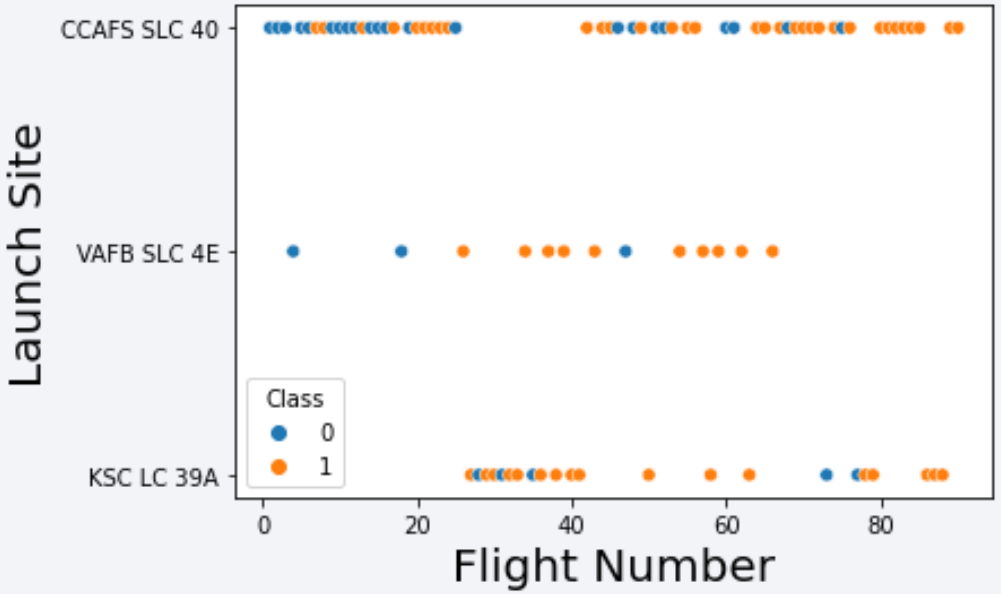
GitHub ML URL GitHub ML v2 URL

Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

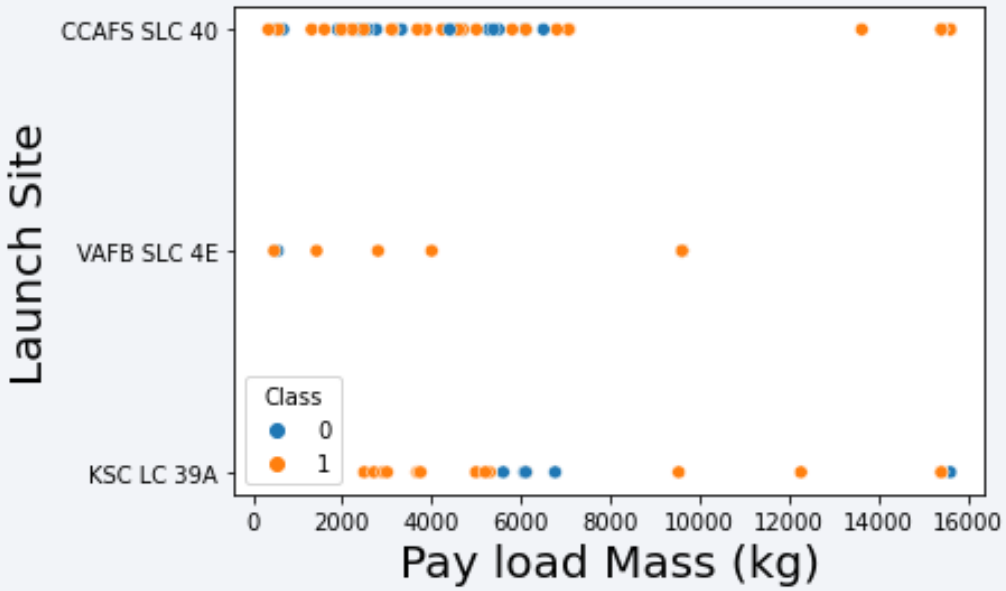


**Flight Number is the index/ID of each launch*

The bigger the Flight Number (or the more recent launches), the more successful the launches are (in orange).

Seemingly, VAFB SLC 4E launch site has the most successful launches. Despite a slightly more count of failure on KSC LC 39A, it has relatively more successes.

Payload vs. Launch Site



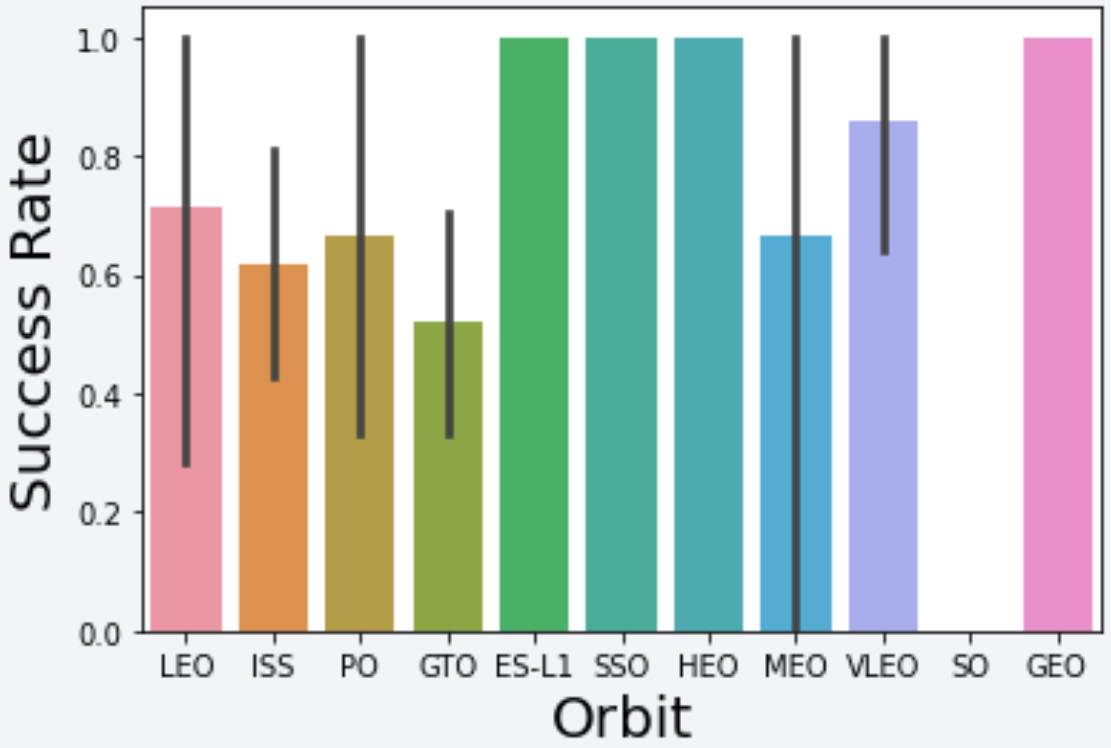
If payload mass is >8000kg, the success rate of the launch is almost 100%.

