IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mahendra Patil
9-Sep-2023

Outline

Executive Summary

Summary of Methodologies:

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis (Classification)

Summary of Results:

- Exploratory Data Analysis (EDA) results
- Geospatial analytics
- Interactive dashboard
- Predictive analysis of classification models

Introduction

- SpaceX Falcon 9 rocket launch costs around 62 million dollars compared to others quoting upward of 165 million dollars. SpaceX launch is affordable due to re-use of the first stage. Total cost of launch thus depends on the probability of landing of the first stage. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
- Problem this project intends to solve is to use various data science methodologies to find out various factors that contribute to outcome of launch and determine the best predicative model to predict the probability of first stage landing successfully.

Section 1

Methodology

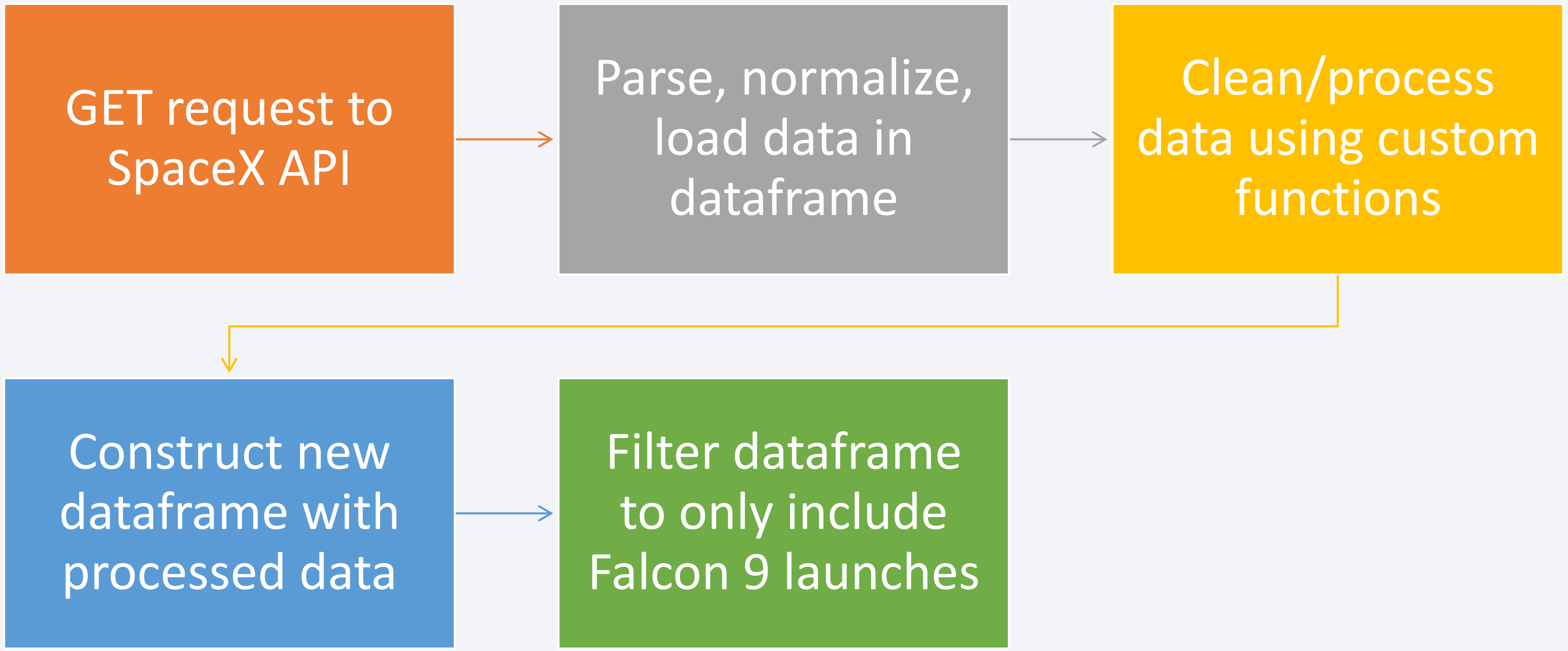
Methodology

Executive Summary

- Data collection methodology:
  - GET request to the SpaceX REST API & Web scrapping on SpaceX Wikipedia (HTML) page
- Perform data wrangling

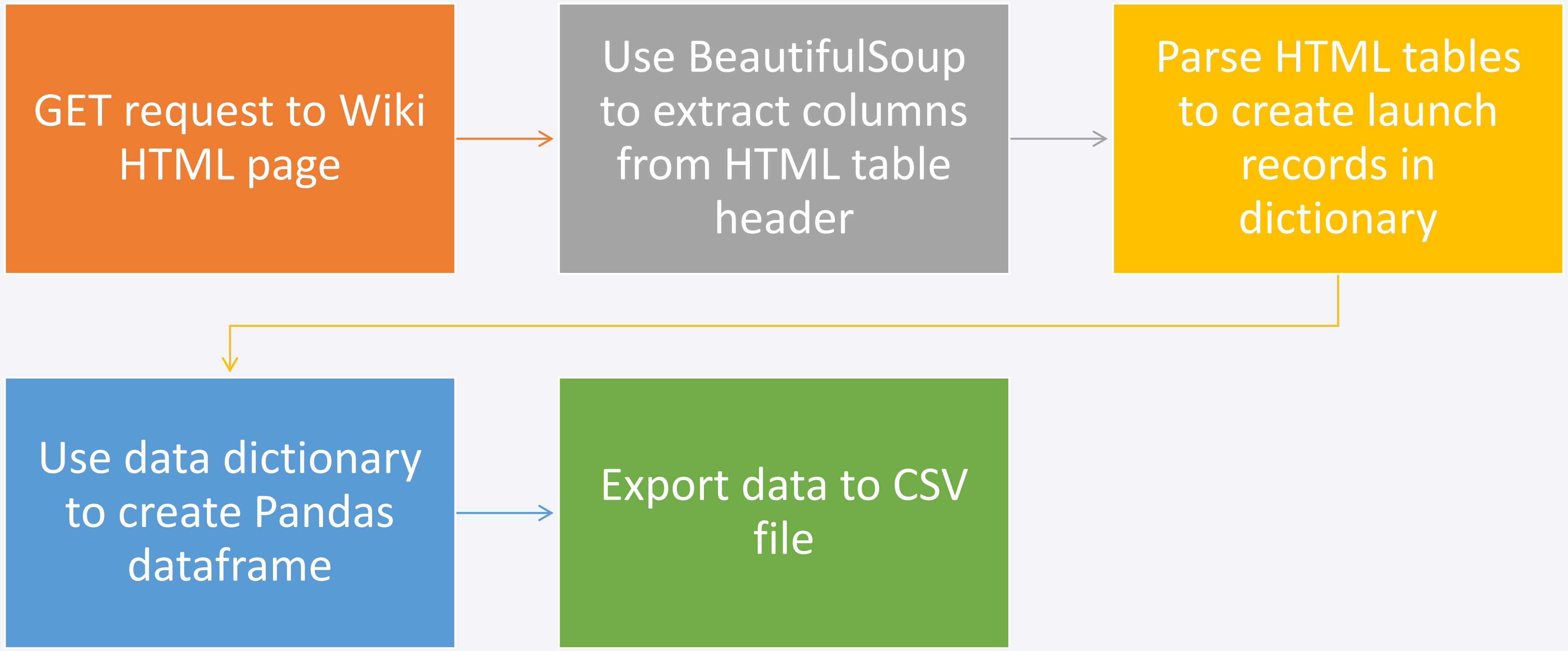
  Create a landing outcome label from Outcome column
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Use Scikit-Learn to standardize data, split into training and test data set.
  - Implement Cross Validation using GridSerachCV to find best hyperparameters using training dataset for various classification models - Logistic Regression, SVM, Decision Tree, and KNN.
  - Use best hyperparameters on these models with test data, evaluate the model performance using accuracy and confusion matrix.

Data Collection – SpaceX API


GET request to SpaceX API
Parse, normalize, load data in dataframe
Clean/process data using custom functions
Construct new dataframe with processed data
Filter dataframe to only include Falcon 9 launches

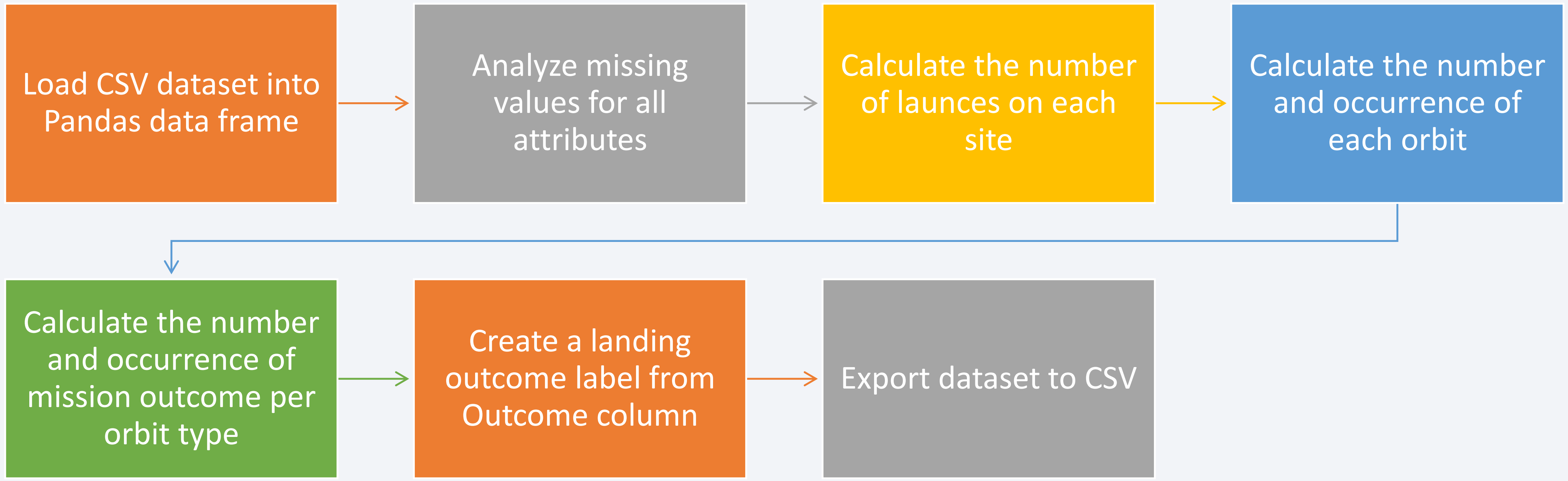
[GitHub URL]

Data Collection - Scraping


GET request to Wiki HTML page
Use BeautifulSoup to extract columns from HTML table header
Parse HTML tables to create launch records in dictionary
Use data dictionary to create Pandas dataframe
Export data to CSV file

GitHub URL

Data Wrangling


Load CSV dataset into Pandas data frame
Analyze missing values for all attributes
Calculate the number of launces on each site
Calculate the number and occurrence of each orbit
Calculate the number and occurrence of mission outcome per orbit type
Create a landing outcome label from Outcome column
Export dataset to CSV

GitHub URL

EDA with Data Visualization

- Scatter Charts:
  These are used to visualize the correlation between features / independent variables. Here they were used for Flight Number and Launch Site, Payload and Launch Site, Orbit Type and Flight Number, Payload and Orbit Type

- Bar Chart:
  These are used to compare different categorical and numerical features / variables. Here it was used for Success Rate and Orbit Type

- Line Charts:
  These are used to depict the change in value of feature / variable over time. Here it was used to visualize Success Rate over Years

- GitHub URL

EDA with SQL

Summary of performed SQL queries:

GitHub URL

- 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 successful landing outcome in ground pad was achieved
- List the names of the boosters which have success on drone ship and have payload mass between 4000 and 6000 kgs
- List the total number of successful and failed missions
- List the names of the booster versions which have carried the maximum payload mass
- Listing the failed landing outcomes on drone ships, their booster versions, and launch site names for year 2015
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

