IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Cristian Pop
6th January 2023

Outline

Executive Summary

- Summary of methodologies
  - **Data Collection** – SpaceX API and Webscraping Wikipedia page
  - **Data wrangling –** Pandas and Numpy to remove missing values from the dataset and other data manipulations
  - **Exploratory Data Analysis -** Perform exploratory data analysis (EDA) using visualization and SQL
  - **Interactive visual analytics (dashboard and maps) -** using Folium and Plotly Dash
  - **Predictive analysis** – using classification models
- Summary of all results
  - The best model was the Decision Tree with a classification accuracy score of 0.875 on the training data
  - We believe our prediction model will have ~87.5% accuracy on predicting the recovery of stage 1 given a series of parameters
  - The success ratio of successful stage 1 landings should improve over time.
  - Launches in orbits ES-L1, GEO, HEO and SSO had a success rate of 100%
  - Launch sites are located close to coastlines and highways, but away from cities
  - 41.7% of the total successful launches were from the KSC LC-39A launch site
  - Most of the successful launches are concentrated in the 2000-4000kg Payload mass range

Introduction

- Project background and context
    - Space X 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 Space X 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 space X for a rocket launch.
- Problems you want to find answers
    - What is the probability of stage 1 to land for SpaceX?
    - Did SpaceX improve their rate of stage 1 retrieval over time?

Section 1

Methodology

Methodology

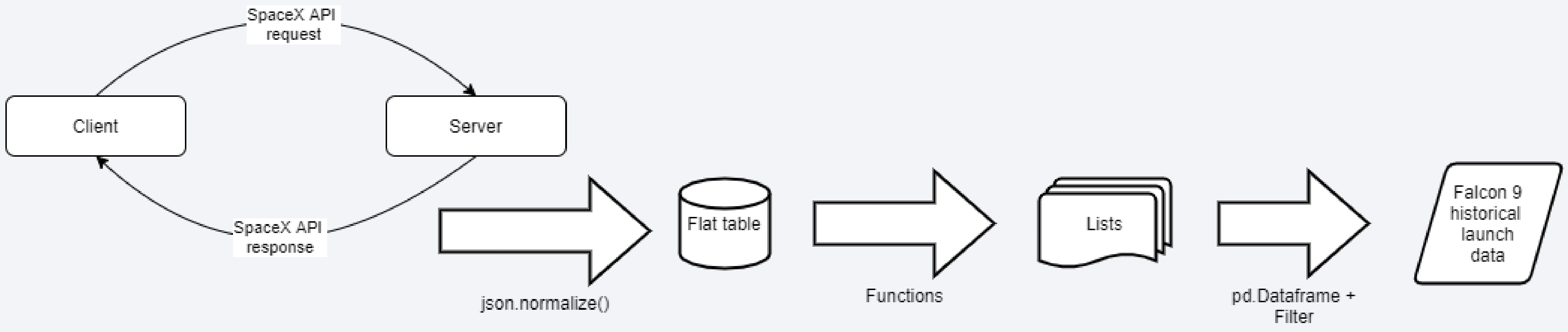
Executive Summary

- Data collection methodology:
    - Historical rocket launch data using SpaceX API and Wikipedia web scraping
- Perform data wrangling
    - Missing data regarding the payload mass were replaced using mean values using pandas
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - Tuned the parameters using cross-validation technique
    - Models used: Linear Regression, Decision Tree, SMV, K-Nearest Neighbour

Data Collection

- Historical rocket launch data using SpaceX API
  - Method: SpaceX API
  - Called the API and normalized the json into a flat table
  - Defined functions to get information for the IDs in the response from different API endpoints
  - Stored the obtained info into lists and then created a pandas dataframe
  - Filtered the dataframe to only include Falcon 9 launches
- Web scraped launch records HTML table from Wikipedia
  - Extract a Falcon 9 launch records HTML table from Wikipedia
  - Parse the table and convert it into a Pandas data frame

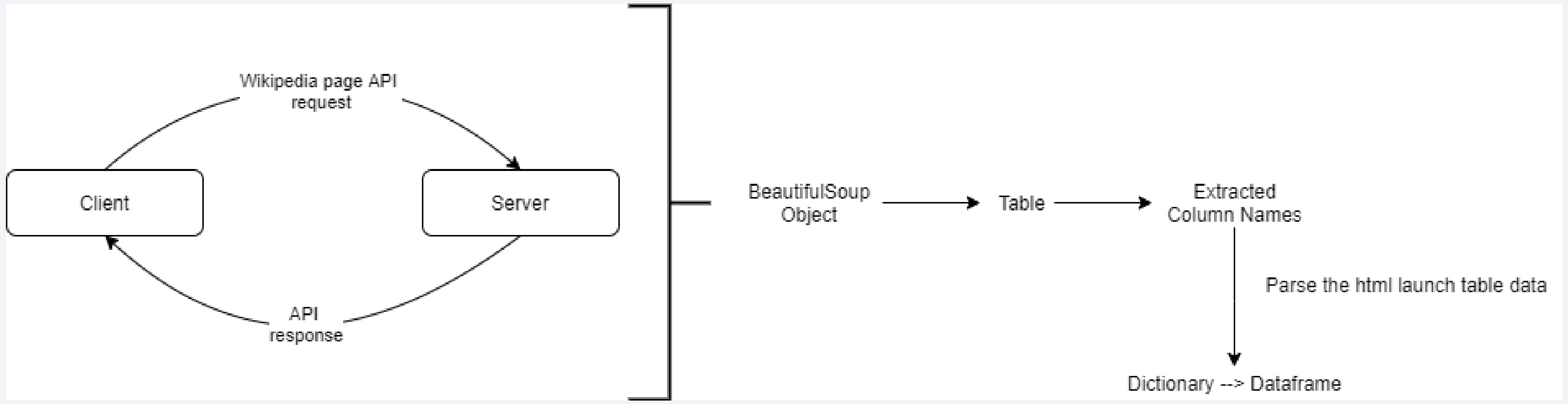
Data Collection – SpaceX API Flowchart



Github reference: https://github.com/cristianrpop/ibm-data_science/blob/main/applied-data-science-capstone/Data%20Collection.ipynb