However, the correlation between these two are weaker (relatively) as compared to the Flight Number.

This may be due to a mix of successes and failures at mass <8000kg.

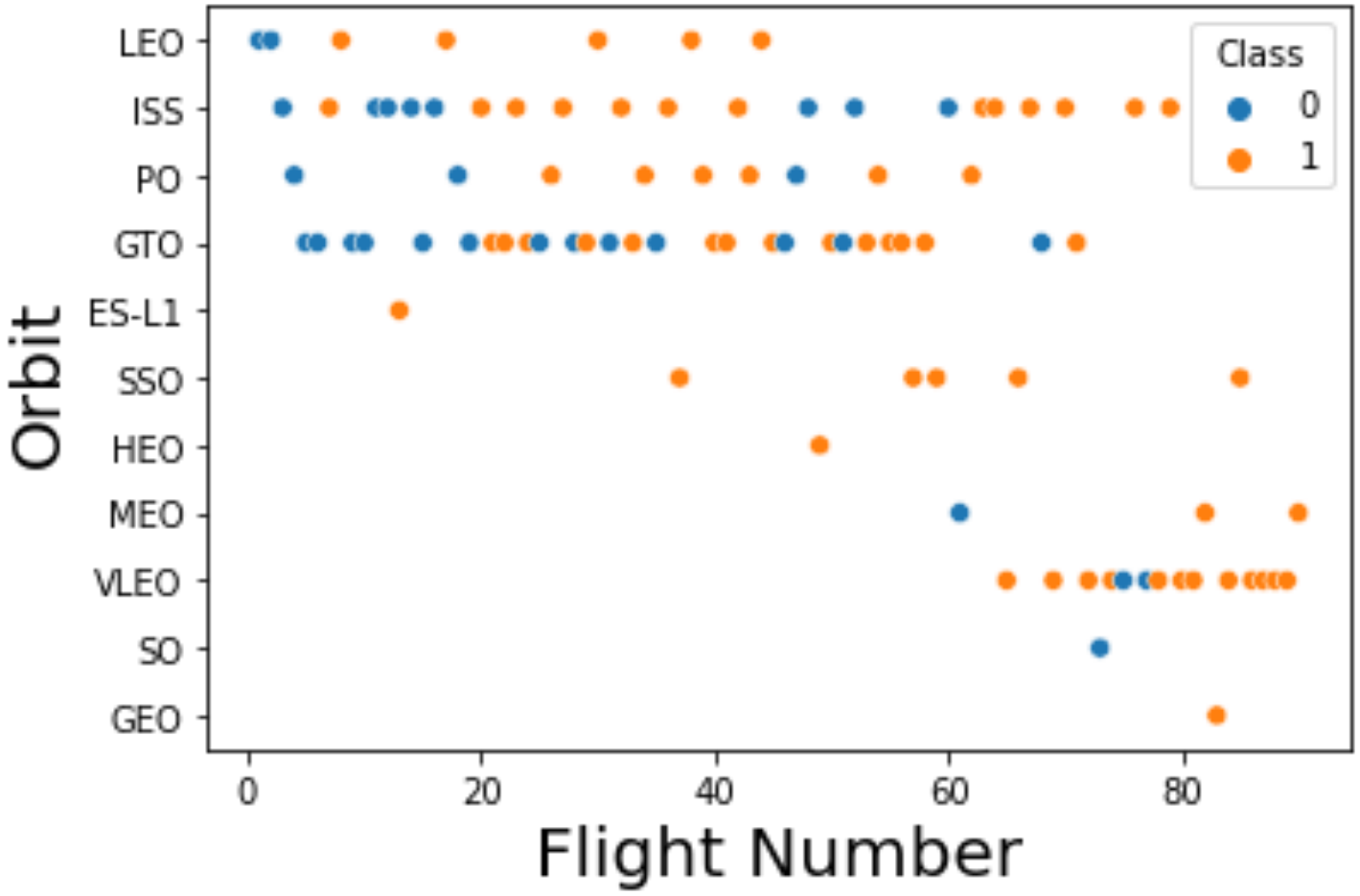
Success Rate vs. Orbit Type

- ES-L1, SSO, HEO, and GEO has the best success rate at 100%.



