IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Daniel Gross
18.1.2023

Outline

Executive Summary

Summary of methodologies

- Launches data from the SpaceX API.
- Data collection and analysis using Python, SQL.
- Data visualization
- Creating a machine learning prediction model with Python.

Summary of all results

All prediction models perform the same with r^2 score of 0.8334

Introduction

Project background and context

- Apply data science for a private space launch company.
- Data from SpaceX API.

Problems you want to find answers

- Predict if the Falcon 9 first stage will land successfully and reduce costs.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collection and web-scraping of launches from the SpaceX API and Wikipedia.
- Perform data wrangling
  - Using Pandas and Numpy. Data was cleaned and processed into numbers to fit prediction models. One-hot encoding was applied.
- Perform exploratory data analysis (EDA) using visualization and SQL

Methodology

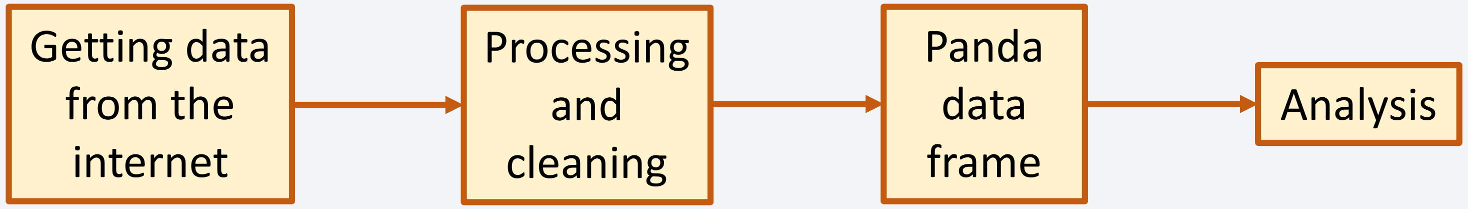
Executive Summary

- Perform interactive visual analytics using Folium and Plotly Dash.

- Perform predictive analysis using classification models
  - Using Pythons’s libraries for KNN, decision tree, SVC and logistic regression prediction models.
  - Deviding the data to a train set to fit the model and a test set to test and evaluate the model’s performance.

Data Collection

- Data sets were collected with the following methods:
  - Get request from the SpaceX API, decoding the response as json with .json() and turning it into a panda data frame using .json_normalize(), and processing it by cleaning the data, replacing missing values, etc.
  - Additionally, web scraping from Wikipedia using BeautifulSoup for Falcon 9 launch records HTTP table. Then it was converted into a panda data frame for analysis with Python.

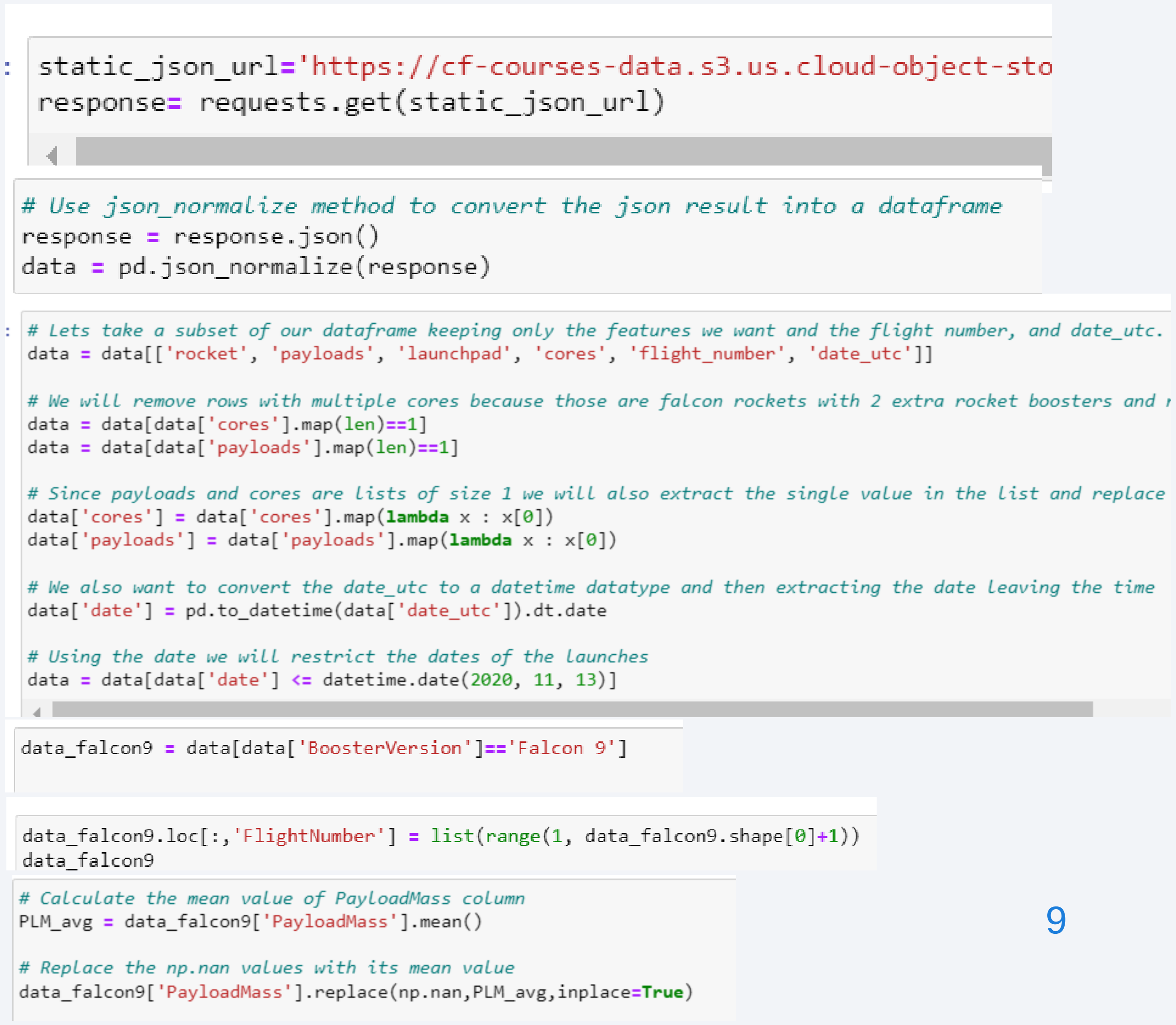


Data Collection – SpaceX API

- Using the get request to the SpaceX API to collect data, clean the requested data and basic data wrangling.

- GitHub URL:

https://github.com/danigross5/IBM_Data_Science_peer_grade-capstone_project-/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



```
static_json_url='https://cf-courses-data.s3.us.cloud-object-sto
response= requests.get(static_json_url)
```

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

```
# 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 r
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
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)]
```

```
data_falcon9 = data[data['BoosterVersion']=='Falcon 9']
```

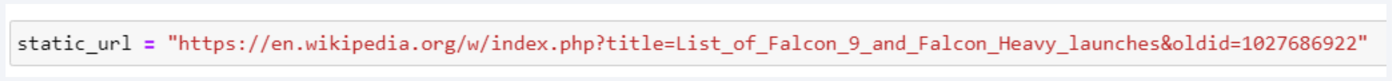
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
# Calculate the mean value of PayloadMass column
PLM_avg = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan,PLM_avg,inplace=True)
```

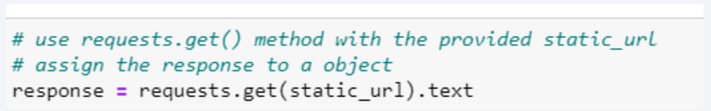
Data Collection - Scraping

- Web scrapping Falcon 9 launch records with BeautifulSoup.
- Parsed the table and converted it into a pandas data frame.
- GitHub URL:

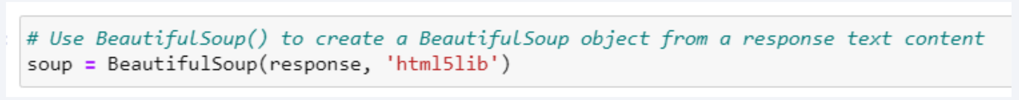
https://github.com/danigross5/IBM_Data_Science_peer_grade-capstone_project-/blob/main/jupyter-labs-webscraping.ipynb



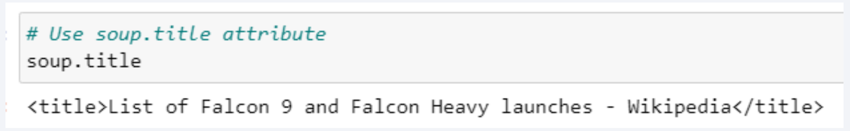
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```



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

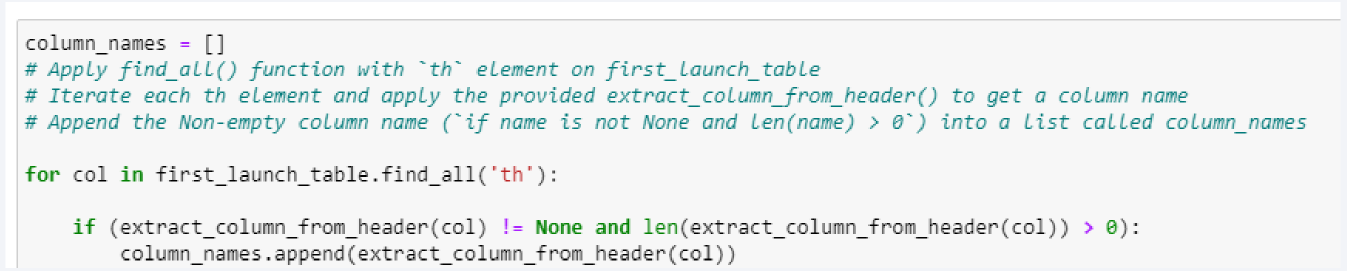


```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html5lib')
```



```
# Use soup.title attribute
soup.title
```

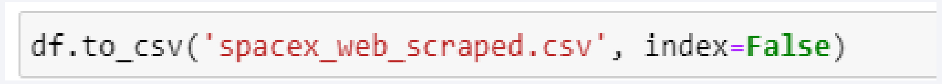
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```



```
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 col in first_launch_table.find_all('th'):

    if (extract_column_from_header(col) != None and len(extract_column_from_header(col)) > 0):
        column_names.append(extract_column_from_header(col))
```

Parsing the HTML table

Exporting the data



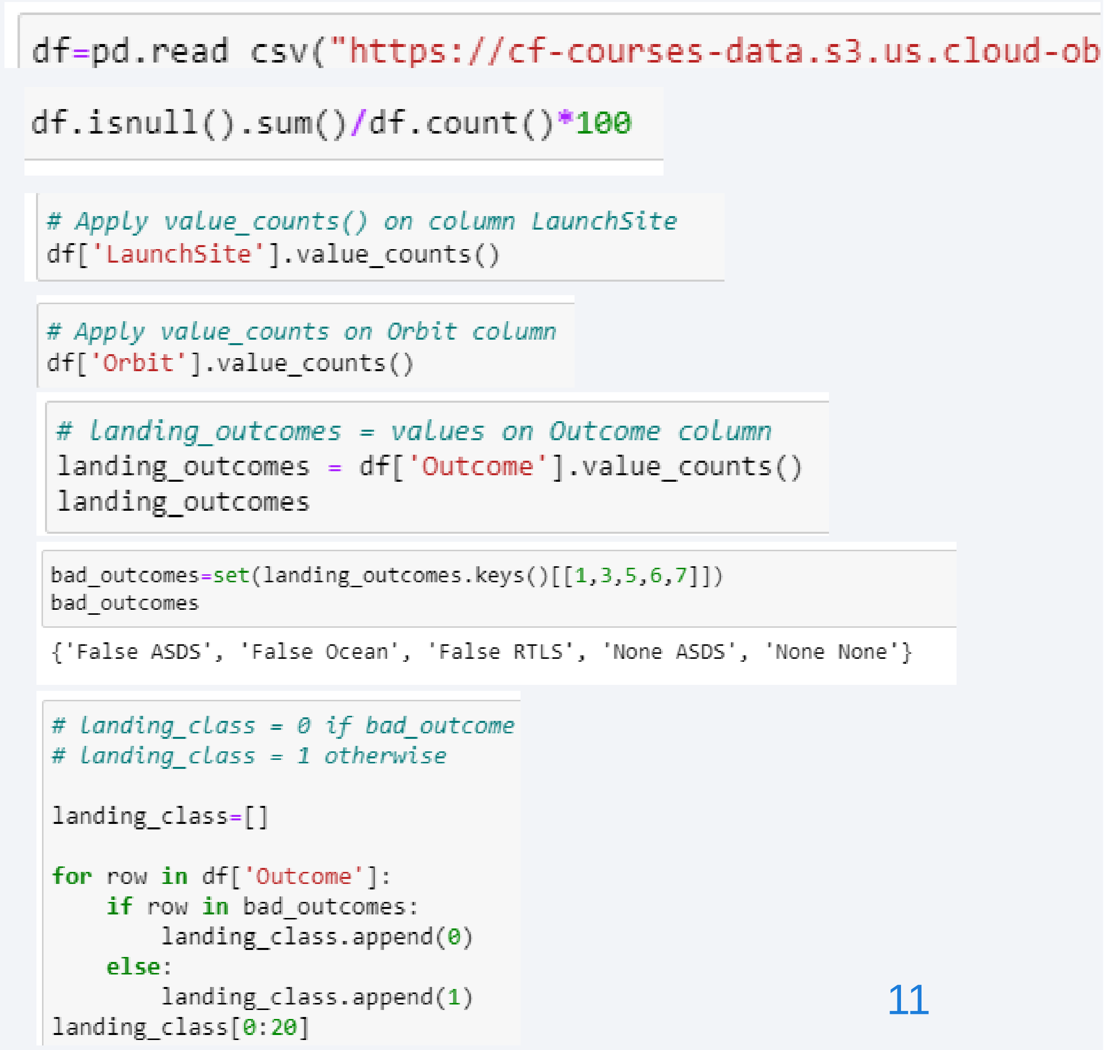
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- The objectives were to perform exploratory data analysis and determine training labels.
- The tasks were:
  - To calculate the number of launches on each site.
  - To calculate the number and occurrence of each orbit.
  - To calculate the number and occurrence of mission outcome per orbit type.
  - To create a landing outcome label from Outcome column

GitHub URL:

https://github.com/danigross5/IBM_Data_Science_peer_grade-capstone_project-/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-ob
```