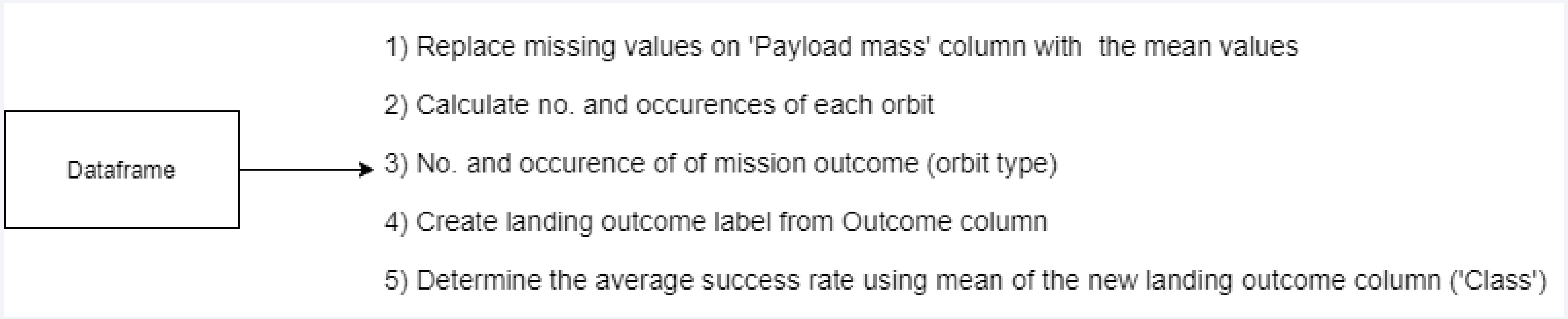
Data Collection - Scraping

- Webscraping library used: BeautifulSoup
- Request the Falcon9 Launch Wiki page from its URL
- Extract all column/variable names from the HTML table header
- Create a data frame by parsing the launch HTML tables
- Github URL: https://github.com/cristianrpop/ibm-data_science/blob/ea37f6426516910c7b12663c63a55cd8a34c8c71/applied-data-science-capstone/Data%20Collection%20-%20Web%20Scraping.ipynb



Data Wrangling

- Missing values on the PayloadMass column were replaced using the mean of the values
- Github URL: https://github.com/cristianrpop/ibm-data_science/blob/ea37f64265169l0c7b12663c63a55cd8a34c8c71/applied-data-science-capstone/Data%20Wrangling.ipynb



1) Replace missing values on 'Payload mass' column with the mean values
2) Calculate no. and occurences of each orbit
3) No. and occurence of of mission outcome (orbit type)
4) Create landing outcome label from Outcome column
5) Determine the average success rate using mean of the new landing outcome column ('Class')

EDA with Data Visualization

Library used: matplotlib (plotting), seaborn (visualization)

1) Scatter plot: Flight Number (x) vs Pay load Mass (y) with the “Class” column as hue -> success rate increased with flight number

2) Scatter plot: Flight Number (x) vs Launch Site (y) with the “Class” column as hue -> Visualize the relationship between Flight Number and Launch Site
   - There were no rockets launched from the ‘CCAFS SLC-40’ site between flights 25 and 40, then the success rate greatly improved at this site

3) Scatter plot: Pay load Mass(x) vs Launch Site (y) with the “Class” column as hue -> for the VAFB-SLC launchsite there were no rockets launched for heavypayload mass (greater than 10000).

4) Bar plot: Class (x) vs ‘Avg. of Class for each Orbit’ column (y)-> Orbits ES-L1, GEO, HEO and SSO have the highest avg. mission success rate (100%) while SO has the lowest with 0%

EDA with Data Visualization

5) Scatter plot: Flight Number (x) vs Orbit (y) with ‘Class’ column as hue-> In the LEO orbit, success rate seems to be connected to flight number while on the GTO orbit, it is not so obvious

6) Scatter plot: Pay load mass (x) vs Orbit (y) with ‘Class’ column as hue-> With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS

7) Line plot: year (x) vs Class (y) > Success rate has kept increasing from 2013 to 2020

Github URL: https://github.com/cristianrpop/ibm-data_science/blob/ea37f6426516910c7b12663c63a55cd8a34c8c71/applied-data-science-capstone/EDA%20with%20Visualization.ipynb

EDA with SQL

- `select distinct launch_site from spacex` - Display the names of the unique launch sites in the space mission
- `select * FROM spacex WHERE launch_site LIKE 'CCA%' LIMIT 5` - Display 5 records where launch sites begin with the string 'CCA'
- `SELECT SUM(payload_mass__kg_) AS "Total_Payload_NASA" FROM spacex WHERE spacex.customer LIKE '%NASA%'` - Display the total payload mass carried by boosters launched by NASA (CRS)
- `SELECT AVG(payload_mass__kg_) AS "average_payload_F9v1.1" FROM spacex WHERE booster_version LIKE '%F9 v1.1%'` - Display average payload mass carried by booster version F9 v1.1
- `SELECT MIN(DATE) AS "First Successful Landing On Ground Pad" FROM spacex WHERE landing__outcome = 'Success (ground pad)'` - List the date when the first successful landing outcome in ground pad was achieved.