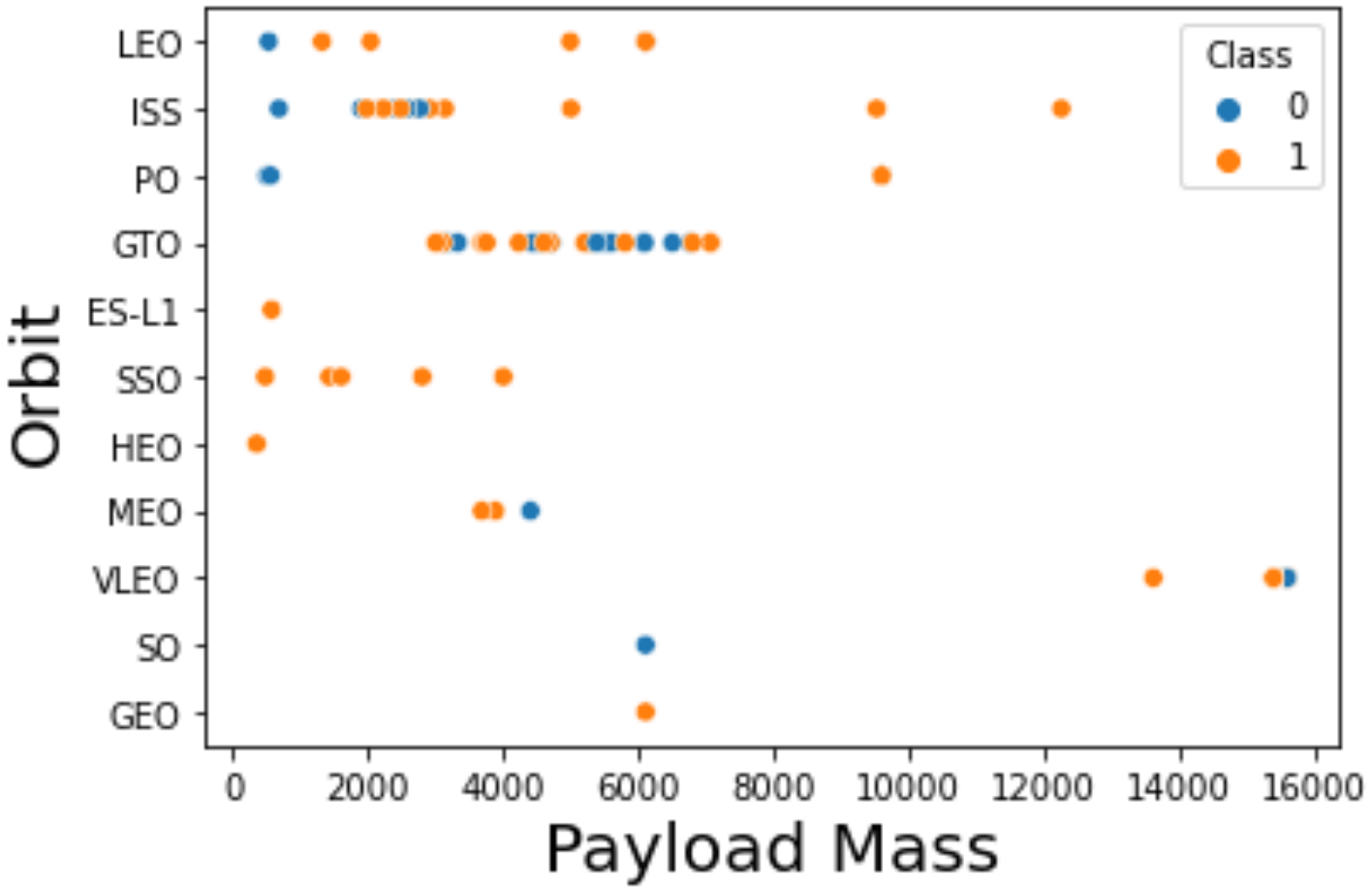
Flight Number vs. Orbit Type

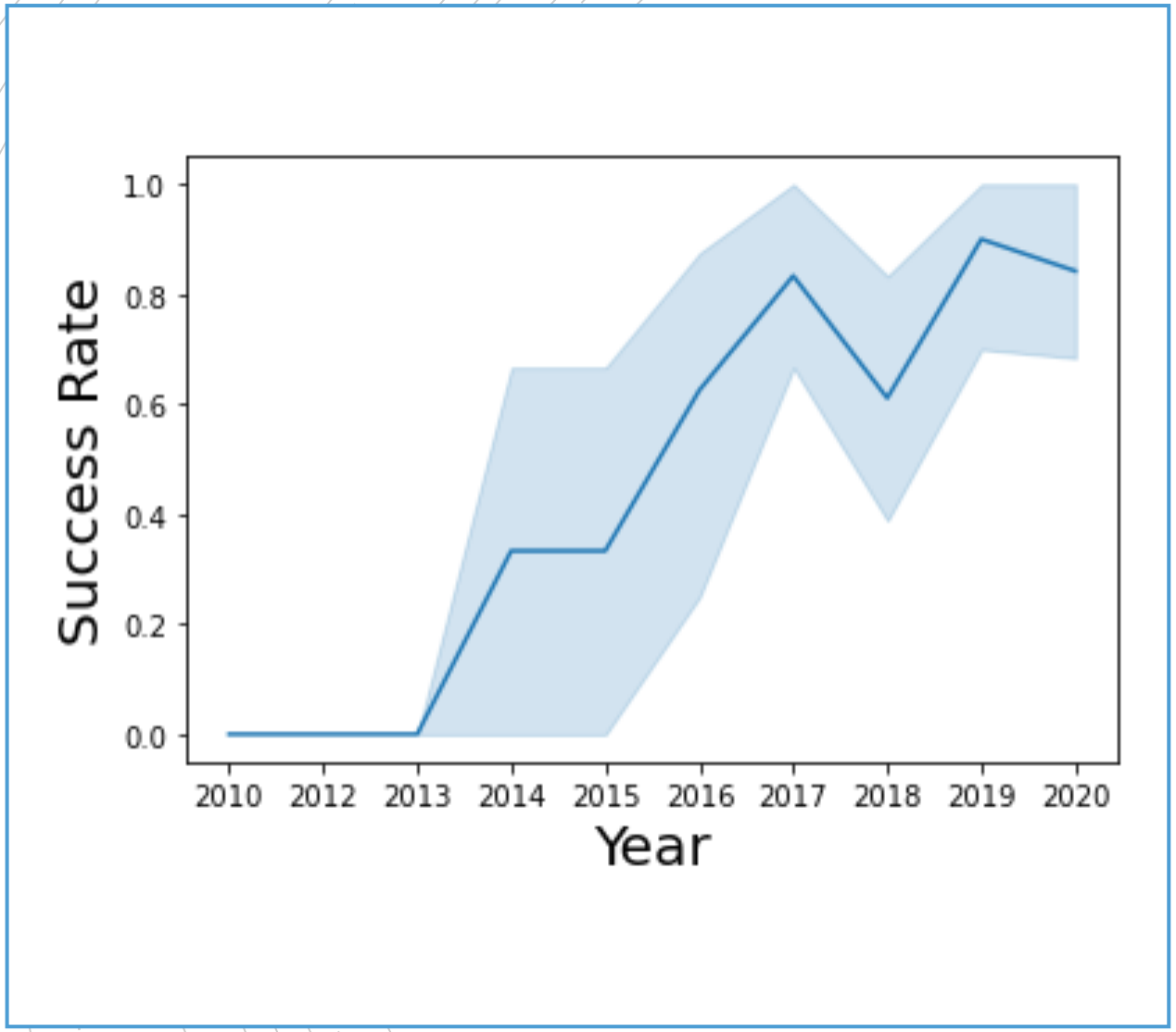
- As observed earlier, ES-L1, SSO, HEO, and GEO has the best success rate at 100%
- Excluding ES-L1, the other 3 orbits are part of later launches.
- Earlier Flight Numbers were also seen with higher failure rates.
- Weaker correlation between Flight Number and Orbit type.



Payload vs. Orbit Type

- As observed earlier, ES-L1, SSO, HEO, and GEO has the best success rate at 100%
- These also have the lower payload masses not more than about 6000kg.
- Weak correlation between Payload mass and Orbit type.
- However, more information, like height of orbit above equator may help to see if other factors like gravity pull may impact the success rates.





Launch Success Yearly Trend

- As the year goes, since 2013, the success rates increases, except for a dip in 2018, and 2020.
- This is also supported by the correlation between Flight Number and Success Rate.

All Launch Site Names

%sql select distinct(Launch_site) from spacexds

- As per the keyword “distinct”, this function returns the unique value on a list of data.