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

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

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

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

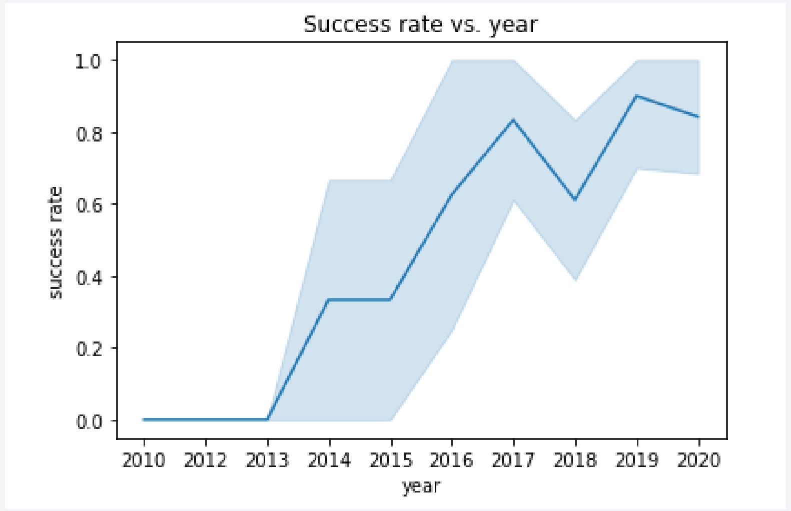
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

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

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

landing_class=[]

for row in df['Outcome']:
    if row in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
landing_class[0:20]
```

EDA with Data Visualization

- Plotted charts to analyze the effect of factors on success rates :
  - Flight number vs. payload mass
  - Flight number vs. launch site
  - Launch site vs. payload mass.
  - Orbit
  - Flight number vs. orbit
  - Payload mass vs. orbit
  - Year



Example: Year vs. success rate

GitHub URL:

https://github.com/danigross5/IBM_Data_Science_peer_grade-capstone_project-/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

Using EDA with SQL following queries in the Jupiter notebook:

- *Names of the unique launch sites.*
- *5 records where launch sites begin with the string 'CCA'.*
- *Total payload mass carried by boosters launched by NASA (CRS)*
- *Average payload mass carried by booster version F9 v1.1*
- *Date when the first successful landing outcome in ground pad was achieved.*
- *List the names of the boosters which have success in drone ship and have payload mass greater than 4,000 but less than 6,000.*
- *total number of successful and failure mission outcomes*
- *names of the booster versions which have carried the maximum payload mass.*
- *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.*

GitHub URL:

https://github.com/danigross5/IBM_Data_Science_peer_grade-capstone_project-/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- All launch sites has been marked. Map objects such as markers, circles, lines were added to mark the success or failure of launches for each launch site on the folium map.
- Launch outcomes (0 for failure and 1 for success) were assigned.
- Color-labeled marker clusters to identified which launch sites have relatively high success rate.
- The distances between a launch site to costlines, highways, railways and cities were calculated to determine whether a certain distance is been kept.

GitHub URL:

https://github.com/danigross5/IBM_Data_Science_peer_grade-capstone_project-/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Interactive dashboard with Plotly dash were built:
- A pie charts to show the total launches by a certain launch site.
- A scatter plot to show the relationship between outcome and payload mass for the different booster version.

GitHub URL:

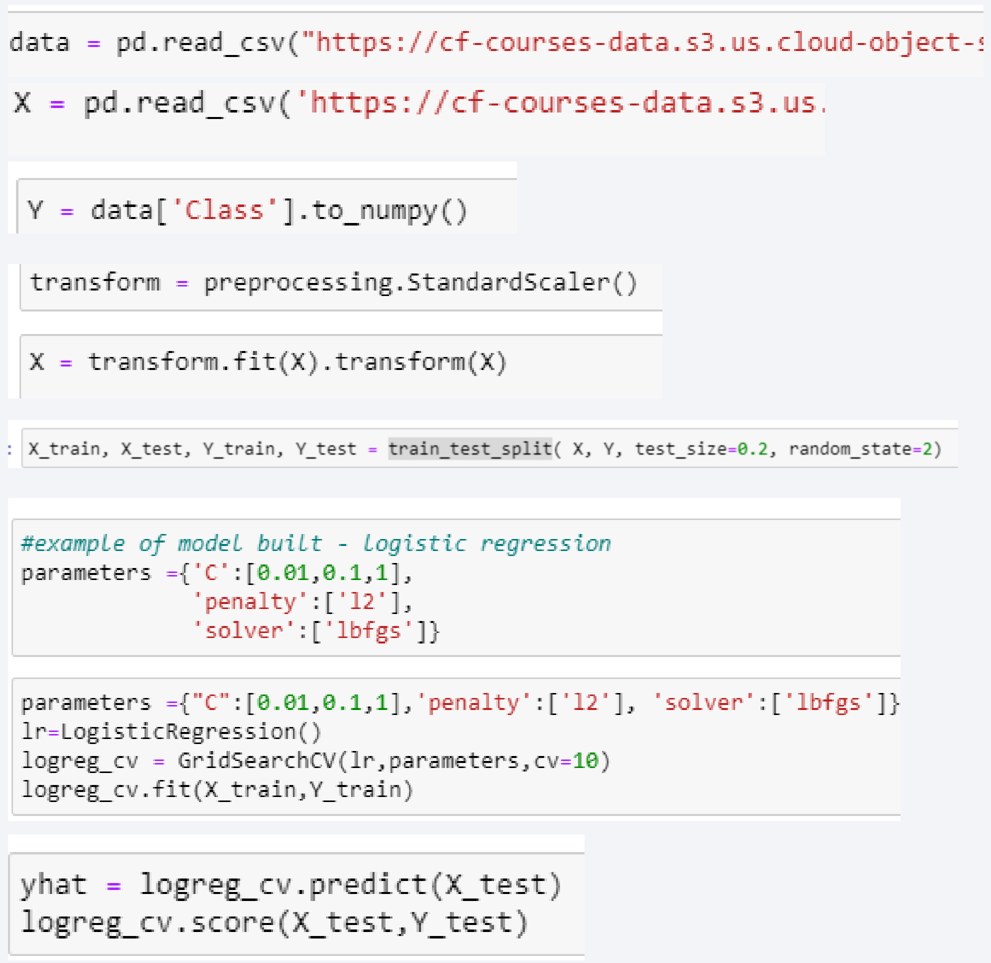
https://github.com/danigross5/IBM_Data_Science_peer_grade-capstone_project-/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- You need present your model development process using key phrases and flowchart
- Data loading and standardizing using numpy, pandas and sklearn.
- Splitting it to train and test data sets with train_test_split.
- Different machine learning models were built and tuned using GridSearchCV.
- Accuracy was the metric for the model. The model was improved using feature engineering and algorithm tuning.
- The best performing classification model was found based on accuracy scores.

GitHub URL:

https://github.com/danigross5/IBM_Data_Science_peer_grade-capstone_project-/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb



```
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-s
X = pd.read_csv('https://cf-courses-data.s3.us.
```

```
Y = data['Class'].to_numpy()
```

```
transform = preprocessing.StandardScaler()
```

```
X = transform.fit(X).transform(X)
```

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
```

```
#example of model built - logistic regression
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr,parameters,cv=10)
logreg_cv.fit(X_train,Y_train)
```

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

Results

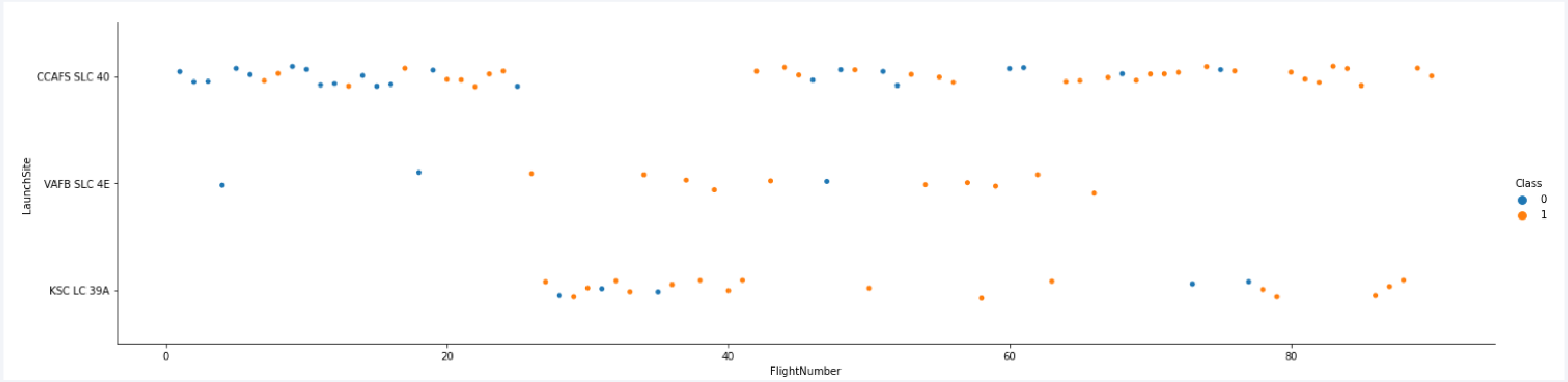
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

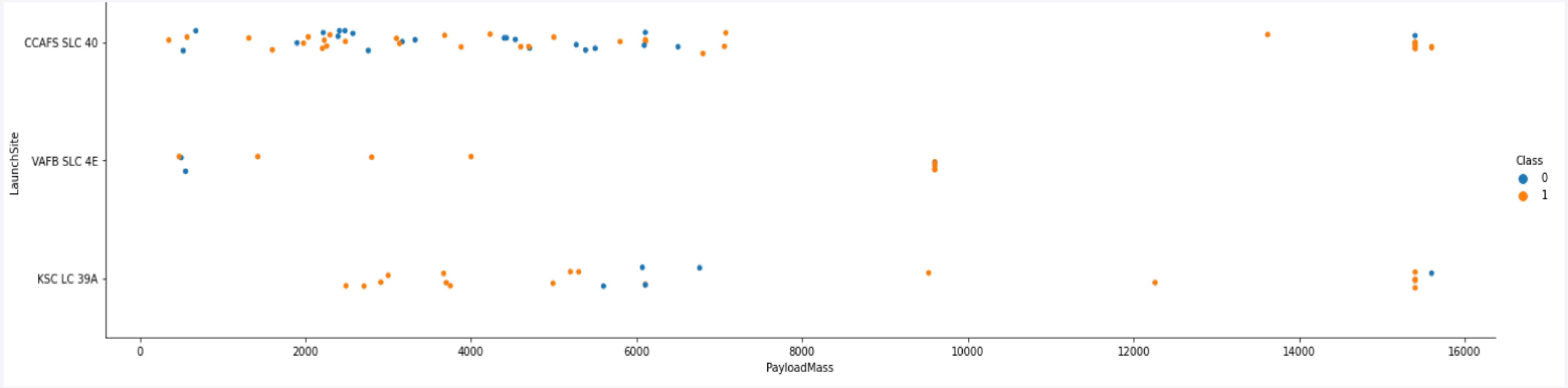
Insights drawn from EDA

Flight Number vs. Launch Site

The more launches are executed at a specific site, the higher success rate is.



Payload vs. Launch Site



The greater the payload mass, the greater mission success rate is.