EDA with SQL

- `SELECT booster_version, payload_mass__kg_ FROM spacex WHERE landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 AND 6000` - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- `SELECT COUNT(*) AS "Total Successful Missions", (SELECT COUNT(*) FROM spacex WHERE mission_outcome LIKE '%Failure%') AS "Total Failed Missions" FROM spacex WHERE mission_outcome LIKE '%Success%'` - List the total number of successful and failure mission outcomes
- `SELECT booster_version FROM spacex WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM spacex)` - List the names of the booster_versions which have carried the maximum payload mass.
- `SELECT booster_version, launch_site, landing__outcome FROM spacex WHERE landing__outcome = 'Failure (drone ship)' AND YEAR(DATE) = 2015` - List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

EDA with SQL

- `SELECT landing__outcome, COUNT(*) AS "Landing Outcomes" FROM spacex WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing__outcome ORDER BY "Landing Outcomes" DESC` - 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

- Github URL: https://github.com/cristianrpop/ibm-data_science/blob/094b52aa9906816b0896d3d82a4202ab1fbeb05d/applied-data-science-capstone/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Circle – highlight NASA Johnson Space Center location
  - Popup – show the location name when clicking on the circle
  - Marker – icon showing the name of the center
- Four Circles and Markers – one for each launch site on the map
- Market_Cluster with Markers for each launch record – result: easily identify which launch sites have relatively high success rates
  - Green marker for successful launch outcomes
  - Red marker for failed launch outcomes
- MousePosition() – added to the map to see the coordinates of any point hovered on the map
- Marker – added to the map to see the value of the distance in km plotted on the map below the coastline
- PolyLine – between VAFB SLC-4E launch site and the closest point on the coastline to visually see the distance
- The same approach as above was used again to illustrate the distance between VAFB SLC-4E and the closest city (Lompoc)
- Github URL: https://github.com/cristianrpop/ibm-data_science/blob/57cd4a6018f662d708e54e2e92502a2fba8fc6ad/applied-data-science-capstone/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- App Layout
    - dcc.Dropdown object – showing the user input options (“ALL Sites” and the names of each launch site)
    - dcc.Graph object – added a pie chart to show the total successful launches count for all sites
    - dcc.RangeSlider object – added a slider to select payload range
    - dcc.Graph object – added a scatter chart to show the correlation between payload and launch success
- Callback function #1
    - Set the ‘site-dropdown’ as input and the ‘success-pie-chart’ as Output in the function decorator
    - get_pie_chart() function – creates the pie chart depending on the input selected by user
- Callback function #2
    - Set the ‘site-dropdown’ and ‘payload-slider’ as inputs and the ‘success-payload-scatter-chart’ as Output in the function decorator
    - get_scatter_chart() function – creates the scatter chart depending on the inputs selected by user in the dropdown and slider objects created

Build a Dashboard with Plotly Dash

- Plots utility
  - Determine which site has the largest successful launches and the highest launch success rate
  - Analyse the payload range(s) launch success rates
  - Extract the Falcon9 rocket model (booster version) with the highest launch success rate
  - Other data analysis

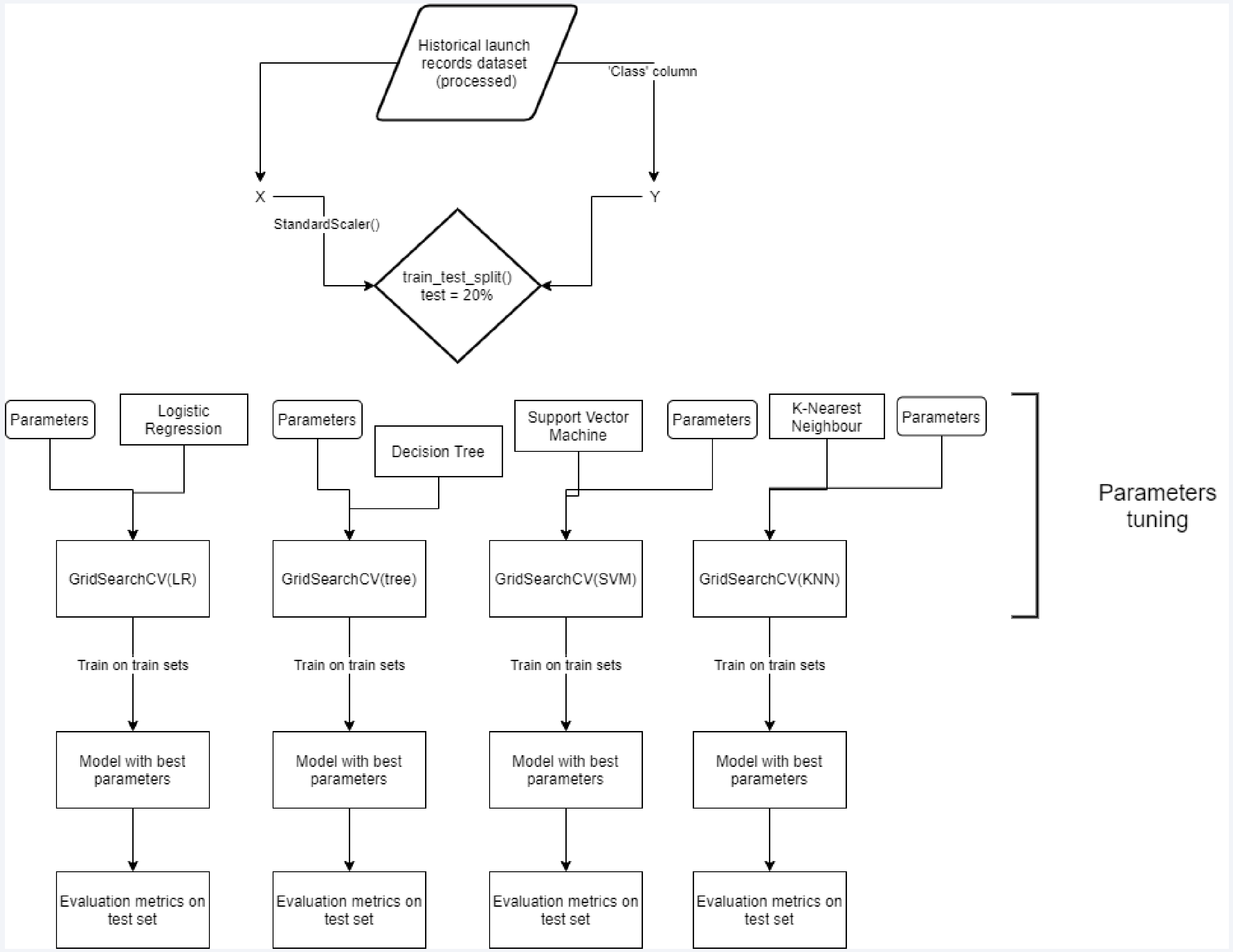
- Github URL: https://github.com/cristianrpop/ibm-data_science/blob/4bbc0ca90091915c3b37f38fa774e9c05ac7bb2f/applied-data-science-capstone/spacex_dash_app%20(1).py