Out[11]:

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

Launch Site Names Begin with 'CCA'

```
In [17]: %sql select * from spacexds where launch_site like 'CCA%' limit 5
```

Out[17]:

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

- ”Limit 5” returns the first 5 rows of the query.
- ”Like” keyword is used to search text containing the sub-string “CCA”.
- A “%” wildcard is coupled with “like”, and place after “CCA” means the string of text to search must begin with “CCA”, and can be anything thereafter

Total Payload Mass

```
In [18]: %sql select sum(payload_mass__kg_) from spacexds where customer = 'NASA (CRS)'
```

Out[18]:

1
45596

- “Sum” is used to aggregate, here to add up all the payload mass.
- Condition “where” is used, and the condition is set that the customer must be “NASA (CRS)”

Average Payload Mass by F9 v1.1

```
In [19]: %sql select avg(payload_mass__kg_) from spacexds where booster_version like 'F9 v1.1%'
```

Out[19]:

1
2534

- “Avg” is a function to average a list of values.
- “where” is used to identify the condition, i.e., given by “booster_version”.
- “like” keyword and wildcard ”%” are used, where version must begin with “F9 v1.1”

First Successful Ground Landing Date

```
In [25]: %sql select min(date) from spacexds where landing__outcome = 'Success (ground pad)'
```

Out[25]:

1
2015-12-22

- “min” is to sort and identify the lowest value, and for datetime, would refer to the earliest.
- “where” condition is set the be a successful ground pad landing.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
In [27]: %sql select distinct(booster_version) from spacexds where payload_mass__kg_ between 4001 and 5999 and landing__outcome
         = 'Success (drone ship)'
```

Out[27]:

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

- “distinct” is used to identify the unique booster versions, rather than duplicating and lengthening the list.
- Condition is also set, using “where” on what to look for (i.e., payload mass and successful drone ship landing), and
- “between” is used to set a range. Since question in the lab requested for greater/less than, value of 4000 and 6000 are omitted.

Total Number of Successful and Failure Mission Outcomes

```
In [30]: %sql select mission_outcome, count(*) as count from spacexds group by mission_outcome
```

Out[30]:

mission_outcome	COUNT
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

- “group by” keyword is used to let SQL know the requirement to categorise the values per “mission outcome” together.
- “count(*) is used to count the number of observations.
- “mission_outcome” attribute is included, so that the count can be allocated to the right attributes.

Boosters Carried Maximum Payload

```
In [31]: %sql select distinct(booster_version) from spacexds where payload_mass__kg_ = (select max(payload_mass__kg_) from spac
exds)
```

Out[31]:

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

- ”distinct” used to return unique (booster version) values, shorting the result
- Subquery is used on the condition, where it returns the value of the “max” payload mass.
- The main query will then filter observations with the max payload mass.

2015 Launch Records

```
In [33]: %sql select landing__outcome, booster_version, launch_site from spacexds where year(date) = 2015 AND landing__outcome
         = 'Failure (drone ship)'
```

Out[33]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- The necessary columns are identified and given as part of the query to return.
- Conditions are also set, by “where” keyword.
  - Year of the date are extracted, and to match if they are in 2015.
  - Landing Outcome is fixed as a filter, where we are looking at failure landing on drone ships.

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