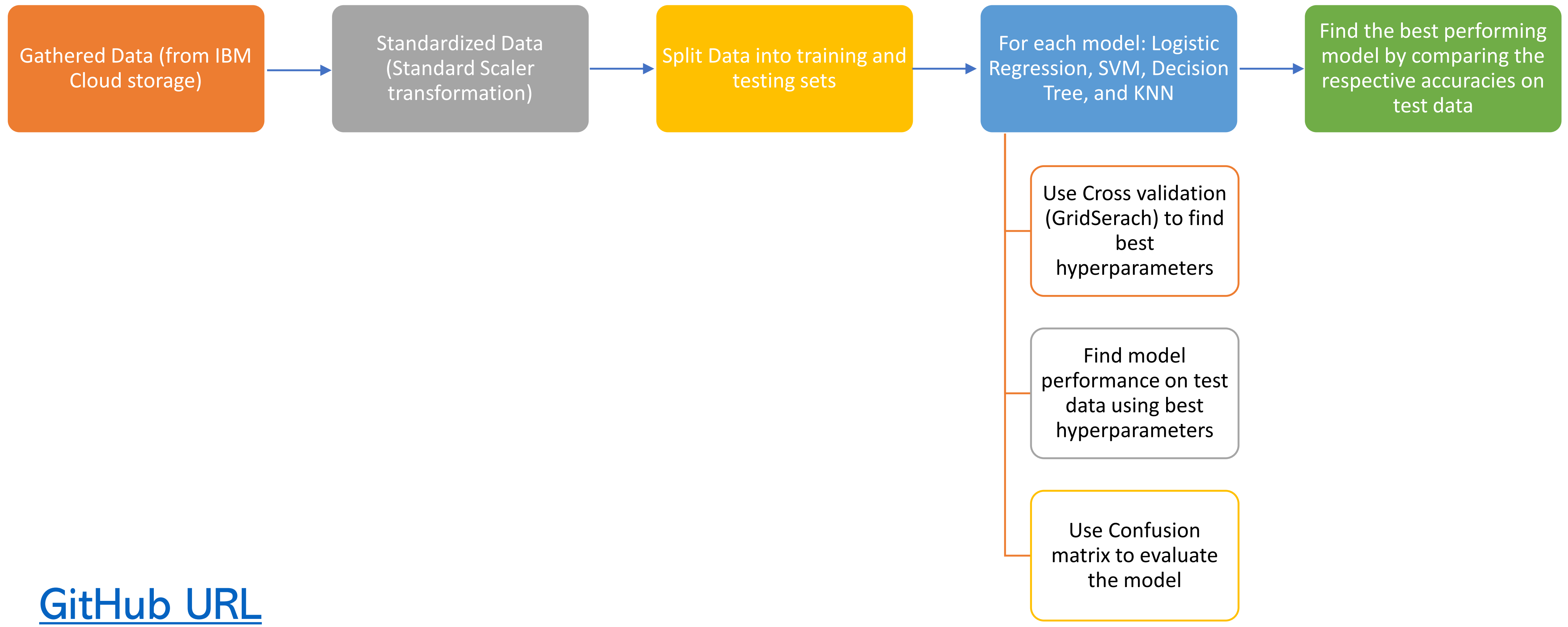
Build an Interactive Map with Folium

- Markers of all Launch Sites: Added Marker with Circle, Popup Label and Text Label of NASA Johnson Space Center using its latitude and longitude coordinates as a start location. Added Markers with Circle, Popup Label and Text Label of all Launch Sites using their latitude and longitude coordinates to show their geographical locations and proximity to Equator and coasts.
- Coloured Markers of the launch outcomes for each Launch Site: Added coloured Markers of success (Green) and failed (Red) launches using Marker Cluster to identify which launch sites have relatively high success rates.
- Distances between a Launch Site to its proximities: Added coloured Lines to show distances between the Launch Site KSC LC-39A (as an example) and its proximities like Railway, Highway, Coastline and Closest City
- GitHub URL

Build a Dashboard with Plotly Dash

- Pie Chart, Scatter Plot, and Slider are added to dashboard.
- If all sites are selected in the dropdown, pie chart depicts distribution of successful launches across all launch sites
- If specific site is selected in the dropdown, pie chart depicts distribution of successful vs failed launches for the selected launch site
- Scatter plot is used to visualize the relationship between Outcome and Payload mass for various booster versions.
- Slider could be used to interactively visualize scatter plot by applying filter on various ranges of Payload mass
- GitHub URL

Predictive Analysis (Classification)


Gathered Data (from IBM Cloud storage)
Standardized Data (Standard Scaler transformation)
Split Data into training and testing sets
For each model: Logistic Regression, SVM, Decision Tree, and KNN
Find the best performing model by comparing the respective accuracies on test data
Use Cross validation (GridSerach) to find best hyperparameters
Find model performance on test data using best hyperparameters
Use Confusion matrix to evaluate the model

GitHub URL

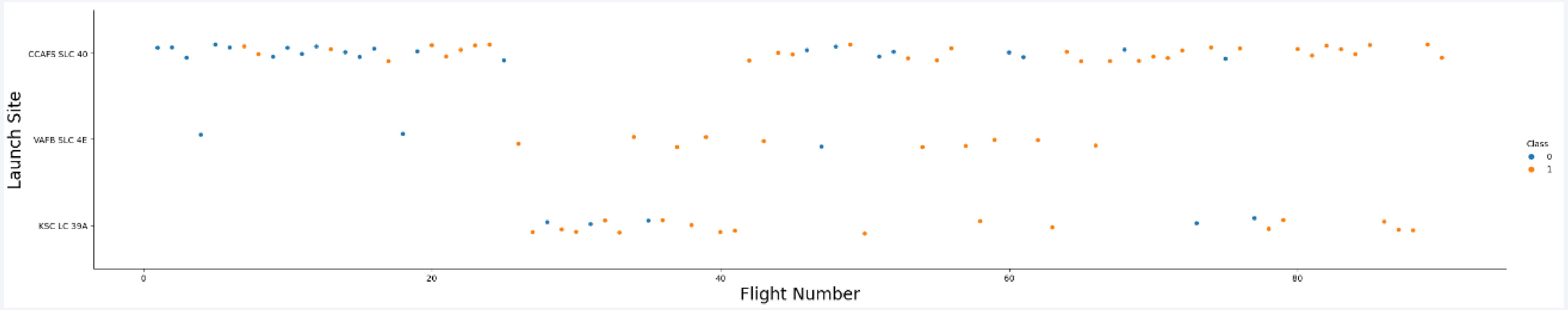
Results

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

Section 2

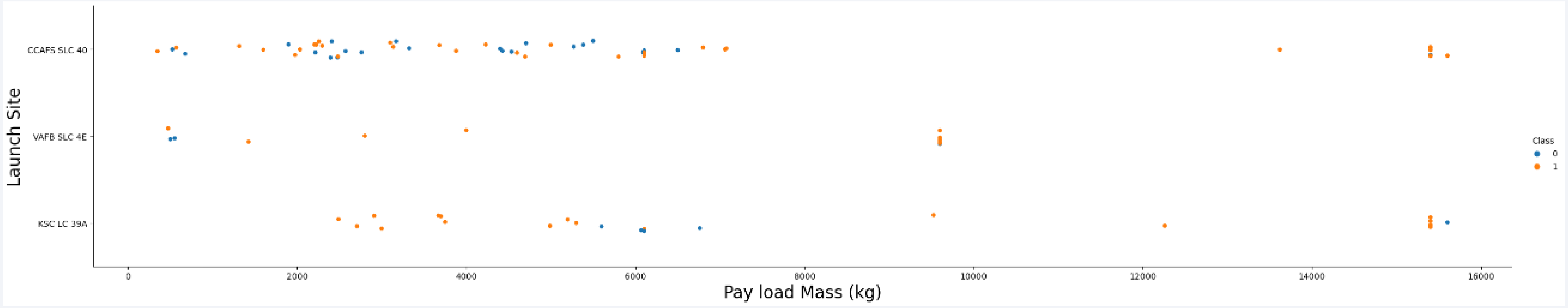
Insights drawn from EDA

Flight Number vs. Launch Site



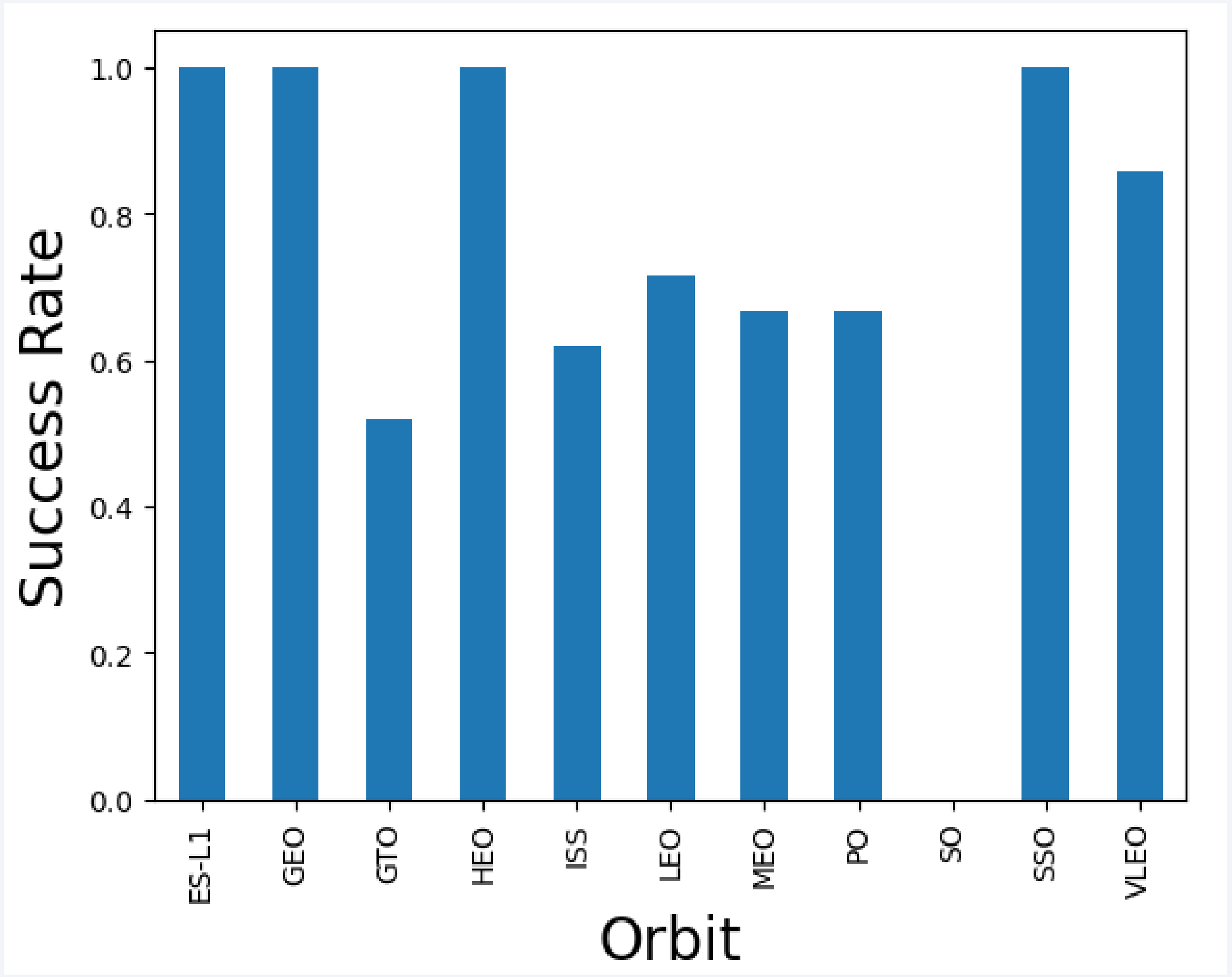
- Class 0 (blue) represents failure while Class 1 (orange) represents successful launch
- Success rate has increased with increase in flight number i.e. recent launches
- CCAFS SLC 40 saw most of the early launches with mixed results mostly failing early on.
- KSC LC 39A has seen no launches early on (till flight number 25)

Payload vs. Launch Site



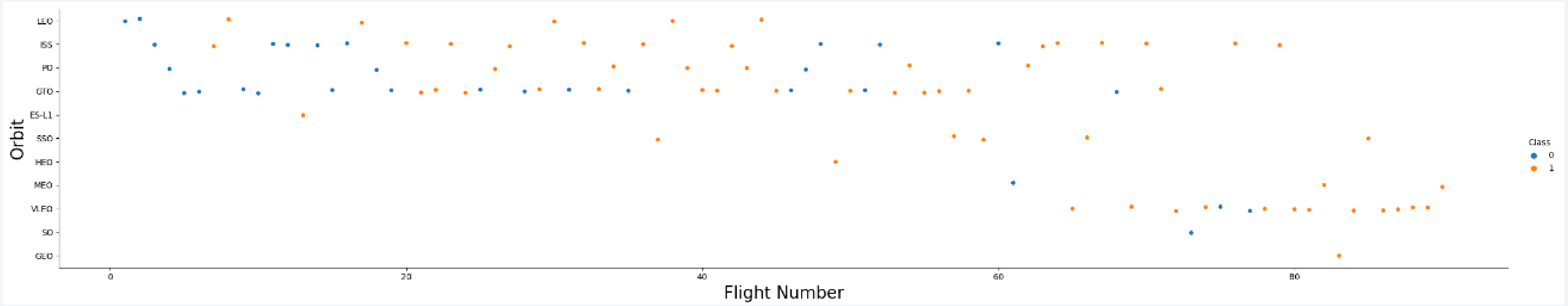
- Class 0 (blue) represents failure while Class 1 (orange) represents successful launch
- This scatter plot shows that there is no definitive correlation between Payload and Launch site.
- Majority of the launches have been with payload smaller than 7000 kgs and greater than 15000 kgs with very few launches in the mid-range.
- CCAFS SLC 40 site is mostly used for relatively lighter payloads.

Success Rate vs. Orbit Type



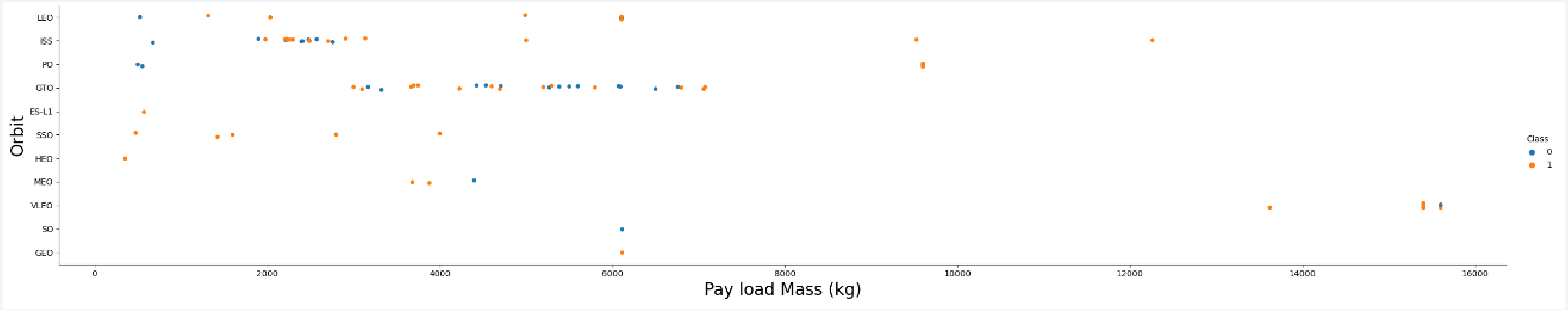
- Launches for orbits ES-L1, GEO, HEO, and SSO have been hugely successful (~100% success rate)
- SO has the least success rate due to just 1 attempt that ended in failure.
- Success rate for GTO is around 50% but also have seen most launches.