Predictive Analysis (Classification)

- Assigned target “Class” column to variable Y after converting it in a numpy array.
- Standardized the data in variable X using the StandardScaler() object.
- Split the data (X and Y) into training and test data using train_test_split() function
- Found best hyperparameters for SVM, Classification Trees and Logistic Regression using the GridSearchCV cross-validation technique.
- Used the accuracy score on training and test data to compare the models’ performance.

- Github URL: https://github.com/cristianrpop/ibm-data_science/blob/cd09ccfe0ef5a566fda11711464f301c7b95e354/applied-data-science-capstone/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Predictive Analysis (Classification) - Flowchart



Results

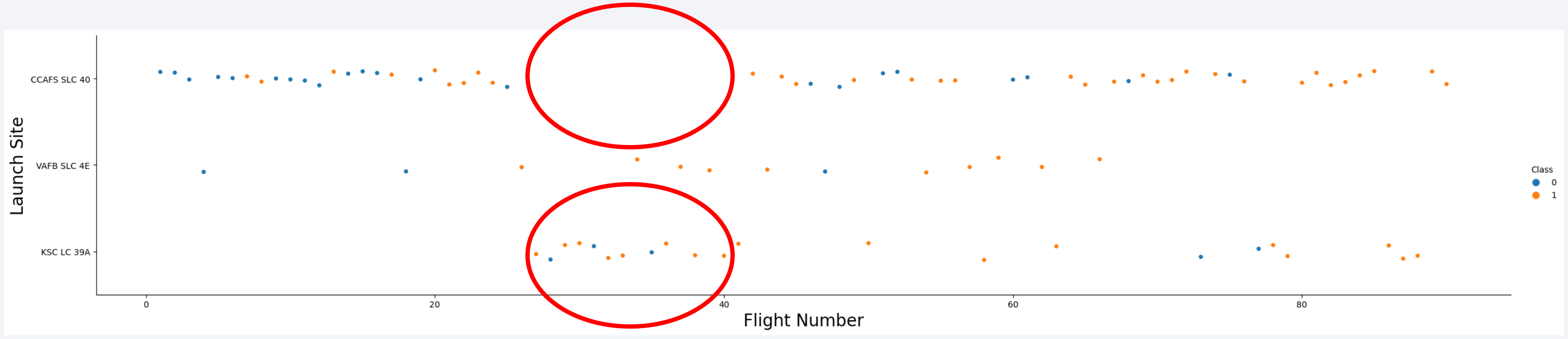
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

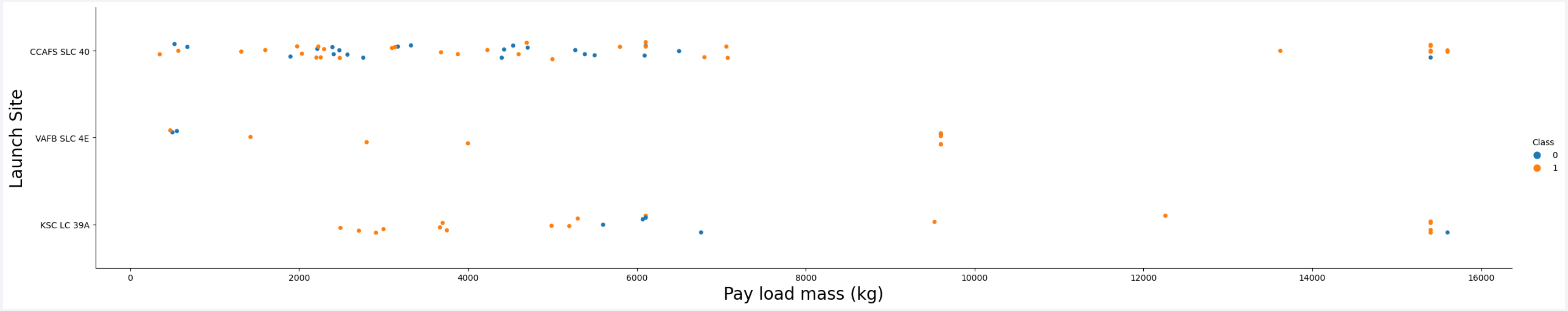
Flight Number vs. Launch Site

- Scatter plot: Flight Number (x) vs Launch Site (y) with the “Class” column as hue -> Visualize the relationship between Flight Number and Launch Site
  - There were no rockets launched from the ‘CCAFS SLC-40’ site between flights 25 and 40, then the success rate greatly improved at this site



