IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Clement Gavaskar Kodamala
05/27/2022

Outline

Executive Summary

- ❖Summary of methodologies
  - ➢ Data Collection Through API
  - ➢ Data Collection with Web Scraping
  - ➢ Data Wrangling
  - ➢ Exploratory Data Analysis with SQL
  - ➢ Exploratory Data Analysis with Data Visualization
  - ➢ Interactive Visual Analysis with Folium
  - ➢ Machine Learning Model and Prediction
- ❖Summary of all results
  - ➢ Exploratory Data Analysis Results
  - ➢ Interactive Dashboards for Insights
  - ➢ Predictive Analytics Results

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. The goal of this project is to build a data science machine learning model to determine if the first stage can land successfully and in what circumstances or with what features and to determine the cost of the launch.

Problems you want to find answers

- ✓ Can the first stage be landing successfully for reuse?
- ✓ What features determine the successful landing?
- ✓ Identify the interdependencies and correlation of features for a successful landing?
- ✓ What are the environmental and operational setup needed to ensure a successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology
  - The SpaceX data was collected from an API, specially the SPACEX REST API. The REST API returns us a JSON format data and we normalized that to fit in our tables of analysis. We used web scraping from the SPACEX related wiki pages for falcon9 launch data.
- Perform data wrangling
  - Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models. Reviewed different attributes that impacts the outcome and landing outcomes are converted into classes (either 1 or 0). Applied one-hot encoding to categorical features for use in predictive analytics machine learning models.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build a predictive model, using machine learning techniques, to help business function more efficiently. Split the data into training testing data. Train and test different classification models. Using Hyperparameter grid search, Identify the best model suitable for accuracy.

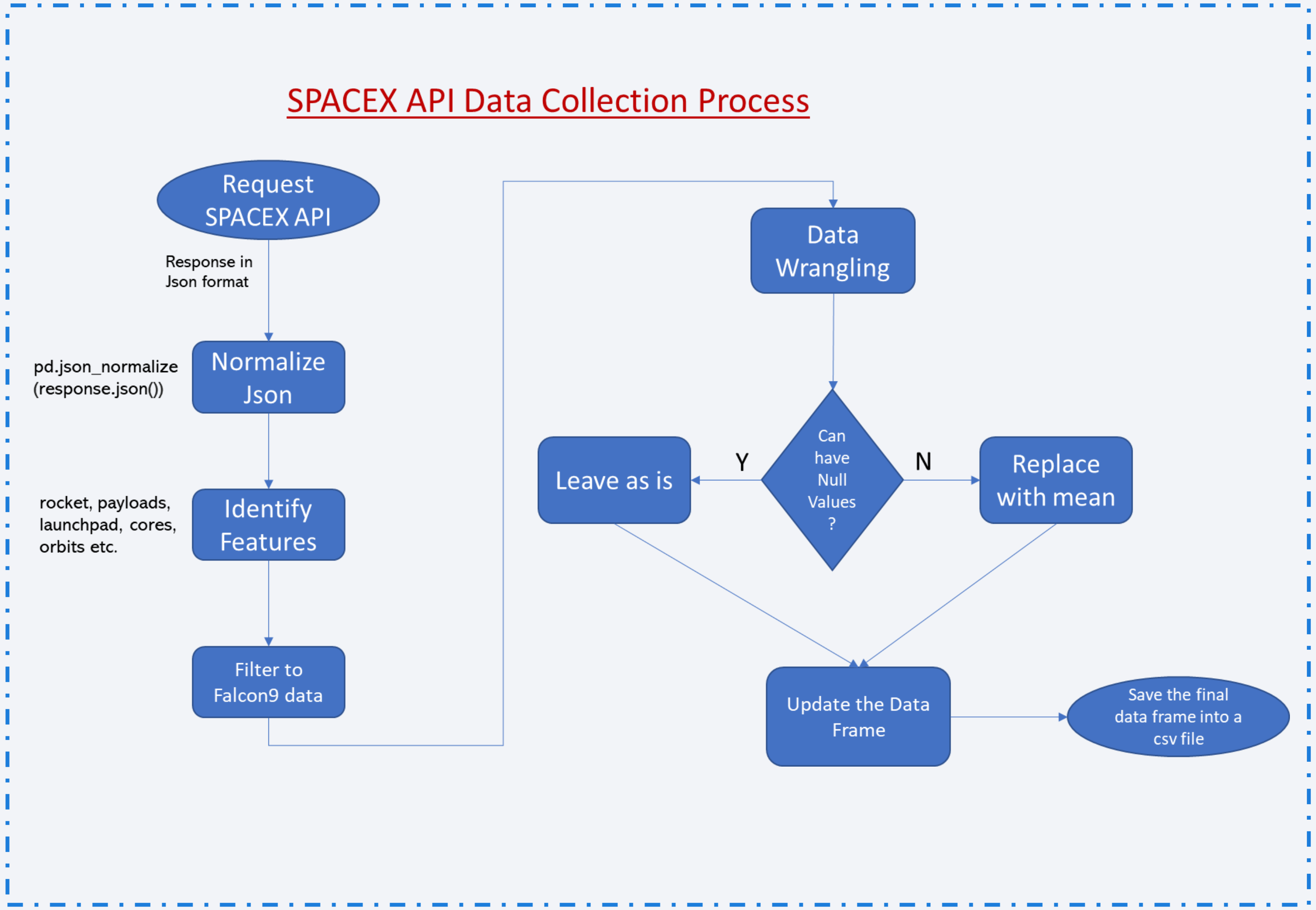
Data Collection

- The data was collected using 2 methods
  - SPACEX REST API:
    - https://api.spacexdata.com/v4/launches/past
    - Request to the SpaceX API
    - Clean the requested data by removing or replacing null values not good for data analysis
  - Web scraping of SPACEX related WIKI pages:
    - https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
    - Extract a Falcon 9 launch records HTML table from Wikipedia
    - Parse the table and convert it into a Pandas data frame using BeautifulSoup

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, cleaned the requested data by filtering to falcon 9 launches and removing null values under booster version columns for predictive analysis.
- GitHub link to the notebook:

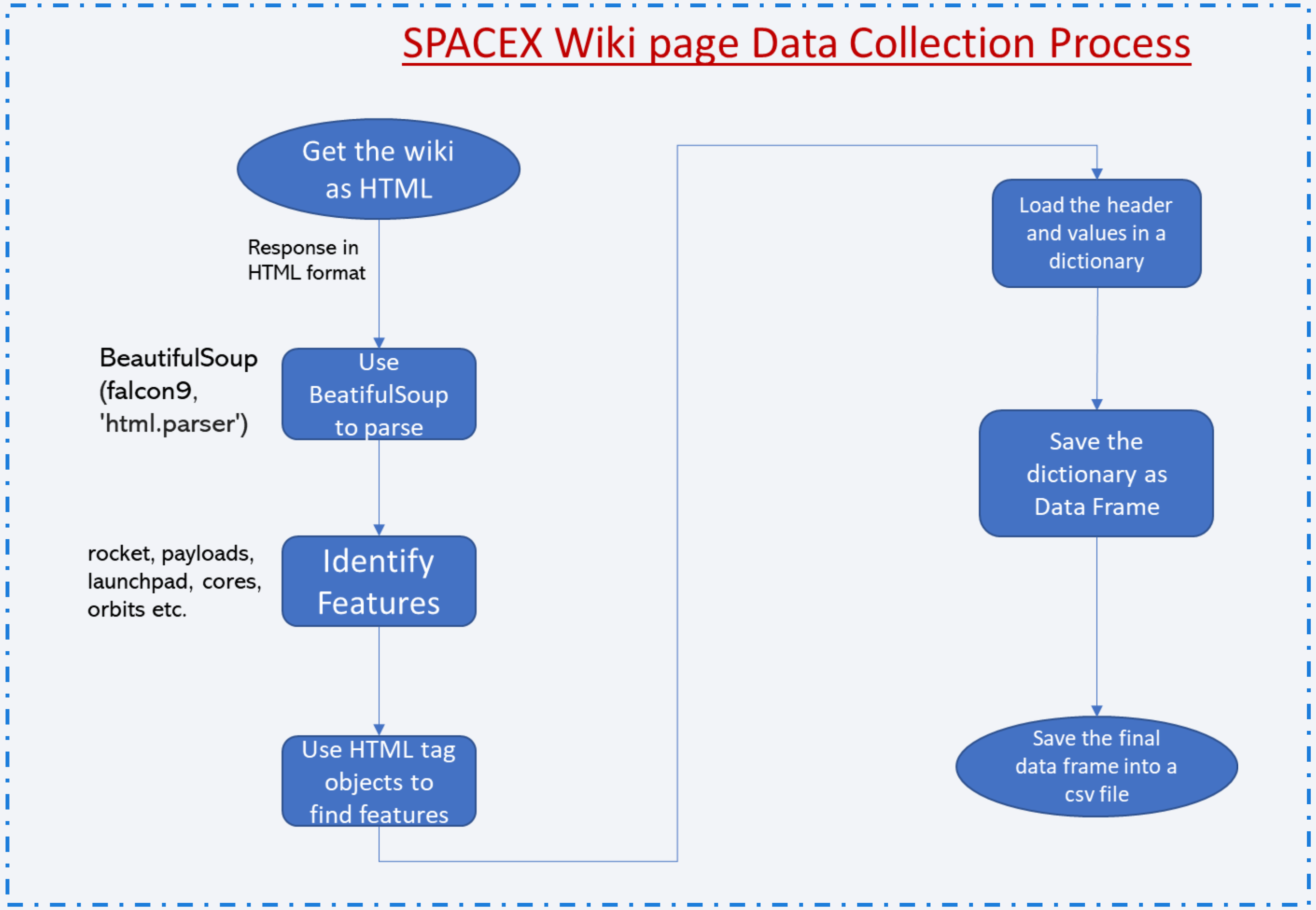
https://github.com/gkclement/IBMDataScienceProfessionalCetrificate/blob/3a5aed9d1d06d725bee6336de6260228e4600a84/Data%20Collection%20API.ipynb



Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is

https://github.com/gkclement/IBMDataScienceProfessionalCetrificate/blob/3a5aed9d1d06d725bee6336de6260228e4600a84/Data%20Collection%20with%20Web%20Scraping.ipynb

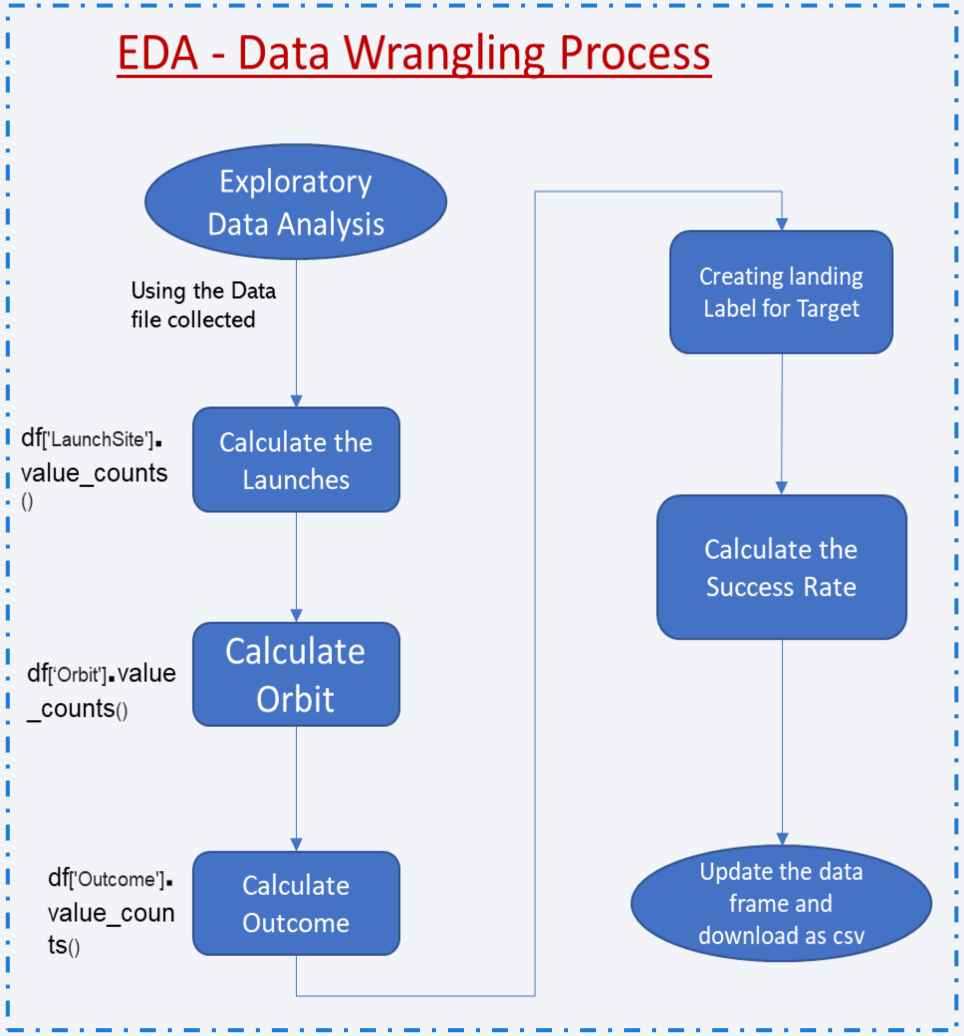


SPACEX Wiki page Data Collection Process
Get the wiki as HTML
Response in HTML format
BeautifulSoup (falcon9, 'html.parser')
Use BeatifulSoup to parse
rocket, payloads, launchpad, cores, orbits etc.
Identify Features
Use HTML tag objects to find features
Load the header and values in a dictionary
Save the dictionary as Data Frame
Save the final data frame into a csv file

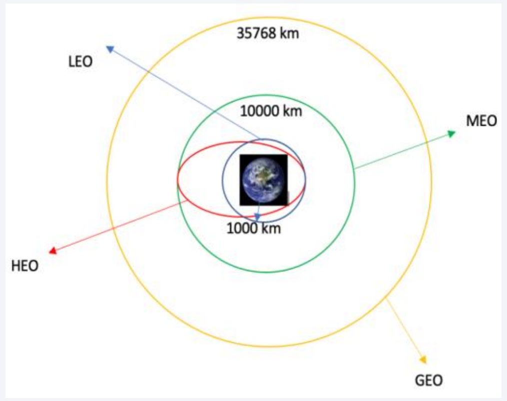
Data Wrangling

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is

https://github.com/gkclement/IBMDataScienceProfessionalCetrificate/blob/3a5aed9d1d06d725bee6336de6260228e4600a84/Data%20Wrangling.ipynb


EDA - Data Wrangling Process
Exploratory Data Analysis
Using the Data file collected
df['LaunchSite'].value_counts()
Calculate the Launches
df['Orbit'].value_counts()
Calculate Orbit
df['Outcome'].value_counts()
Calculate Outcome
Creating landing Label for Target
Calculate the Success Rate
Update the data frame and download as csv

Orbit Types



EDA with Data Visualization

- We did exploratory Data Analysis
- Prepared Data Feature Engineering
- Using Pandas and Matplotlib
- Identified the relation between the features that impacts the success rate
- Create dummy variables to categorical columns
- Cast all numerical columns to flaot64
- The link to the notebook is:

https://github.com/gkclement/IBMDataScienceProfessionalCetrificate/blob/3a5aed9d1d06d725bee6336de6260228e4600a84/EDA%20with%20Visualization.ipynb

EDA with SQL

- **Identified the names of the unique launch sites in the space mission**
  - **%sql** select distinct launch_site from SPACEXTBL
- **Identified 5 records where launch sites begin with the string 'CCA'**
  - **%sql** select * from SPACEXTBL where launch_site like 'CCA%' limit 5
- **Identified the total payload mass carried by boosters launched by NASA (CRS)**
  - %sql select sum(payload_mass__kg_) as total_payload_mass from SPACEXTBL where customer = 'NASA (CRS)'
- **Identified average payload mass carried by booster version F9 v1.1**
  - %sql select avg(payload_mass__kg_) as avg_payload_mass from SPACEXTBL where booster_version='F9 v1.1'
- **List the date when the first successful landing outcome in ground pad was achieved.**
  - %sql select min(DATE) as FIRST_SUCCESS from SPACEXTBL where landing__outcome='Success (ground pad)'
- **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 landing__outcome='Success (drone ship)' and (payload_mass__kg_ > 4000 and payload_mass__kg_ < 6000)

EDA with SQL

- List the total number of successful and failure mission outcomes
  - %sql select mission_outcome, count(mission_outcome) as total_count from SPACEXTBL group by mission_outcome
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
  - %sql select distinct booster_version from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) as max_payload_mass from SPACEXTBL)
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - %sql select DATE, booster_version, launch_site, landing__outcome from SPACEXTBL where landing__outcome='Failure (drone ship)' and substr(DATE,1,4) = '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
  - %sql select landing__outcome, count(landing__outcome) as cnt_land_out from SPACEXTBL where DATE between '2010-06-04' and '2017-03-20' group by landing__outcome order by count(landing__outcome)
- Link to the Notebook is:
- https://github.com/gkclement/IBMDataScienceProfessionalCetrificate/blob/3a5aed9d1d06d725bee6336de6260228e4600a84/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Circled the Launch Sites with Circle in the folium map added a label to the site.
- Marked the Success or failure in each site using marker color to the launch_outcome.
- Created clusters of the Launch sites for each flight or launch with success rate
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- Using add_child(lines) marked the proximity and the distance in km with a line drawn between site and the cities, coastlines and railways
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites in close proximity to railways?
  - Are launch sites in close proximity to highways?
  - Are launch sites in close proximity to coastline?
  - Do launch sites keep certain distance away from cities?