Flight Number vs. Orbit Type



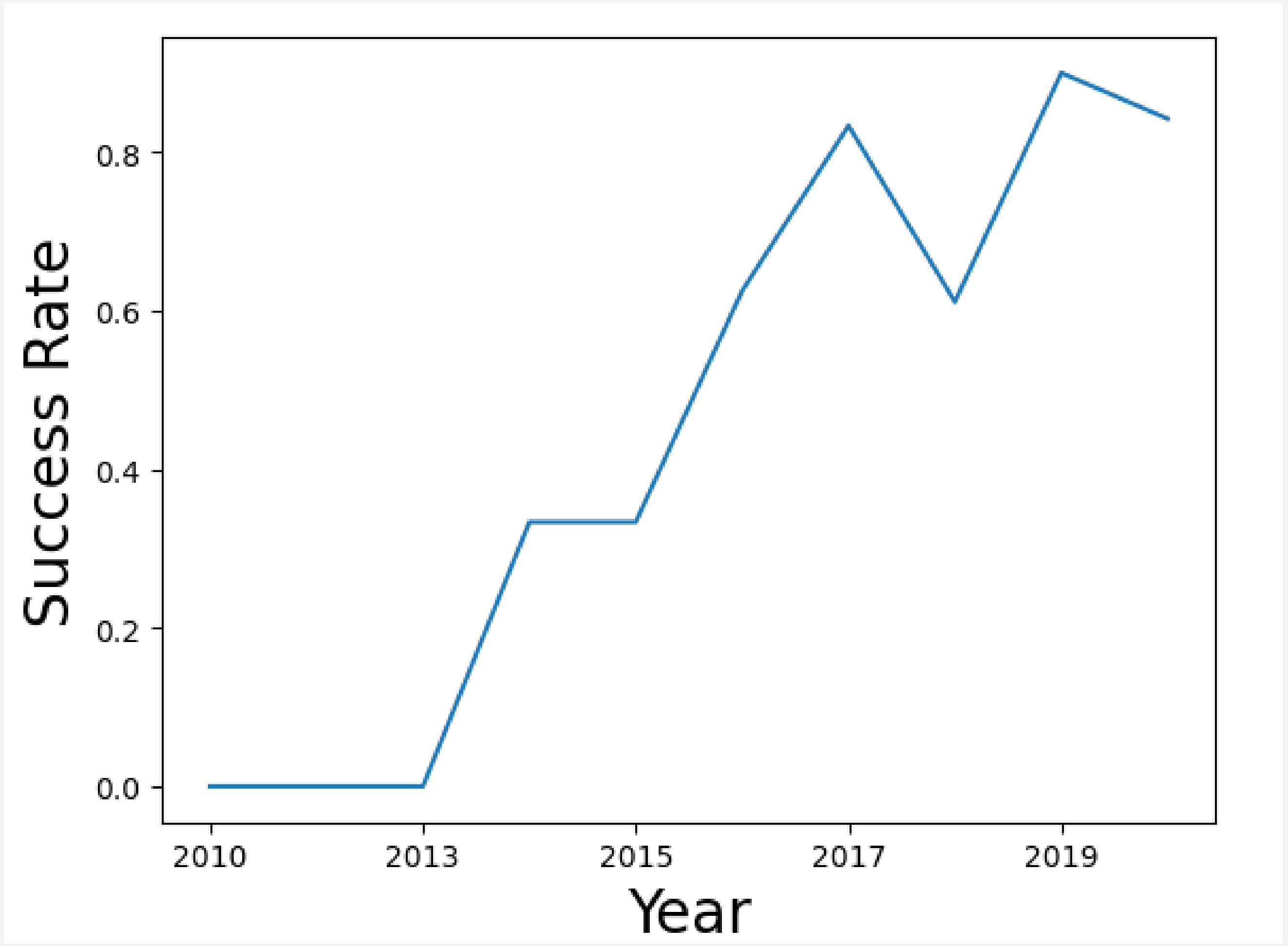
- Class 0 (blue) represents failure while Class 1 (orange) represents successful launch
- In general success rate has improved with more flights, especially LEO which has seen successful launches after just 2 early failures.
- VLFO orbit launches have seen significant success rate recently (larger flight number)

Payload vs. Orbit Type



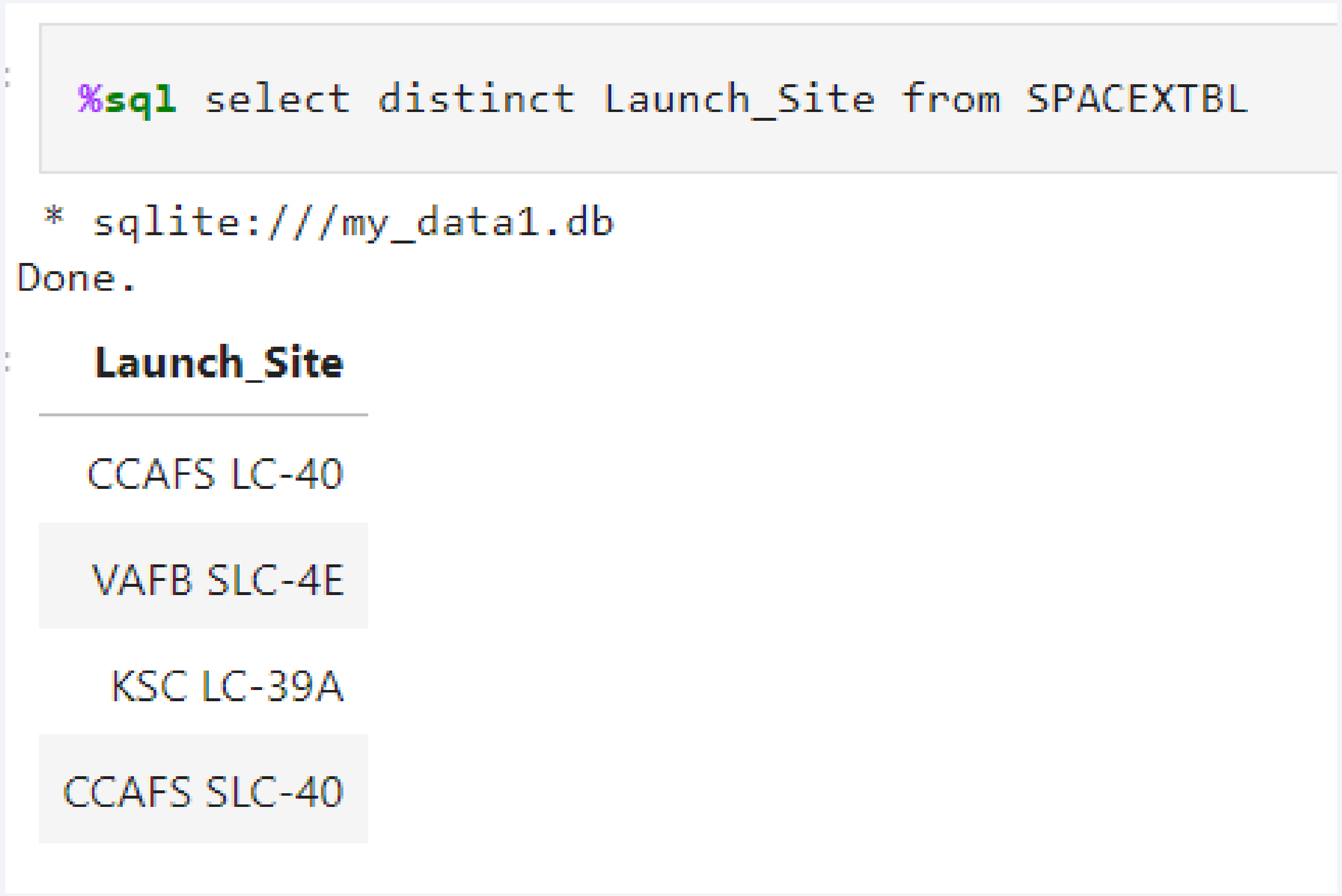
- Class 0 (blue) represents failure while Class 1 (orange) represents successful launch
- PO, ISS, and LEO orbits have seen significant success for heavy payloads
- VLEO orbit is generally used for heavy payload launches.
- SSO orbit launches have been hugely successful for smaller payloads.

Launch Success Yearly Trend



- Launch success rate has been steadily increasing since 2013 barring a deep in 2018 only to be bounced back to higher level after 2019.
- Recent success rate is greater than 80%

All Launch Site Names



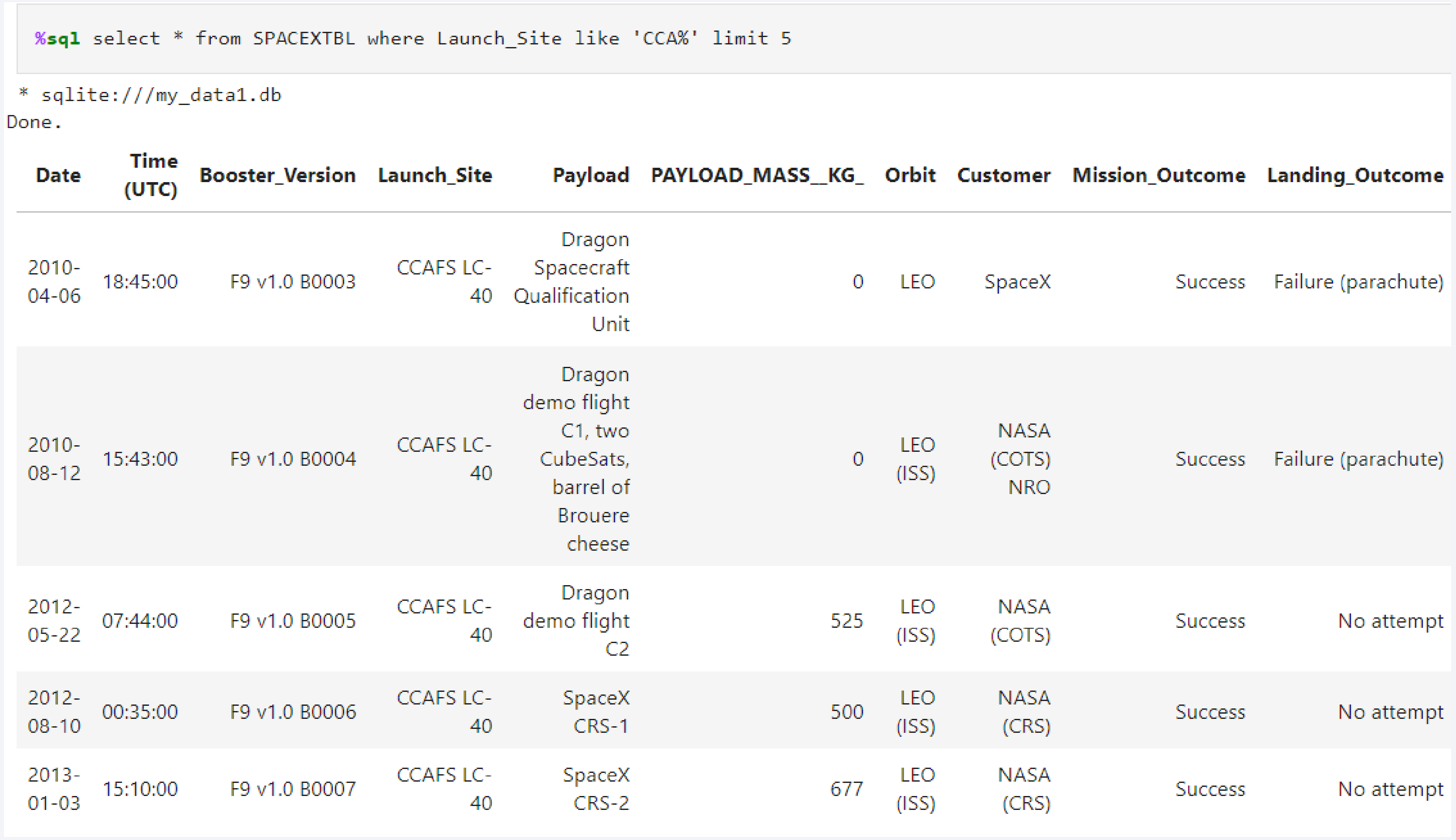
```
%sql select distinct Launch_Site from SPACEXTBL
```

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

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

- This query displays the list of all 4 launch sites using SELECT and DISTINT function.