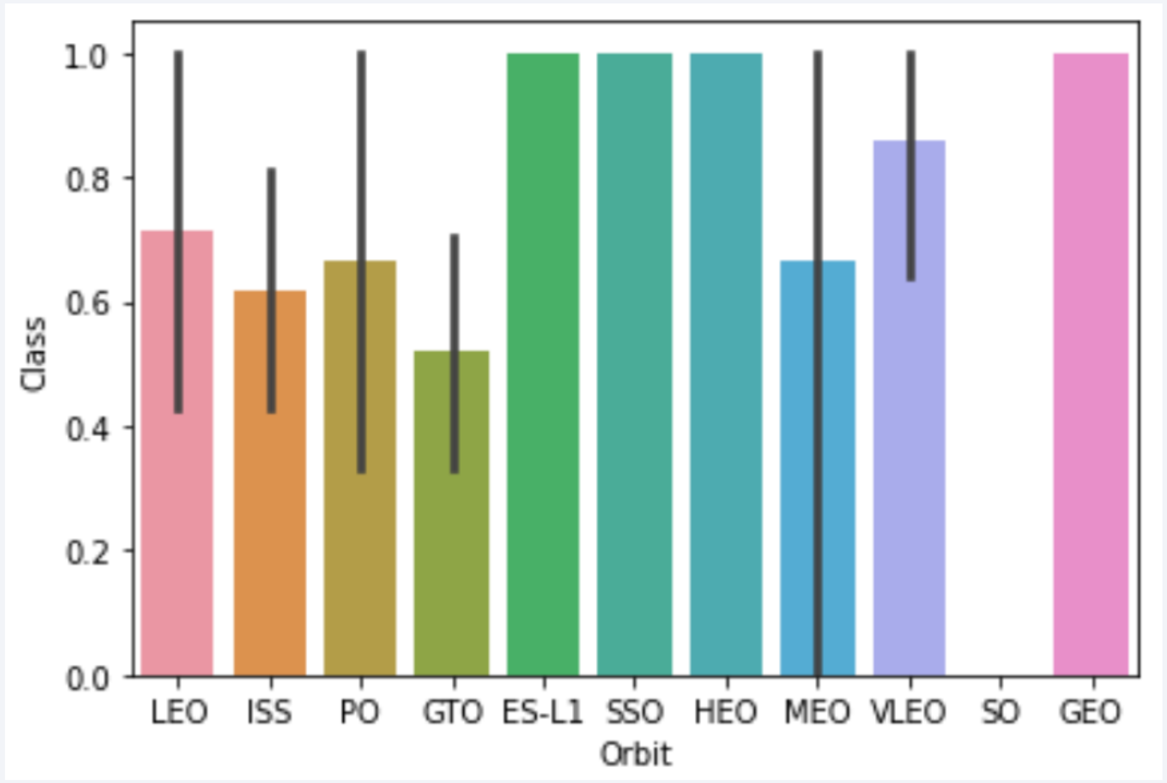
Success Rate vs. Orbit Type

The highest success rates (100%) are for the orbits:

- ES-L1
- SSO
- HEO
- GEO

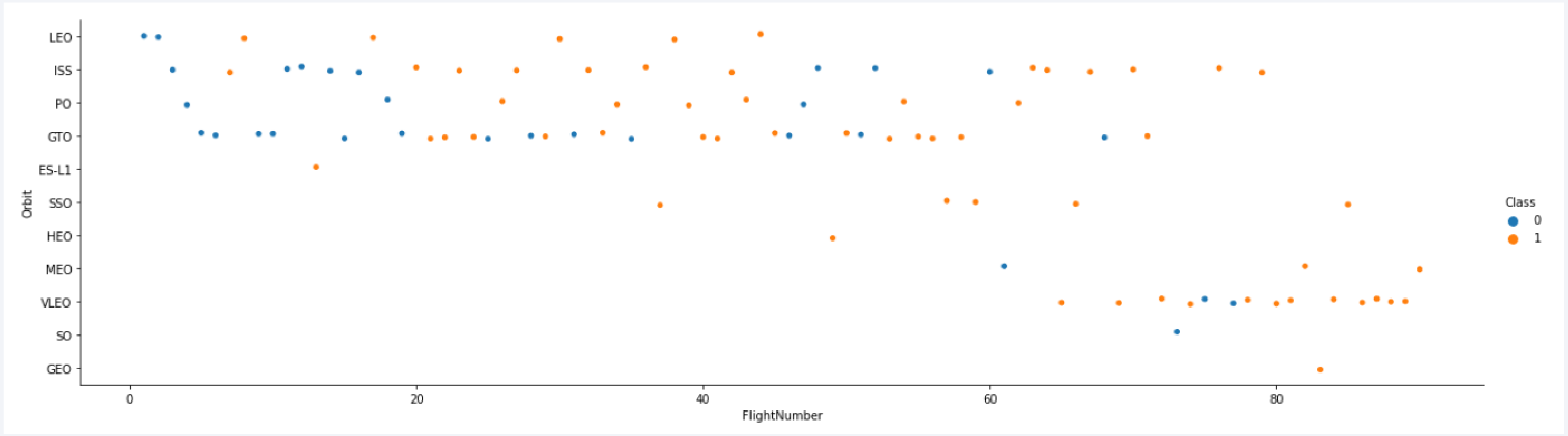
Lowest success rate (~50%) is for orbit:

- GTO

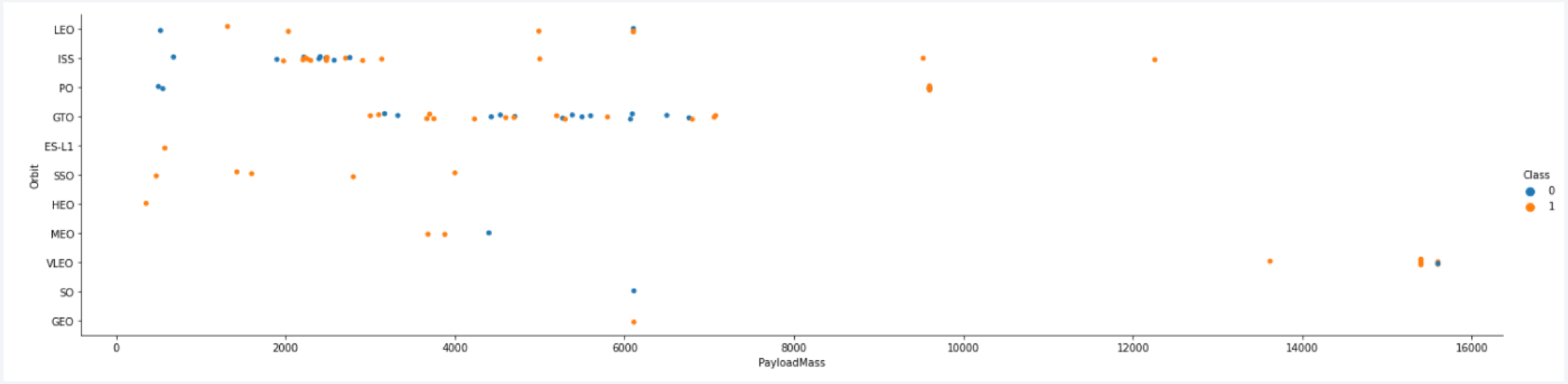


Flight Number vs. Orbit Type

It is shown that generally the more launches are executed for an orbit, the higher the success rate is.



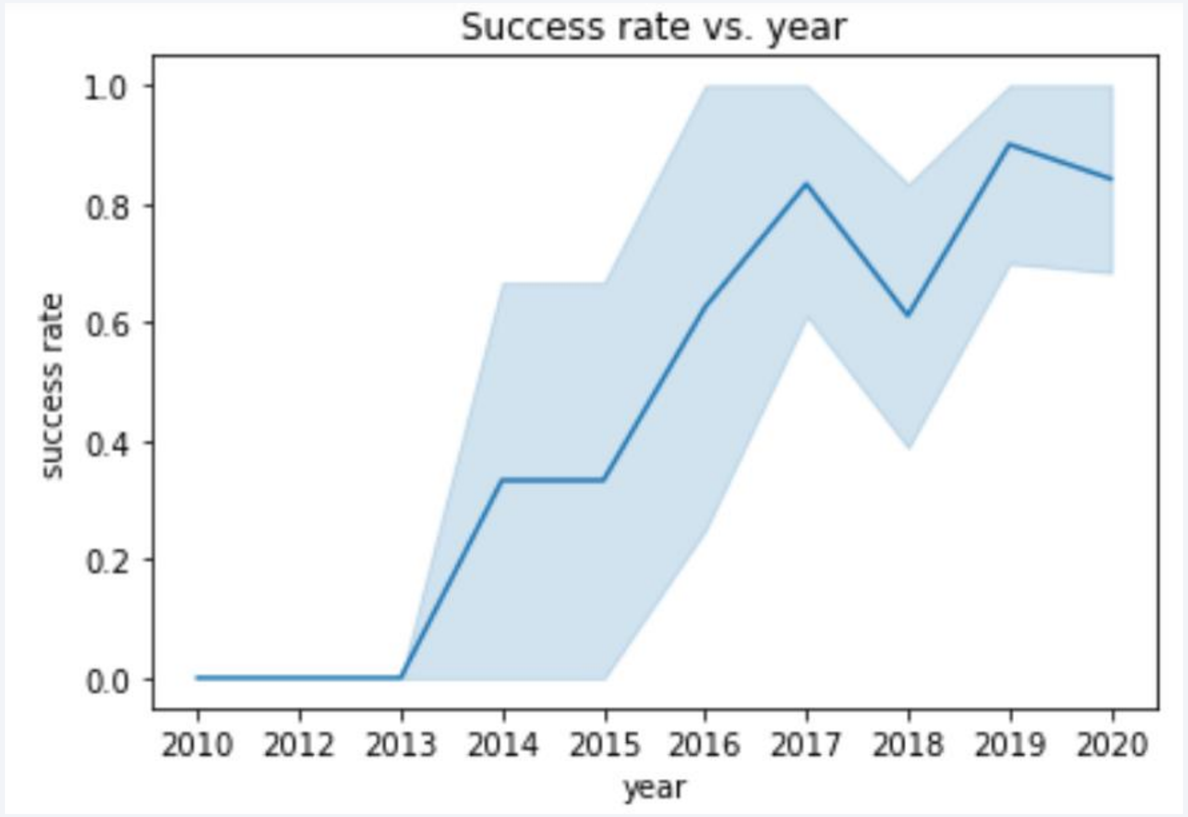
Payload vs. Orbit Type



It is shown that the higher payload for an orbit, the higher the success rate is.

Launch Success Yearly Trend

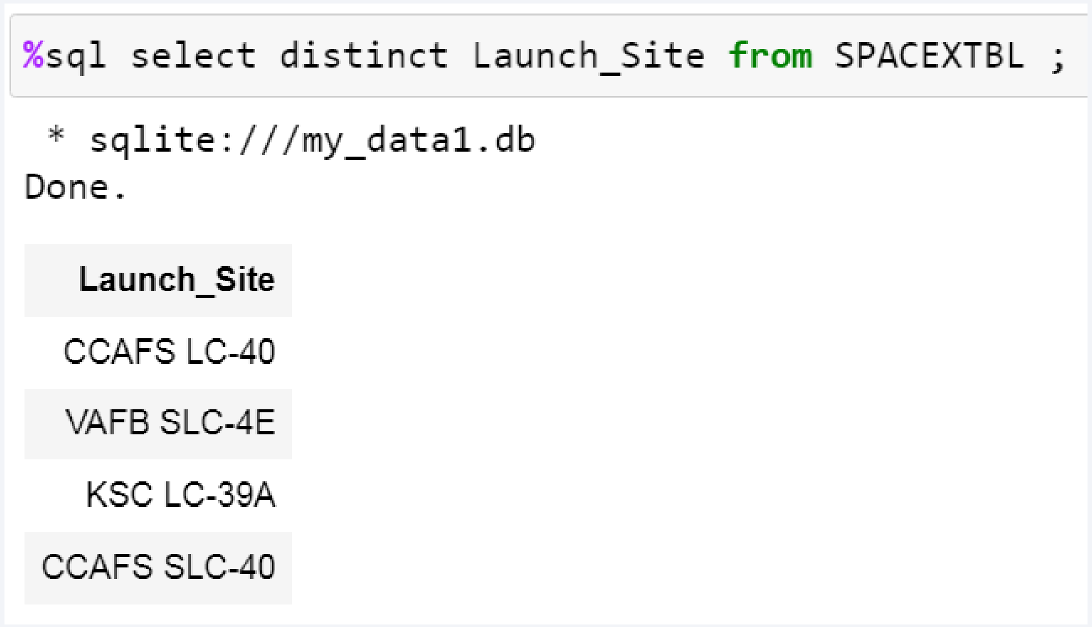
The general trend is increasing success rate from 2013 to 2020



All Launch Site Names

SQL 1/10

The key word “DISTINCT” was used to show unique launch sites.



```
%sql select distinct Launch_Site from SPACEXTBL ;
```