Observation: Launch Sites are in close proximity to coastlines and highways and railways, but Cities are a bit far.

- Link to the Notebook is:
- https://github.com/gkclement/IBMDataScienceProfessionalCetrificate/blob/3a5aed9d1d06d725bee6336de6260228e4600a84/Interactive%20Visual%20Analytics%20with%20Folium%20.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash. This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart.
- We plotted pie charts showing the total launches by a certain launch site or all combined.
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the python file in GitHub:
- https://github.com/gkclement/IBMDataScienceProfessionalCetrificate/blob/3a5aed9d1d06d725bee6336de6260228e4600a84/spacex_dash_app.py

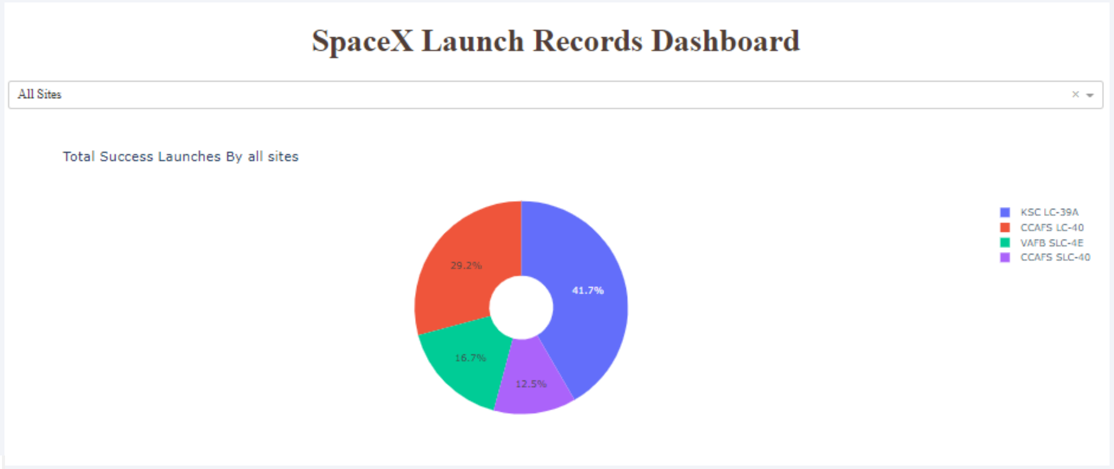
Build a Dashboard with Plotly Dash

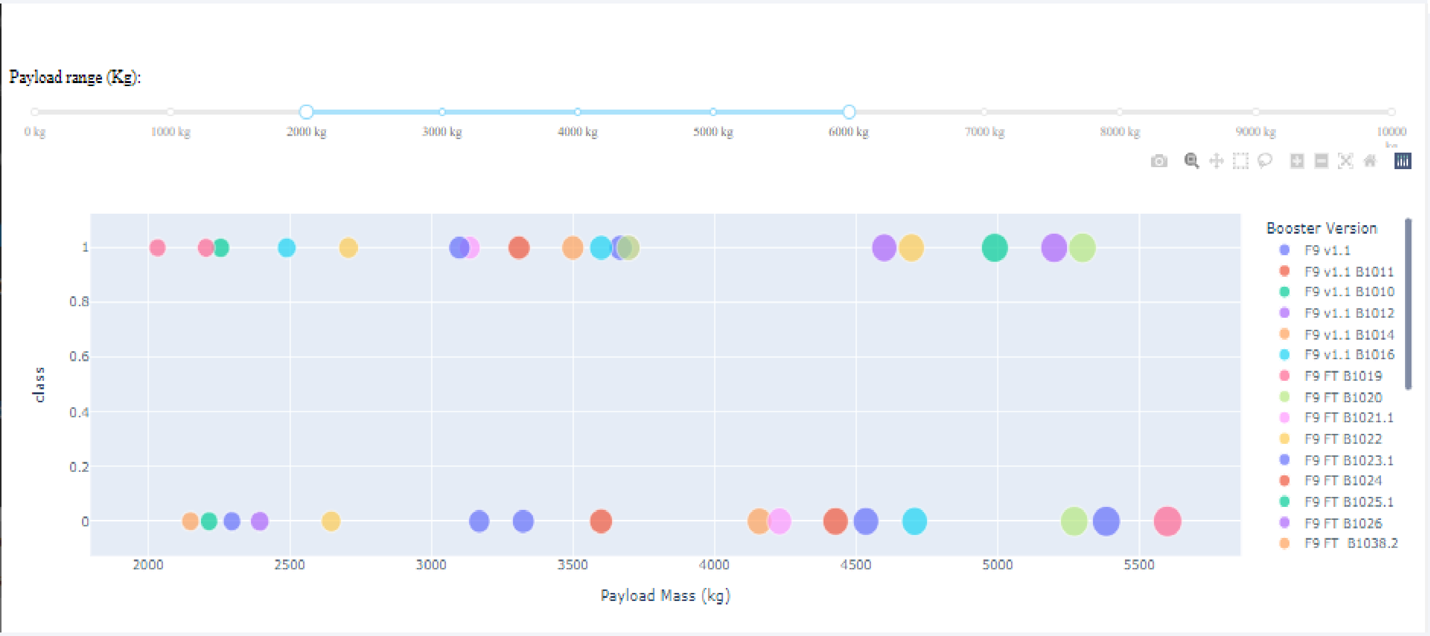
Using the Dashboard, we can answer the following questions:

1. Which site has the largest successful launches?
   – KSC-LC-39A has the highest successful launches
2. Which site has the highest launch success rate?
   - KSC-LC-39A has the highest success rate
3. Which payload range(s) has the highest launch success rate?
   – 2000 kgs to 6000 kgs pay load mass have the highest success rate
4. Which payload range(s) has the lowest launch success rate?
   – less than 2000 kgs
5. Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate?
   - F9 v1.1

Build a Dashboard with Plotly Dash

Pie Chart with All Launch sites in the Drop down as options.





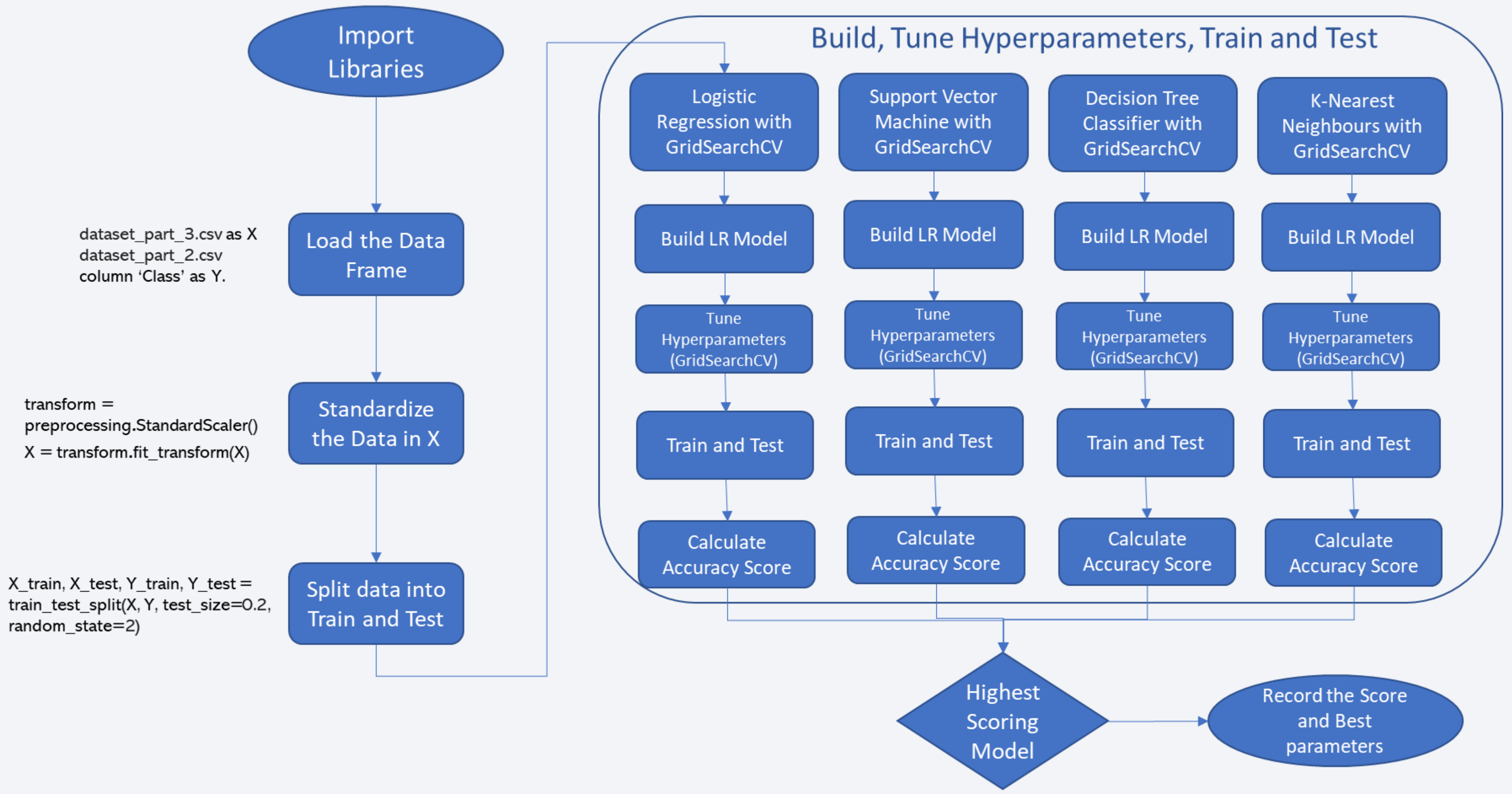
Payload vs Success rate with slide bar that we can select a specific payload range.

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model based on the highest best score which is closure to 1.
- Based on the test scores we found that Decision Tree Classification model is the best for predicting SpaceX – falcon 9 launches with the best score as 0.8732.
- The link to the notebook is:
    - https://github.com/gkclement/IBMDataScienceProfessionalCetrificate/blob/3a5aed9d1d06d725bee6336de6260228e4600a84/Machine%20Learning%20Prediction.ipynb

Predictive Analysis (Classification)

Best Machine Learning Model Identification



Import Libraries
dataset_part_3.csv as X
dataset_part_2.csv
column 'Class' as Y.
Load the Data Frame
transform =
preprocessing.StandardScaler()
X = transform.fit_transform(X)
Standardize the Data in X
X_train, X_test, Y_train, Y_test =
train_test_split(X, Y, test_size=0.2,
random_state=2)
Split data into Train and Test
Build, Tune Hyperparameters, Train and Test
Logistic Regression with GridSearchCV
Support Vector Machine with GridSearchCV
Decision Tree Classifier with GridSearchCV
K-Nearest Neighbours with GridSearchCV
Build LR Model
Build LR Model
Build LR Model
Build LR Model
Tune Hyperparameters (GridSearchCV)
Tune Hyperparameters (GridSearchCV)
Tune Hyperparameters (GridSearchCV)
Tune Hyperparameters (GridSearchCV)
Train and Test
Train and Test
Train and Test
Train and Test
Calculate Accuracy Score
Calculate Accuracy Score
Calculate Accuracy Score
Calculate Accuracy Score
Highest Scoring Model
Record the Score and Best parameters

Results

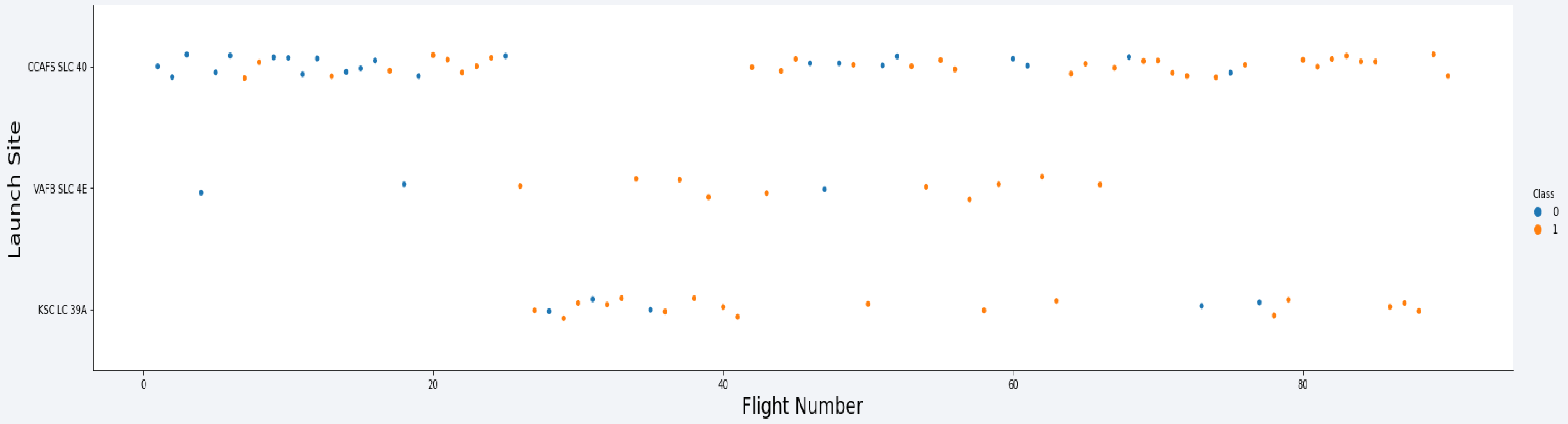
- Exploratory data analysis results
  - Identified the Features for Successful landing:
  - FlightNumber,Date,BoosterVersion,PayloadMass,Orbit,LaunchSite,Outcome,Flights, GridFins,Reused,Legs,LandingPad,Block,ReusedCount,Serial,Longitude,Latitude
- Interactive analytics demo in screenshots
  - Identified the Best suitable environments for successful landing:
  - On Coastal lines, far from Cities, close proximity to railways and highways.
- Predictive analysis results
  - Decision Tree Classification Model is best suitable to predict the Falcon9 launches success or failure with the best rate as 0.8732.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

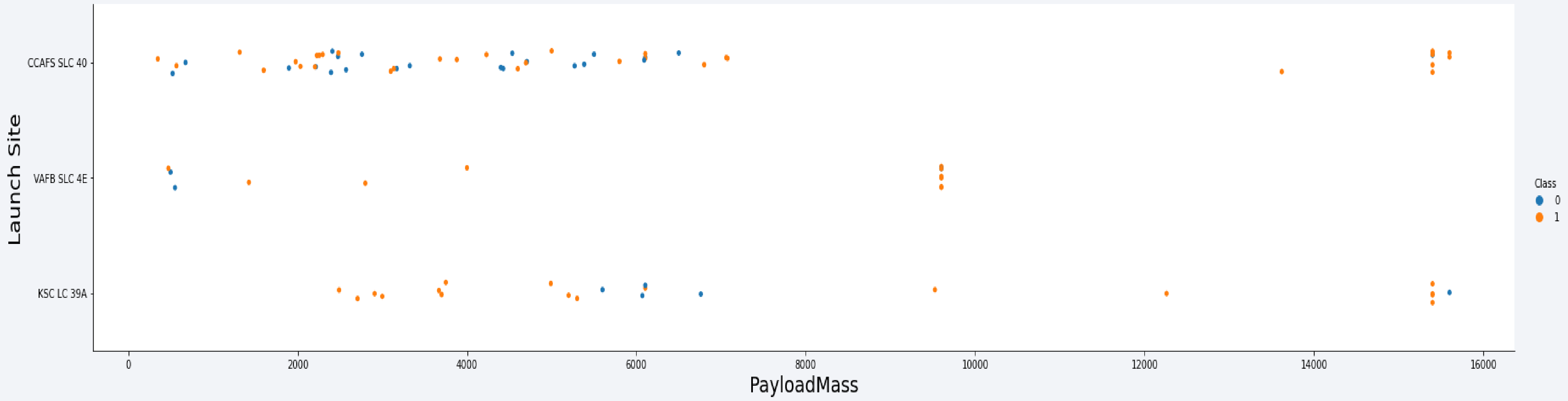
Visualized the relationship between Flight Number and Launch Site