Launch Site Names Begin with 'CCA'



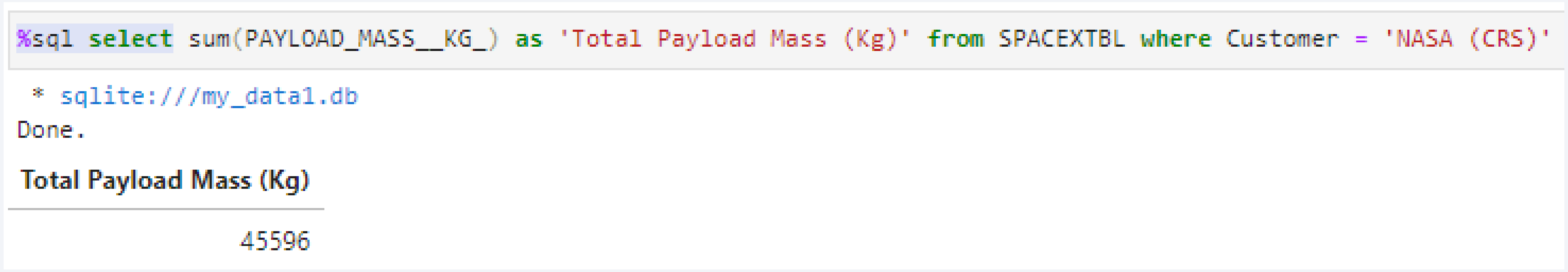
```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- LIKE clause filters the results to include only records starting with CCA
- LIMIT clause limits the number of records to 5

Total Payload Mass



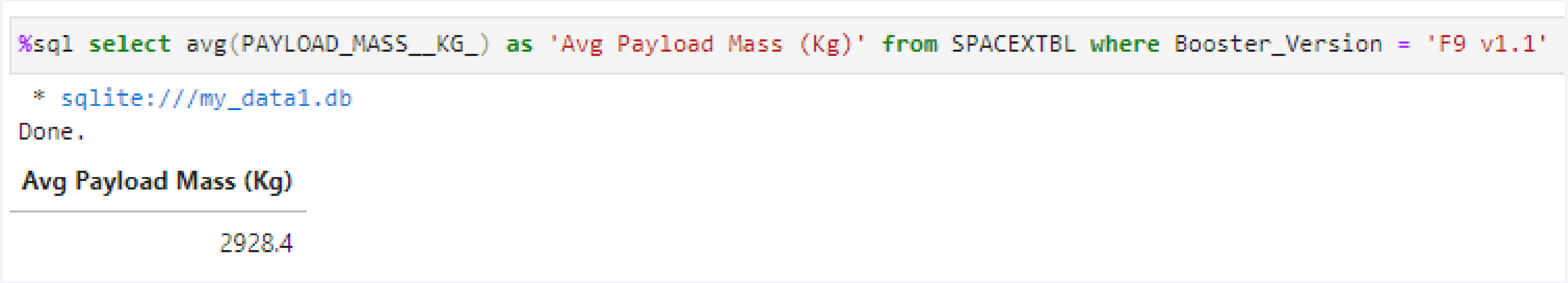
```
%sql select sum(PAYLOAD_MASS__KG_) as 'Total Payload Mass (Kg)' from SPACEXTBL where Customer = 'NASA (CRS)'

 * sqlite:///my_data1.db
Done.

Total Payload Mass (Kg)
                  45596
```

- This query sums up the payload carried by boosters from NASA
- WHERE clause limits records to customer ‘NASA (CRS)’
- SELECT cause along with SUM function displays Total payload mass.

Average Payload Mass by F9 v1.1



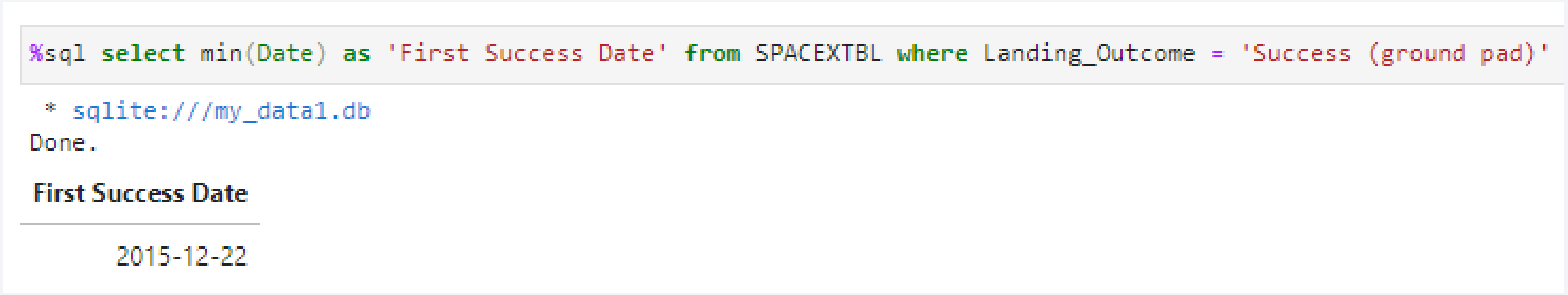
```
%sql select avg(PAYLOAD_MASS__KG_) as 'Avg Payload Mass (Kg)' from SPACEXTBL where Booster_Version = 'F9 v1.1'
```

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

Avg Payload Mass (Kg)
2928.4

- AVG function calculates the average payload mass
- WHERE clause limits the records to booster version F9 v1.1

First Successful Ground Landing Date



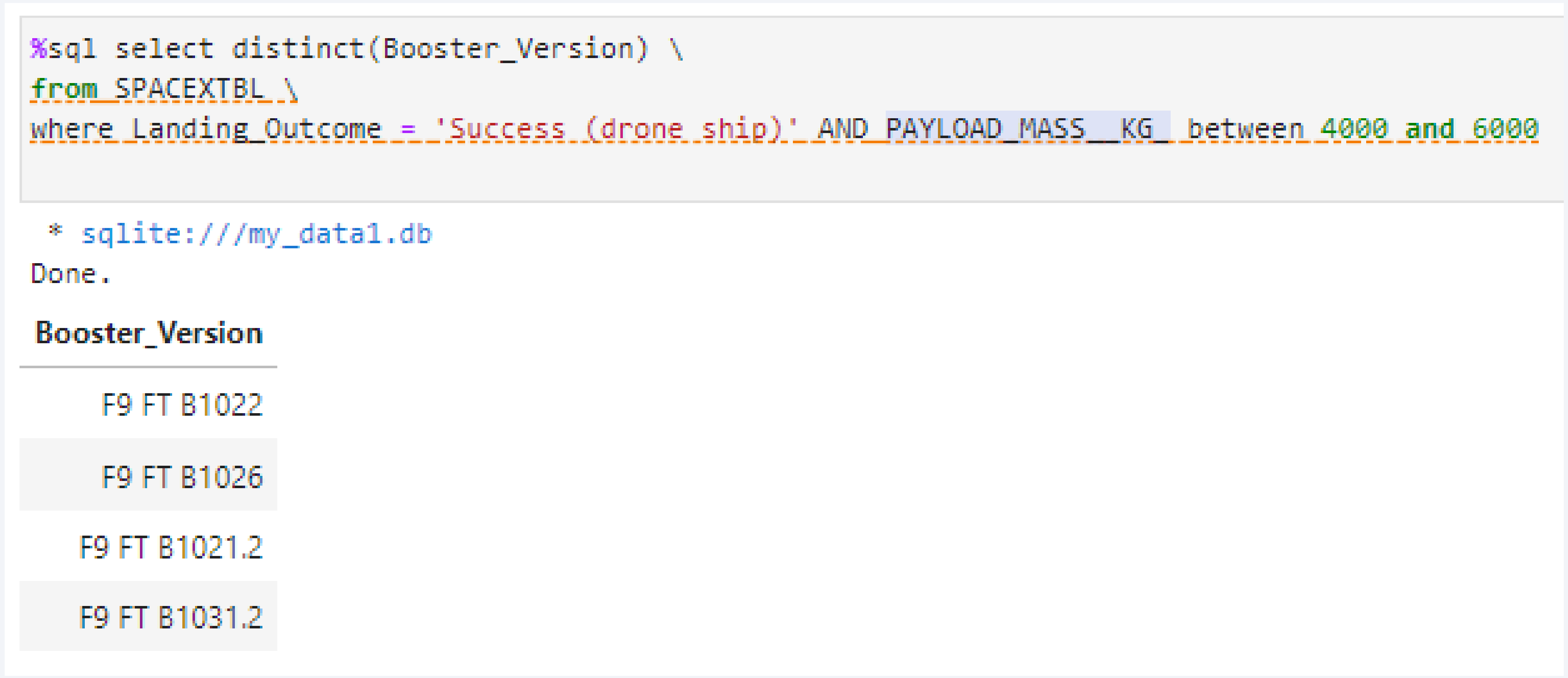
```
%sql select min(Date) as 'First Success Date' from SPACEXTBL where Landing_Outcome = 'Success (ground pad)'
```

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

First Success Date
2015-12-22

- WHERE clause limits the landing outcome to successful landing on ground pad
- MIN function selects the minimum i.e. first landing date

Successful Drone Ship Landing with Payload between 4000 and 6000



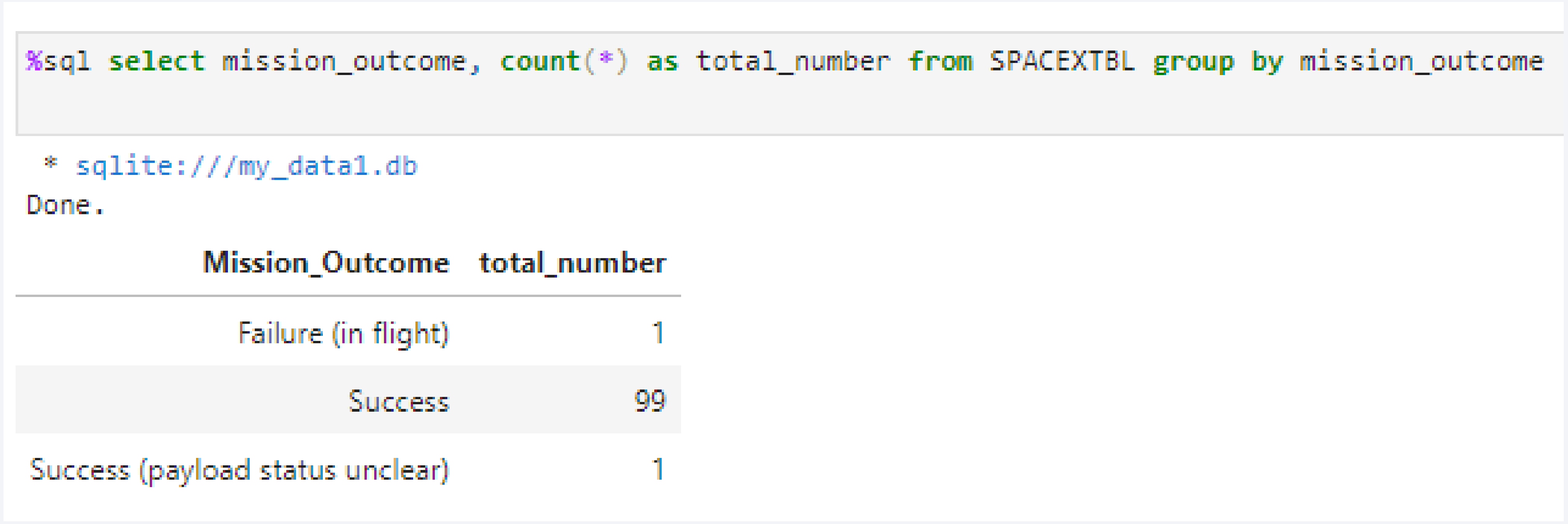
```
%sql select distinct(Booster_Version) \
from SPACEXTBL \
where Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ between 4000 and 6000
```

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

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

- DISTINCT function selects unique booster versions to avoid listing duplicates.
- WHERE clause limits landing outcome to success on drone ship and payload mass between 4000 and 6000 kgs.

Total Number of Successful and Failure Mission Outcomes



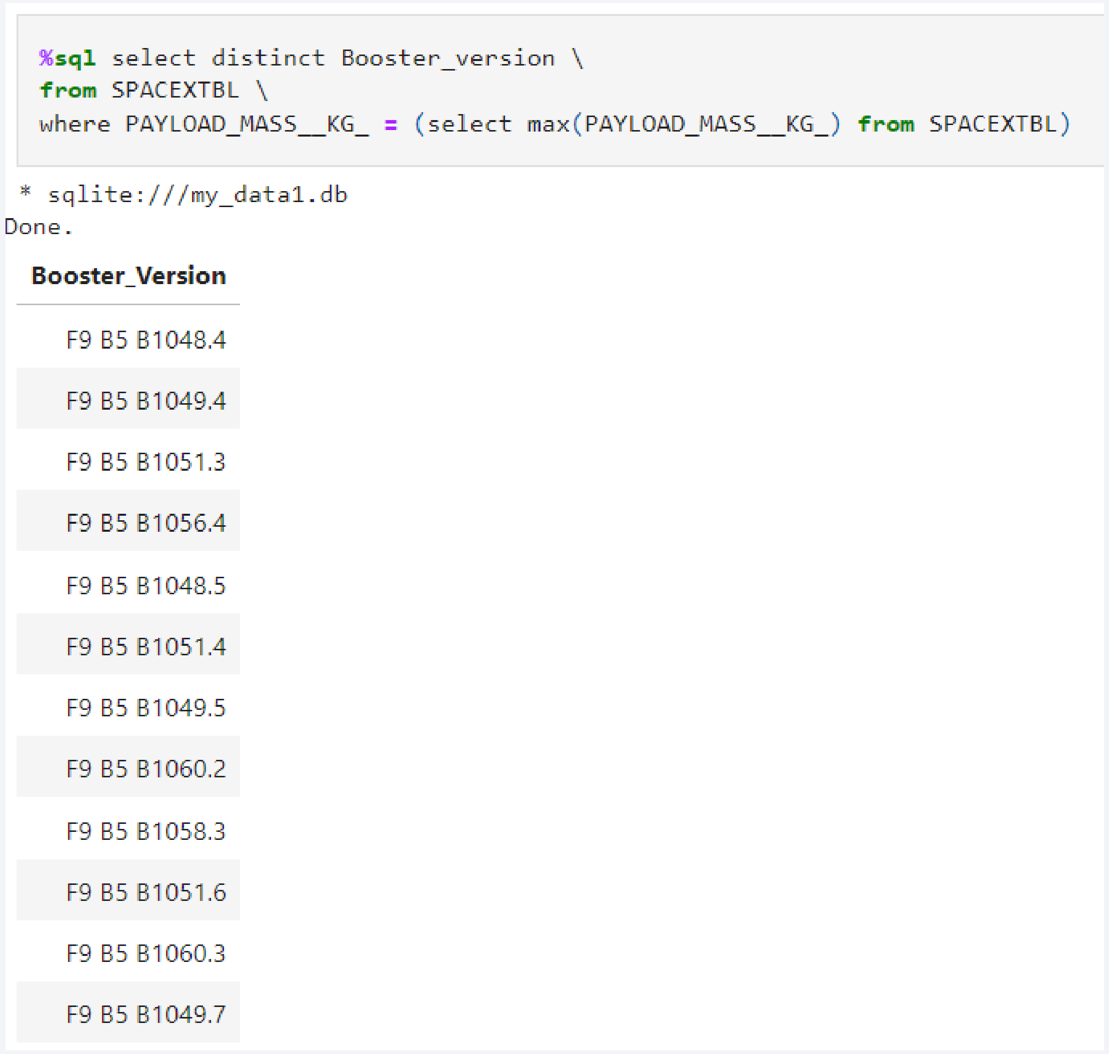
```
%sql select mission_outcome, count(*) as total_number from SPACEXTBL group by mission_outcome
```

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

Mission_Outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

- This query uses GROUP BY clause and COUNT function to get total number of successful and failed missions.
- Only 1 mission failed i.e 99% of launches fulfilled desired mission outcome

Boosters Carried Maximum Payload



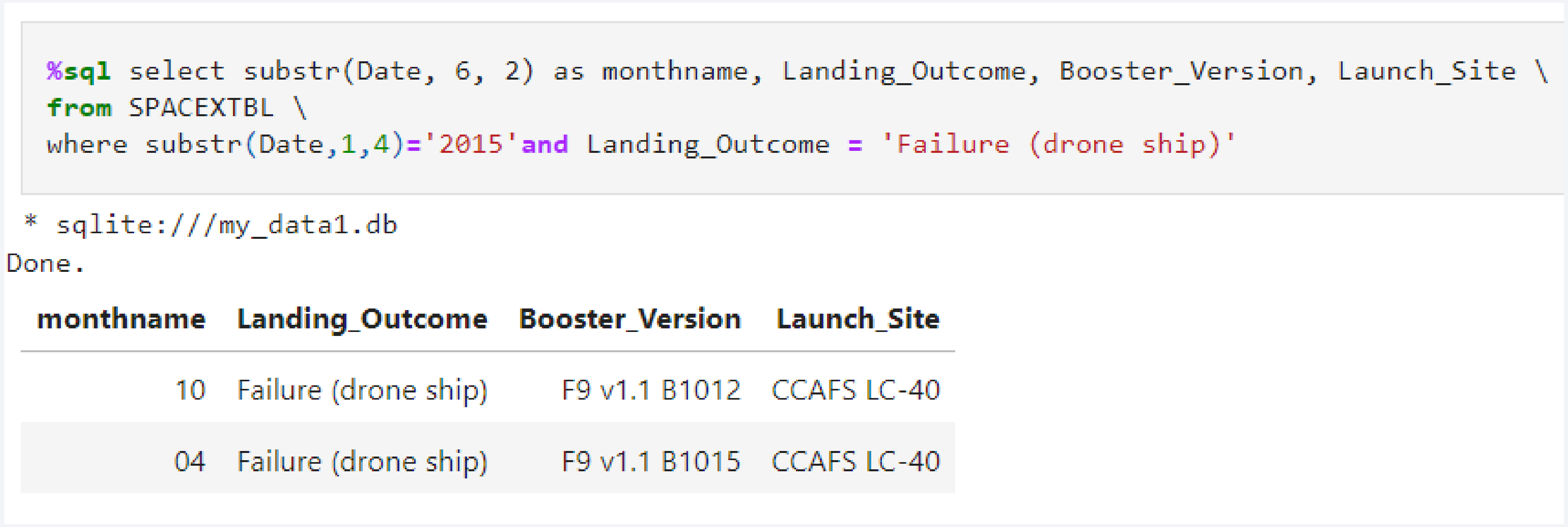
```
%sql select distinct Booster_version \
from SPACEXTBL \
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

```
 * 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

- Subquery calculates the maximum payload in kgs first.
- Main query uses WHERE clause on the result of subquery and then DISTINCT function to list the required boosters.

2015 Launch Records



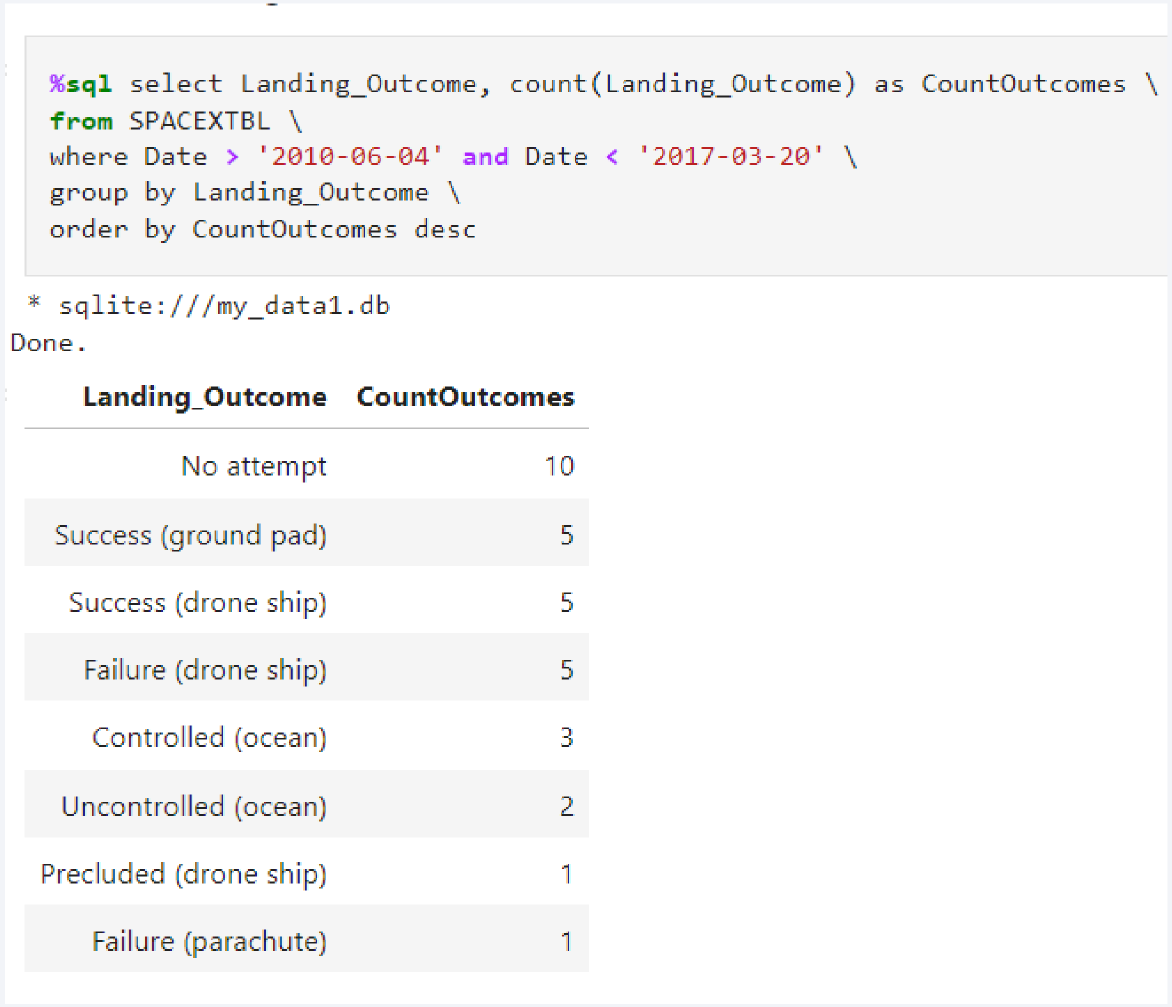
```
%sql select substr(Date, 6, 2) as monthname, Landing_Outcome, Booster_Version, Launch_Site \
from SPACEXTBL \
where substr(Date,1,4)='2015'and Landing_Outcome = 'Failure (drone ship)'
```

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

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

- SUBSTR function is used to extract year from Date and compare against 2015
- Month April (04) and October (10) saw drone ship landing failure each.

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