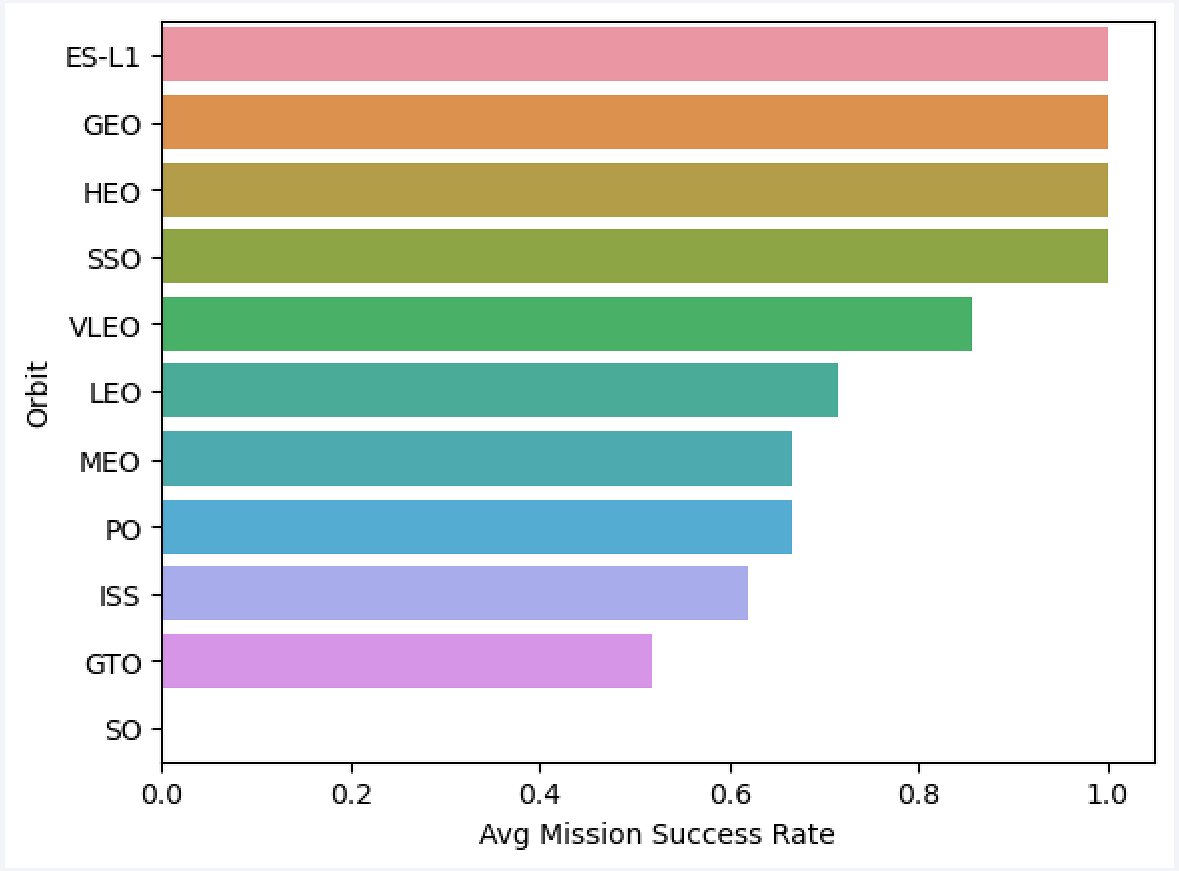
Payload vs. Launch Site

- Scatter plot: Pay load Mass(x) vs Launch Site (y) with the "Class" column as hue -> for the VAFB-SLC launch site there were no rockets launched for heavy payload mass (greater than 10000).



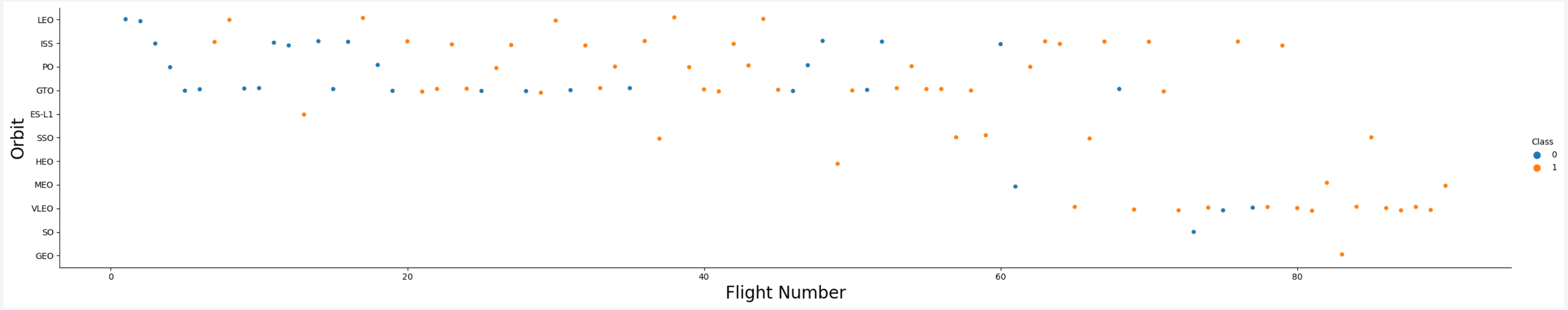
Success Rate vs. Orbit Type

- Bar plot: Class (x) vs ‘Avg. of Class for each Orbit’ column (y)
- Orbits ES-L1, GEO, HEO and SSO have the highest avg. mission success rate (100%) while SO has the lowest with 0%
- 10 out of 11 orbits have a success rate of over 50%