```
In [48]: %sql select landing__outcome, count(*) as count \
         from (select landing__outcome from spacexds where date between '2010-06-04' and '2017-03-20')\
         group by landing__outcome\
         order by count(*) desc
```

Out[48]:

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

- A subquery is used to filter down to the observations between the desired date range and return as a dataset to be queried by the main query.
- The sub-queried dataset, is then grouped by the landing outcomes.
- Ranking on “count” is then performed, by “order by”, and “desc” is to inform SQL to return in descending order.

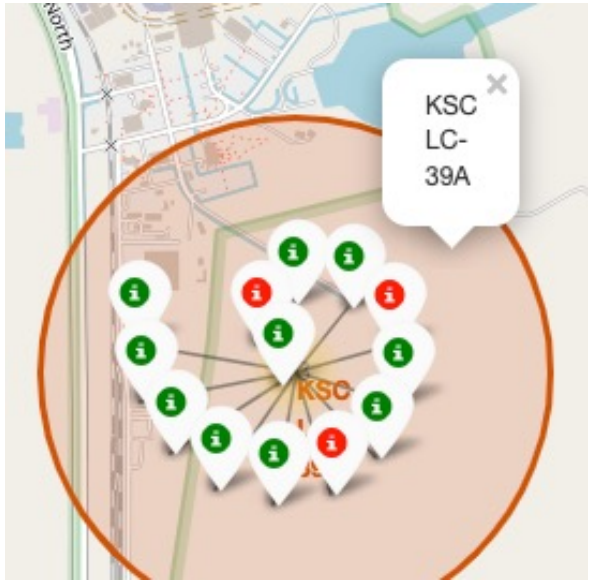

Section 3
Launch Sites
Proximities Analysis

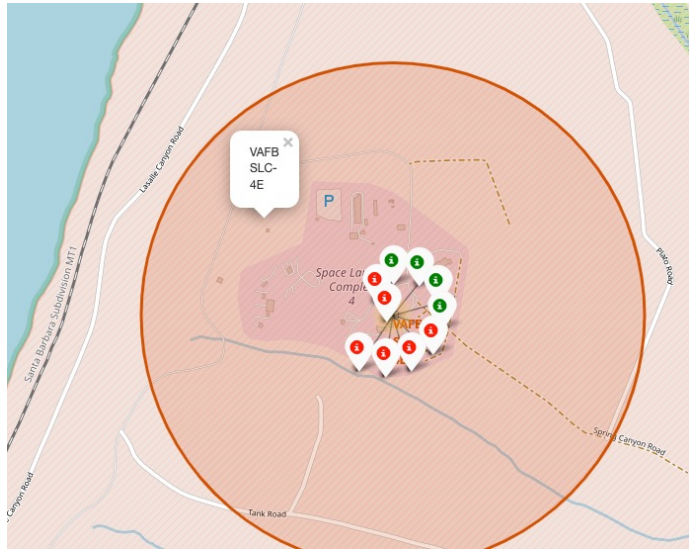
SpaceX Launch Sites on Global Map

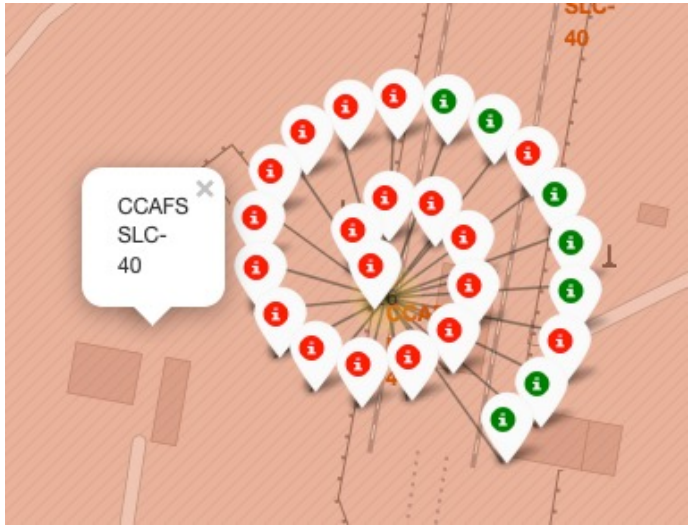
- The launch sites cluster near the coastline of either East or West Americas

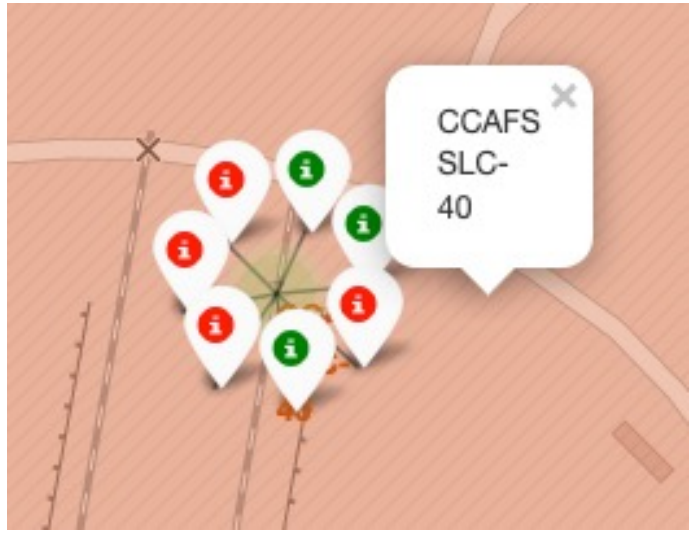
Color labeled launch outcomes

- Greens are successful launches, and red are not.
- Most of the Greens are located at “KSC LC-39A”.



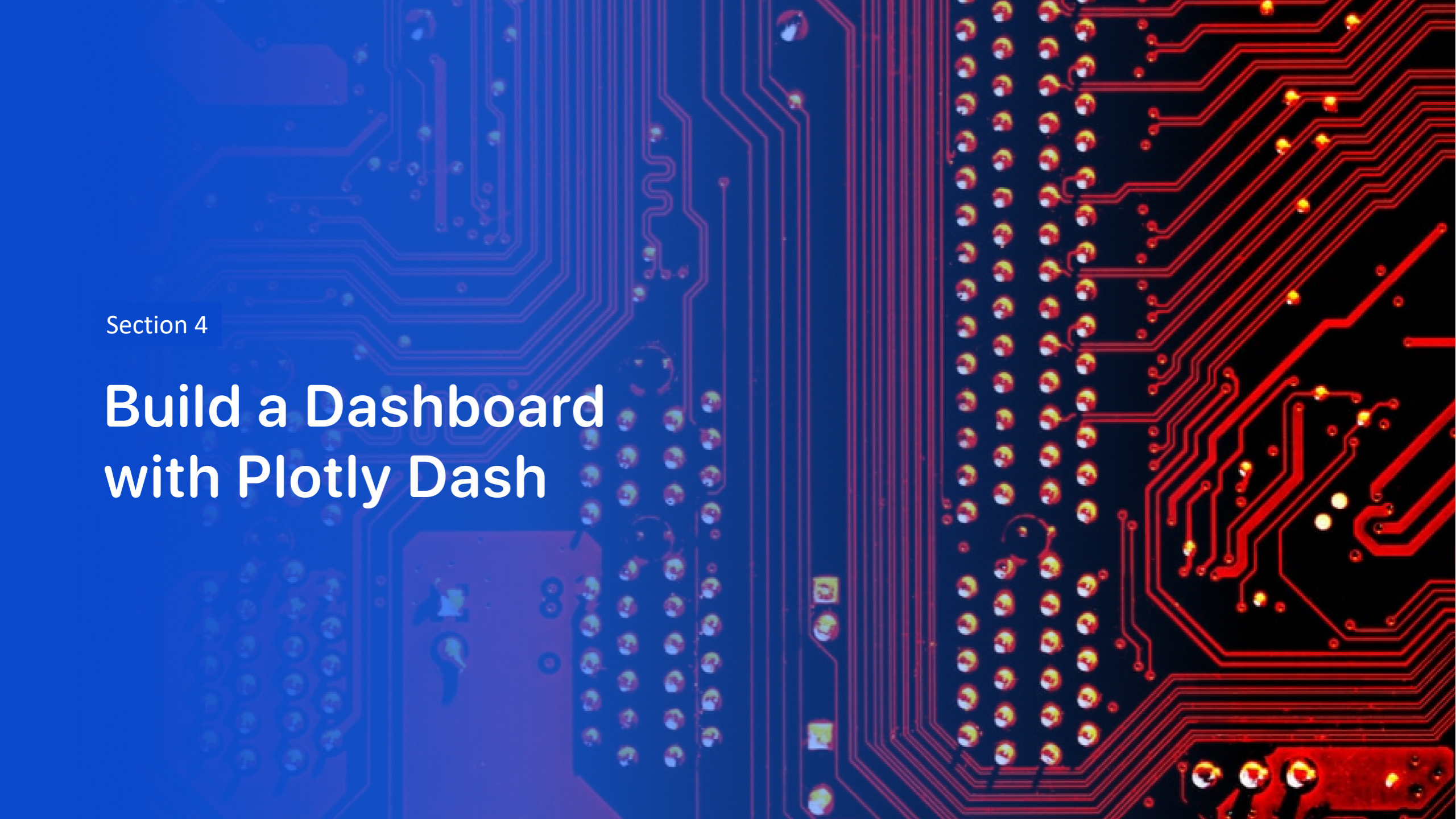




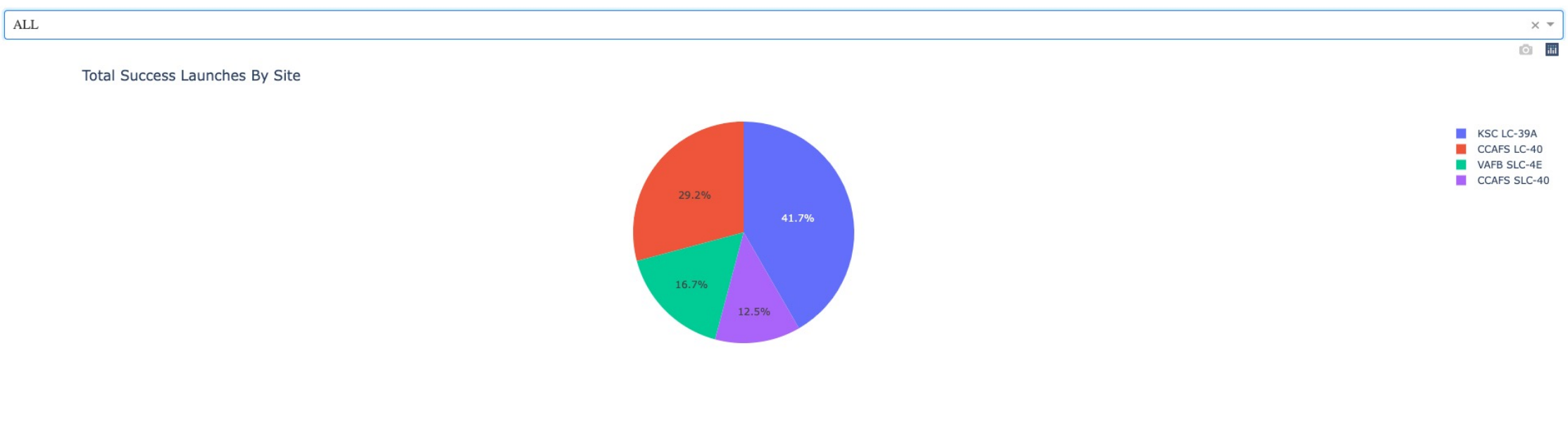


Distances to landmarks from an identified Launch Site

- The launch site coordinates which was used in the lab illustration, marked with an “X”, falls directly on a railway track.
- Therefore, making the railway track the nearest proximity. May be due to trains transporting the parts of the rockets.