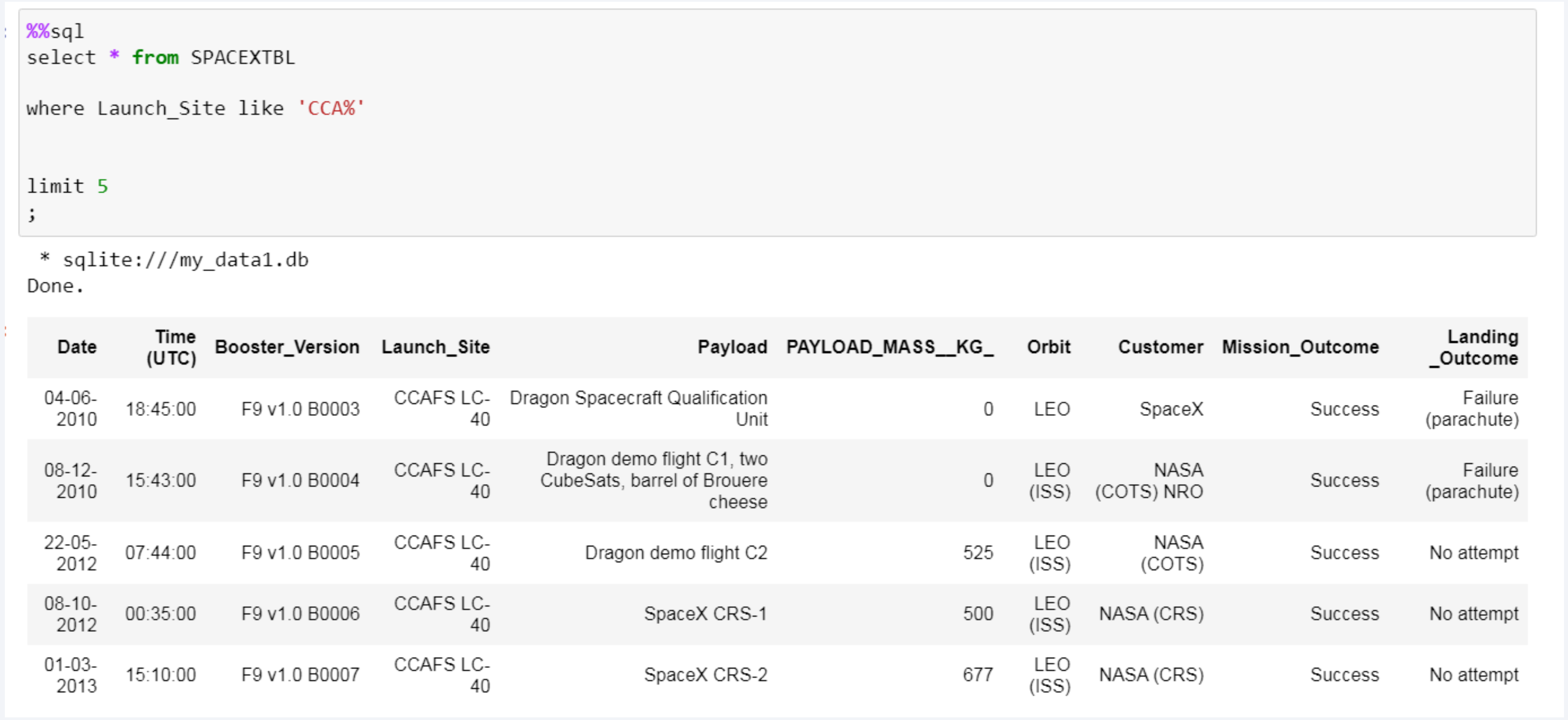
```
 * sqlite:///my_data1.db
Done.
```

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

Launch Site Names Begin with 'CCA'

SQL 2/10

- The “WHERE” key word was used to find launch sites begin with `CCA`
- The key word “LIMIT” was used to find the first 5 rows



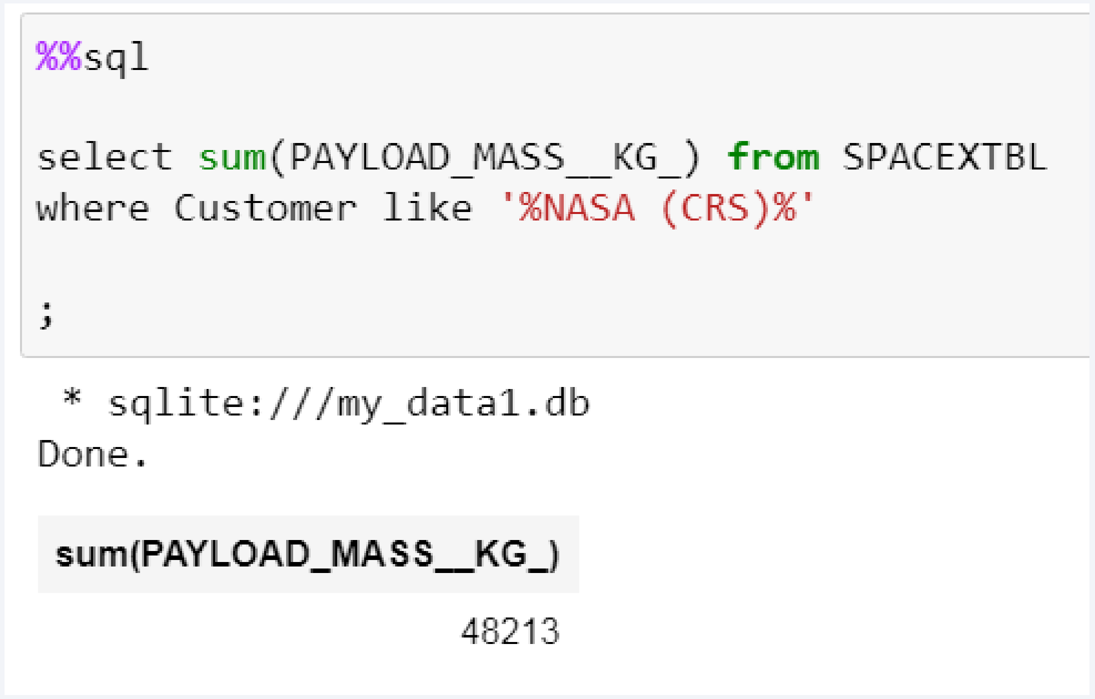
```
%%sql
select * from SPACEXTBL

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

- The “WHERE” clause was used to find the “NASA (CRS)” customer
- The function SUM() was used to calculate the total payload mass (48,213 kg).



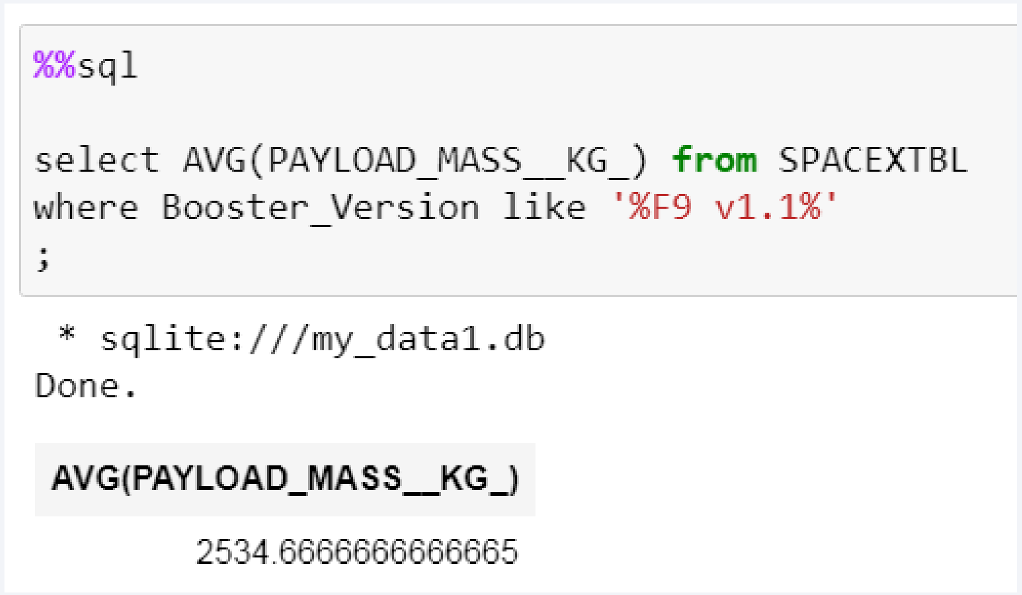
```
%%sql

select sum(PAYLOAD_MASS__KG_) from SPACEXTBL
where Customer like '%NASA (CRS)%'

;
```

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

sum(PAYLOAD_MASS__KG_)
48213

Average Payload Mass by F9 v1.1

- The following query was used to find the average payload mass (2,534 kg) by Falcon 9 v1.1.
- The function AVG() calculates the average.
- The "WHERE" clause was used to find the Falcon 9 v1.1 booster version.



```
%%sql

select AVG(PAYLOAD_MASS__KG_) from SPACEXTBL
where Booster_Version like '%F9 v1.1%'
;
```

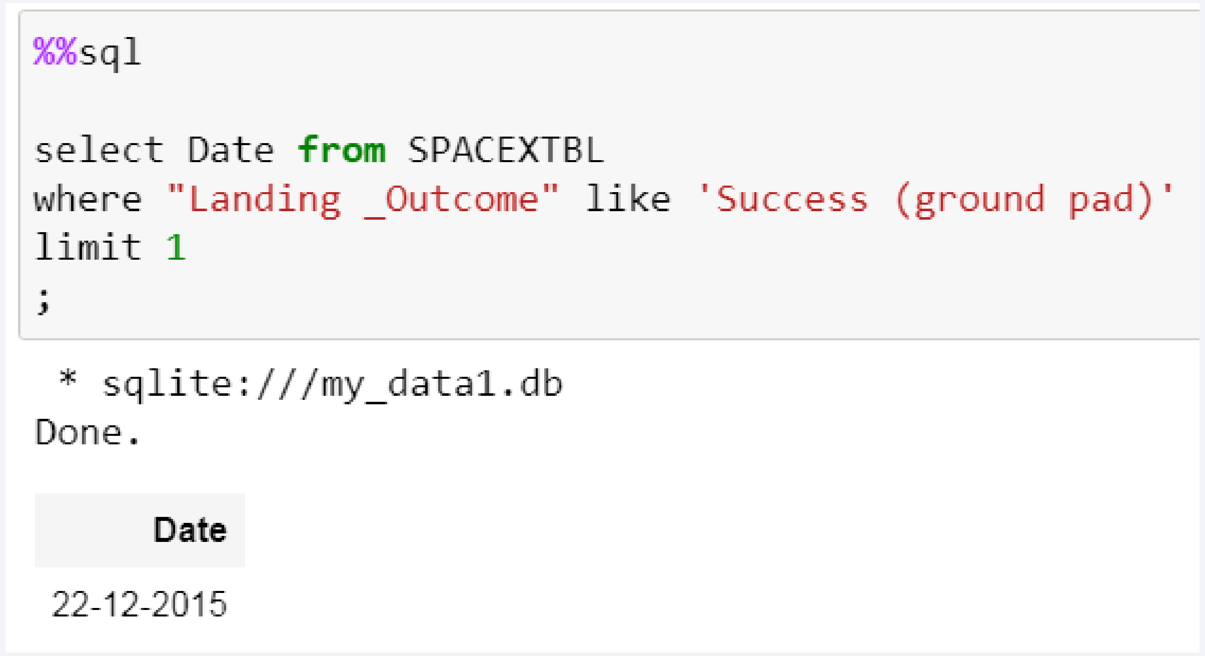
```
 * sqlite:///my_data1.db
Done.
```

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

SQL 5/10

- The “WHERE” clause was used to find the first successful landing at a ground pad (22-12-2015).
- The “LIMIT” was to select the first date (the data frame was ordered by dates)



```
%%sql

select Date from SPACEXTBL
where "Landing _Outcome" like 'Success (ground pad)'
limit 1
;
```

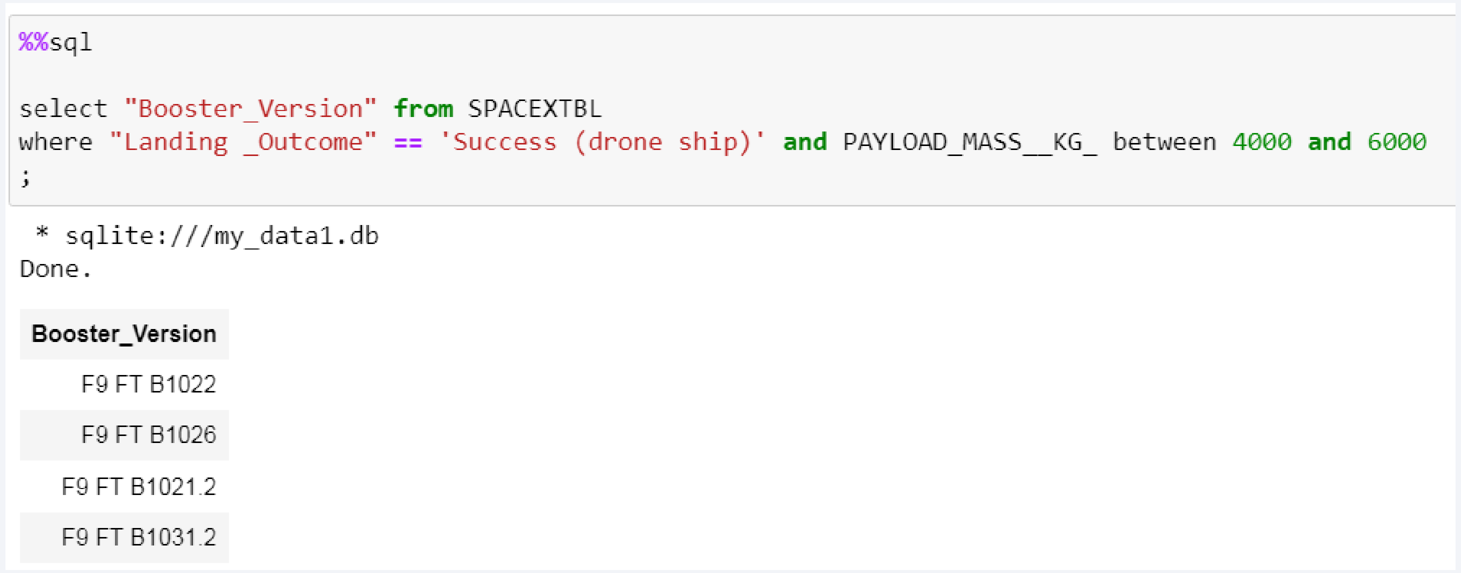
```
 * sqlite:///my_data1.db
Done.
```

Date
22-12-2015

Booster version of successful drone ship landing with payload mass between 4,000 and 6,000 kg

SQL 6/10

- The “WHERE” clause was used to find the successful landing between 4,000 to 6,000 kg.
- The “BETWEEN” clause was used to determine the target range of minimum and maximum payload mass values.
- All boosters were “Falcon 9 - FT”.