Observation: In any Launch Site As the Flight number Increases the success rate increases

Payload vs. Launch Site

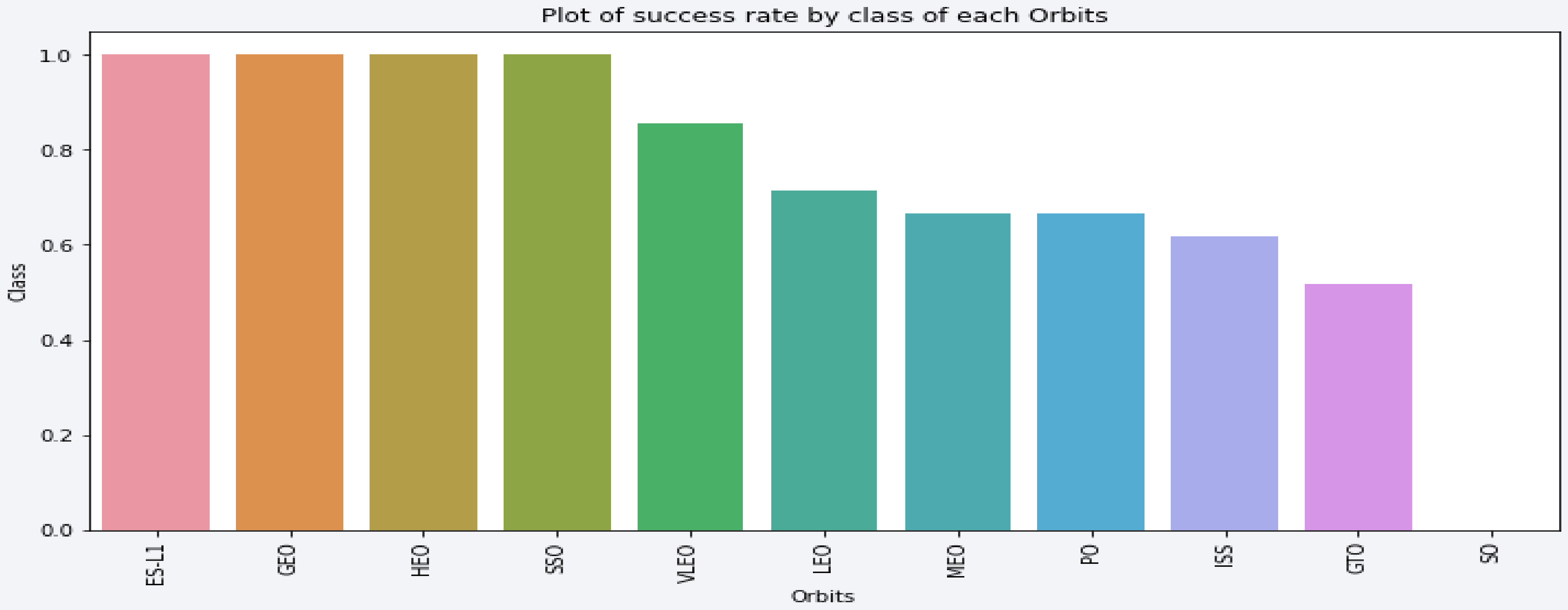
Visualize the relationship between Payload and Launch Site



Observation: We see that Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy pay load mass(greater than 10000).

Success Rate vs. Orbit Type

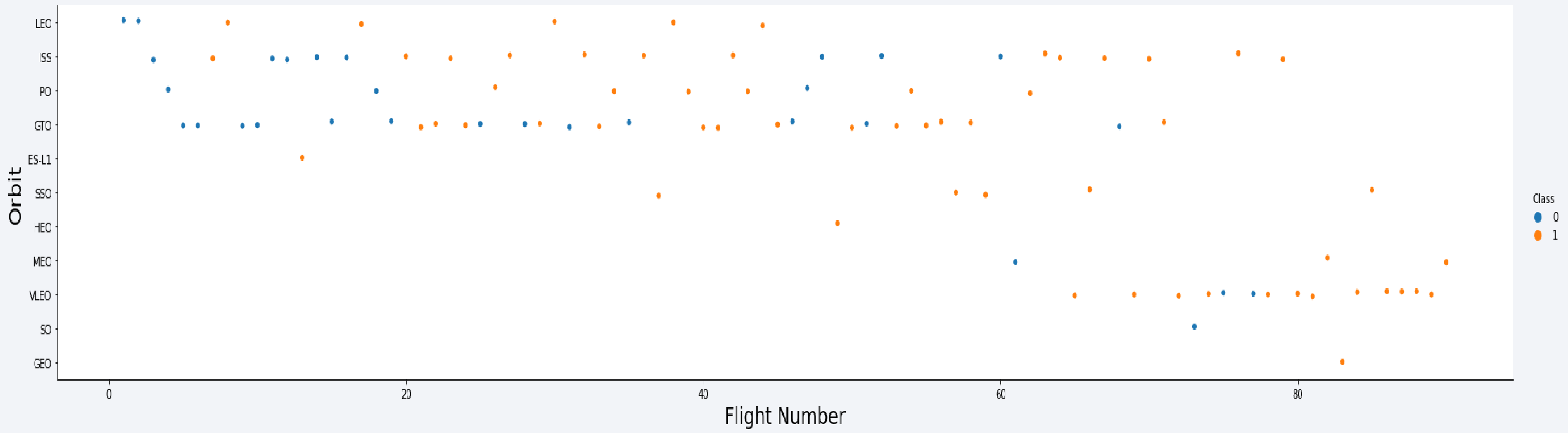
Visualize the relationship between Success Rate and Orbit type



Observation: We see that The plot we observe that Orbits ES-L1, GEO, HEO and SSO has the highest success rate. SO orbit has the least success rate

Flight Number vs. Orbit Type

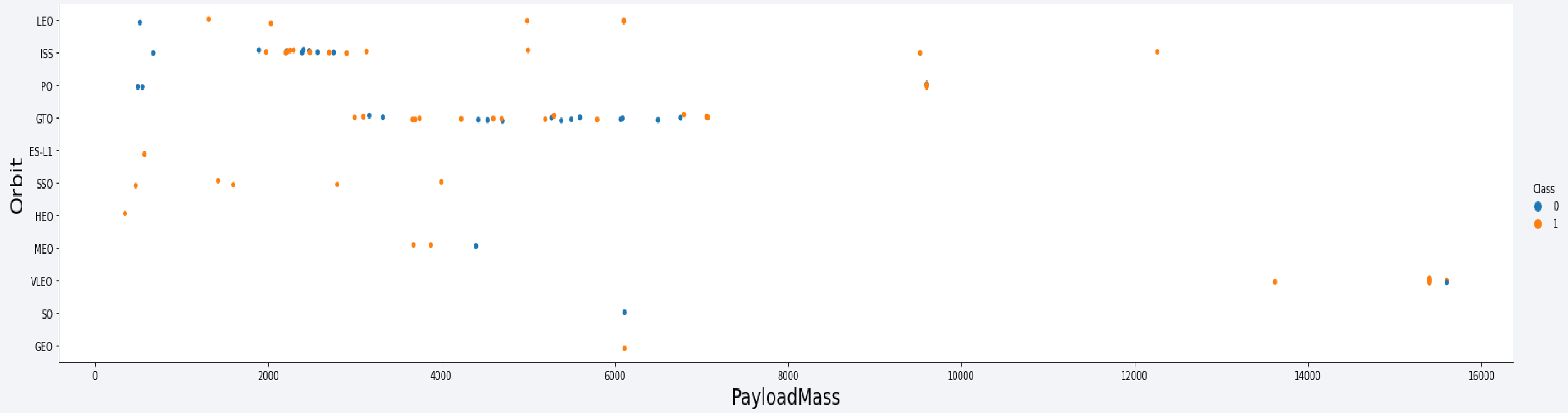
Visualize the relationship between Flight Number and Orbit Type



Observation: You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type

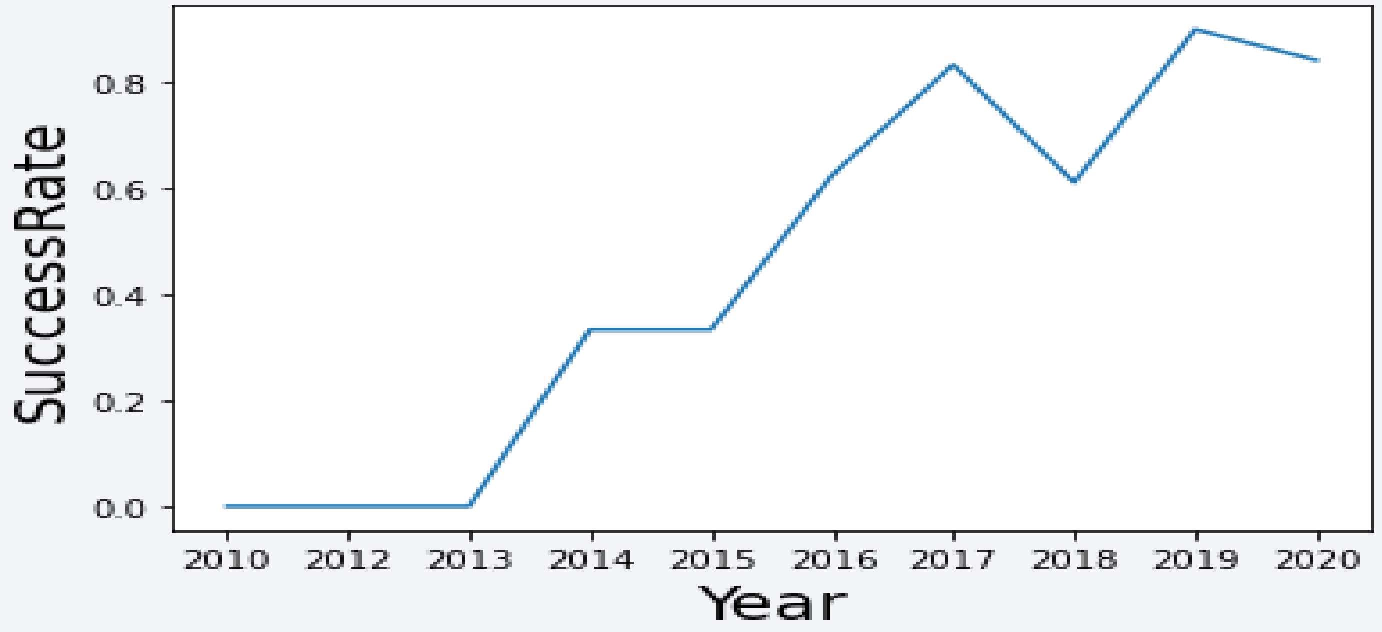
Visualize the relationship between Payload and Orbit type



Observation: We see that with heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS. However, for GTO we cannot distinguish this well as both positive landing rate and negative landing (unsuccessful mission) are both there here.

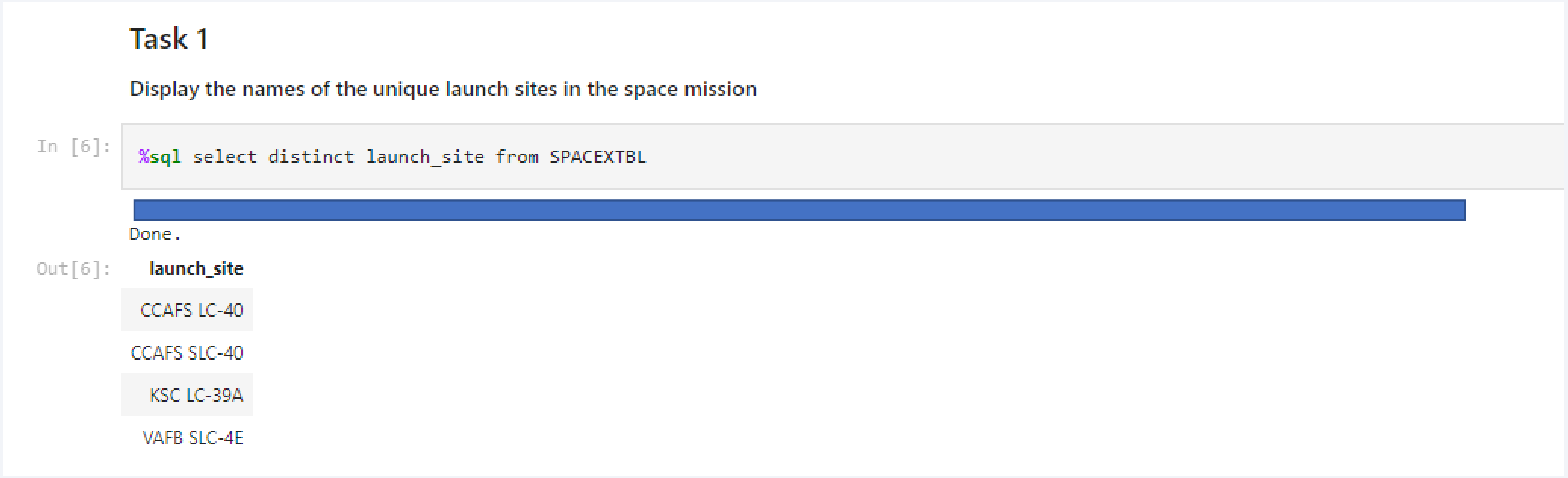
Launch Success Yearly Trend

Visualize Launch Success Yearly Trend



Observation: We see that the success rate since 2013 kept increasing till 2020

All Launch Site Names


Task 1

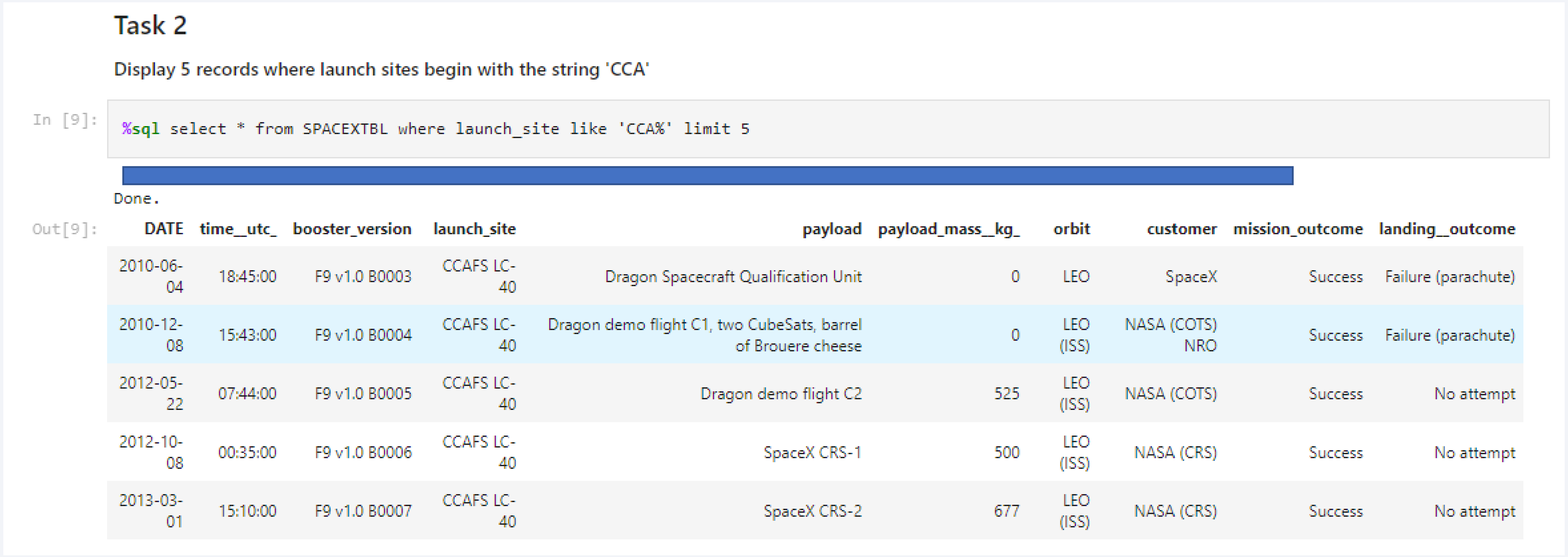
Display the names of the unique launch sites in the space mission

```
In [6]: %sql select distinct launch_site from SPACEXTBL
```

```
Done.
Out[6]:  launch_site
         CCAFS LC-40
         CCAFS SLC-40
         KSC LC-39A
         VAFB SLC-4E
```

- Based on the Query Result we found that there are only 4 launch sites for SpaceX.
- They are CCAFS LC-40, CCAFS SLC-40, KSC LC-39A and VAFB SLC-4E.
- To get the above result we use ‘Distinct’ keyword in the select query for the launch_site column.