```
%sql select Landing_Outcome, count(Landing_Outcome) as CountOutcomes \
from SPACEXTBL \
where Date > '2010-06-04' and Date < '2017-03-20' \
group by Landing_Outcome \
order by CountOutcomes desc
```

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

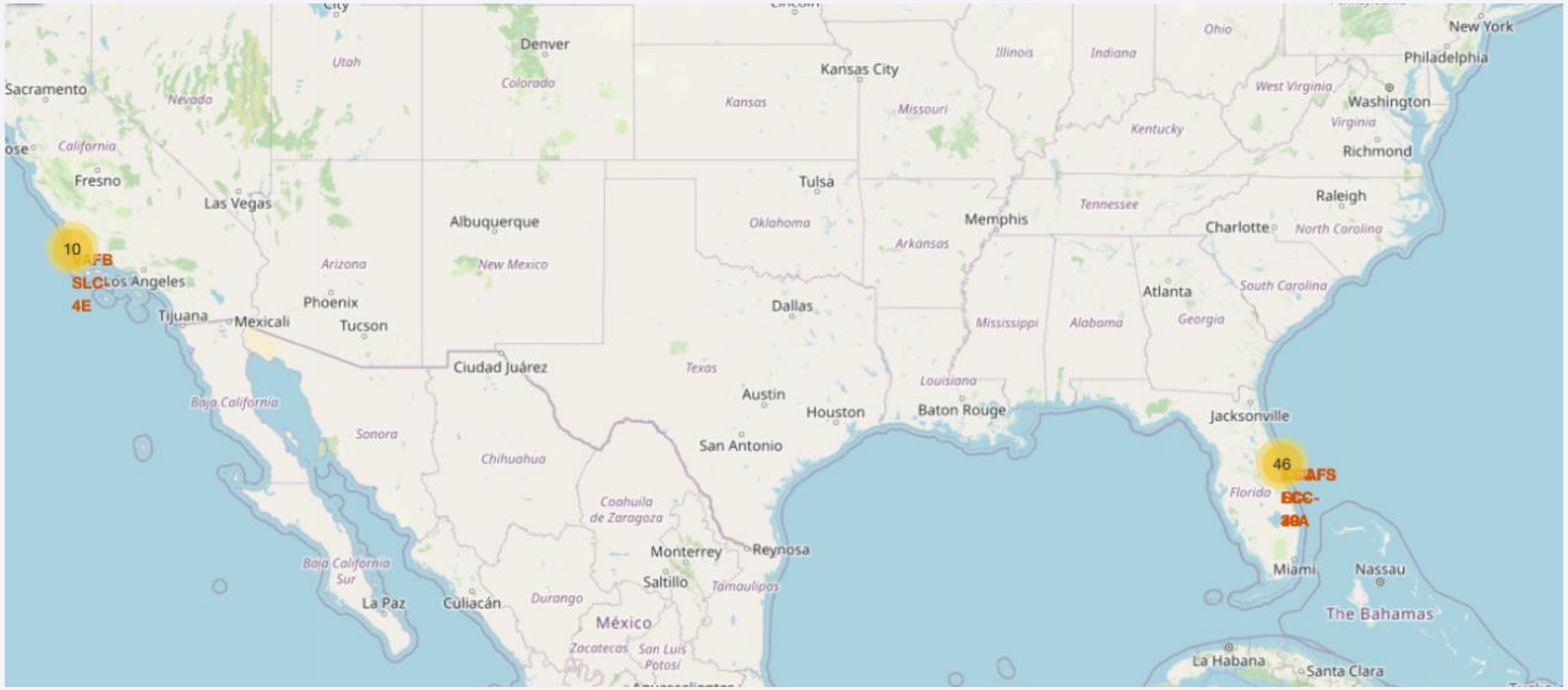
Landing_Outcome	CountOutcomes
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

- WHERE clause selects records between desired dates.
- GROUP BY clause groups the records by Landing_Outcome which are further counted using count function in select clause.
- ORDER BY sorts the resulting records in descending order by count of outcomes.

Section 3

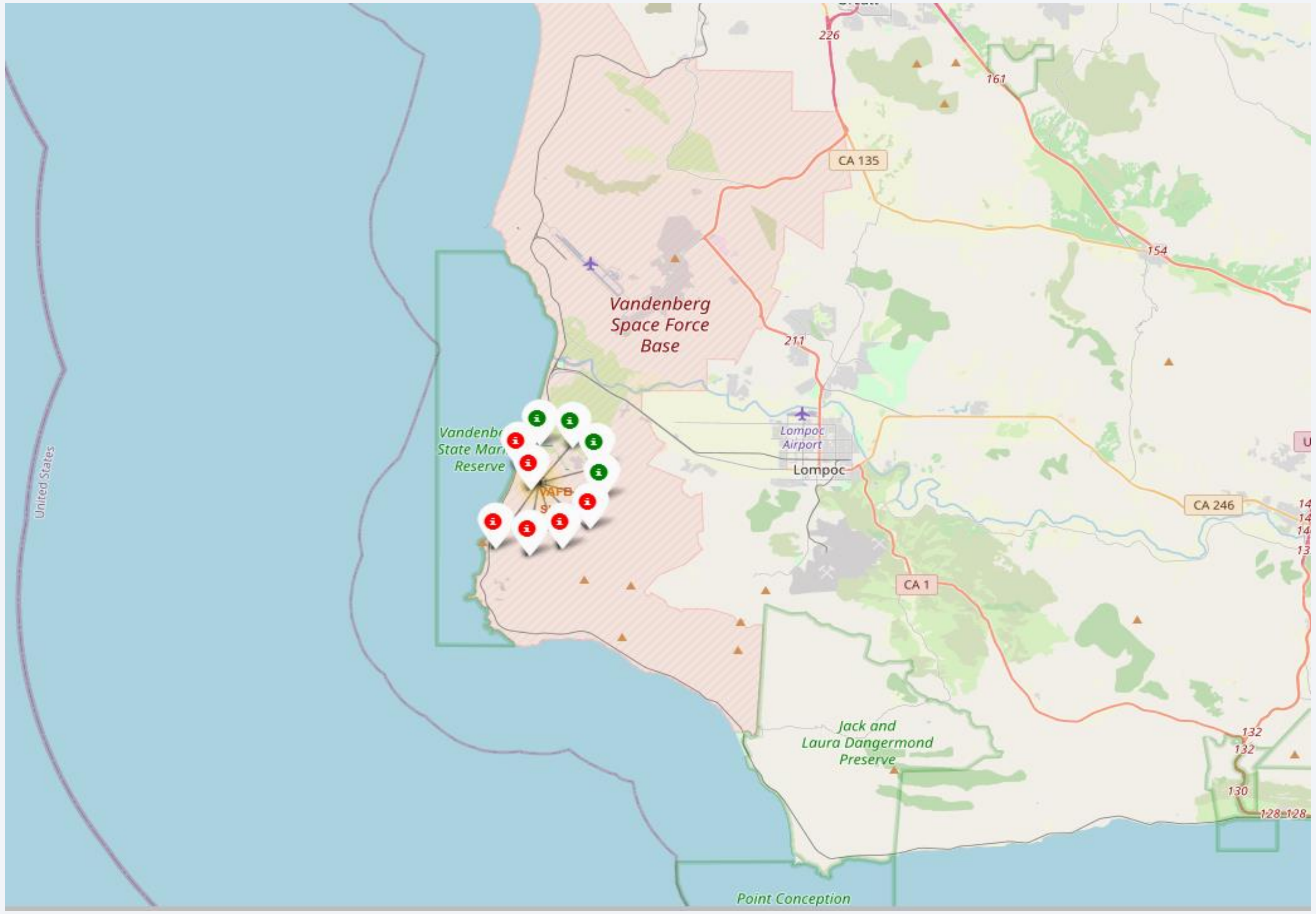
Launch Sites Proximities Analysis

Launch Site Geographical Locations



- Map shows the Launch sites along with the number of launches
- All launch sites are either on east or west coast and mostly in the southern parts closer to the equator. Proximity to the ocean reduces the risk of debris or failed rocket falling into residential areas.

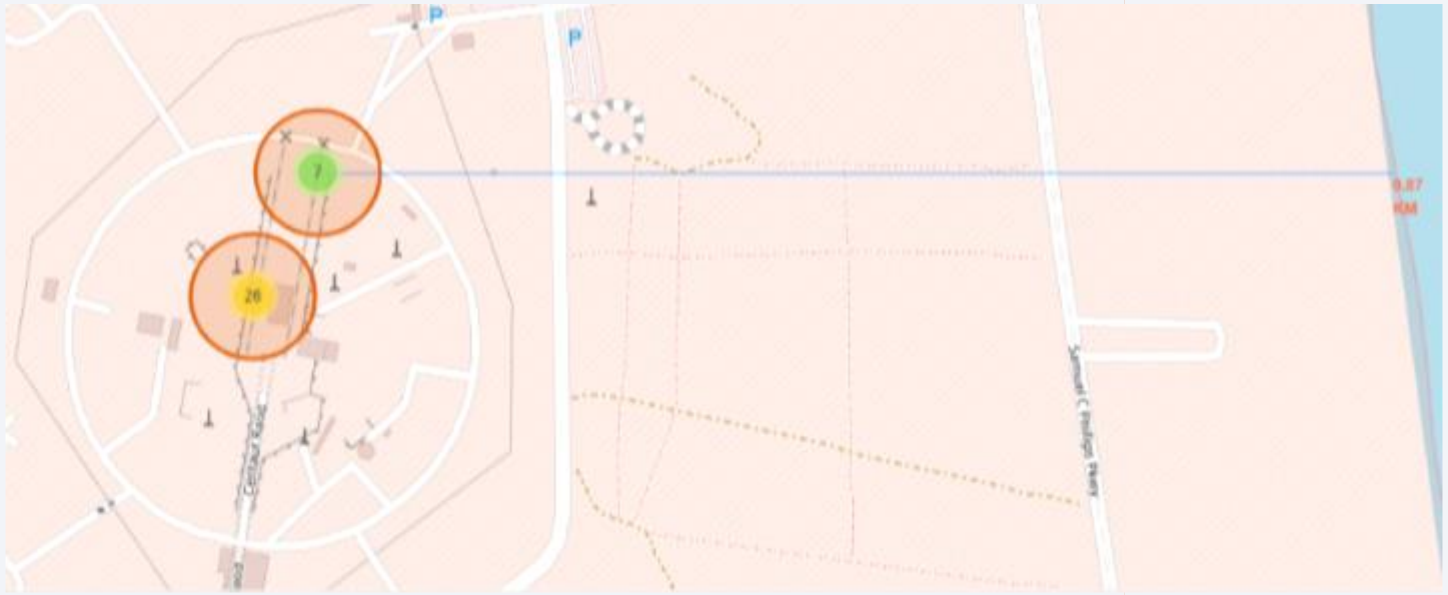
Color coded Successful/Failed Launches

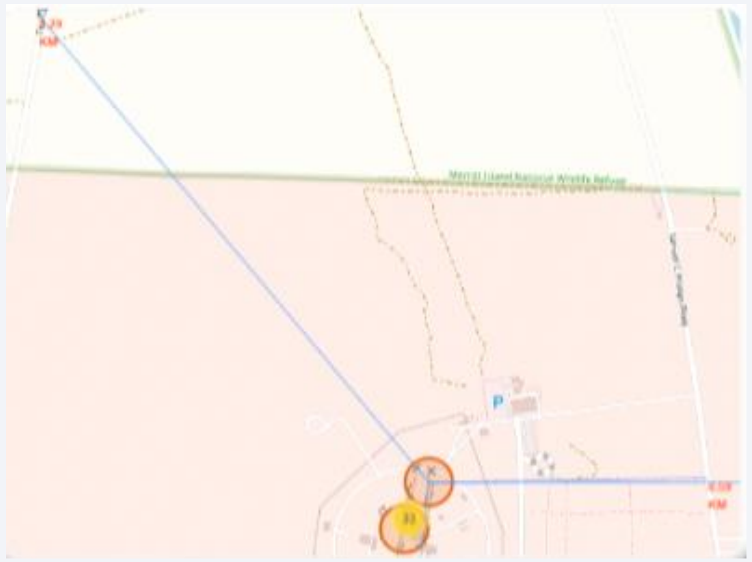


Vandenberg Space Force Base
Vandenberg State Marine Reserve
Lompoc Airport
Lompoc
VAFB
CA 135
CA 246
CA 1
Jack and Laura Dangermond Preserve
Point Conception
United States

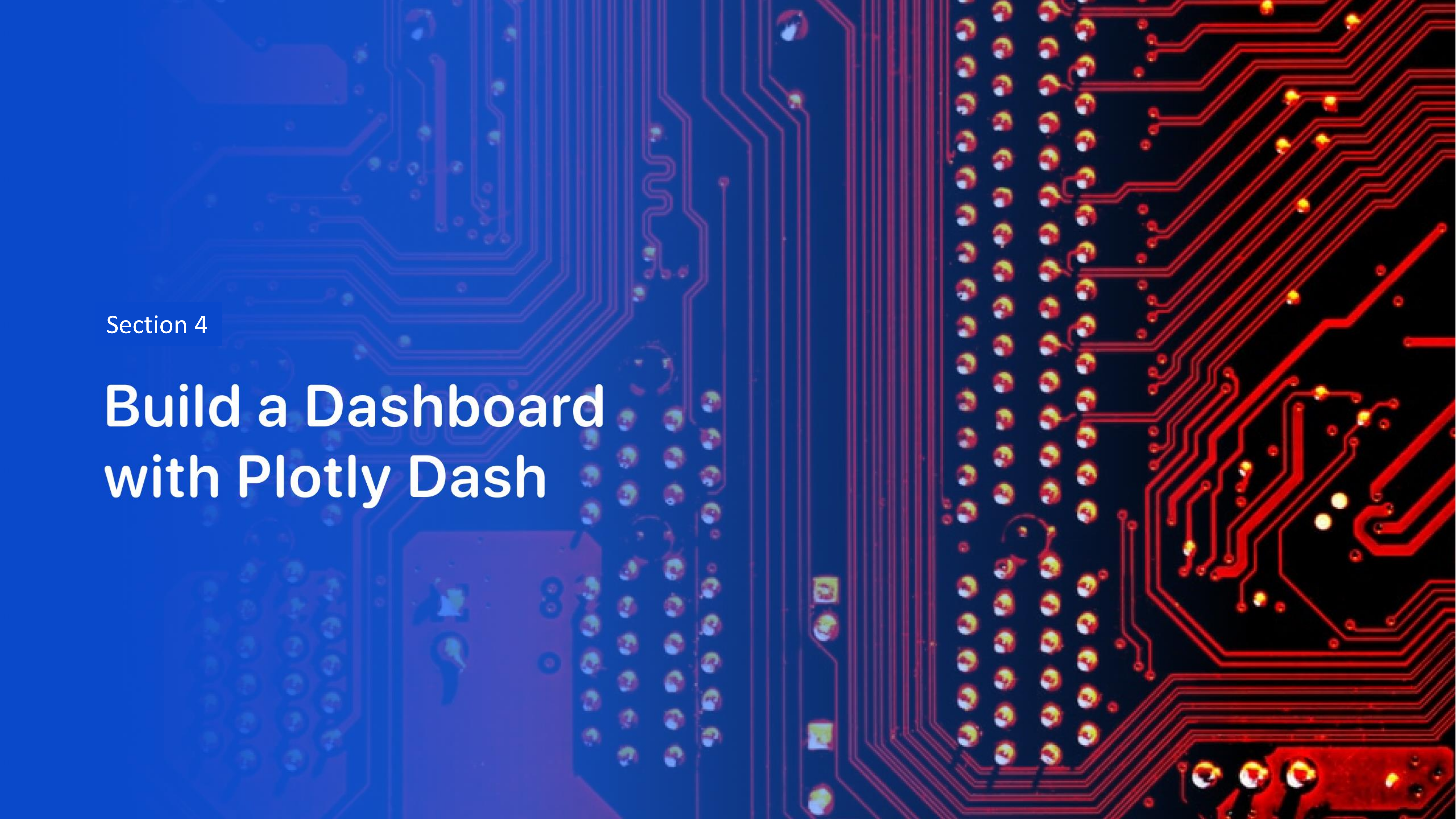
- Clusters on the folium are configured to display success / failure status of launces using color codes.
- Green icon indicates successful and Red icon indicates failed launch.