- The highway and coastlines are also in near proximity, as they were calculated to be less than 1km away. Ideal for launch and landsites.
- The city is the furthest. This makes sense as any explosions from failed launches would be undesirable.

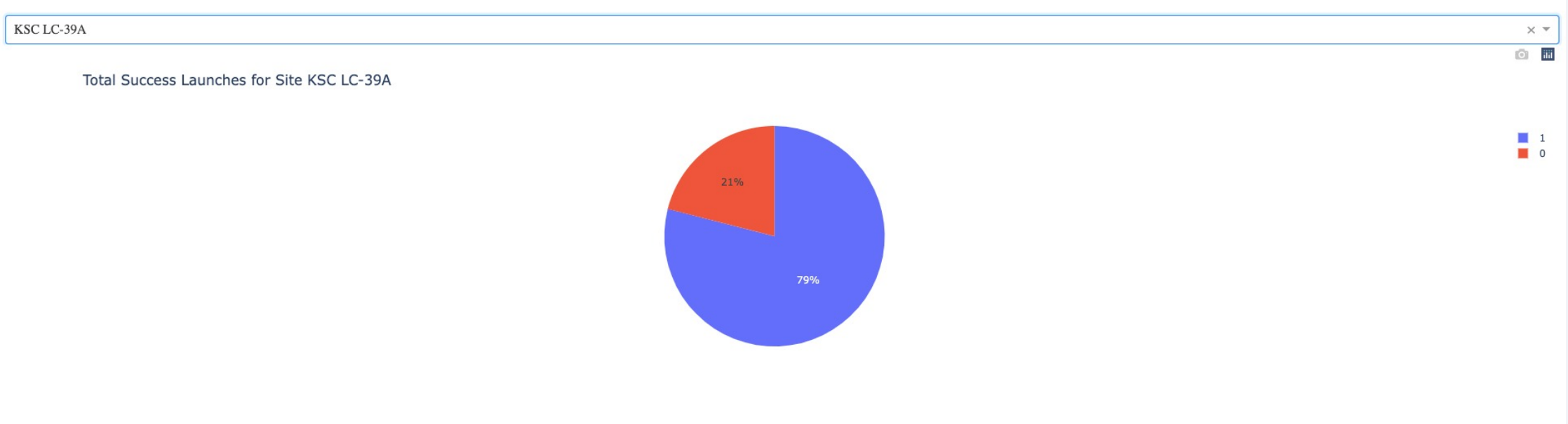

Section 4
Build a Dashboard with Plotly Dash

Dashboard – Pie Chart of Total Success Launches per Sites



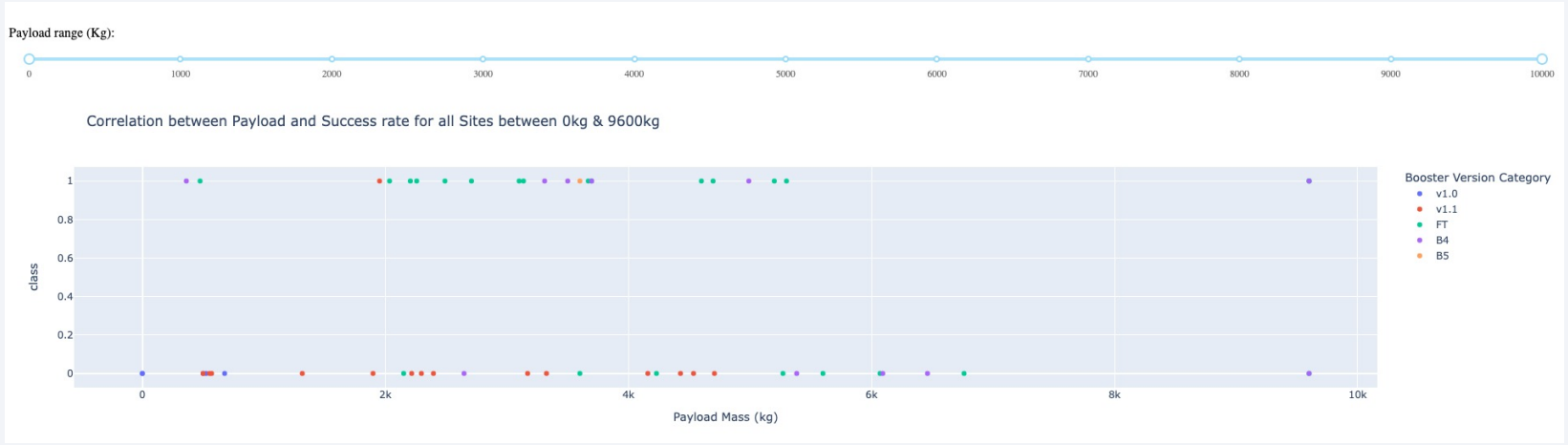
- The pie chart splits up the success counts into each of the site, with a ratio of the total count.
- “KSC LC-39A” site has the highest success launch count.

Dashboard – Pie Chart of Total Success Launches per Sites



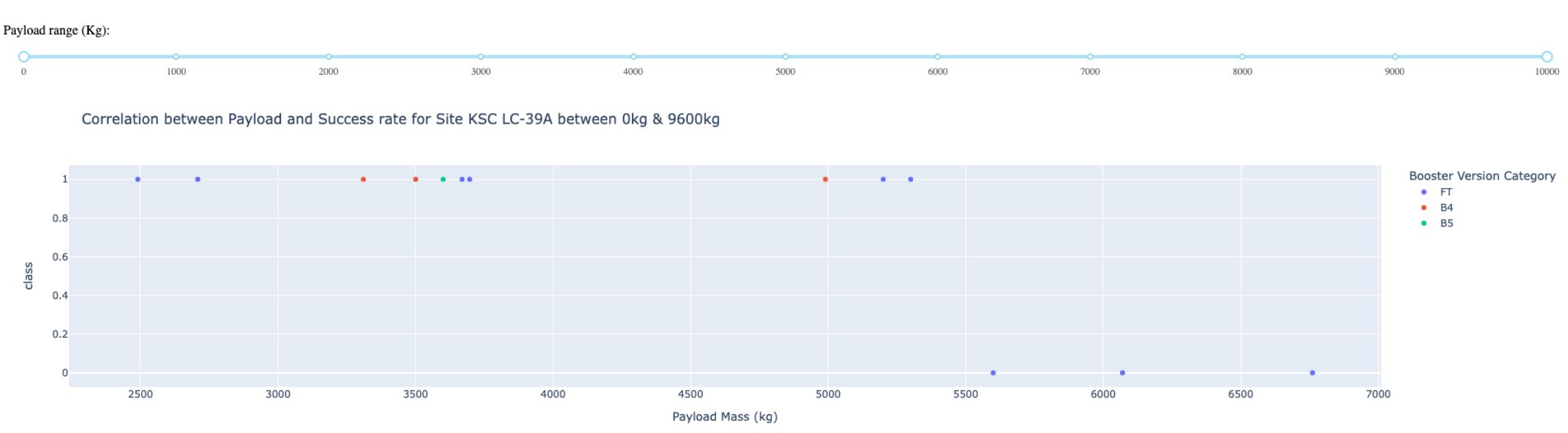
- “KSC LC-39A”, having the highest success launch count, achieved 79% success rate.

Dashboard – Payload Slider and Scatter Plot



- Above 6000kg, the success rate is almost 0%.
- Narrowing down the range between 2000kg and 6000kg, there are more successes than the other range of payload mass.
- As discussed earlier in other scatterplots, payload masses seem to have relative weaker correlation to the success rate.

Dashboard – Payload Slider and Scatter Plot (cont'd)



- At the most successful launch site, the lighter payload mass, from around 2500kg to below 5500kg has successful launches

<Dashboard Screenshot 3>

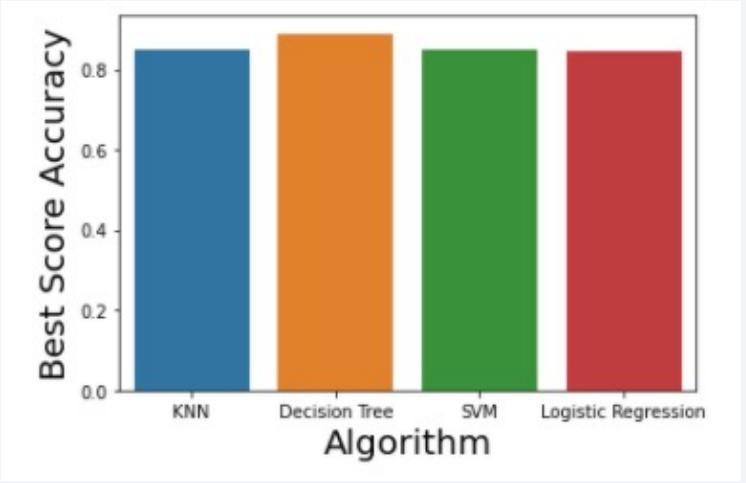
- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