Launch Site Names Begin with 'CCA'



Task 2

Display 5 records where launch sites begin with the string 'CCA'

```
In [9]: %sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5
```

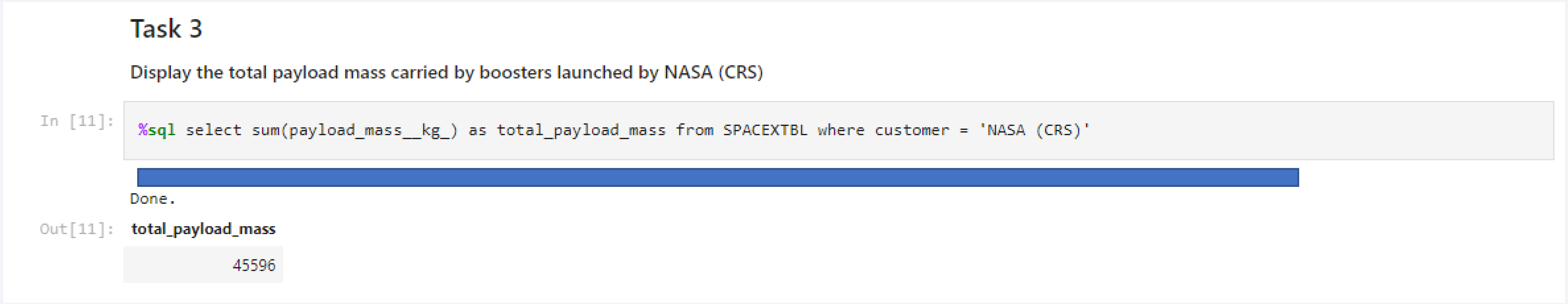
```
Done.
```

Out[9]:

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

- We used like calue for the launch_site with 'CCA%' to display the launch sites begin with 'CCA' and used Limit clause to limit to first 5 records.

Total Payload Mass



Task 3

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

```
In [11]: %sql select sum(payload_mass__kg_) as total_payload_mass from SPACEXTBL where customer = 'NASA (CRS)'
```

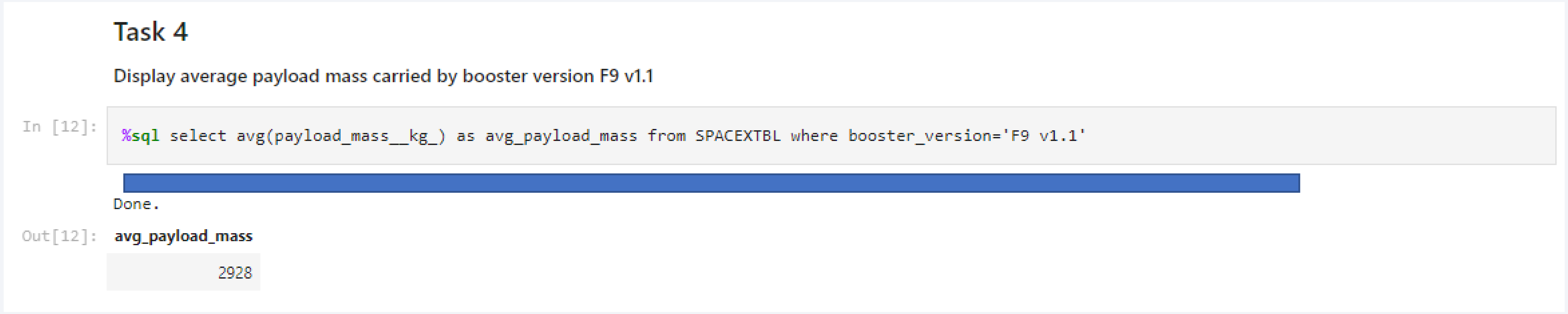
```
Done.
```

Out[11]:

total_payload_mass
45596

- We used sum() function on the records filtered using where clause with customer = ‘NASA (CRS)’
- The total payload mass used by the customer is 45596 from all the launches.

Average Payload Mass by F9 v1.1



Task 4

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

```
In [12]: %sql select avg(payload_mass__kg_) as avg_payload_mass from SPACEXTBL where booster_version='F9 v1.1'
```

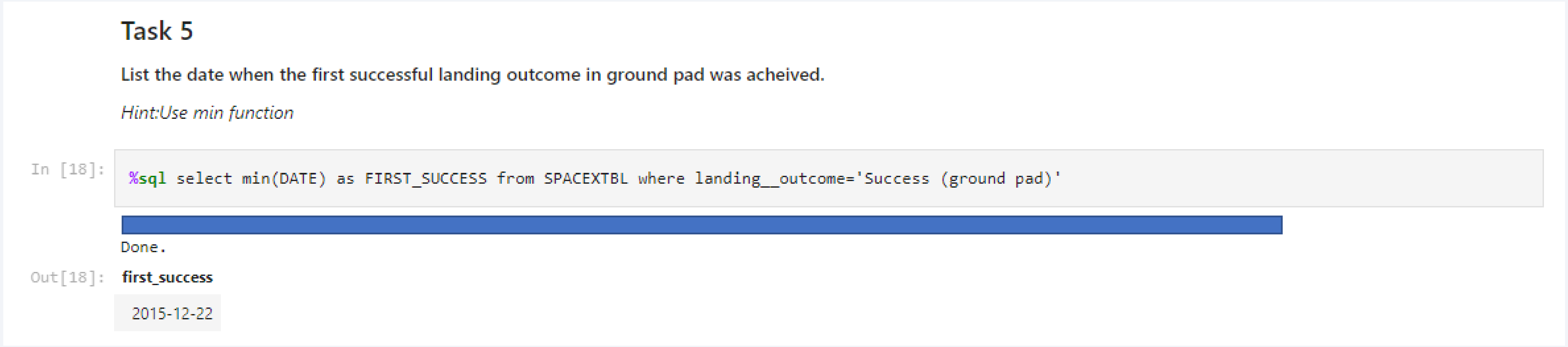
```
Done.
```

Out[12]:

avg_payload_mass
2928

- We used avg() function on the records filtered using where clause with booster_version = ‘F9 v1.1’
- The average payload mass used by the F9 v1.1 booster version is 2928.

First Successful Ground Landing Date



Task 5

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

Hint:Use min function

```
In [18]: %sql select min(DATE) as FIRST_SUCCESS from SPACEXTBL where landing__outcome='Success (ground pad)'
```

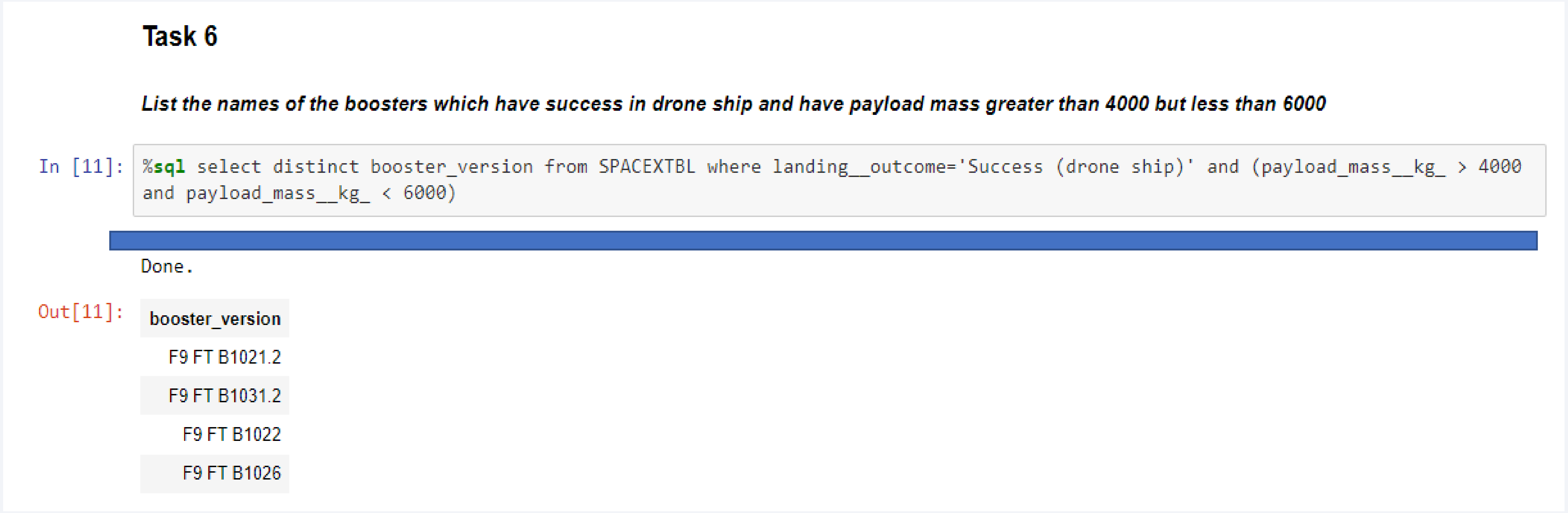
```
Done.
```

Out[18]:

first_success
2015-12-22

- We used min(DATE) function on the records filtered using where clause with landing__outcome='Success (ground pad)'
- The first successful landing on ground pad was on 22nd December 2015.

Successful Drone Ship Landing with Payload between 4000 and 6000



Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
In [11]: %sql select distinct booster_version from SPACEXTBL where landing__outcome='Success (drone ship)' and (payload_mass__kg_ > 4000
         and payload_mass__kg_ < 6000)
```

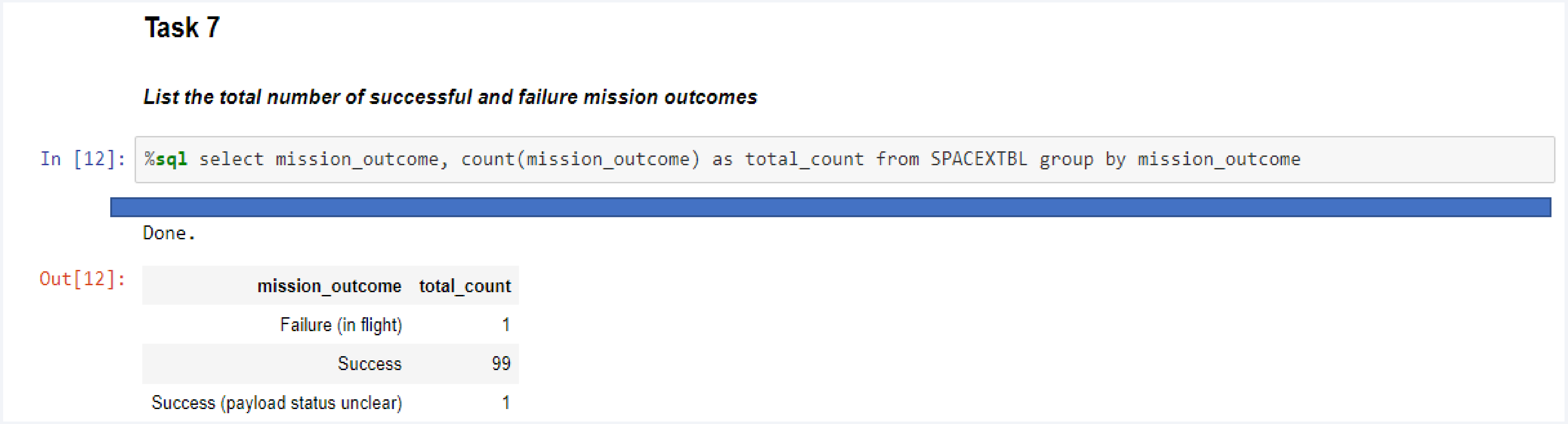
```
Done.
```

Out[11]:

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

- We used distinct booster version keyword on the records filtered using where clause with landing__outcome='Success (drone ship)' and payload mass range as (payload_mass__kg_ > 4000 and payload_mass__kg_ < 6000)

Total Number of Successful and Failure Mission Outcomes



Task 7

List the total number of successful and failure mission outcomes

```
In [12]: %sql select mission_outcome, count(mission_outcome) as total_count from SPACEXTBL group by mission_outcome
```

```
Done.
```

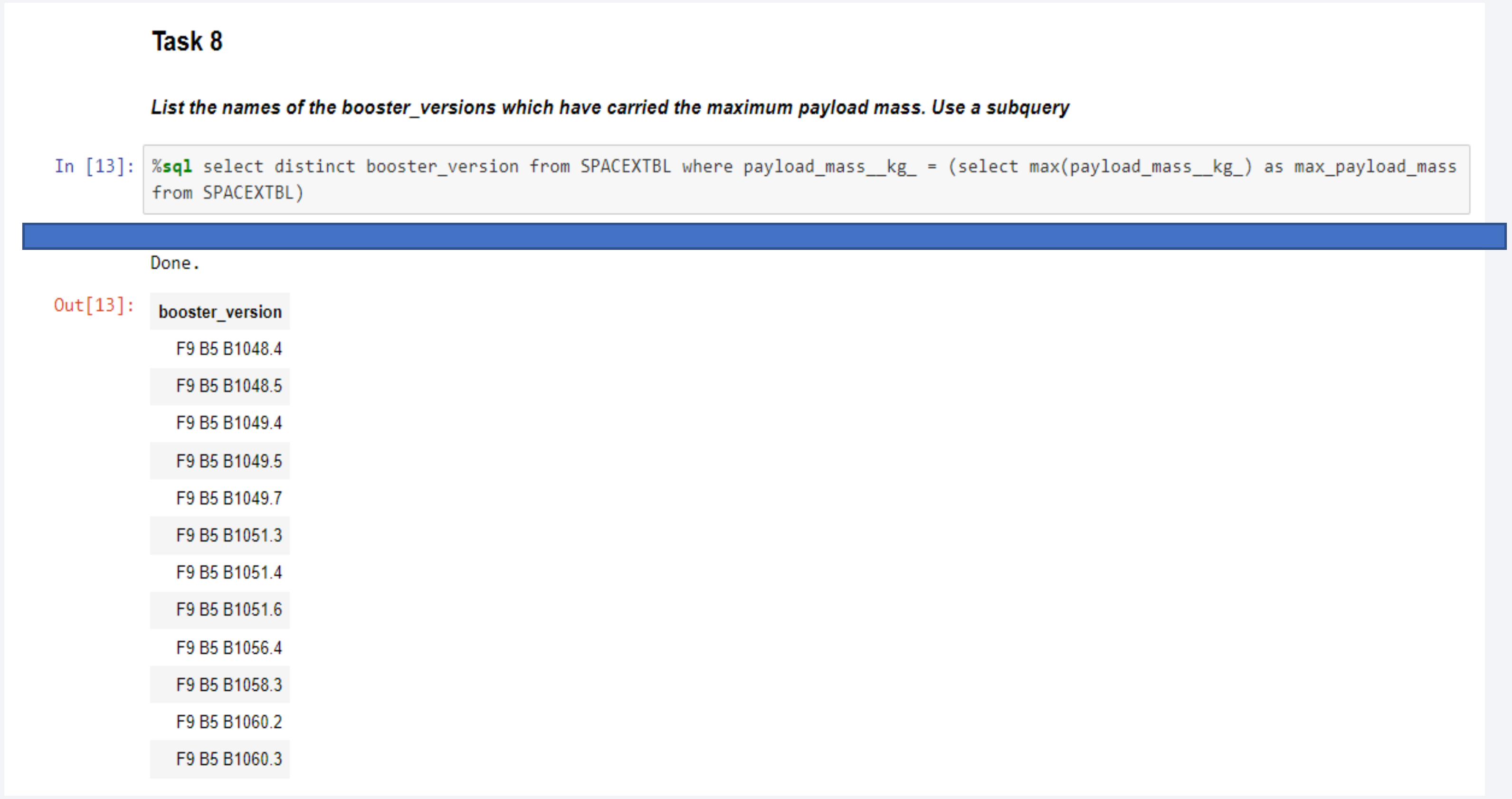
Out[12]:

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

- We selected mission_outcome and count(mission_outcome) on the records group by mission_outcome.
- There are 99 successful mission outcomes, 1 failure in flight and 1 success where payload status unclear.

Boosters Carried Maximum Payload

- We selected distinct booster versions where payload mass is equal to the maximum payload mass used among all launches.
- Used a subquery to find the max payload mass in the table and applied that in the where clause to filter the records.
- Then found the distinct booster versions on those records.



Task 8

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

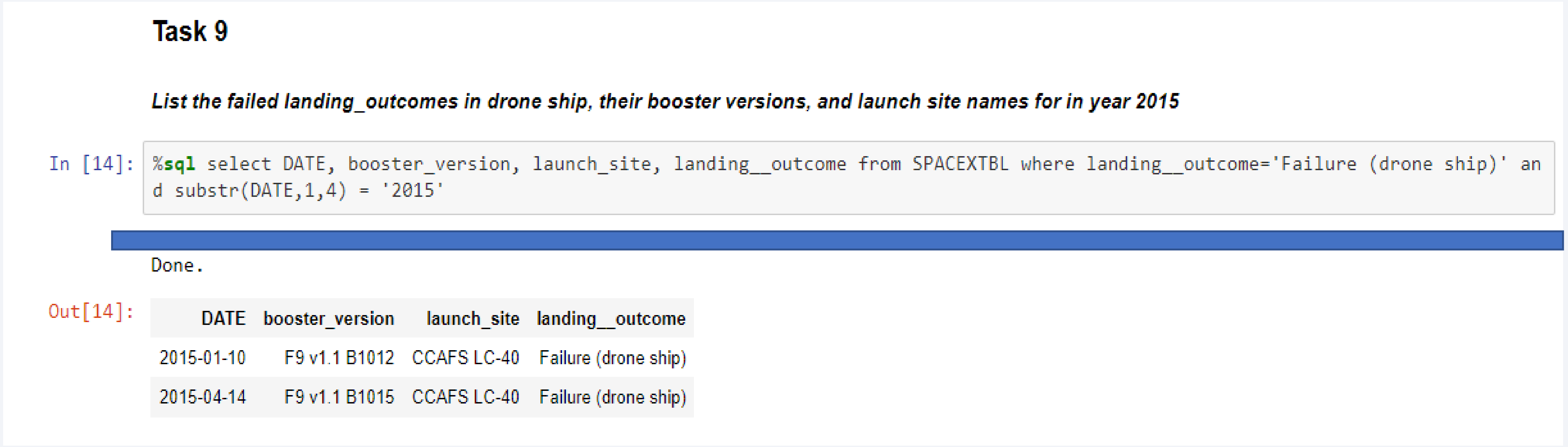
```
In [13]: %sql select distinct booster_version from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) as max_payload_mass
         from SPACEXTBL)
```