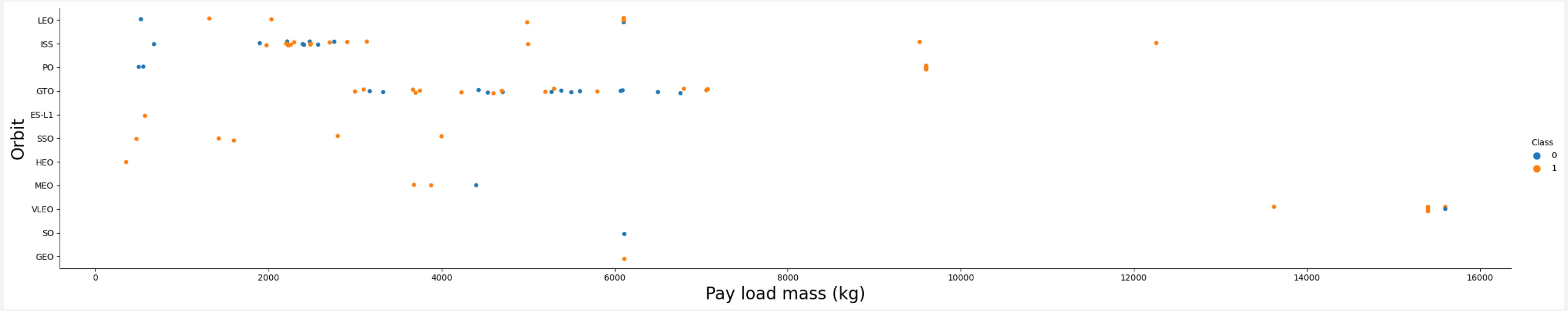
Flight Number vs. Orbit Type

- Scatter plot: Flight Number (x) vs Orbit (y) with ‘Class’ column as hue
- In the LEO orbit, success rate seems to be connected to flight number while on the GTO orbit, it seems to be no relationship between the two variables
- Notice that for the last 6 Orbits on the chart SpaceX started commencing missions only after the 50th flight. We believe this could be a result of a lack of technology or feasibility, but it requires further investigation.



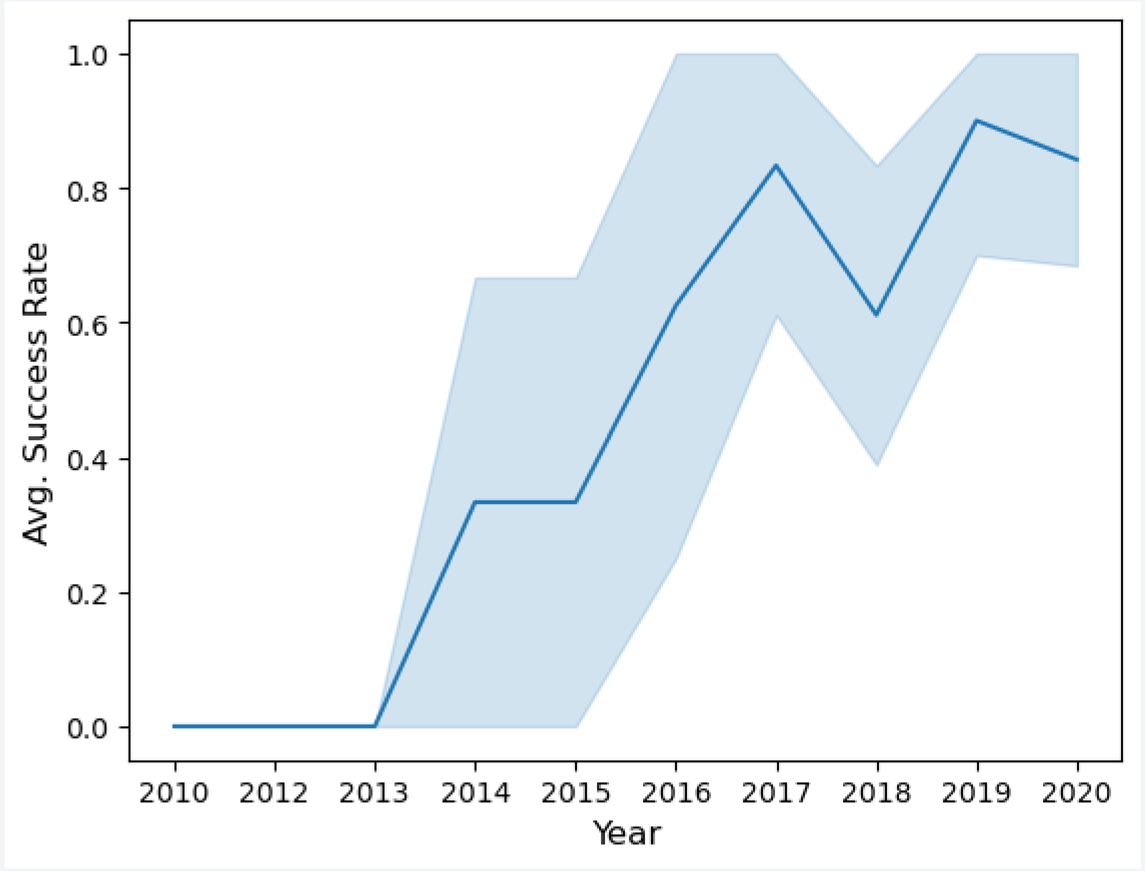
Payload vs. Orbit Type

- Scatter plot: Pay load mass (x) vs Orbit (y) with ‘Class’ column as hue
- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS



Launch Success Yearly Trend

- Line plot: year (x) vs Class (y)
- Average success rate has kept increasing from 2013 to 2020



All Launch Site Names

- `select distinct launch_site from spacex`
- SpaceX has 4 launch sites

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

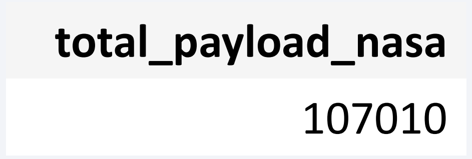
Launch Site Names Begin with 'CCA'