- This example shows 4 successful and 6 failed launches at VAFB SLC-4E site

Proximity of Launch Site With Points Of Interest





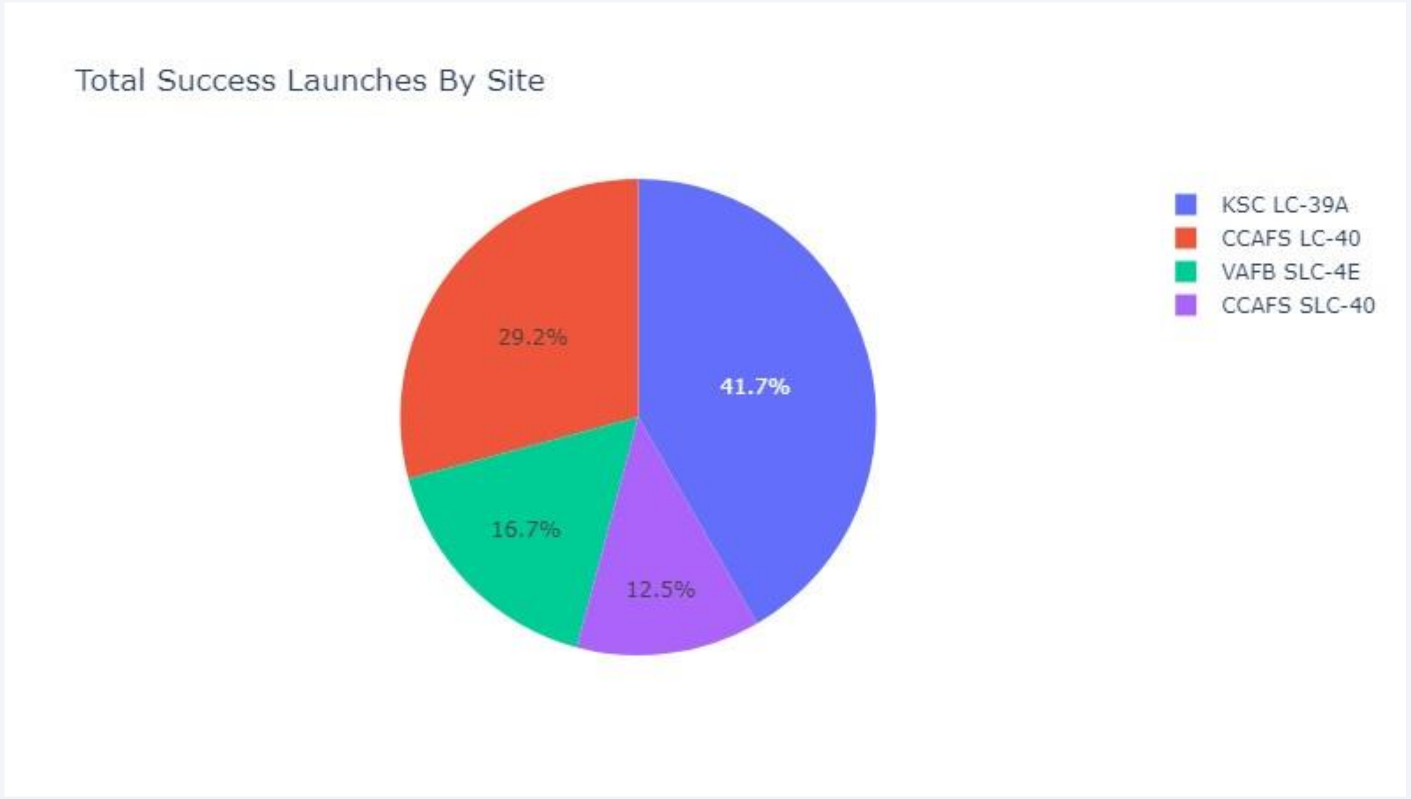
- Launch sites are located closer to railway and highway as Rocket launches involve transportation of heavy material and personnel.
- Launch sites are located closer to the coastline to reduces the risk of debris or failed rocket falling into residential areas.



Section 4

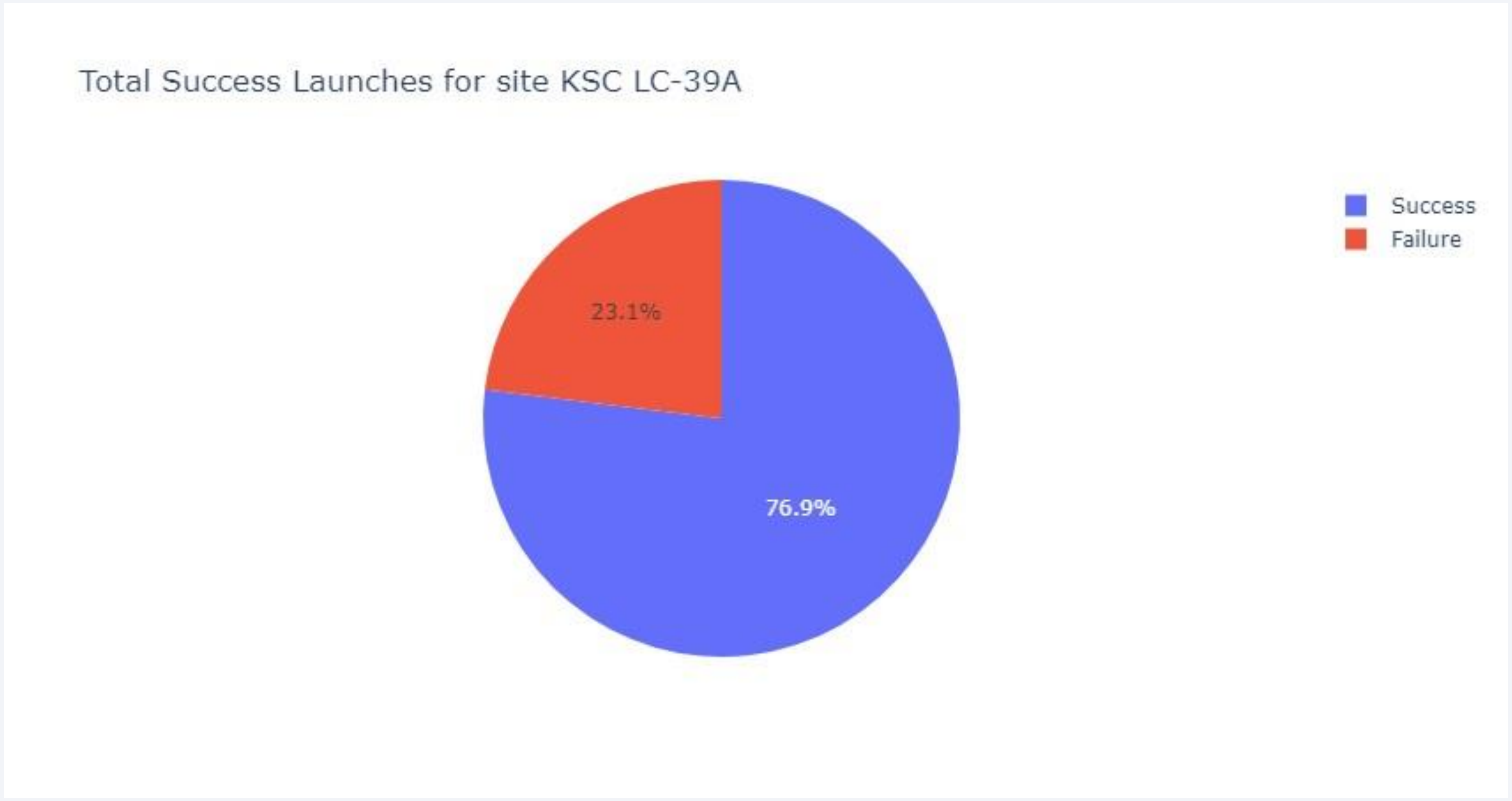
Build a Dashboard with Plotly Dash

Success Launches By Site



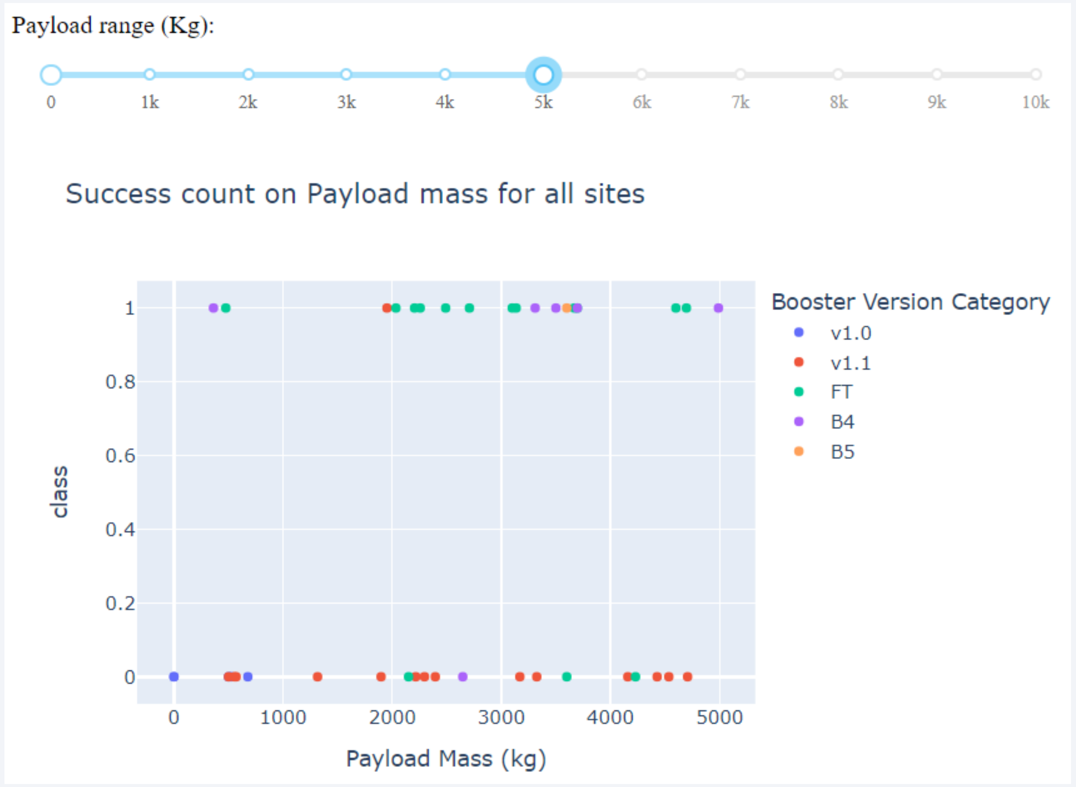
- This is an aggregate chart indicating share of each site in successful launches, not to be confused with success rate of individual site.
- The launch site “KSC LC-39A” has major (41.7%) share of successful launches followed by CCASFS LC-40, VAFG SLC-4E, CCAFS SLC-40 in that order.

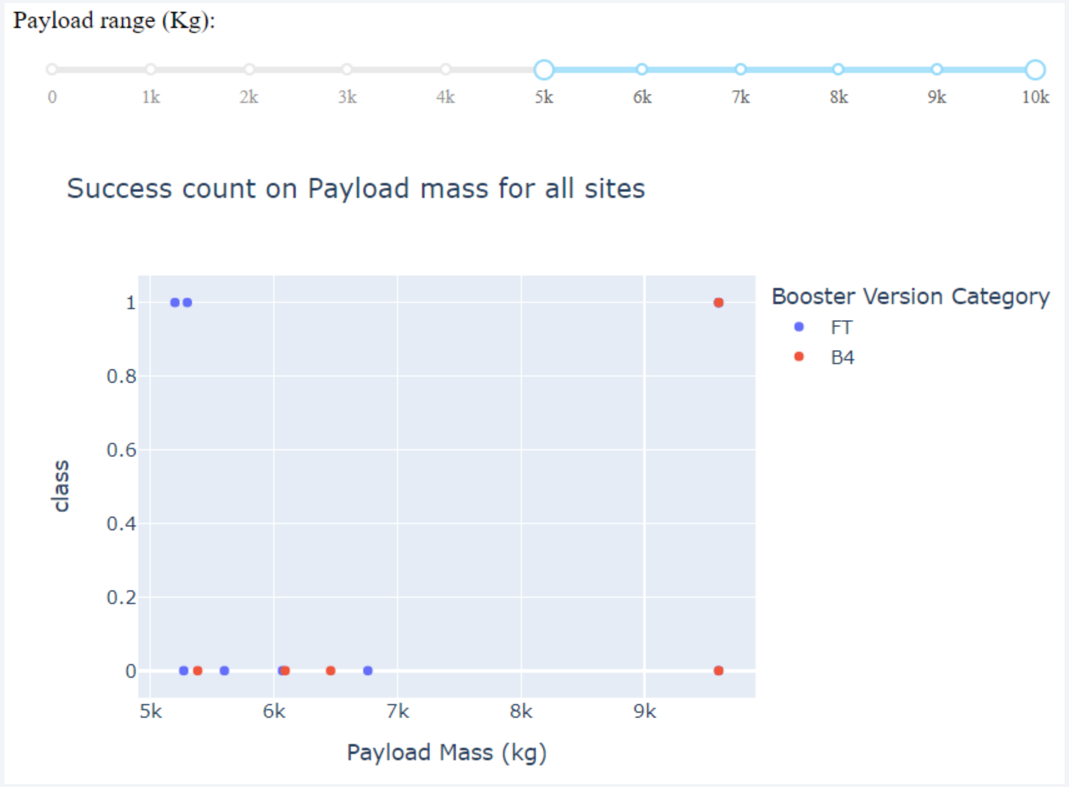
Most Successful Launch Site



- This chart shows success / failure percentage for most successful site KSC LC-39A
- There is 76.9% chance that mission launched from this site will be successful

Payload Ranges vs Launch Success Rate



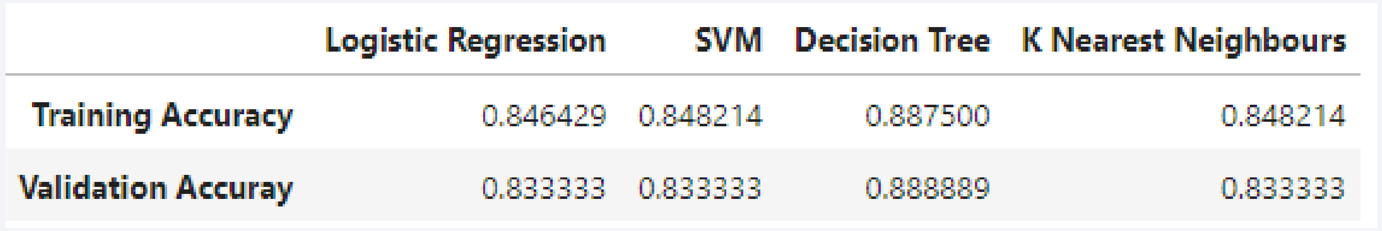


- Success rate of Launches for Smaller to medium payloads (< 5000 kgs) is higher than heavy payloads (5000-10000 kgs)
- Boost version FT has been one of the most successful boosters.

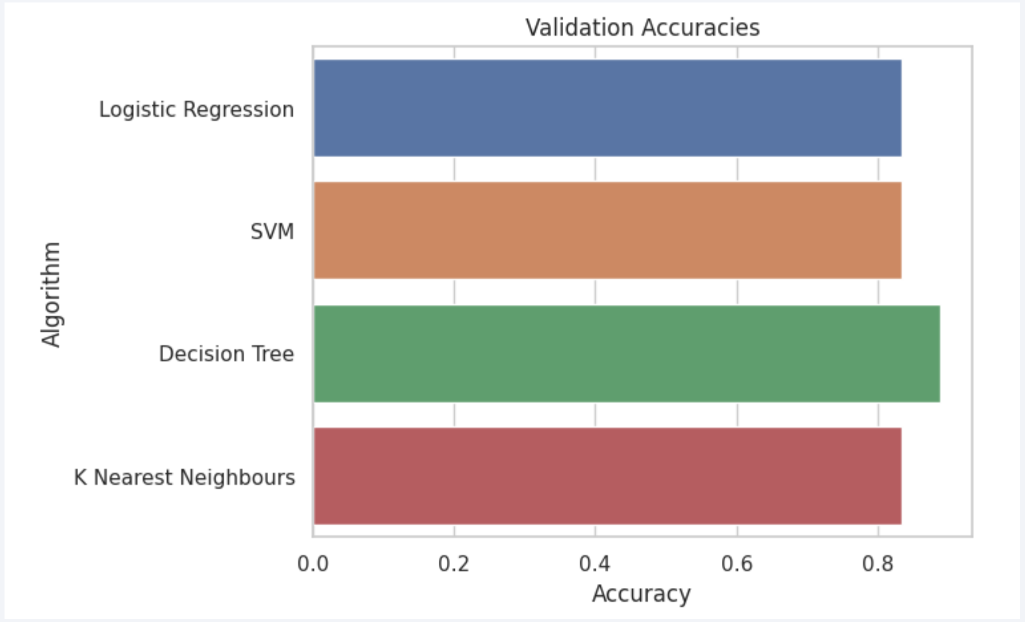
Section 5

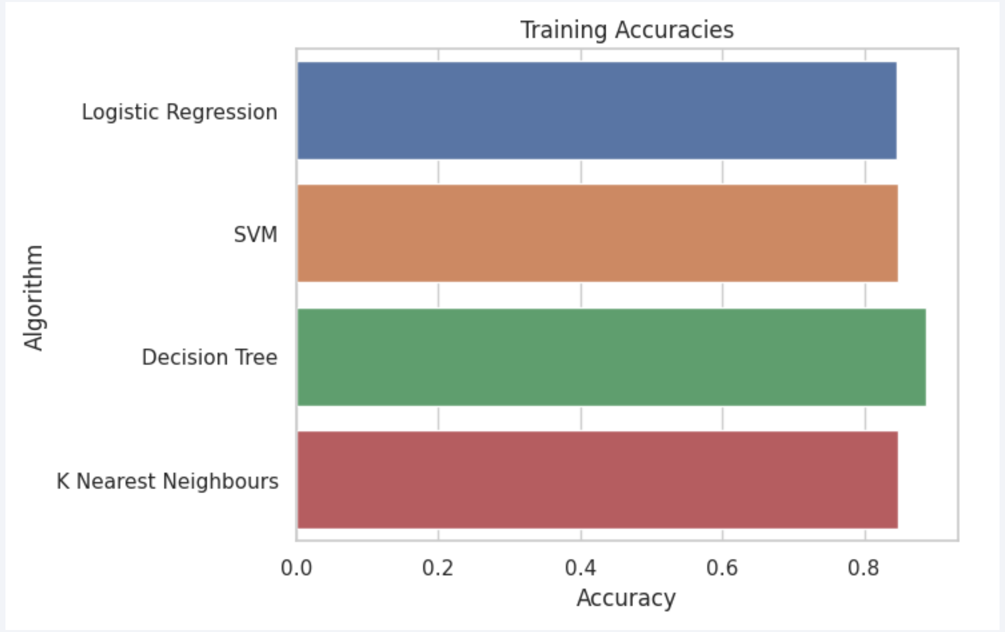
Predictive Analysis (Classification)

Classification Accuracy



	Logistic Regression	SVM	Decision Tree	K Nearest Neighbours