```
Done.
```

Out[13]:

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

2015 Launch Records



Task 9

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

```
In [14]: %sql select DATE, booster_version, launch_site, landing__outcome from SPACEXTBL where landing__outcome='Failure (drone ship)' an
         d substr(DATE,1,4) = '2015'
```

```
Done.
```

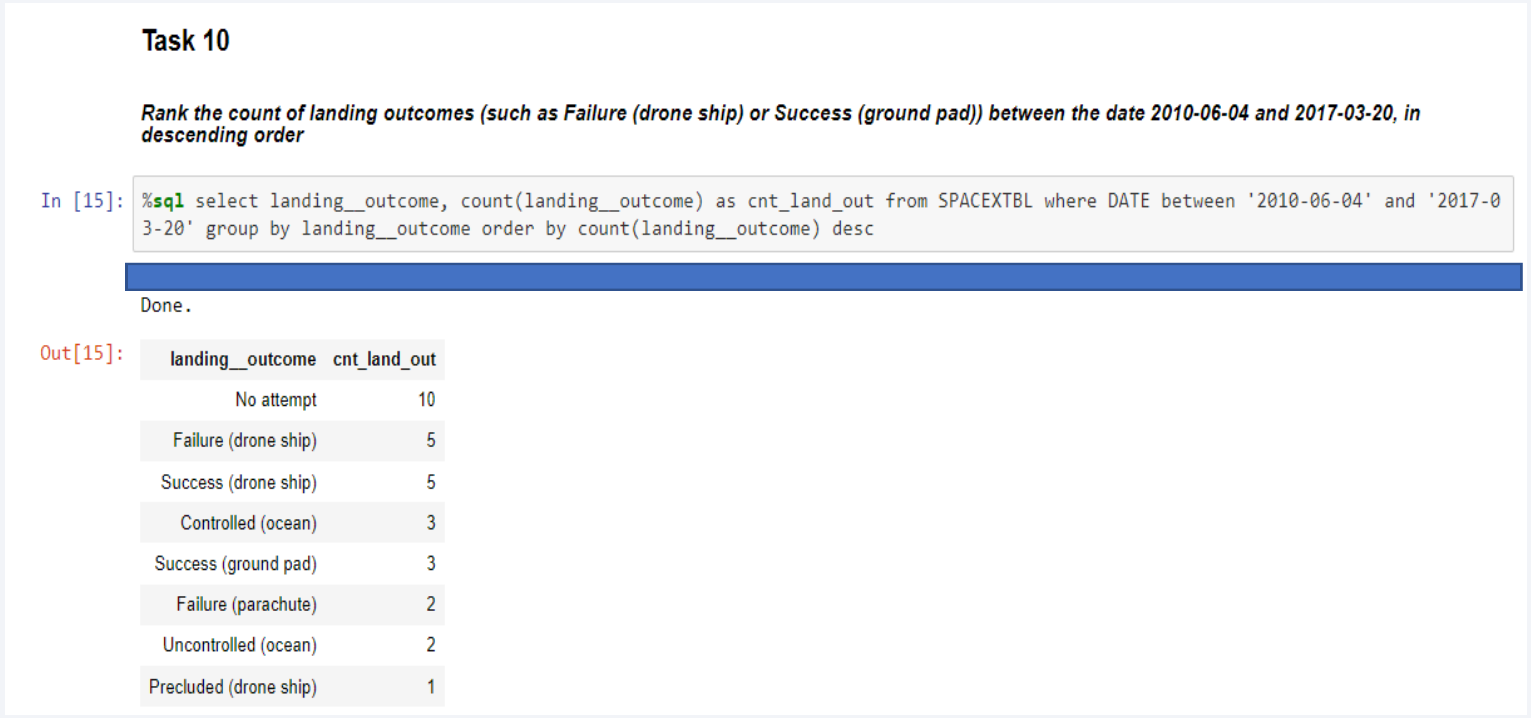
Out[14]:

DATE	booster_version	launch_site	landing__outcome
2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

- We selected date, booster version, launch_siteand landing_outcome filtered the records using where clause with landing__outcome='Failure (drone ship)'.
- Used substr() function to extract the year part of the date to filter the launches in the year 2015.
- There are 2 launches from CCAFS LC-40 site using F9 v1.1 B1012 booster version which are failed on dates 10th Jan 2015 and 14th April 2015.

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

- We selected landing__outcome and count(landing__outcome using a group by clause on landing__outcome to get the count of each landing outcome
- Used where clause to filter the records between the date range 2010-06-04 and 2017-03-20.
- Used order by clause on the count to display them in the descending order.
- Based on this the highest rank is for the “No attempt” with count 10.
- Failure (drone ship) and success (drone ship) are at the second rank and so forth.



Task 10

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

```
In [15]: %sql select landing__outcome, count(landing__outcome) as cnt_land_out from SPACEXTBL where DATE between '2010-06-04' and '2017-0
3-20' group by landing__outcome order by count(landing__outcome) desc
```