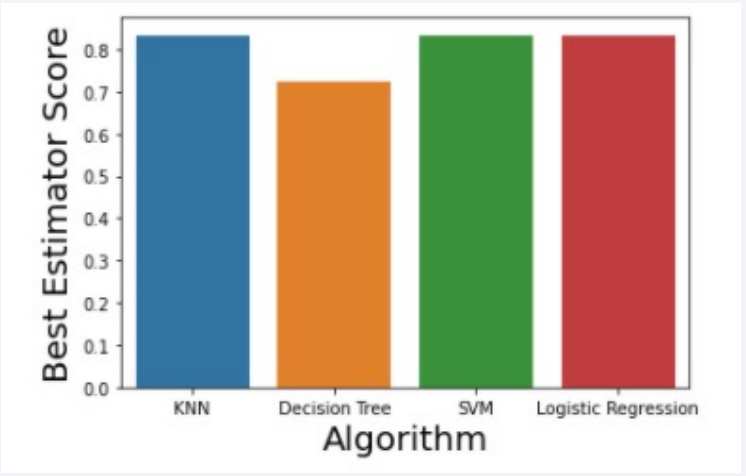
Section 5

Predictive Analysis (Classification)

Classification Accuracy

	Algorithm	Best Score Accuracy	Best Estimator Score
0	KNN	0.848214	0.833333
1	Decision Tree	0.889286	0.722222
2	SVM	0.848214	0.833333
3	Logistic Regression	0.846429	0.833333

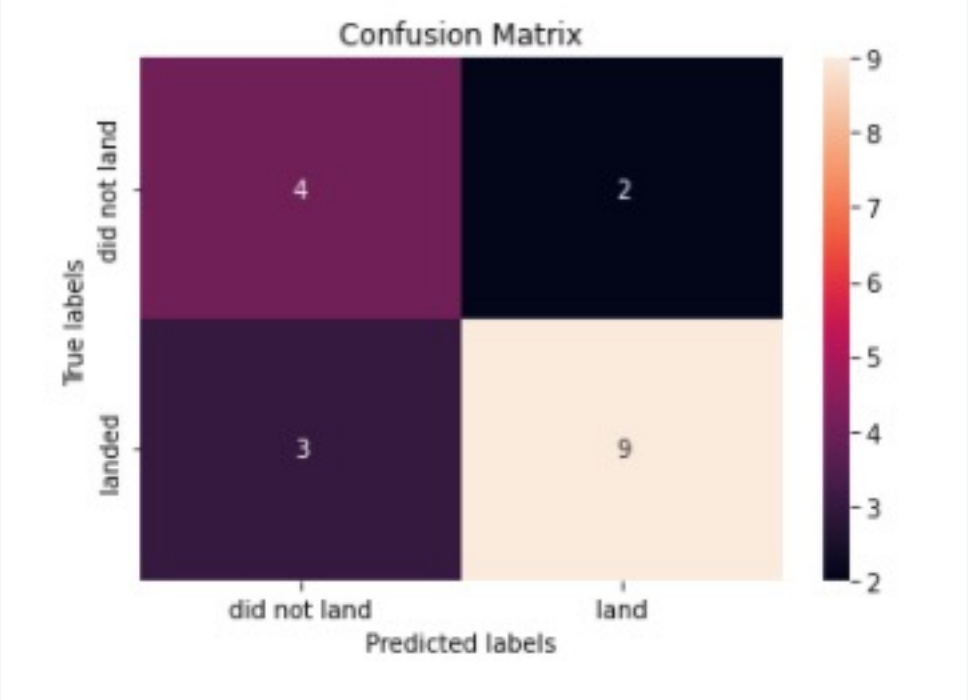




- Mixed results.
- Decision Tree gave the best accuracy score, but the best estimator gave the worst accuracy.
- The other algorithms have similar accuracies on both the best scores and best estimator scoring

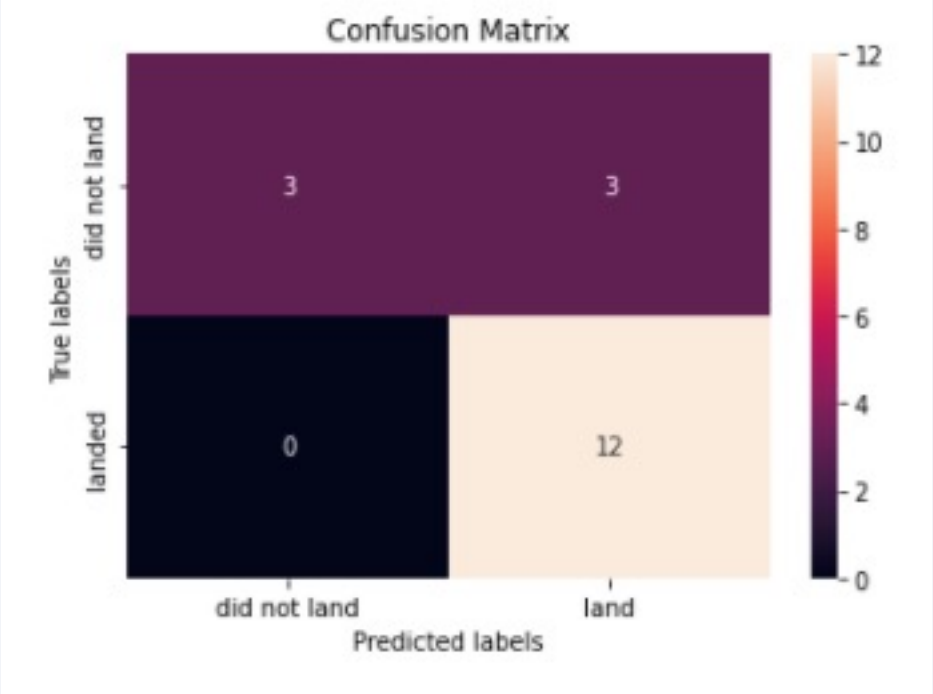
Confusion Matrix

- The confusion matrix shown, displays a high false positive result (i.e., a false success).
- Also, the false negative is also higher as compared to the other 3 algorithm, shown next slide



Confusion Matrix (cont'd)

- The other 3 algorithms with higher accuracy on best estimator, are able to successfully distinguish the true positive and false negative.



Conclusions

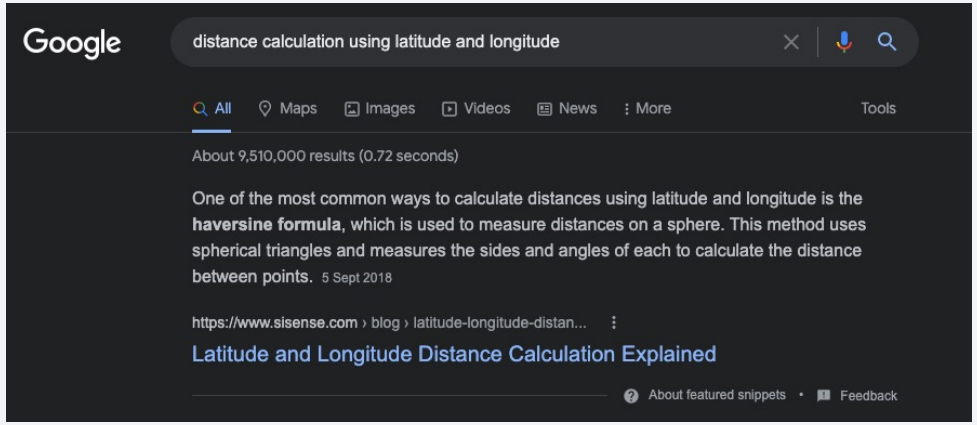
- The success rate has been on an upward trend since 2013, likely due to iteration over the years.
- Based on the scatterplots, it seem that correlation between payload mass and orbits are weak.
- Subsequent scatterplot on payload mass and success rates show mixed result, despite mass between 2000 and 6000kg has relatively higher successes (mixed with failures as well).
- KSC LC 39A launch site has the most successful launches over the years.
- ES-L1, GEO, HEO, SSO orbits have the perfect success rates.
- Decision Tree Classifier Algorithm has the best accuracy score but yield different results on the best estimator score almost every run, likely due to how it works – random.
- The other algorithms scored similar, with minute differences in accuracies.
- Launch sites are nearer to coastlines because of the danger they pose to humans if launch failures.

Conclusions - Improvements

- Despite the earlier findings, there are more analysis required:
  - Correlation, and/or trend between Payload Mass and Flight Number (or over the years).
    - If payload mass have been concentrated within a range at the later years, the out-of-range masses may not have been adequately tested.
  - Orbit count, and maybe the height of orbit types.
    - The higher successes on ES-L1, GEO, HEO, SSO have lower counts, thus may have skewed the overall result.
    - Height may be a factor in determining the success rate – say gravity pull, the atmospheric resistance etc.
  - Location of different launch sites
    - More analysis required to analyse why different launch sites yield different success rates.
  - Identifying the planned failed attempts
    - These may have impacted the accuracy of the result.

Appendix

- IBM Labs – for the python notebook templates and codes.
- Google – for the definition of Haversine formula.



- Stack Overflow – for the various debugging sessions.

Thank you!