- `select * FROM spacex WHERE launch_site LIKE 'CCA%' LIMIT 5`
- Used the LIKE operator in the WHERE clause to filter after the ‘launch_site’ column
- Used the LIMIT operator to show only the first five 5 matching records

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

Total Payload Mass

- The total payload carried by boosters from NASA was 107010 kg
-
```
SELECT SUM(payload_mass__kg_) AS "Total_Payload_NASA" FROM spacex
WHERE spacex.customer LIKE '%NASA%’
```
- Used the SUM() function to get the total amount and the LIKE operator to filter the dataset on the ‘customer’ column



total_payload_nasa
107010

Average Payload Mass by F9 v1.1

-
  ```
  SELECT AVG(payload_mass__kg_) AS "average_payload_F9v1_1"
  FROM spacex WHERE booster_version LIKE '%F9 v1.1%’
  ```
- The average payload mass carried by rockets with booster version F9 v.1.1 has been 2534kg
- Used the AVG() function on the ‘payload_mass_kg’ column
- Used the LIKE operator to filter the ‘booster_version’ column

average_payload_f9v1_1
2534

First Successful Ground Landing Date

- The first SpaceX Falcon9 successful landing outcome on a ground pad has been on 22th December, 2015

-
  ```
  SELECT MIN(DATE) AS First_Successful_Landing_On_Ground_Pad
  FROM spacex WHERE landing__outcome = 'Success (ground pad)’
  ```
- Used the MIN() function to get the minimum date on the filtered query

first_successful_landing_on_ground_pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- `SELECT booster_version, payload_mass__kg_ FROM spacex WHERE landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 AND 6000`
- Used the BETWEEN operator to filter the successful launches by payload mass

booster_version	payload_mass__kg_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Total Number of Successful and Failure Mission Outcomes

- There was only one failed mission which might be an indicator for the high quality work SpaceX is doing
- `SELECT COUNT(*) AS Total_Successful_Missions, (SELECT COUNT(*) FROM spacex WHERE mission_outcome LIKE '%Failure%') AS Total_Failed_Missions FROM spacex WHERE mission_outcome LIKE '%Success%’`
- Used a subquery to calculate the total failed missions

total_successful_missions	total_failed_missions
100	1

Boosters Carried Maximum Payload

- Used a subquery to perform this query
- MAX() function used in the subquery to get the maximum amount of payload mass from the dataset
- `SELECT booster_version FROM spacex WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM spacex)`
- 12 out of 101 (11.88%) booster versions have carried the maximum payload mass

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 Launch Records

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

-
  ```
  SELECT booster_version, launch_site, landing__outcome FROM spacex
  WHERE landing__outcome = 'Failure (drone ship)' AND YEAR(DATE) =
  2015
  ```

- Used the YEAR() function on the DATE column in the WHERE claused in order to filter the dataset to show only launches from year 2015

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

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

- There were 10 rocket launches with no landing attempt in the specified period while the number of successful landings on a drone ship matches the failed ones.

-
  ```
  SELECT landing__outcome, COUNT(*) AS
  Landing_Outcomes FROM spacex WHERE DATE
  BETWEEN '2010-06-04' AND '2017-03-20'
  GROUP BY landing__outcome ORDER BY
  Landing_Outcomes DESC
  ```
- Used the BETWEEN operator to filter the dataset by the required interval
- Used the GROUP BY clause to aggregate the data for each landing outcome

landing__outcome	landing_outcomes
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