Training Accuracy	0.846429	0.848214	0.887500	0.848214
Validation Accuray	0.833333	0.833333	0.888889	0.833333

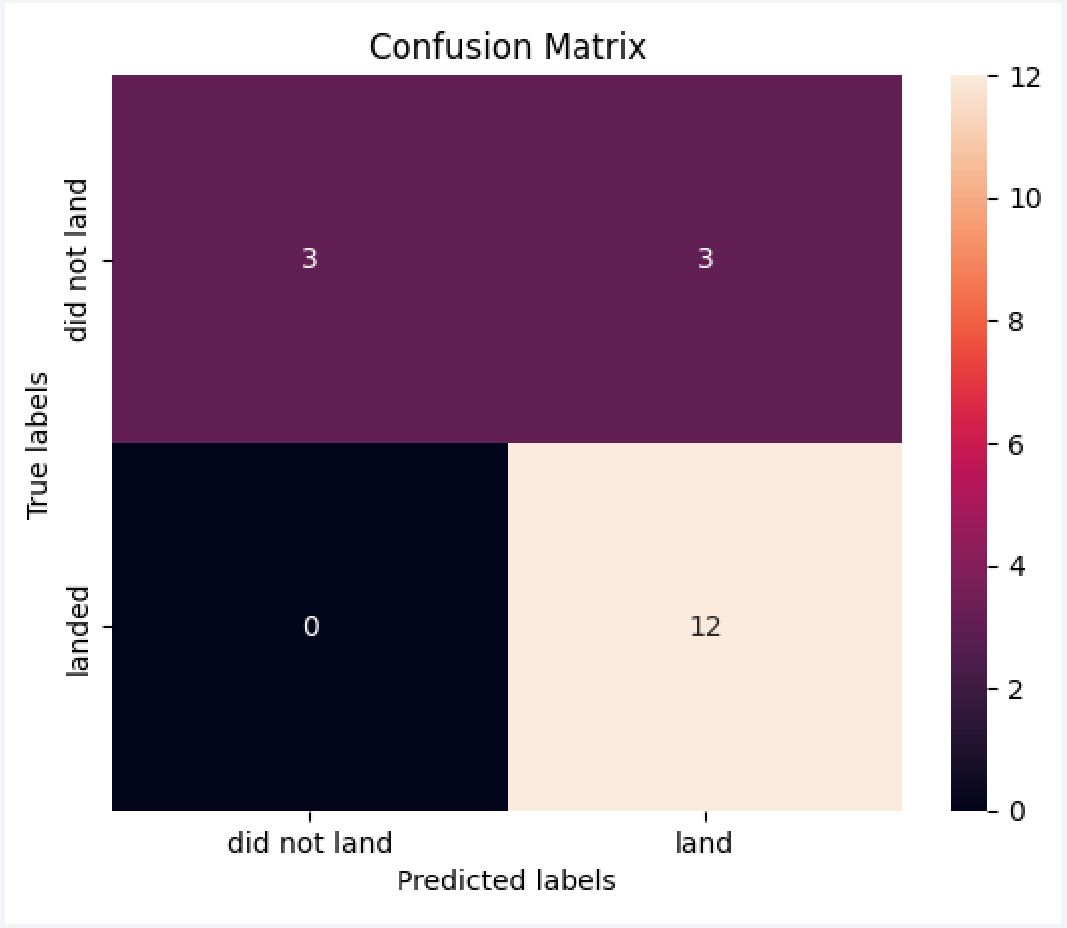




Decision tree model has the highest accuracy for both training and test data

Confusion Matrix

- There are no False Negatives, so model has desired sensitivity (Recall) of 1.
- There are 3 False positives, so model's precision is 0.8 which is not perfect but acceptable for limited dataset.



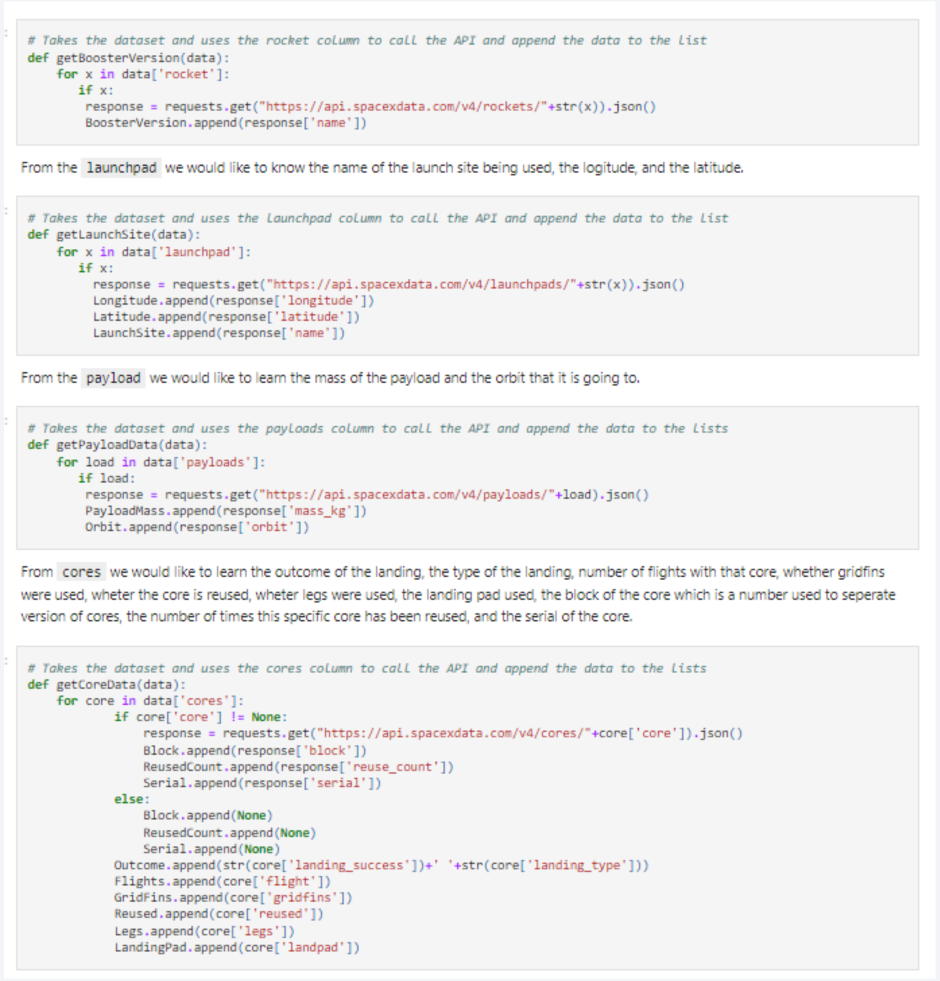
Conclusions

- Majority of the launches are with payload smaller than 7000 kgs and greater than 15000 kgs.
- Launches for orbits ES-L1, GEO, HEO, and SSO have been hugely successful (~100% success rate).
- PO, ISS, and LEO orbits have seen significant success for heavy payloads.
- Launch success rate has been improving since 2013 barring a deep in 2018 only to be bounced back to after 2019, Recent success rate is greater than 80%
- Only 1 mission failed i.e 99% of launches fulfilled desired mission outcome.
- All launch sites are located on coast and mostly in the southern parts closer to the equator. Proximity to the ocean reduces the risk of debris or failed rocket falling into residential areas. Proximity to railway and highway is important factor as Rocket launches involve transportation of heavy material and personnel.
- Success rate for smaller to medium payloads (< 5000 kgs) is higher than heavy payloads (5000-10000 kgs).
- For outcome prediction, Decision tree model has the highest accuracy for both training and test data.

Appendix

Custom Functions

Project GitHub URL