```
Done.
```

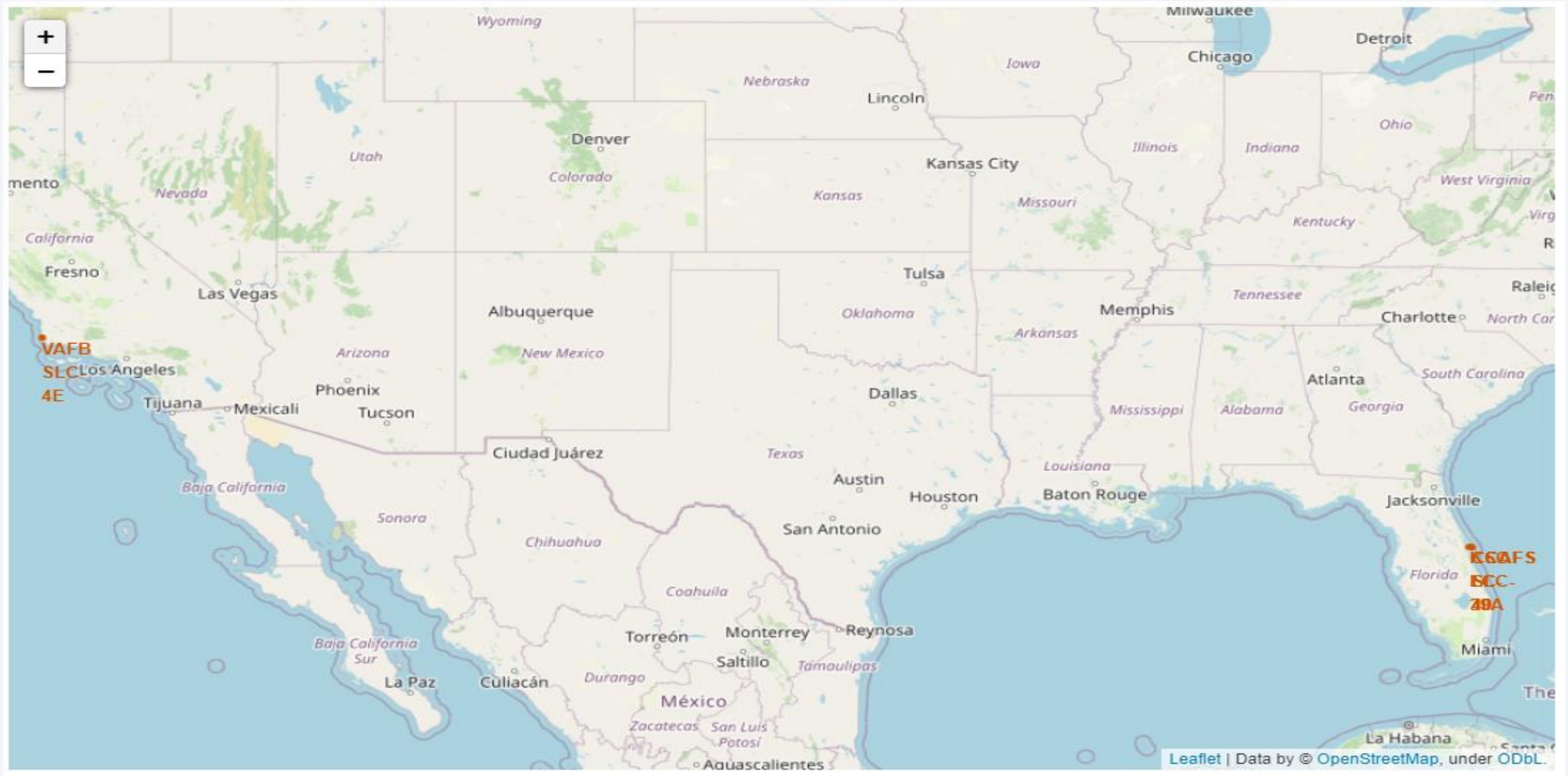
Out[15]:

landing__outcome	cnt_land_out
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

Section 3

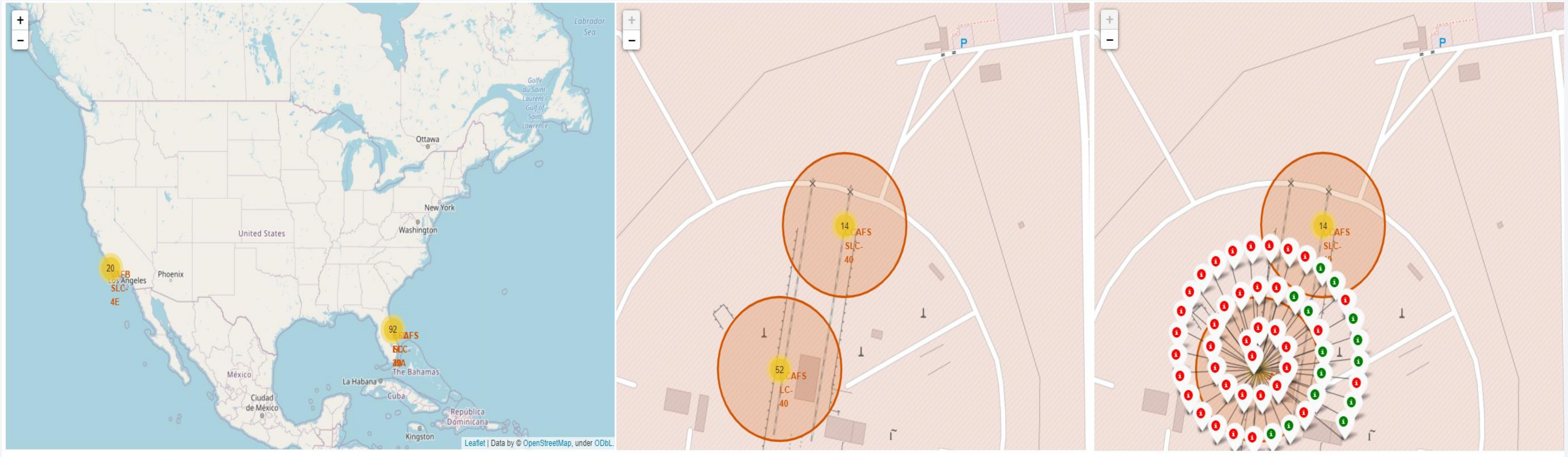
Launch Sites Proximities Analysis

All SpaceX Launch Sites



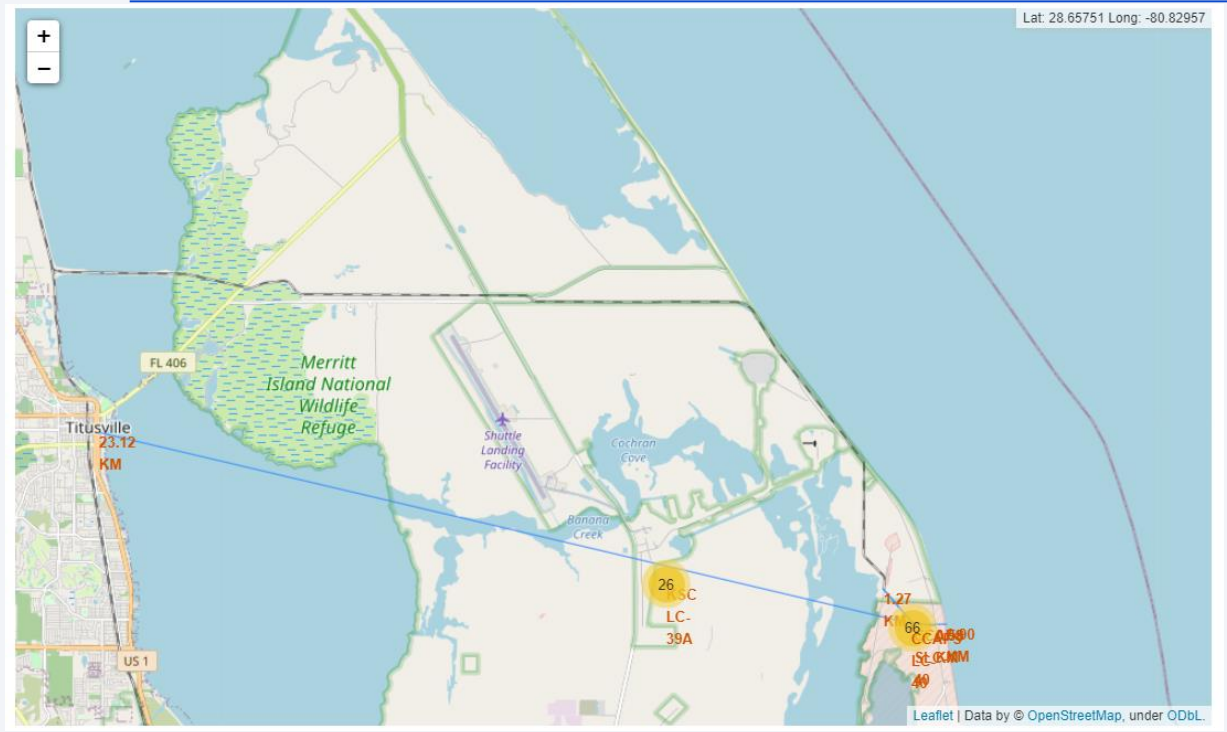
- All SpaceX launch Sites marked in the map using folium maps
- Using the Longitudes and latitudes of the launch sites to make them and circle them with a label as launch site name.

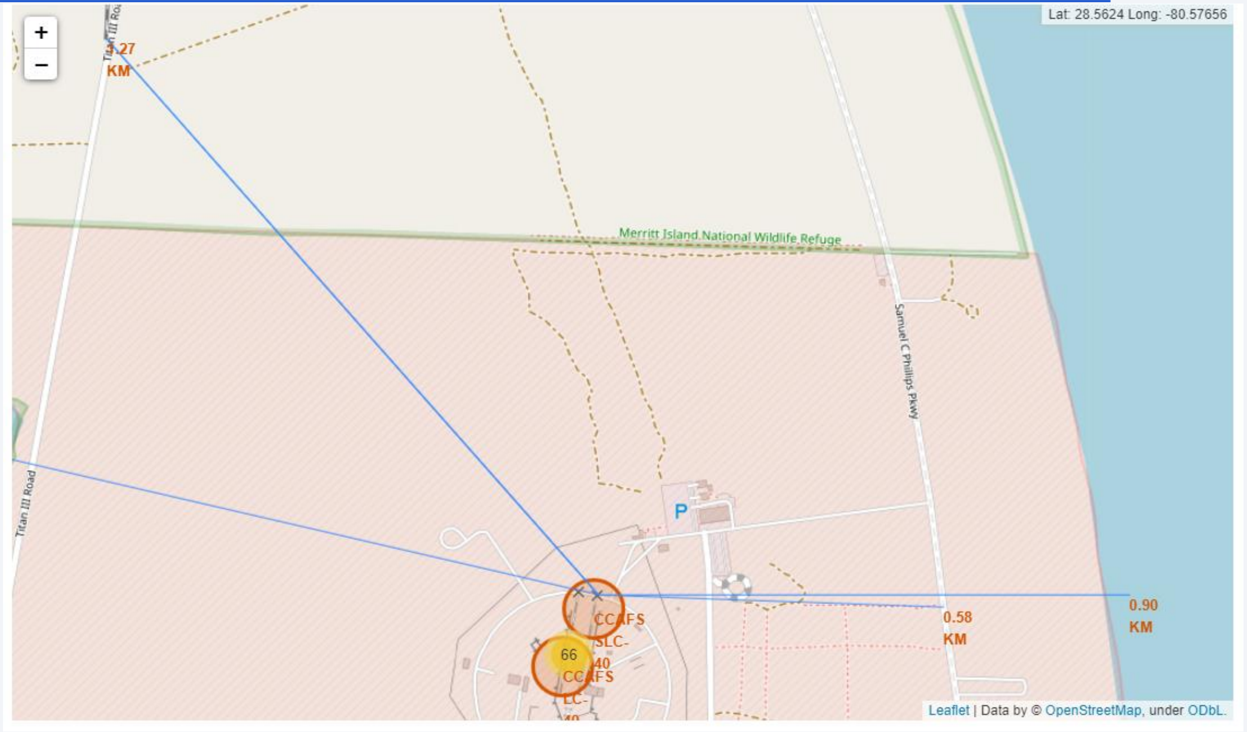
Mark the success/failed launches for each site on the map



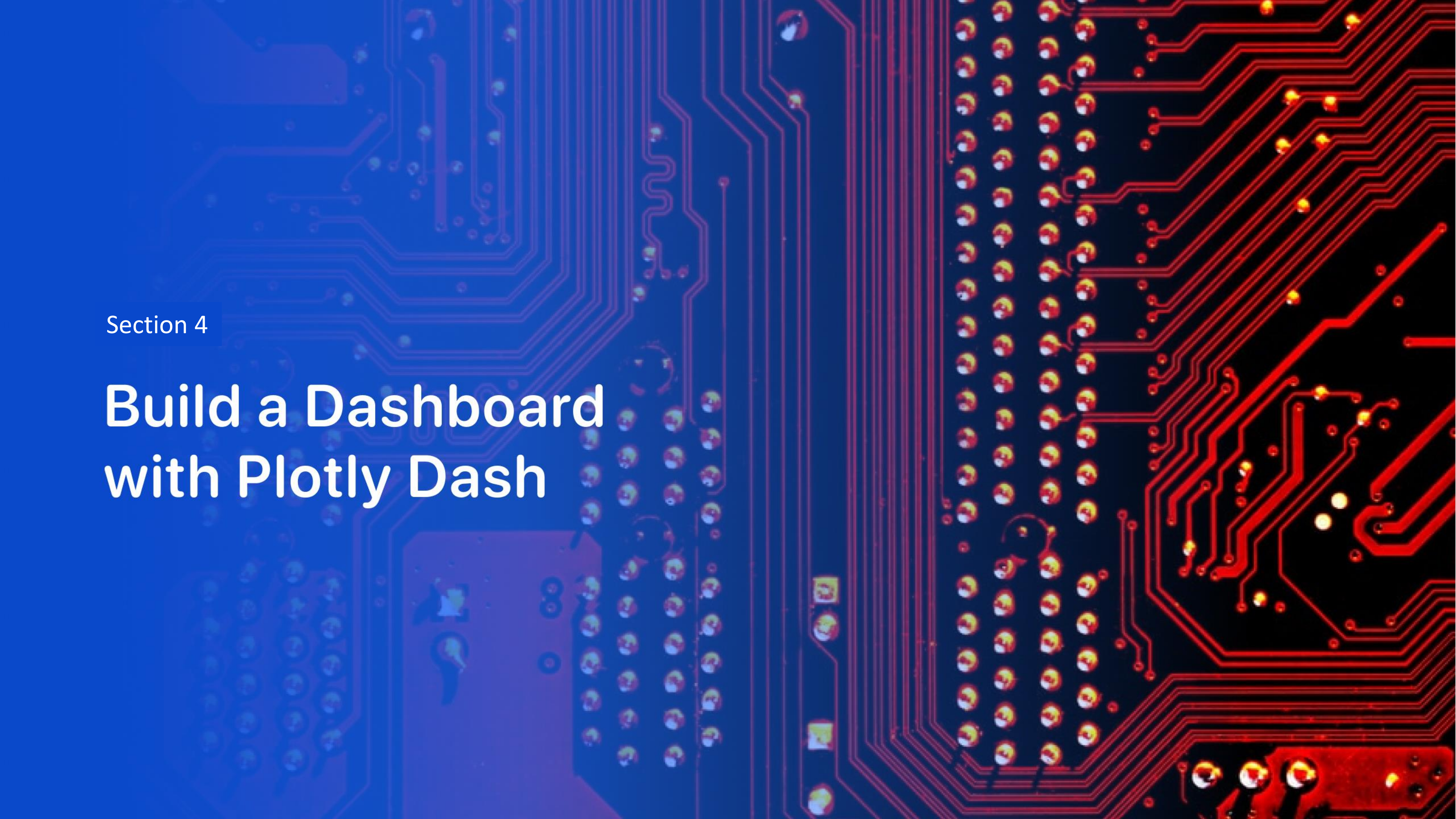
- Marked all launch sites with number of launches as clusters
- We used Folium maps markerCluster function to mark the clusters of launches per site.
- Used launch_outcome to mark success green and failure as red.

Launch Sites Proximity to Railways, Highways, cities and coast lines


Lat: 28.65751 Long: -80.82957
Merritt Island National Wildlife Refuge
Titusville
23.12 KM
FL 406
US 1
Shuttle Landing Facility
Cochran Cove
Banana Creek
26
KSC LC-39A
1.27 KM
66
CCAFS SLC-40
Leaflet | Data by © OpenStreetMap, under ODbL.


Lat: 28.5624 Long: -80.57656
1.27 KM
Titan III Road
Merritt Island National Wildlife Refuge
Samuel C Phillips Pkwy
P
CCAFS SLC-40
66
0.58 KM
0.90 KM
Leaflet | Data by © OpenStreetMap, under ODbL.

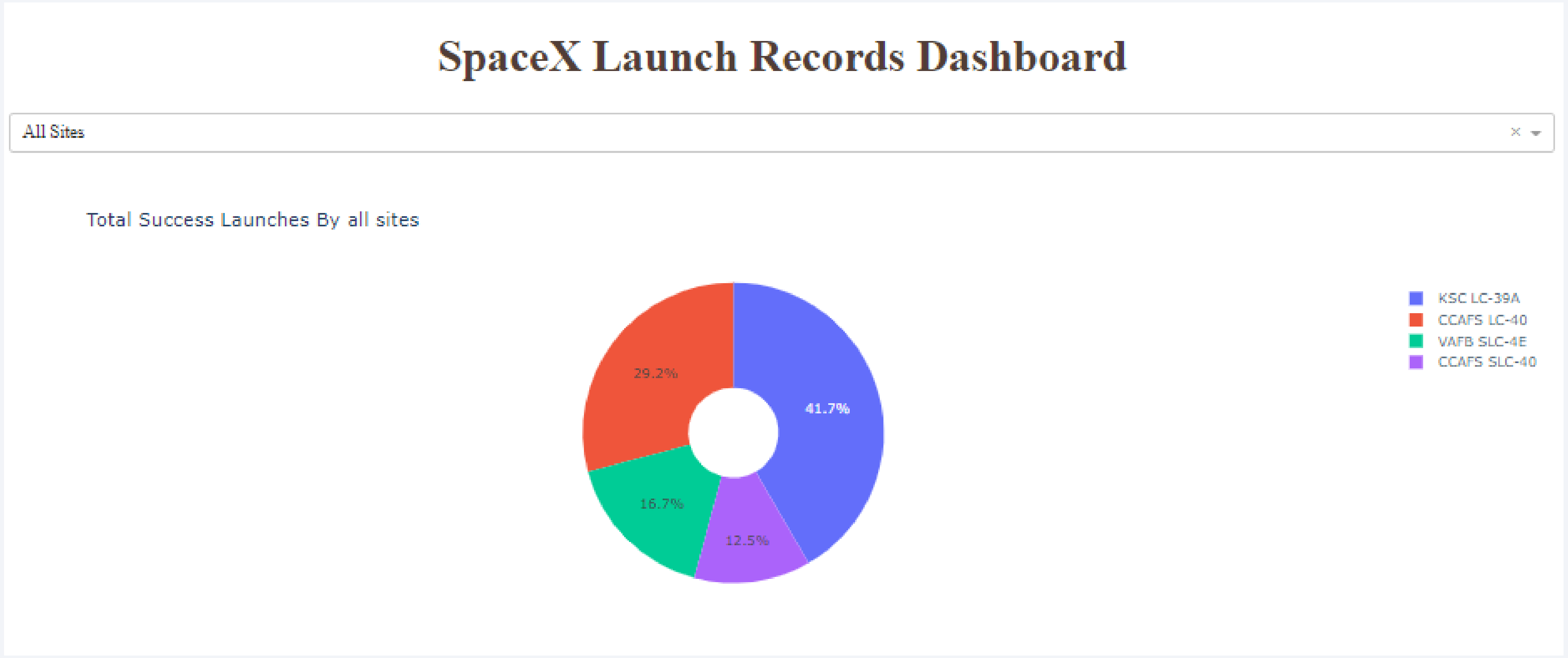
- We used Folium maps PolyLine function to mark and draw lines the highways, railways, cities and coastlines.
- Used a user defined function to calculate the distance between two coordinates.
- Used MousePosition function to display the longitudes and latitudes of a mouse point position.
- Observed that Launch Sites are in close proximity to coastlines and highways and railways, but Cities are a bit far.



Section 4

Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site

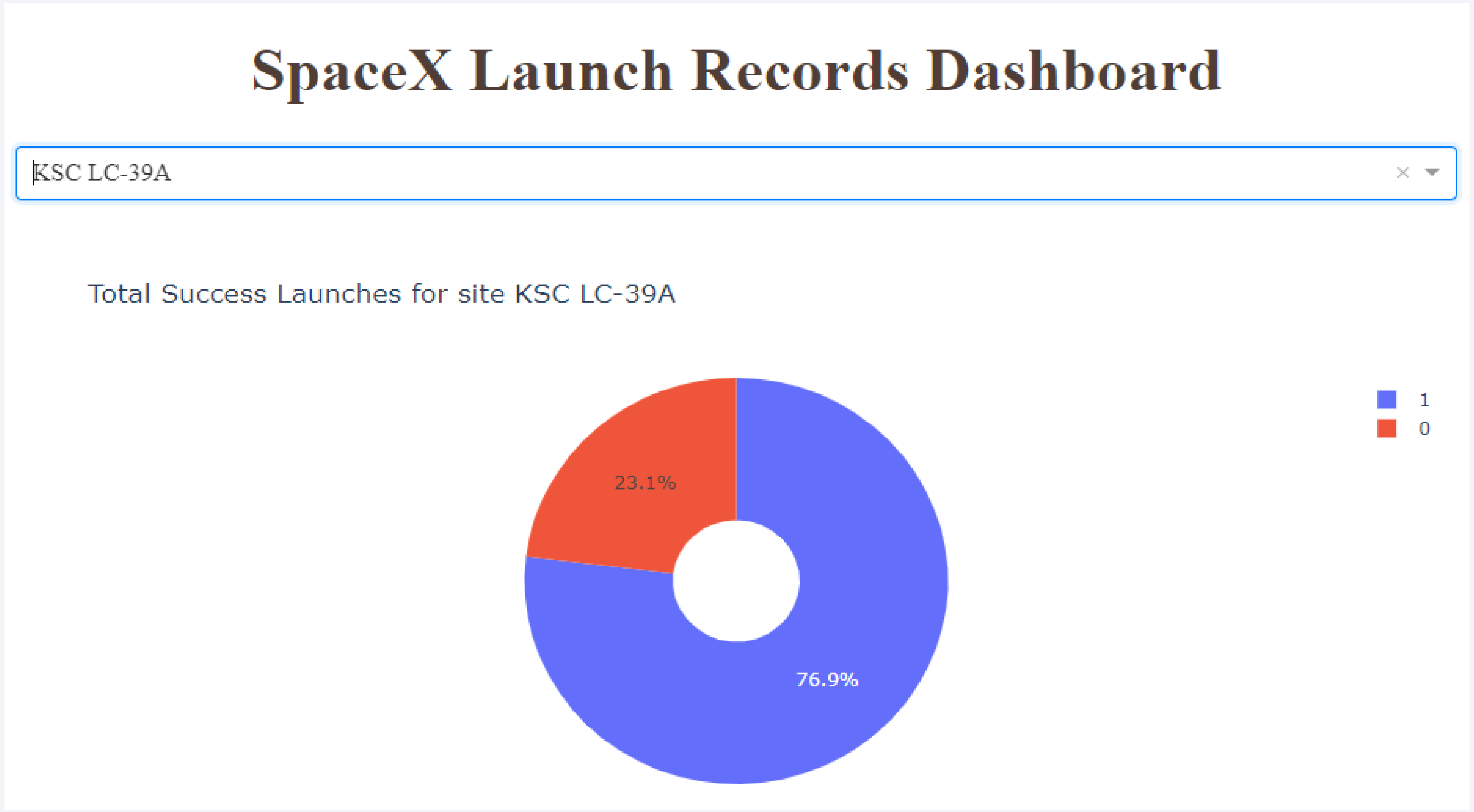

SpaceX Launch Records Dashboard

All Sites

Total Success Launches By all sites

- Pie Chart with All Launch sites in the Drop down as options.
- We observe that KSC LC-39A has the highest success rate.

Pie chart showing the Launch site with the highest launch success ratio

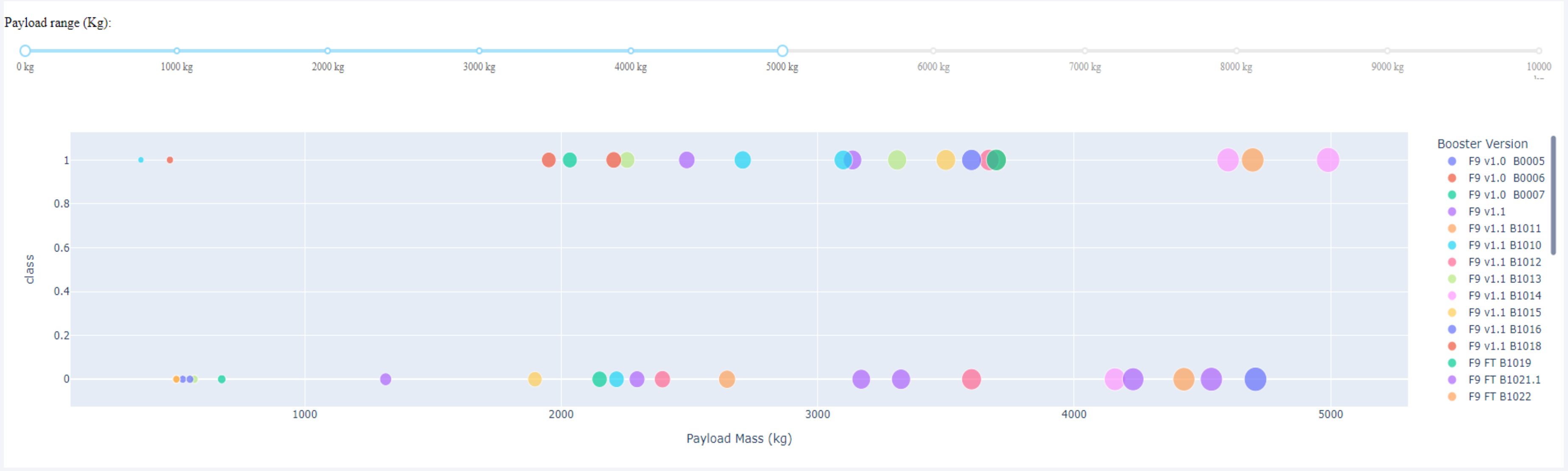

SpaceX Launch Records Dashboard

KSC LC-39A

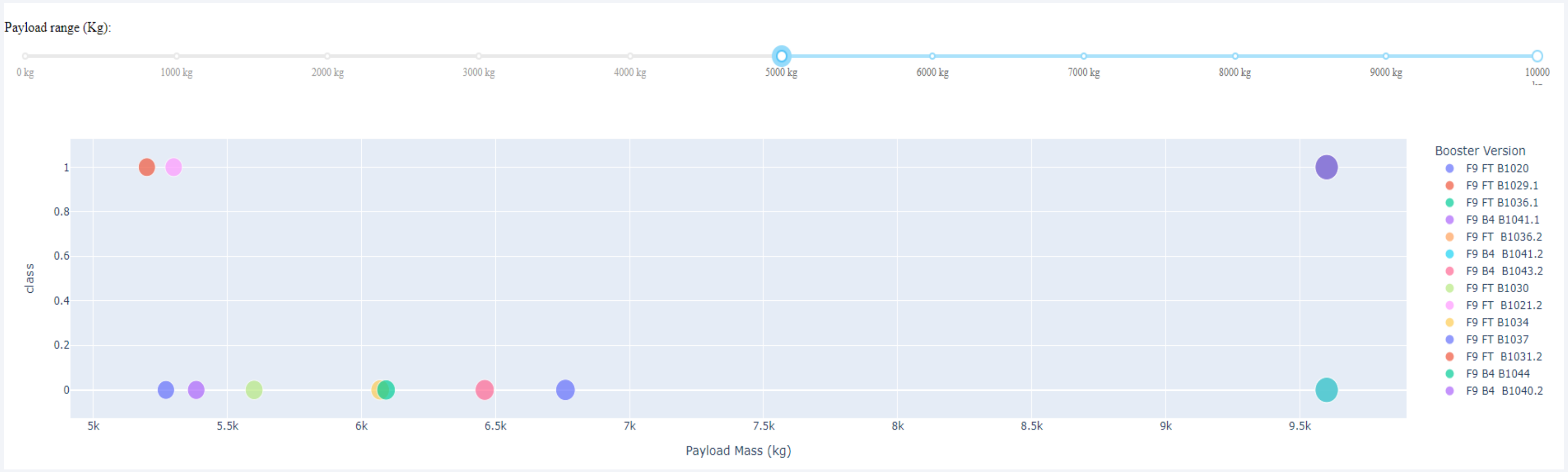
Total Success Launches for site KSC LC-39A

- Pie Chart with KSC LC-39A Launch sites in the Drop down as options.
- We observe that KSC LC-39A has 76.9% successful launches and 23.1% failed launches.

Payload vs Success rate with slide bar



We observe that the success rate for low weighted payloads is higher.

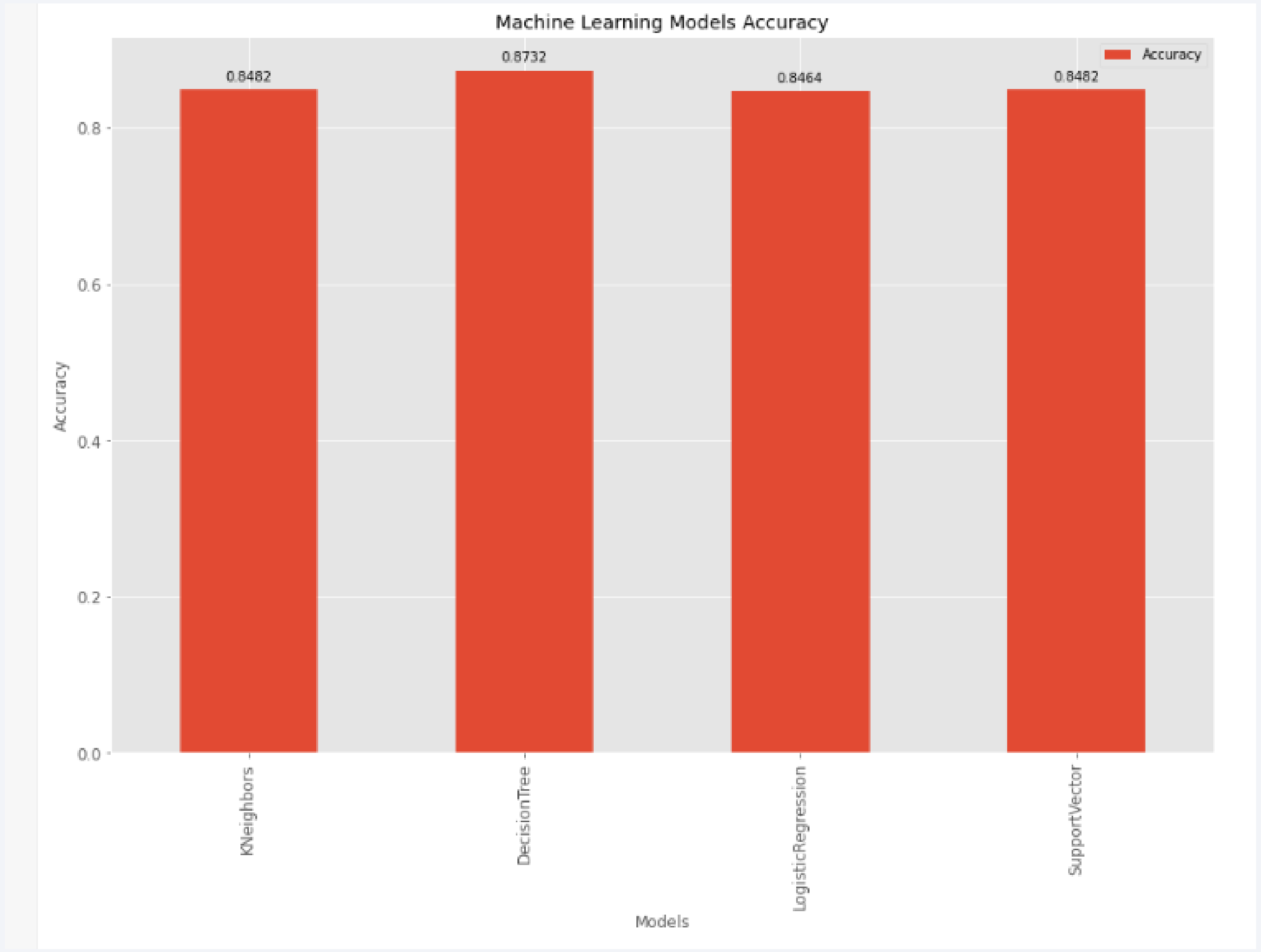


We observe that the success rate for heavy weighted payloads is less.

Section 5

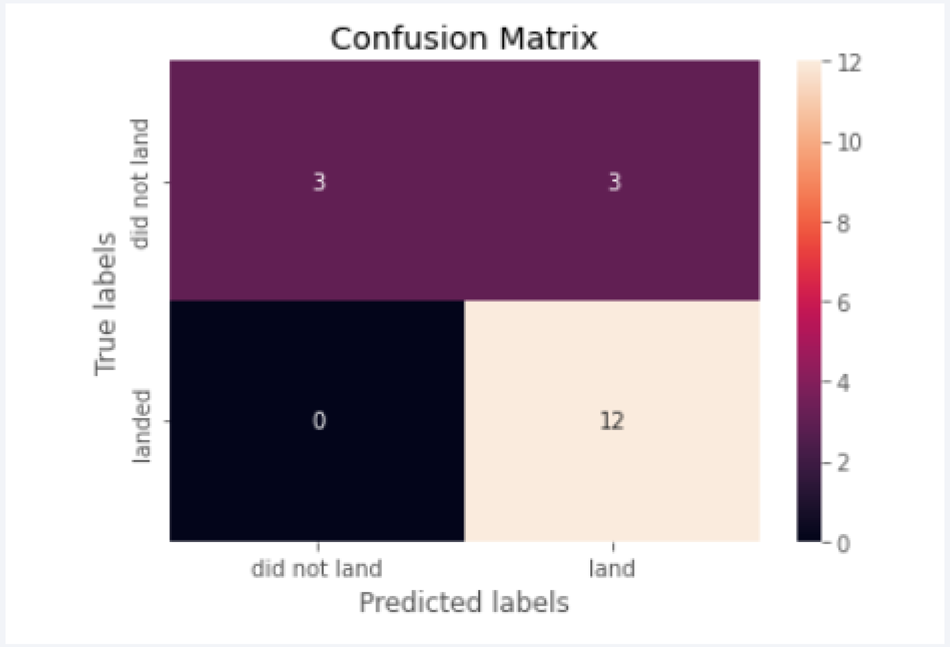
Predictive Analysis (Classification)

Classification Accuracy



We observed that Decision Tree Model has the highest Accuracy score “0.8732”.

Confusion Matrix



The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. A very low percent of false positives .i.e., unsuccessful landing marked as successful landing by the classifier.

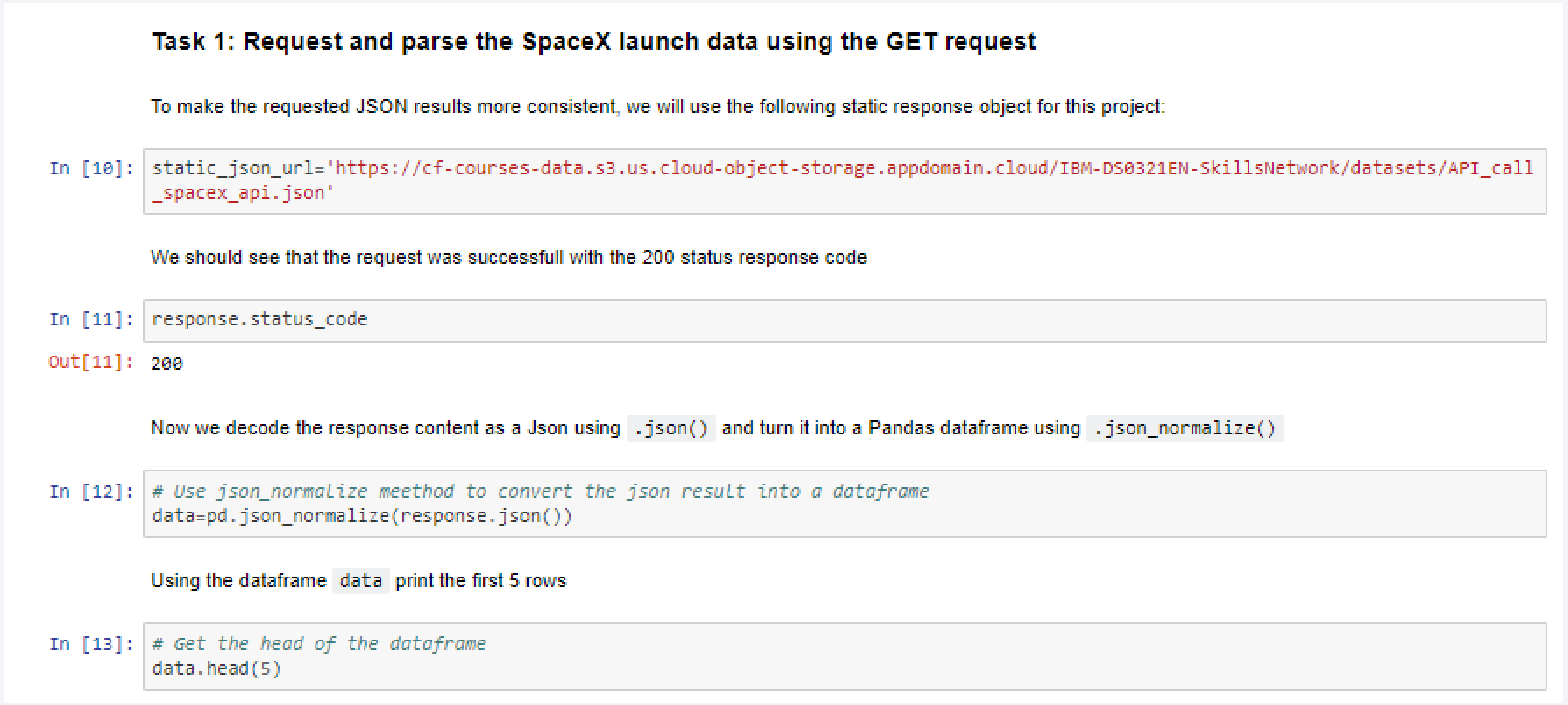
Conclusions

Based on the Analysis we conclude that:

- The first stage can be landing successfully for reuse under certain circumstances.
- The Launch Site, Payload mass, Orbit and number of flights are the main features that determine the successful landing of the first stage for reuse.
- KSC LC-39A is more suitable for landing first stage successfully for reuse.
- Orbits ES-L1, GEO, HEO, SSO, VLEO are the best suitable for successful landing.
- Payload Mass is between 2000 kg and 6000 kg there is a high chance of successful landing
- The Launch Sites should be on a coastal line, proximity to highways and railways but far from cities and residential areas.
- Decision Tree Classifier machine learning Model is best suitable for predicting the successful landing with ~87% accuracy score.
    - Using the parameters: {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}

Appendix-A

Data Collection: Using SpaceX API get Request Json– Code Snippet



Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
In [10]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call
         _spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