```
%%sql

select "Booster_Version" from SPACEXTBL
where "Landing _Outcome" == 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000
;
```

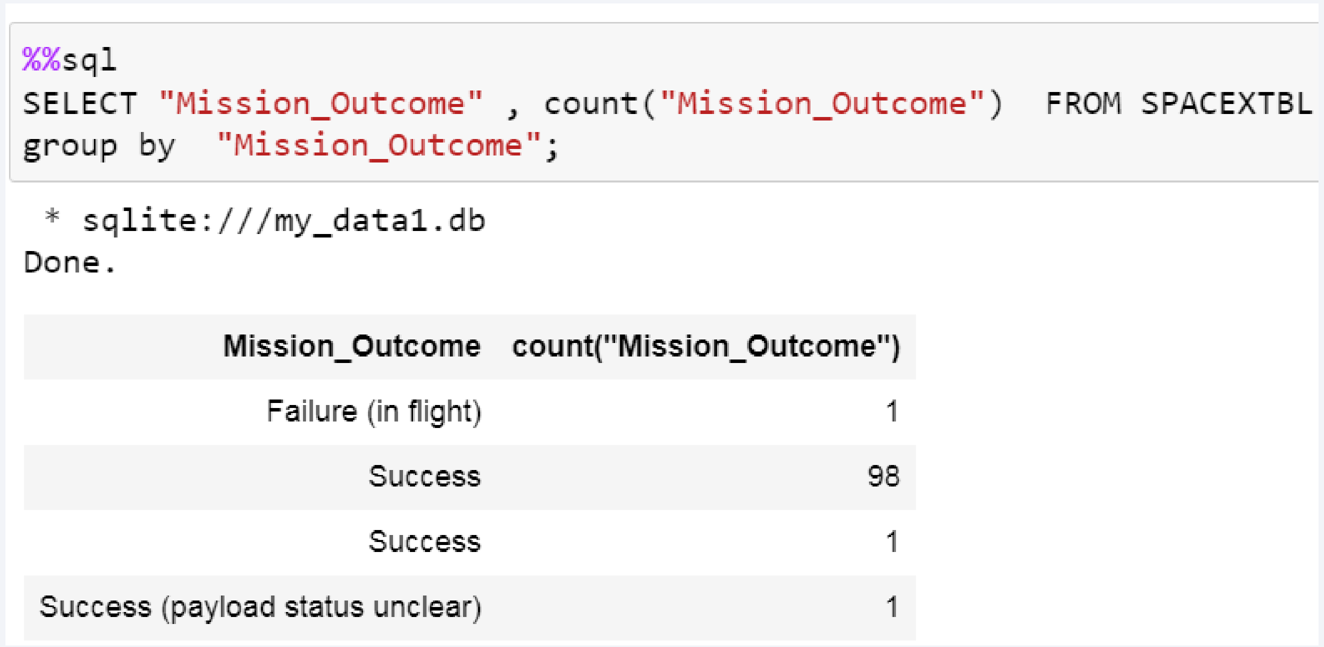
```
 * sqlite:///my_data1.db
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

SQL 7/10

- “COUNT” function was used to count the number of successful and failure mission outcomes.
- “GROUP BY” clause was used to find the different mission outcomes.
- The results are 100 successful missions (99.01%) and 1 failure (0.99%),



```
%%sql
SELECT "Mission_Outcome" , count("Mission_Outcome")  FROM SPACEXTBL
group by  "Mission_Outcome";
```

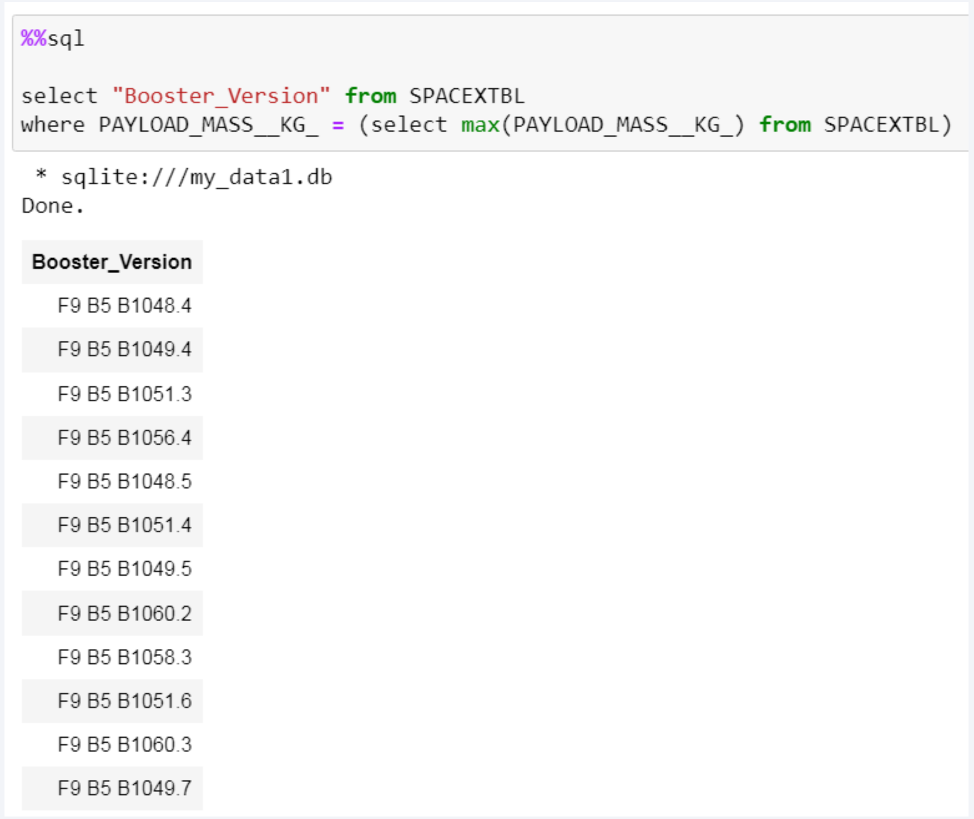
```
 * sqlite:///my_data1.db
Done.
```

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

Boosters Carried Maximum Payload

SQL 8/10

- MAX() function was used to find tha maximal payload value.
- The booster versions carrying the maximal payload mass were “Falcon 9 - B5”



```
%%sql

select "Booster_Version" from SPACEXTBL
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

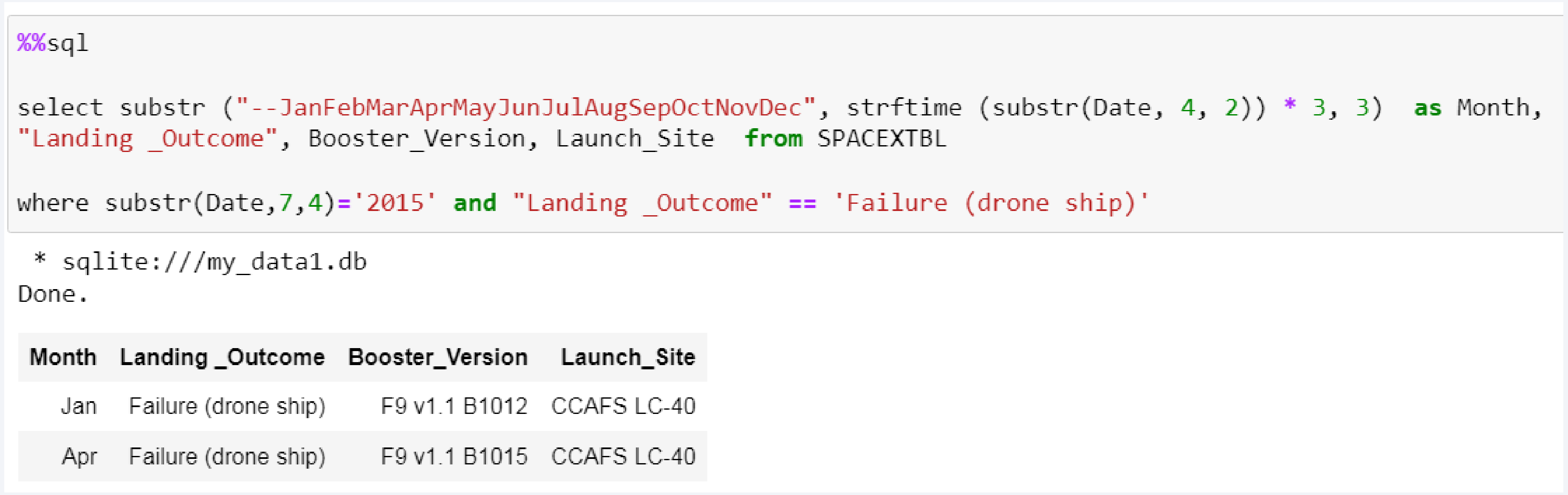
```
 * sqlite:///my_data1.db
Done.
```

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
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 drone ship failure launch records

SQL 9/10

- The list below shows the month, failure landing outcomes in drone ship, booster versions, and launch site names for in year 2015.
- “WHERE” clause was used to find the missions for 2015 that were failure drone ship.



```
%%sql

select substr ("--JanFebMarAprMayJunJulAugSepOctNovDec", strftime (substr(Date, 4, 2)) * 3, 3)  as Month,
"Landing _Outcome", Booster_Version, Launch_Site  from SPACEXTBL

where substr(Date,7,4)='2015' and "Landing _Outcome" == 'Failure (drone ship)'
```

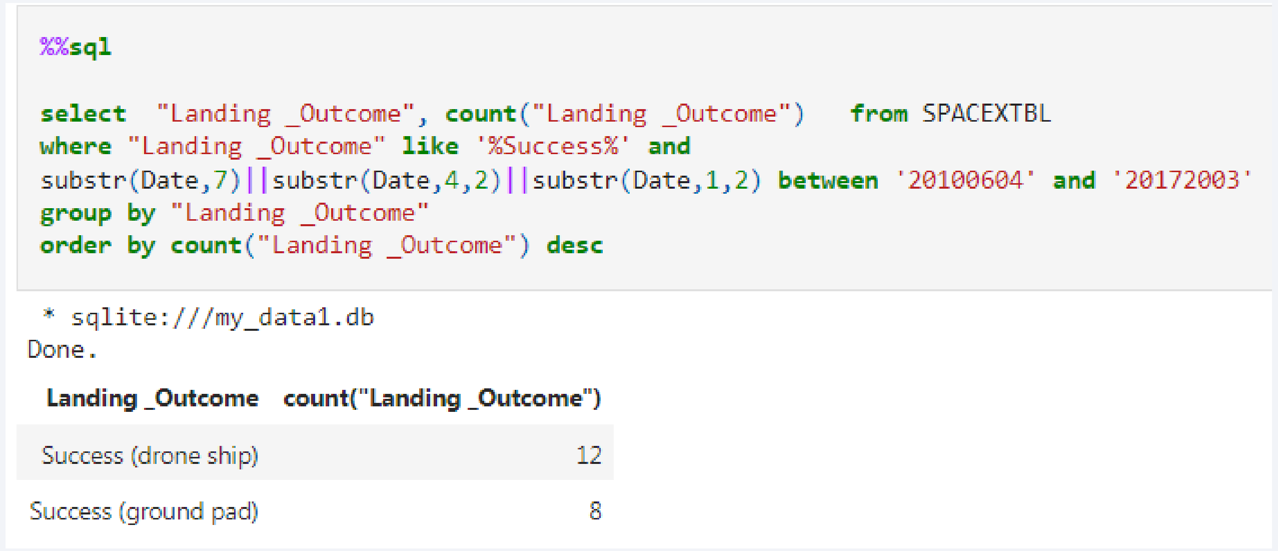
```
 * sqlite:///my_data1.db
Done.
```

Month	Landing _Outcome	Booster_Version	Launch_Site
Jan	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Apr	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

SQL 10/10

- “WHERE” clause was used to find the successful landing outcomes between the dates 2010-06-04 and 2017-03-20.
- “GROUP BY” clause was used to find each different successful outcome.
- “ORDER BY” clause was used to order the outcomes according to the count.