```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

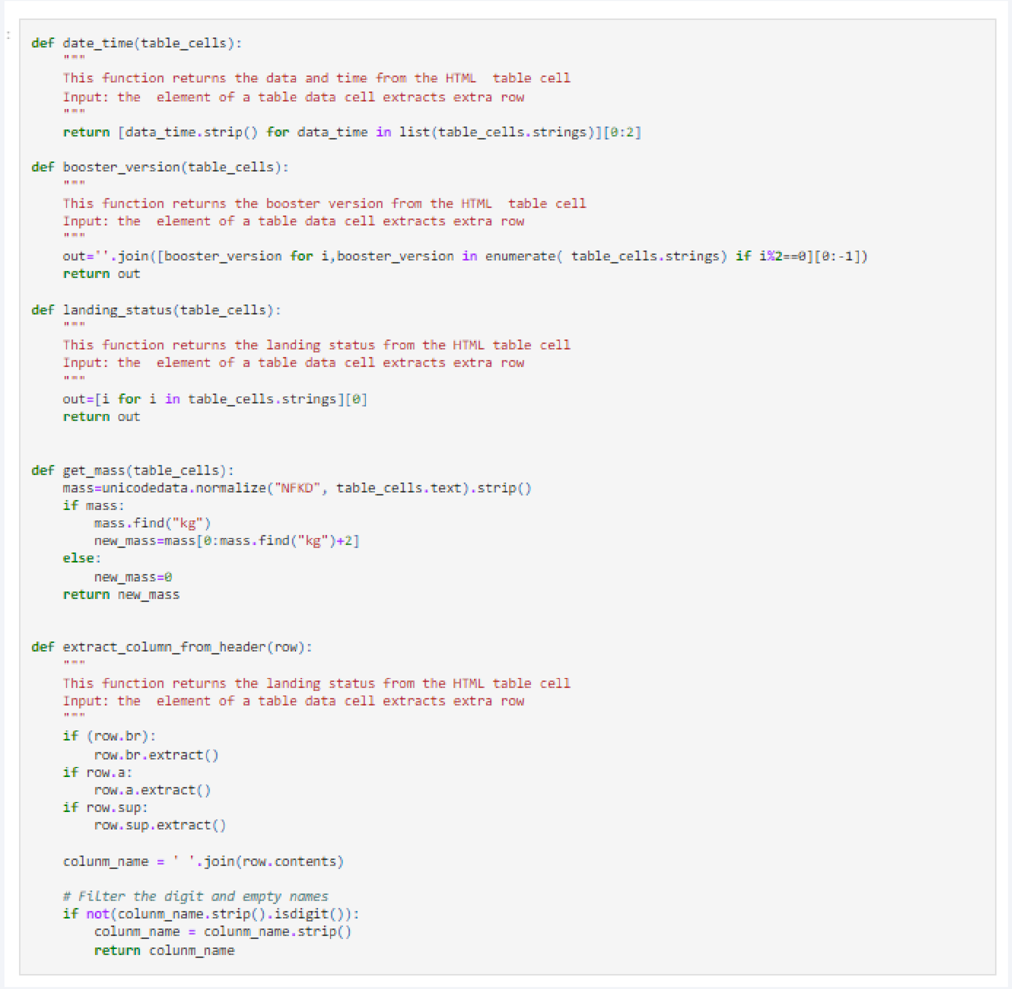
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```



```
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
        return colunm_name
```

Thank you!