```
In [11]: response.status_code

Out[11]: 200
```

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

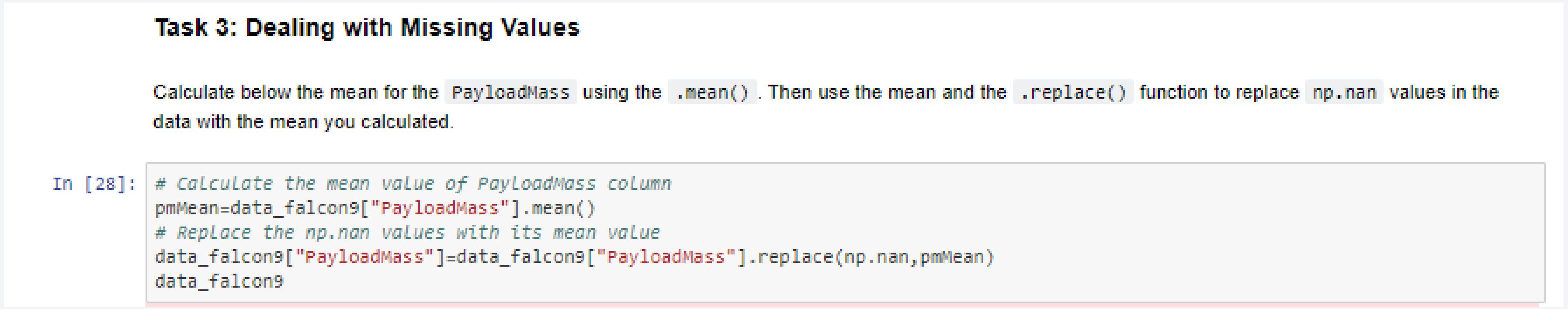
```
In [12]: # Use json_normalize meethod to convert the json result into a dataframe
         data=pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
In [13]: # Get the head of the dataframe
         data.head(5)
```

Appendix-B

Data Wrangling : Dealing with Missing values – Code Snippet

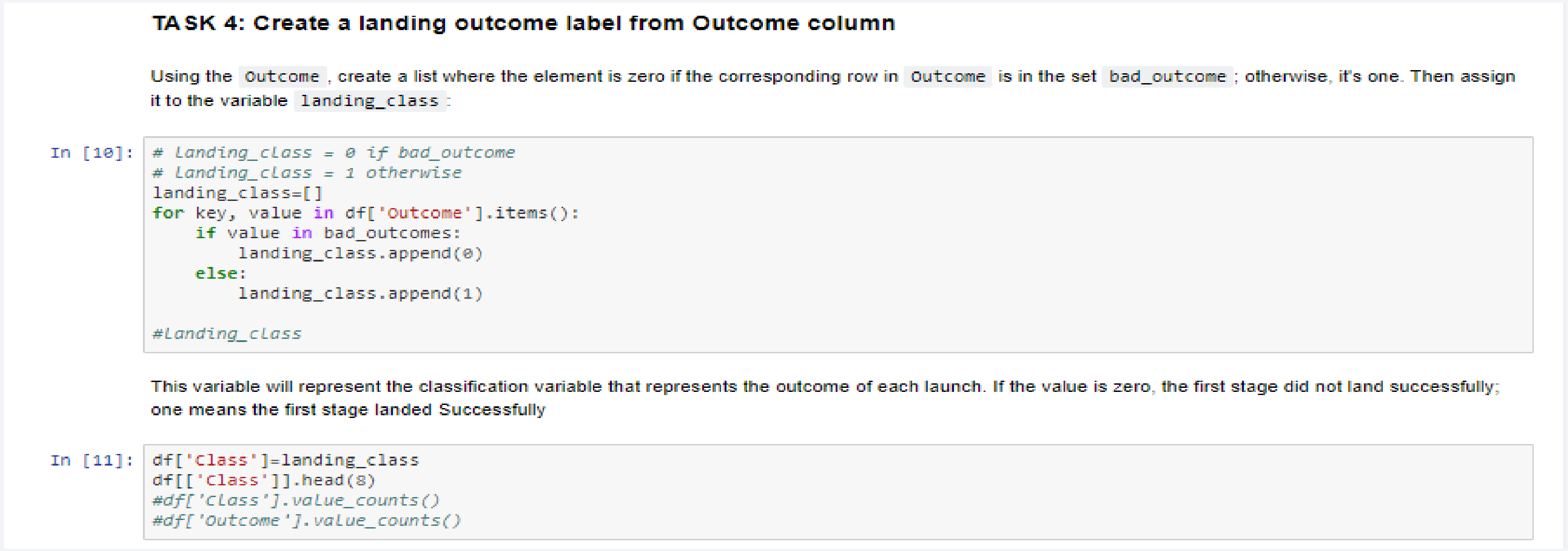


Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [28]:
```
# Calculate the mean value of PayloadMass column
pmMean=data_falcon9["PayloadMass"].mean()
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"]=data_falcon9["PayloadMass"].replace(np.nan,pmMean)
data_falcon9
```

Data Wrangling : Creating landing Outcome Column –Code Snippet



TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

In [10]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class=[]
for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)

#landing_class
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [11]:
```
df['Class']=landing_class
df[['Class']].head(8)
#df['Class'].value_counts()
#df['Outcome'].value_counts()
```

Appendix-C

Web Scrapping Table Extraction - Code snippet:

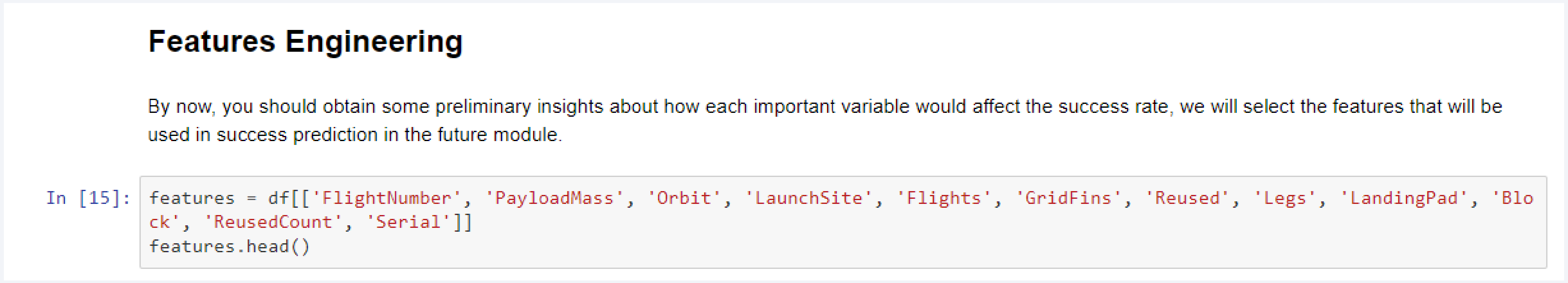
```
extracted_row = 0 #Extract each table for table_number,table in
(soup.find_all('table',"wikitable plainrowheaders collapsible")): # get table row for
rows in table.find_all("tr"): #check to see if first table heading is as number
corresponding to launch a number if rows.th: if rows.th.string:
flight_number=rows.th.string.strip() 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.`