```
%%sql

select  "Landing _Outcome", count("Landing _Outcome")   from SPACEXTBL
where "Landing _Outcome" like '%Success%' and
substr(Date,7)||substr(Date,4,2)||substr(Date,1,2) between '20100604' and '20172003'
group by "Landing _Outcome"
order by count("Landing _Outcome") desc
```

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

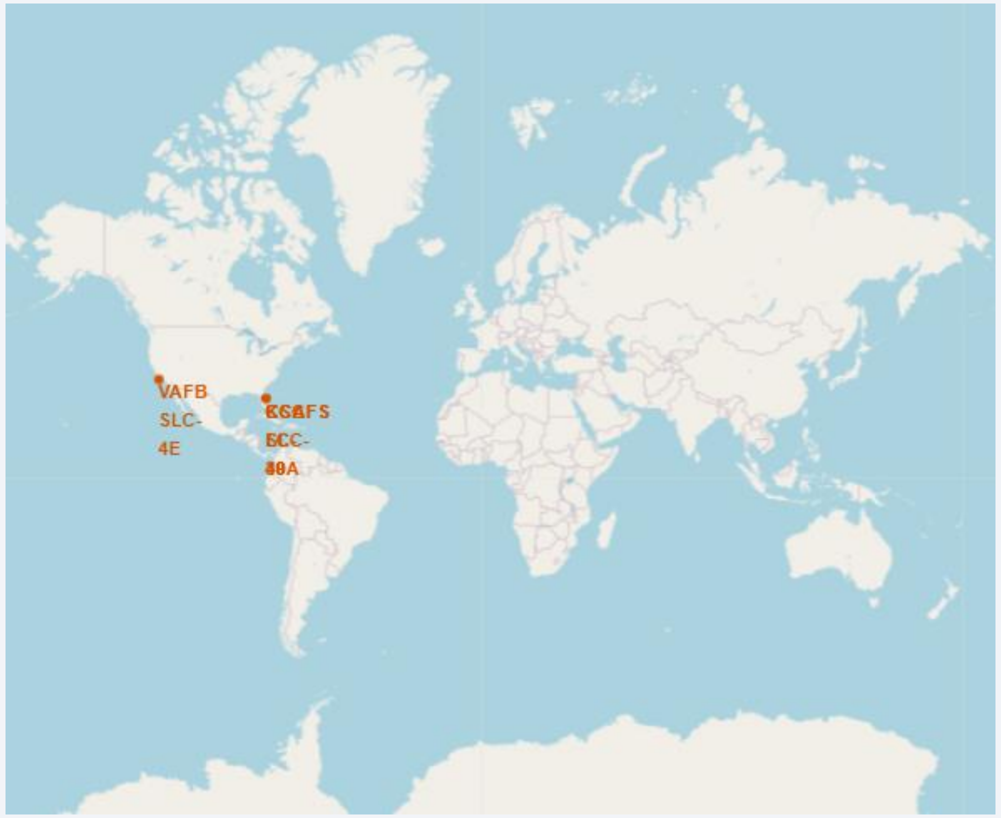
Landing _Outcome	count("Landing _Outcome")
Success (drone ship)	12
Success (ground pad)	8

Section 3

Launch Sites Proximities Analysis

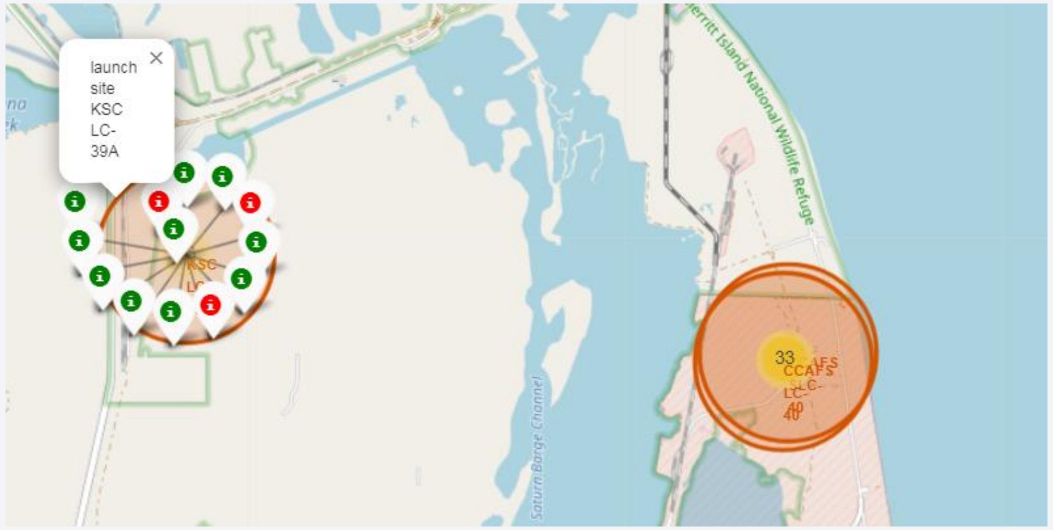
SpaceX Launch Sites

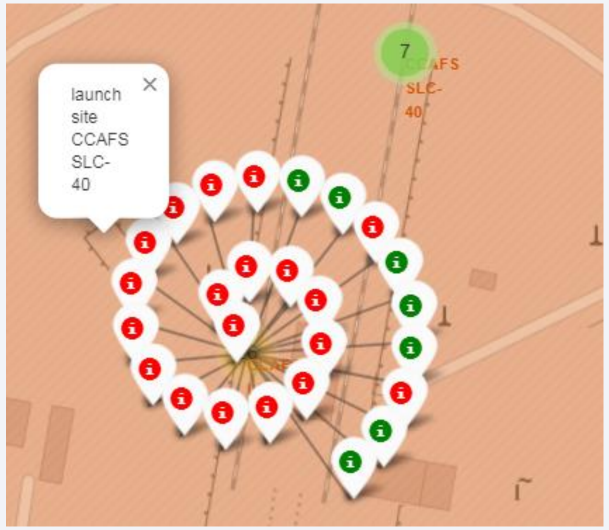
- We can see all SpaceX sites are in both sides of the US (Florida and California) near the sea for both land and sea landing.

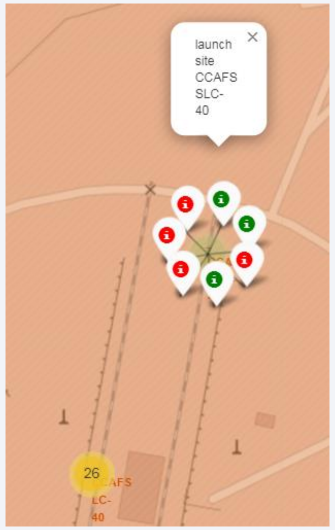


Launch sites and launch outcome with color labels for outcome

Florida





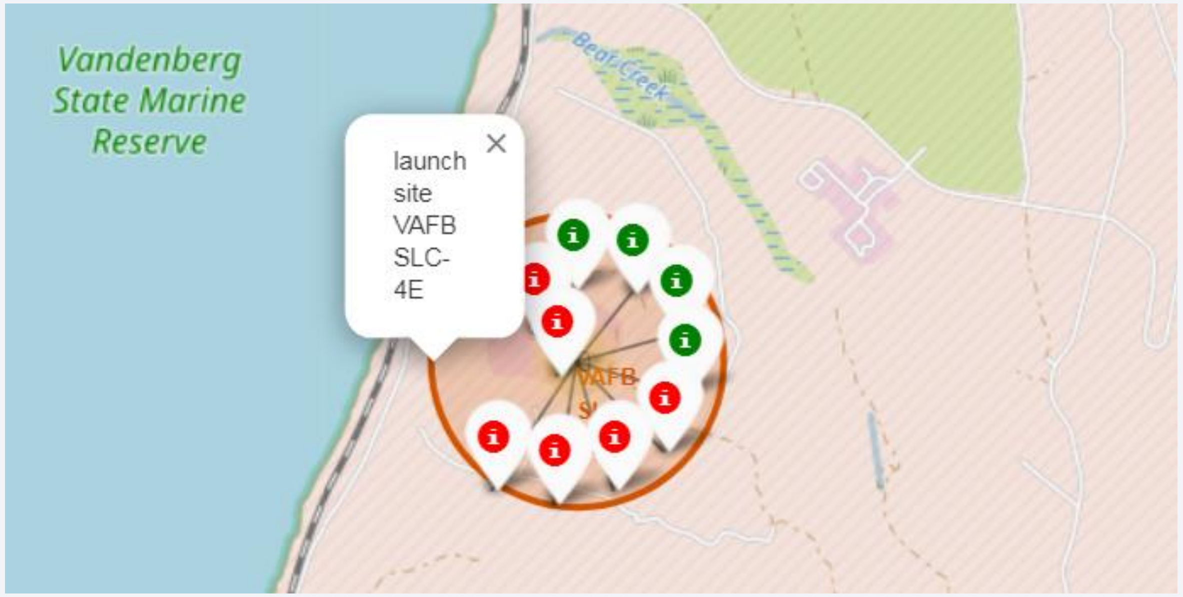


🟢 - Success

🔴 - Failure

Launch sites and launch outcome with color labels for outcome

California

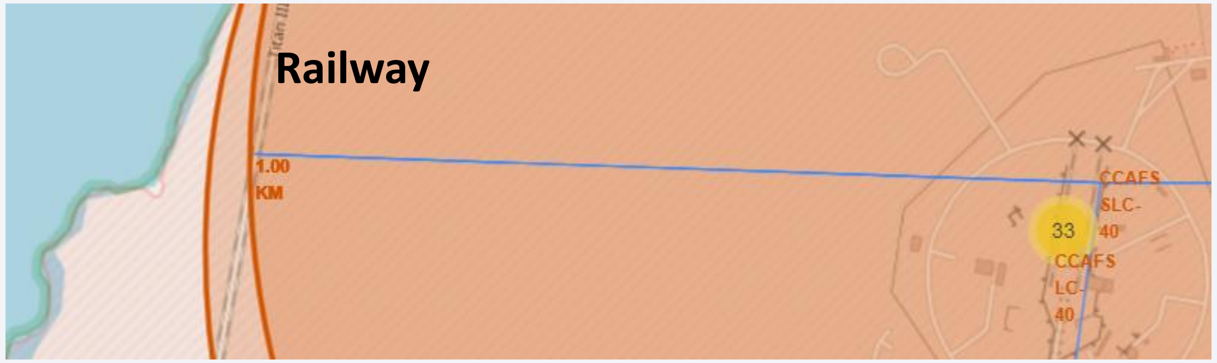


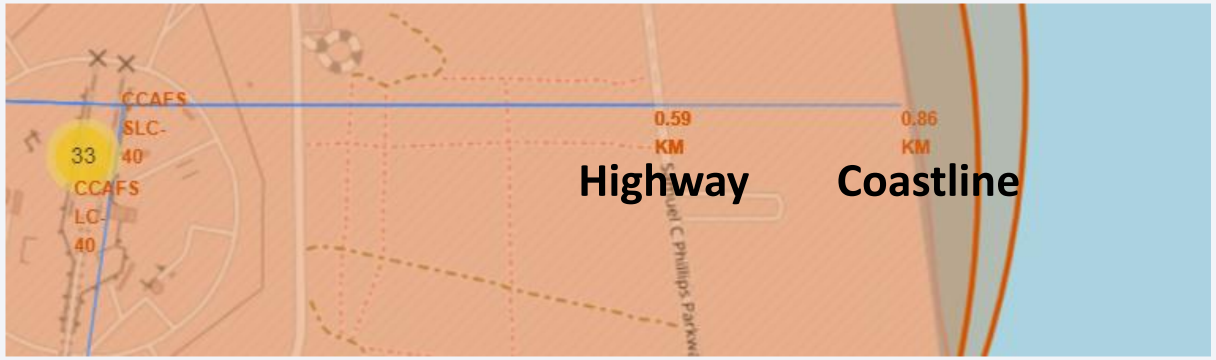
🟢 - Success

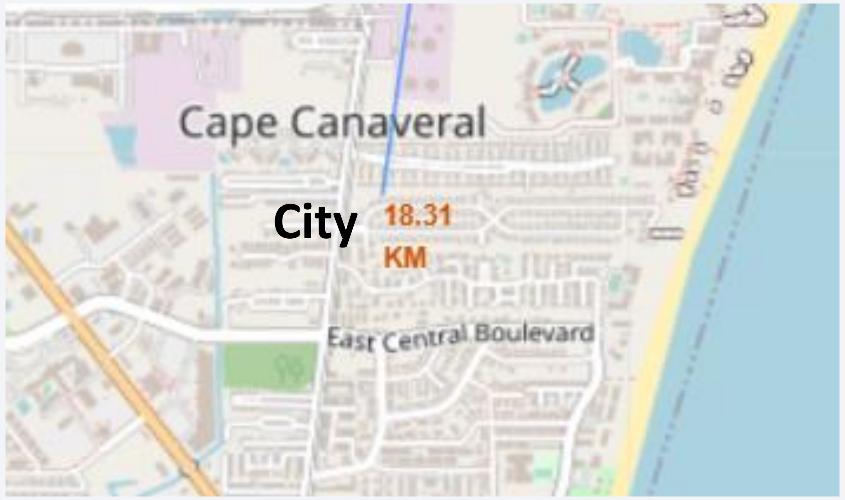
🔴 - Failure

Land marks distances from CCAFS SLC-40

- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed



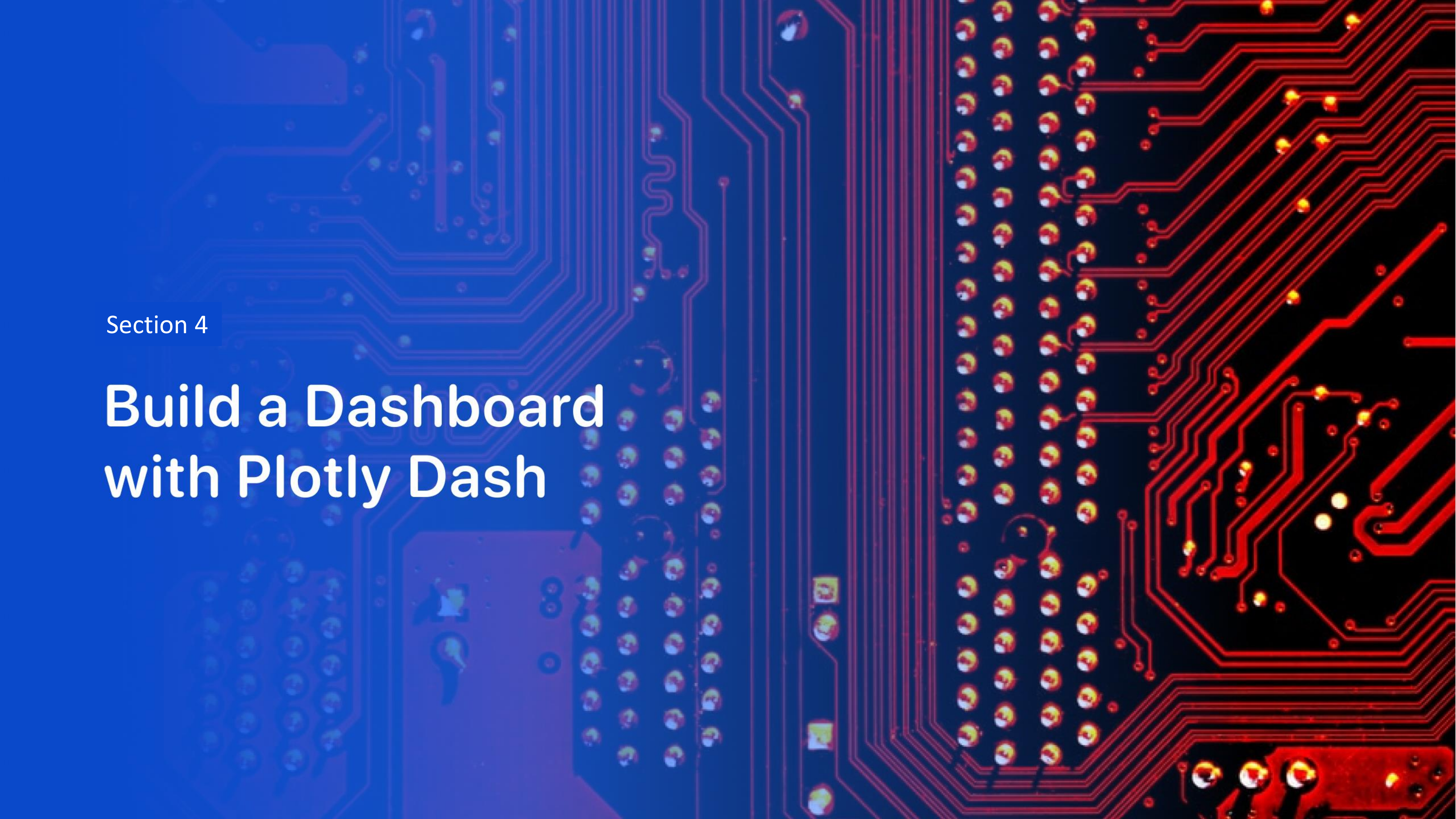




Far from a city (18.3 km) and close (1 km or less) from facilities

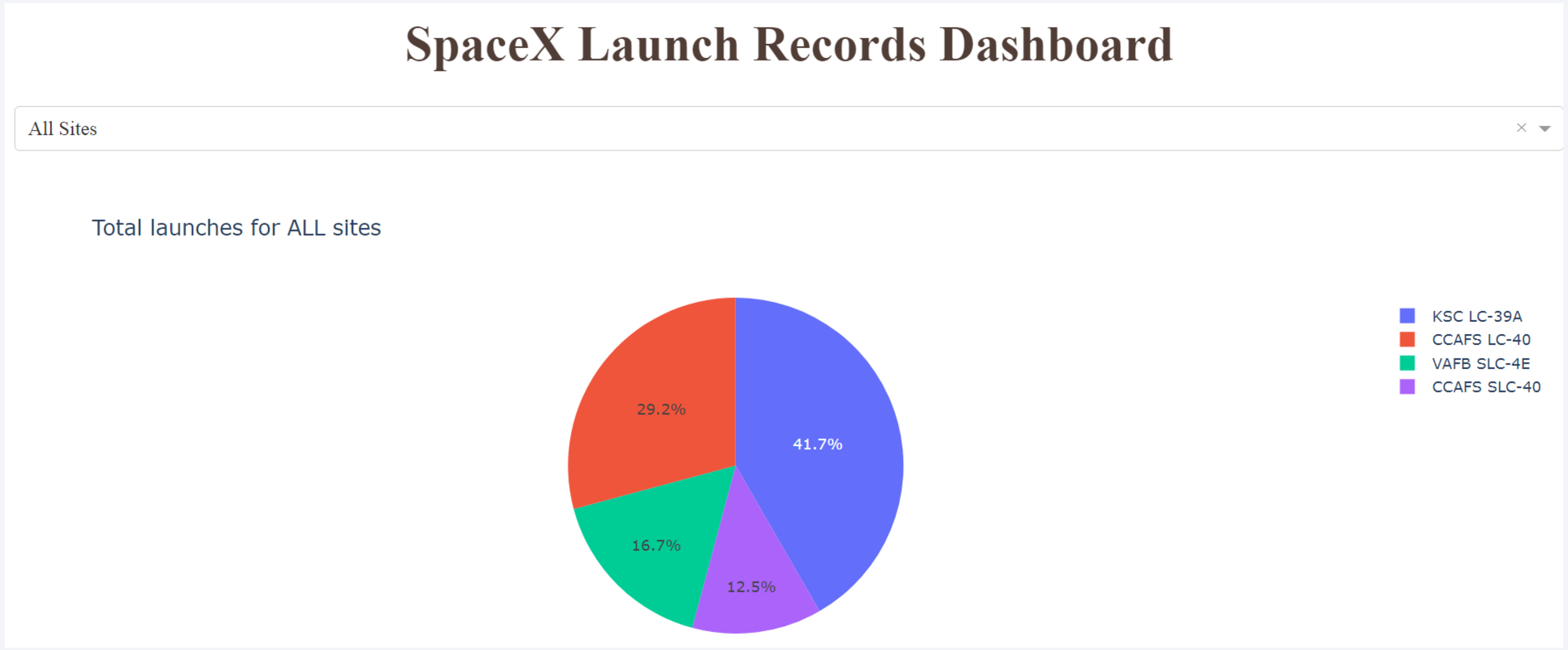
Section 4

Build a Dashboard with Plotly Dash



Launch Success count For All Sites

- A pie chart presenting the total success launches count as a percentage of the total launches.
- KSC LC-39A has the most successful launches from all sites (10 launches, 41.7%).

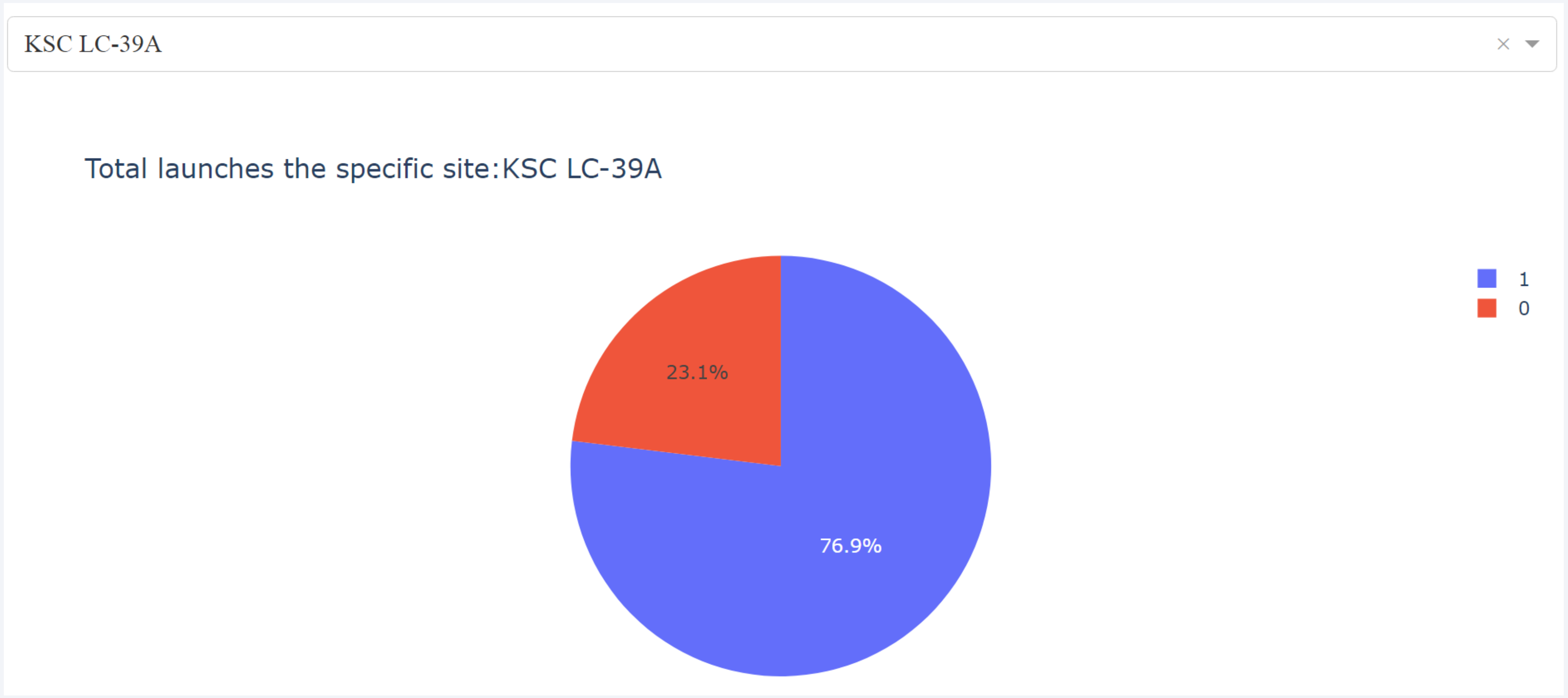

SpaceX Launch Records Dashboard

All Sites

Total launches for ALL sites

Launch Site With The Highest Success Ratio

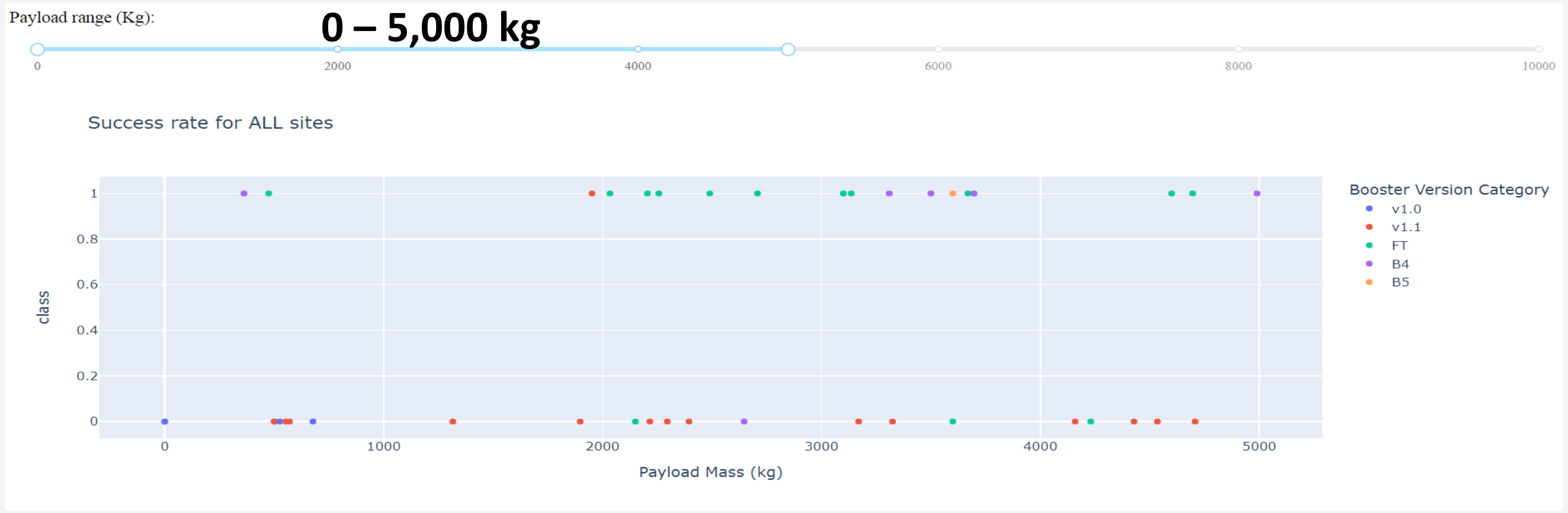
- KSC LC-39A has the highest success ratio among all sites (76.9%).

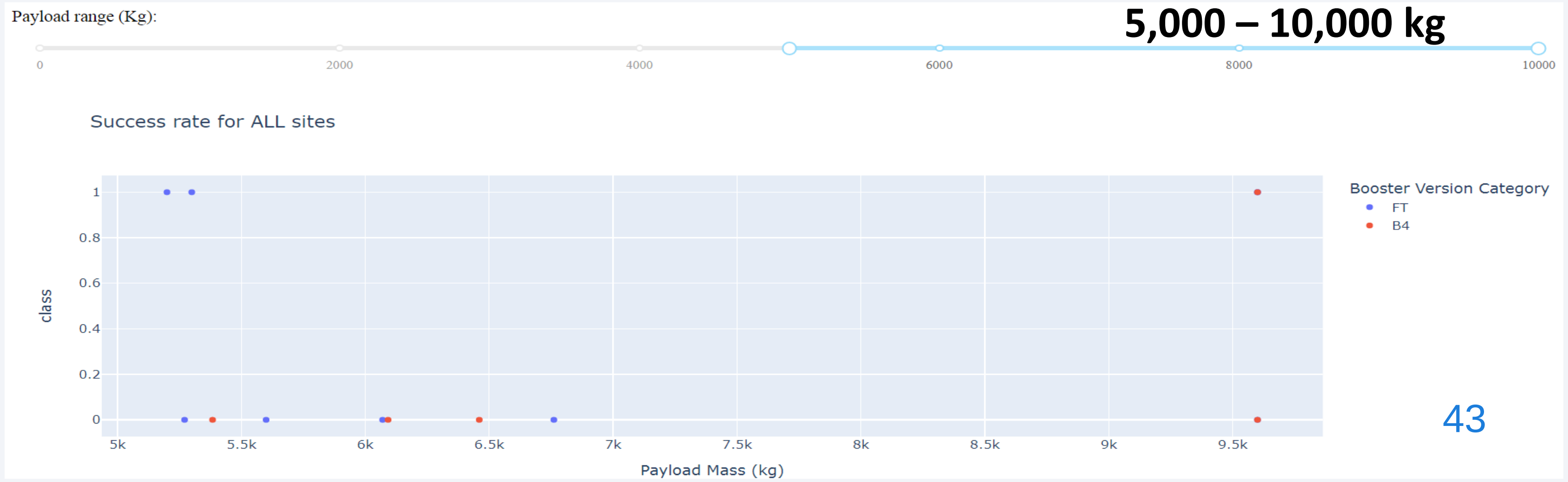

KSC LC-39A

Total launches the specific site:KSC LC-39A

Payload vs. Launch Outcome

- Highest success ratio is between 3,000 – 4,000 kg **(~70%)**
- Lowest success ratio is between 6,000 – 7,000 kg **(0%)**
- Best success ratio is for the booster "B5" **(1/1 – 100%)**



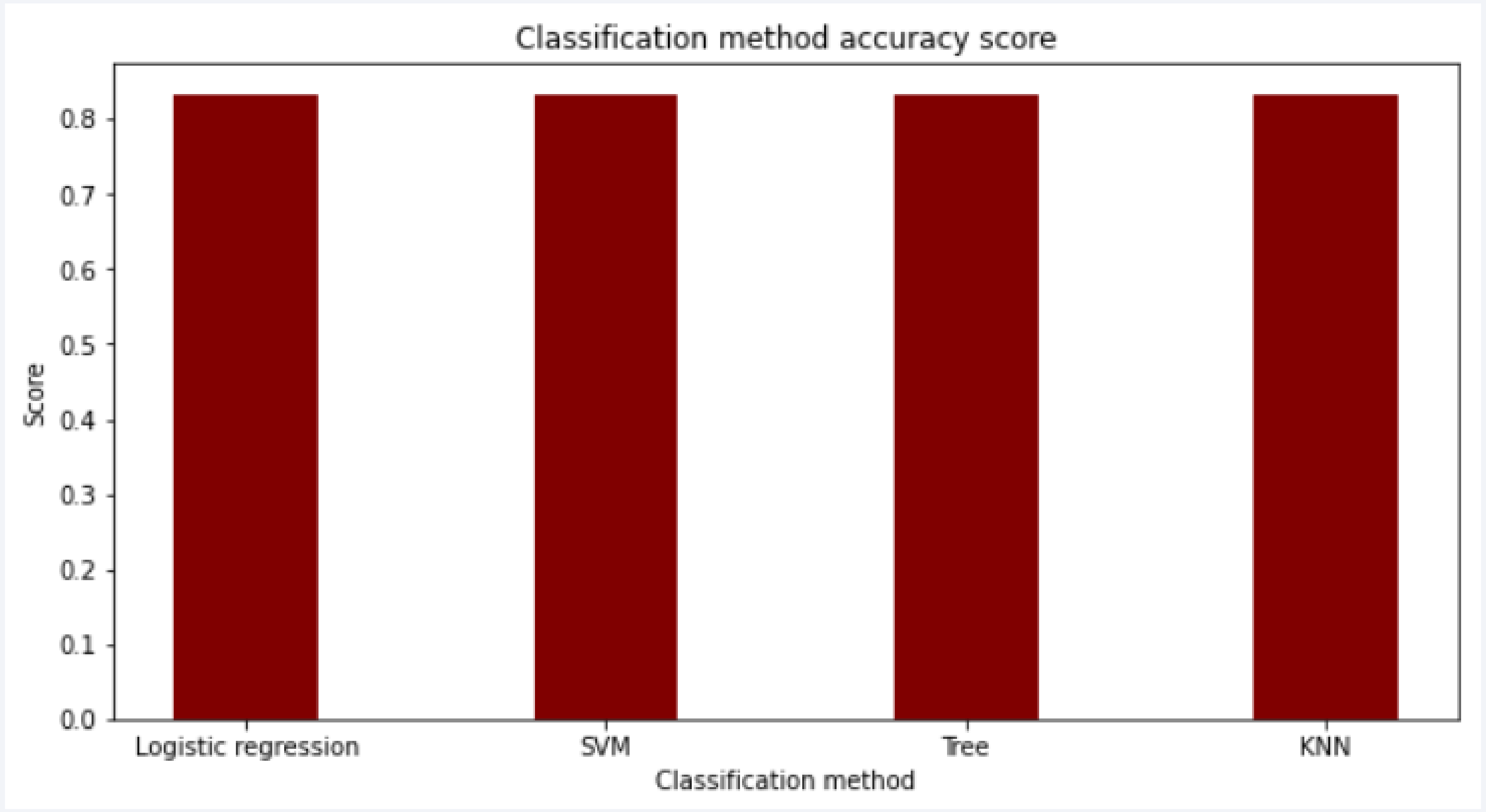


Section 5

Predictive Analysis (Classification)

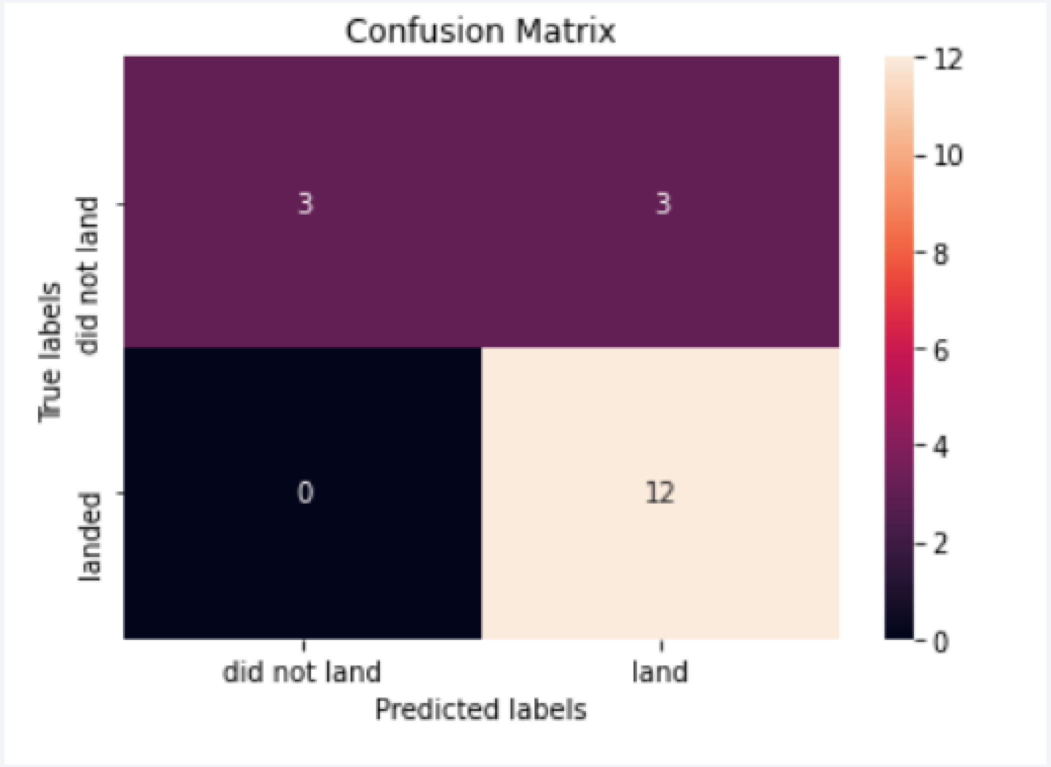