(flight_number) launch_dict['Flight No.'].append(flight_number)
datatimelist=date_time(row[0]) # Date value # TODO: Append the date into
launch_dict with key `Date` date = datatimelist[0].strip(',')

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

(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

(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

(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])

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

(orbit) launch_dict['Orbit'].append(orbit)

# Customer # TODO: Append the customer into launch_dict with key `Customer` if (row[6].a): customer =
row[6].a.string else: customer = row[6].string

(customer) launch_dict['Customer'].append(customer)
```

```
# Launch outcome # TODO: Append the launch_outcome into launch_dict with key
`Launch outcome` launch_outcome =

(row[7].strings)[0]

(launch_outcome) launch_dict['Launch outcome'].append(launch_outcome) # Booster
landing # TODO: Append the launch_outcome into launch_dict with key `Booster
landing` booster_landing = landing_status(row[8])

(booster_landing) launch_dict['Booster landing'].append(booster_landing)
```

Appendix-D

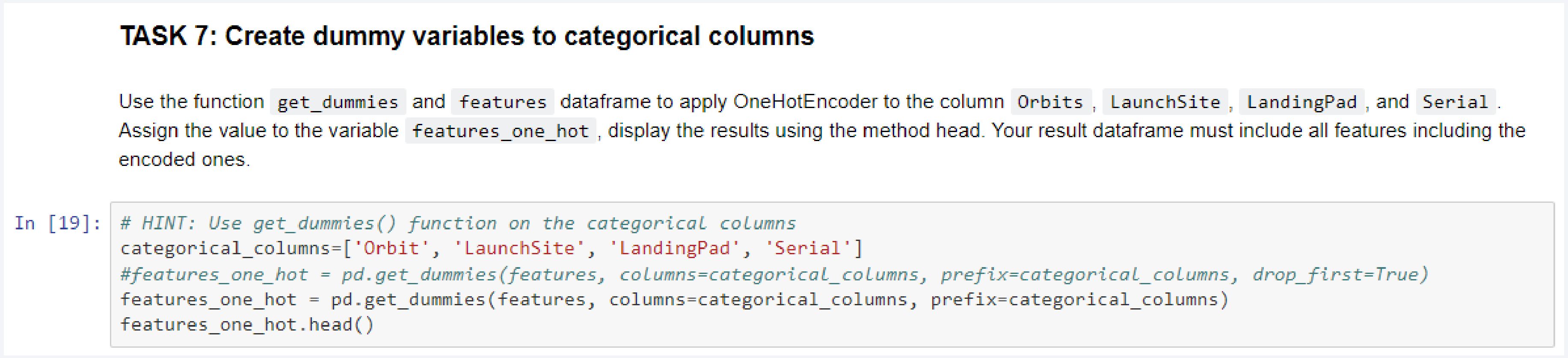
Feature Engineering - Code snippet:



Features Engineering

By now, you should obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

```
In [15]: features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Blo
         ck', 'ReusedCount', 'Serial']]
         features.head()
```



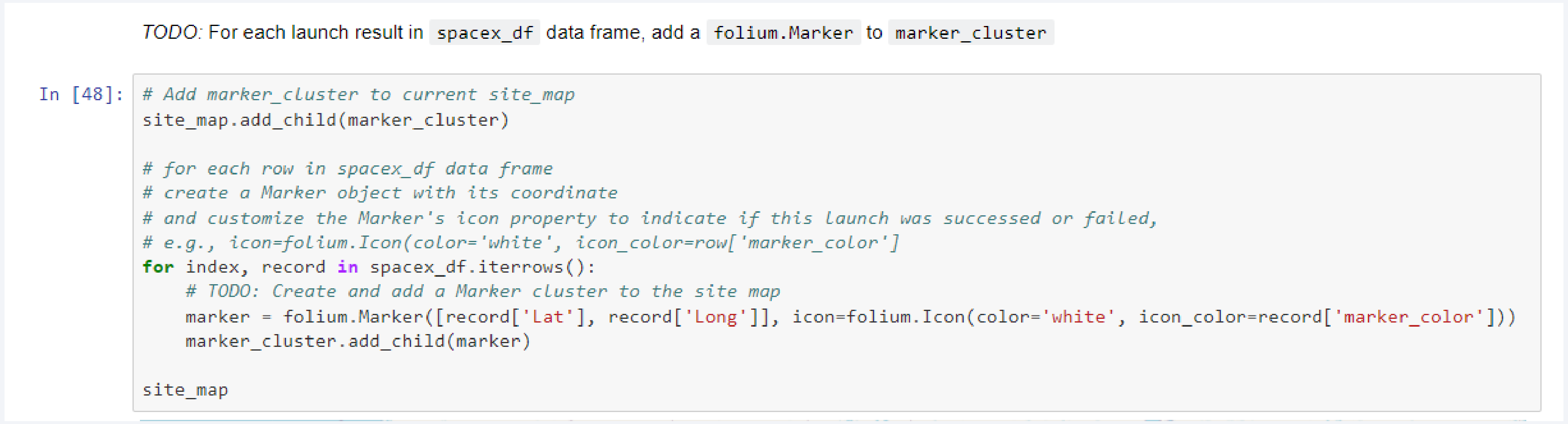
TASK 7: Create dummy variables to categorical columns

Use the function `get_dummies` and `features` dataframe to apply OneHotEncoder to the column `Orbits`, `LaunchSite`, `LandingPad`, and `Serial`. Assign the value to the variable `features_one_hot`, display the results using the method head. Your result dataframe must include all features including the encoded ones.

```
In [19]: # HINT: Use get_dummies() function on the categorical columns
         categorical_columns=['Orbit', 'LaunchSite', 'LandingPad', 'Serial']
         #features_one_hot = pd.get_dummies(features, columns=categorical_columns, prefix=categorical_columns, drop_first=True)
         features_one_hot = pd.get_dummies(features, columns=categorical_columns, prefix=categorical_columns)
         features_one_hot.head()
```

Appendix-E

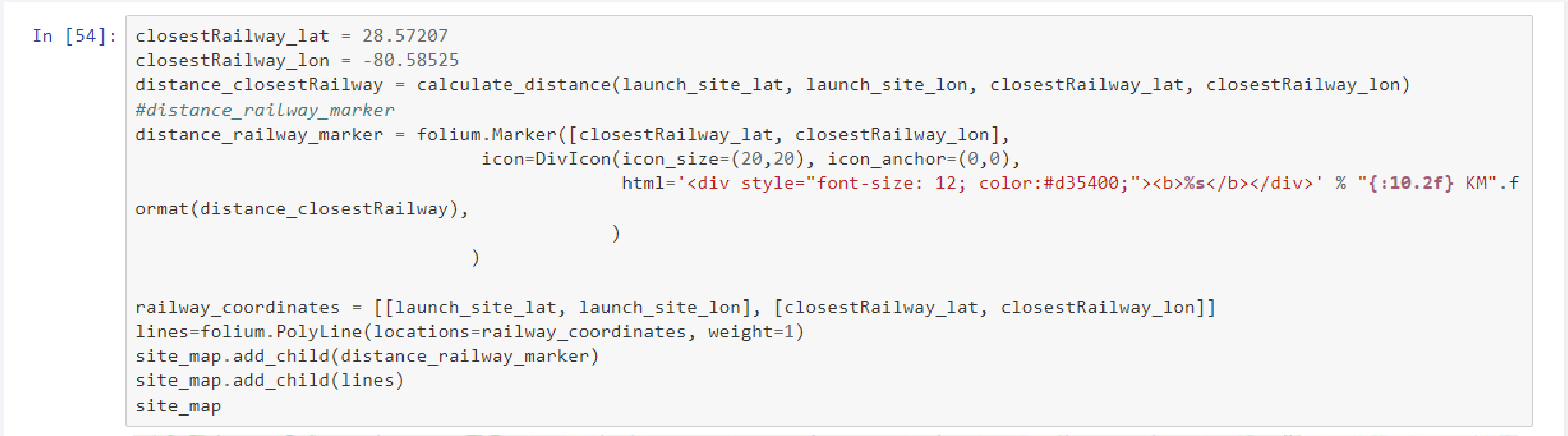
Folium Cluster and Success or Failure Marker - Code snippet:



TODO: For each launch result in `spacex_df` data frame, add a `folium.Marker` to `marker_cluster`

```
In [48]: # Add marker_cluster to current site_map
         site_map.add_child(marker_cluster)

         # for each row in spacex_df data frame
         # create a Marker object with its coordinate
         # and customize the Marker's icon property to indicate if this launch was successed or failed,
         # e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
         for index, record in spacex_df.iterrows():
             # TODO: Create and add a Marker cluster to the site map
             marker = folium.Marker([record['Lat'], record['Long']], icon=folium.Icon(color='white', icon_color=record['marker_color']))
             marker_cluster.add_child(marker)

         site_map
```

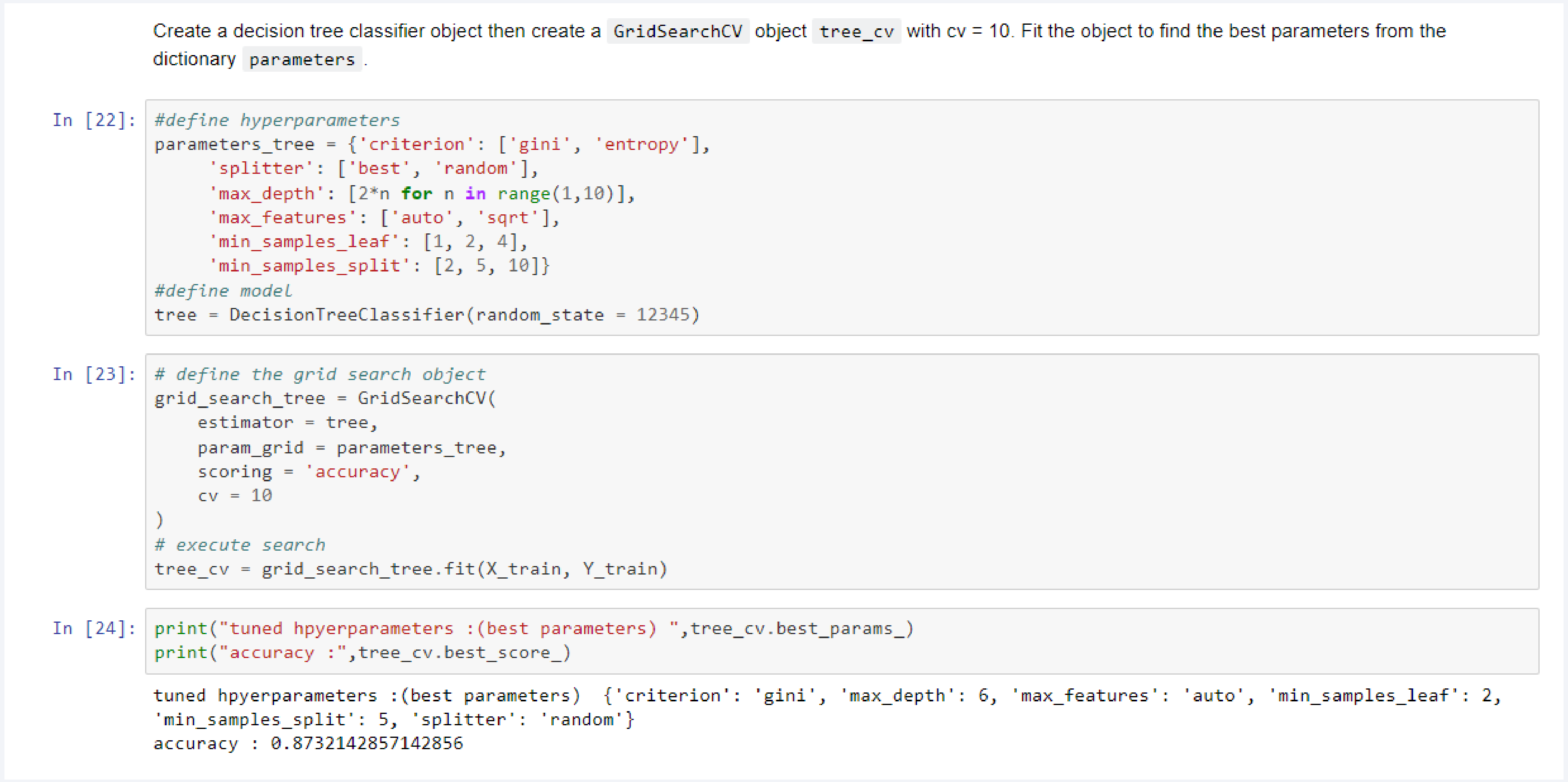
Folium Polylines drawing for proximity and distance calculation Code Snippet:



```
In [54]: closestRailway_lat = 28.57207
         closestRailway_lon = -80.58525
         distance_closestRailway = calculate_distance(launch_site_lat, launch_site_lon, closestRailway_lat, closestRailway_lon)
         #distance_railway_marker
         distance_railway_marker = folium.Marker([closestRailway_lat, closestRailway_lon],
                                     icon=DivIcon(icon_size=(20,20), icon_anchor=(0,0),
                                                 html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".f
         ormat(distance_closestRailway),
                                                 )
                                     )

         railway_coordinates = [[launch_site_lat, launch_site_lon], [closestRailway_lat, closestRailway_lon]]
         lines=folium.PolyLine(locations=railway_coordinates, weight=1)
         site_map.add_child(distance_railway_marker)
         site_map.add_child(lines)
         site_map
```

Appendix-F

Machine Learning model Code snippet:



Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
In [22]: #define hyperparameters
         parameters_tree = {'criterion': ['gini', 'entropy'],
              'splitter': ['best', 'random'],
              'max_depth': [2*n for n in range(1,10)],
              'max_features': ['auto', 'sqrt'],
              'min_samples_leaf': [1, 2, 4],
              'min_samples_split': [2, 5, 10]}
         #define model
         tree = DecisionTreeClassifier(random_state = 12345)
```

```
In [23]: # define the grid search object
         grid_search_tree = GridSearchCV(
             estimator = tree,
             param_grid = parameters_tree,
             scoring = 'accuracy',
             cv = 10
         )
         # execute search
         tree_cv = grid_search_tree.fit(X_train, Y_train)
```

```
In [24]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
         print("accuracy :",tree_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2,
         'min_samples_split': 5, 'splitter': 'random'}
         accuracy : 0.8732142857142856
```

Thank you!