Classification models Accuracy

- All models perform the same



Confusion Matrix

- Since all models performed the same the confusion matrixes are the same.
- The main problem is the false positives. Unsuccessful landing marked as successful landing by the classifier.



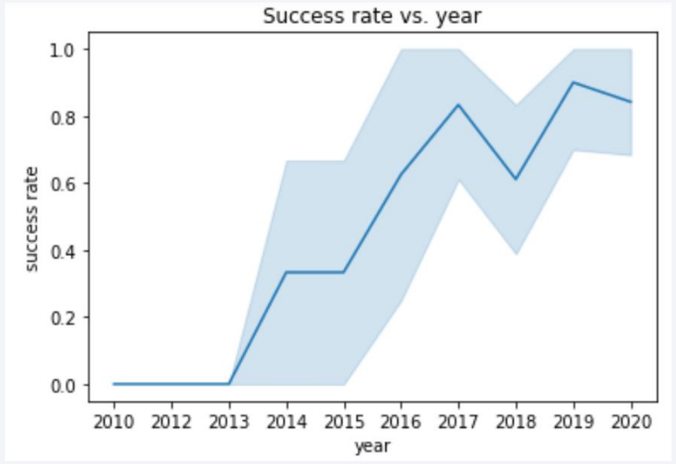
Conclusions

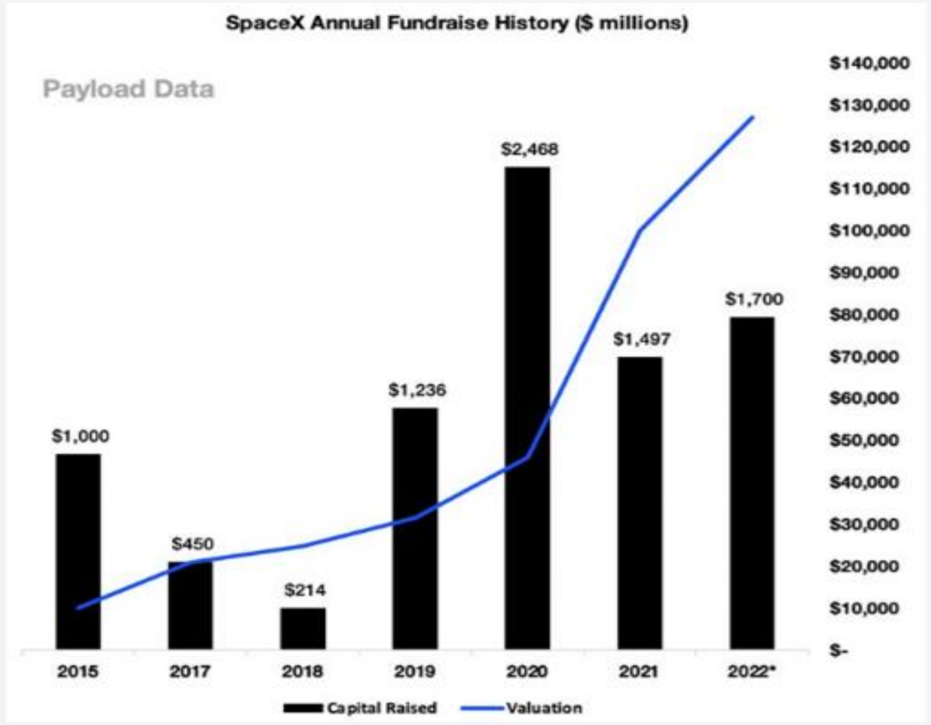
1. The larger the flight amount at a launch site, the greater the success rate at a launch site.
2. Launch success rate started to increase in 2013 till 2020.
3. Orbits ES-L1, GEO, HEO, SSO had the most success rate while GTO is the least successful orbit.
4. KSC LC-39A had the most successful launches of any sites.
5. All classification models perform the same.
6. Sites should be close to facilities and far from cities.

Innovative insight – Low success rate in 2018

There is a decrease in fund raising in 2017 and a decrease in success rate in 2018. Lower funds can result in a less quality working power and equipment maintenance. That can explain this trend.

The lower success rates in 2018 can explain the even lower fundraising in 2018.





https://www.teslarati.com/spacex-127-billion-valuation-new-funding-round-2022/

Thank you!