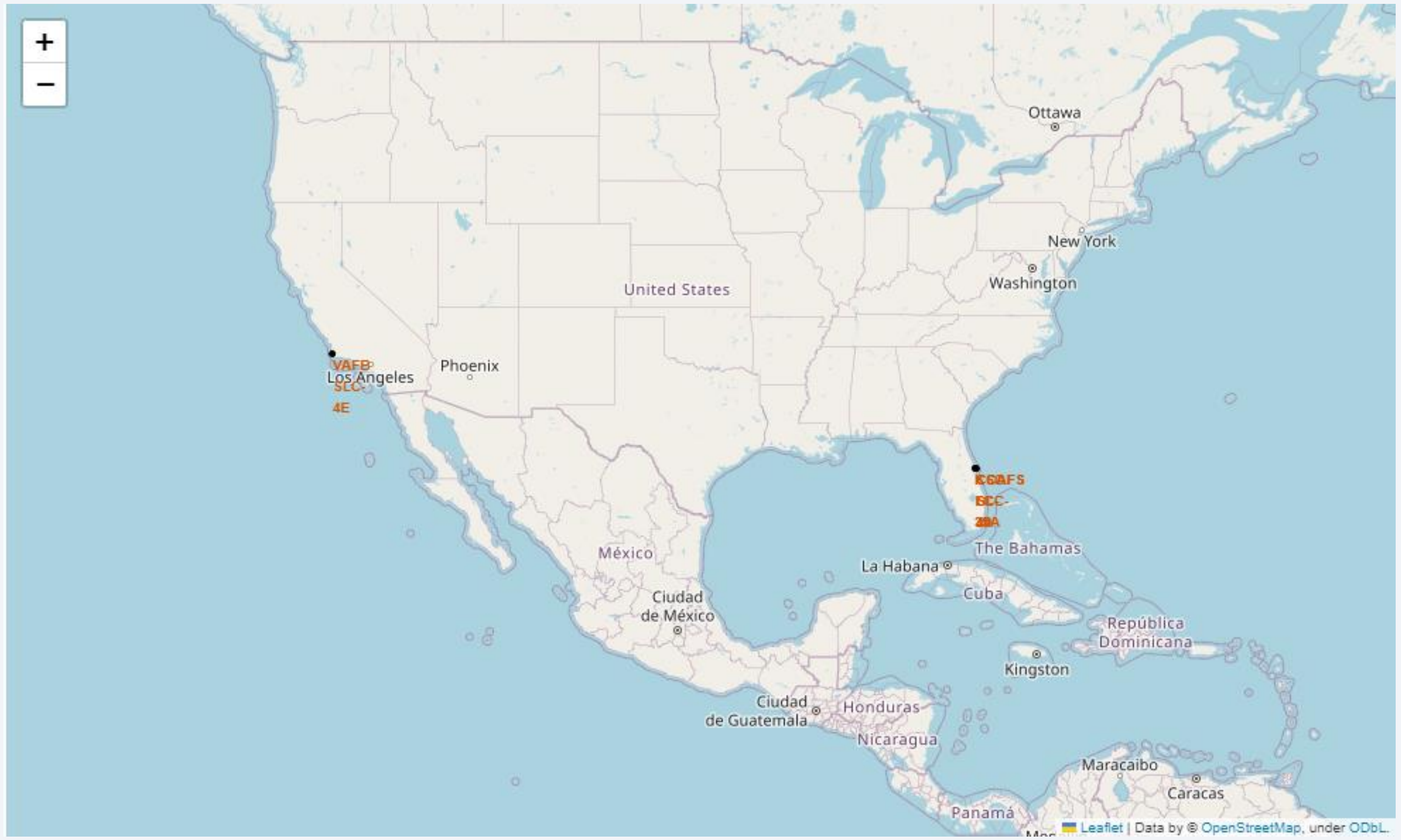
Section 3

Launch Sites Proximities Analysis

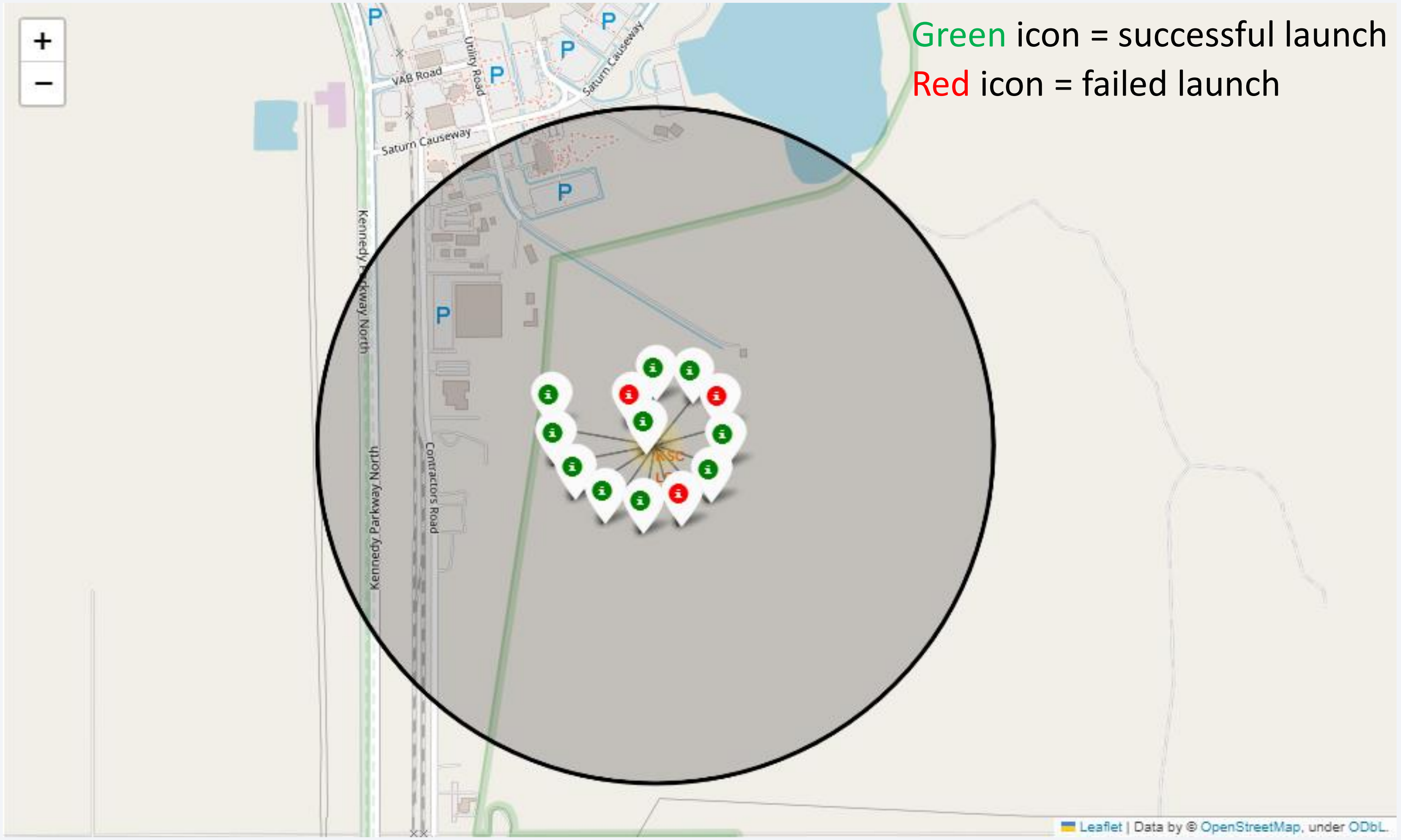
SpaceX Launch Sites Global Locations

- VAFB SLC-4E – located on the left coastline of the United States

- The other 3 launch sites are located on the right coastline of US, close to the Bahamas island, on the Merritt Island



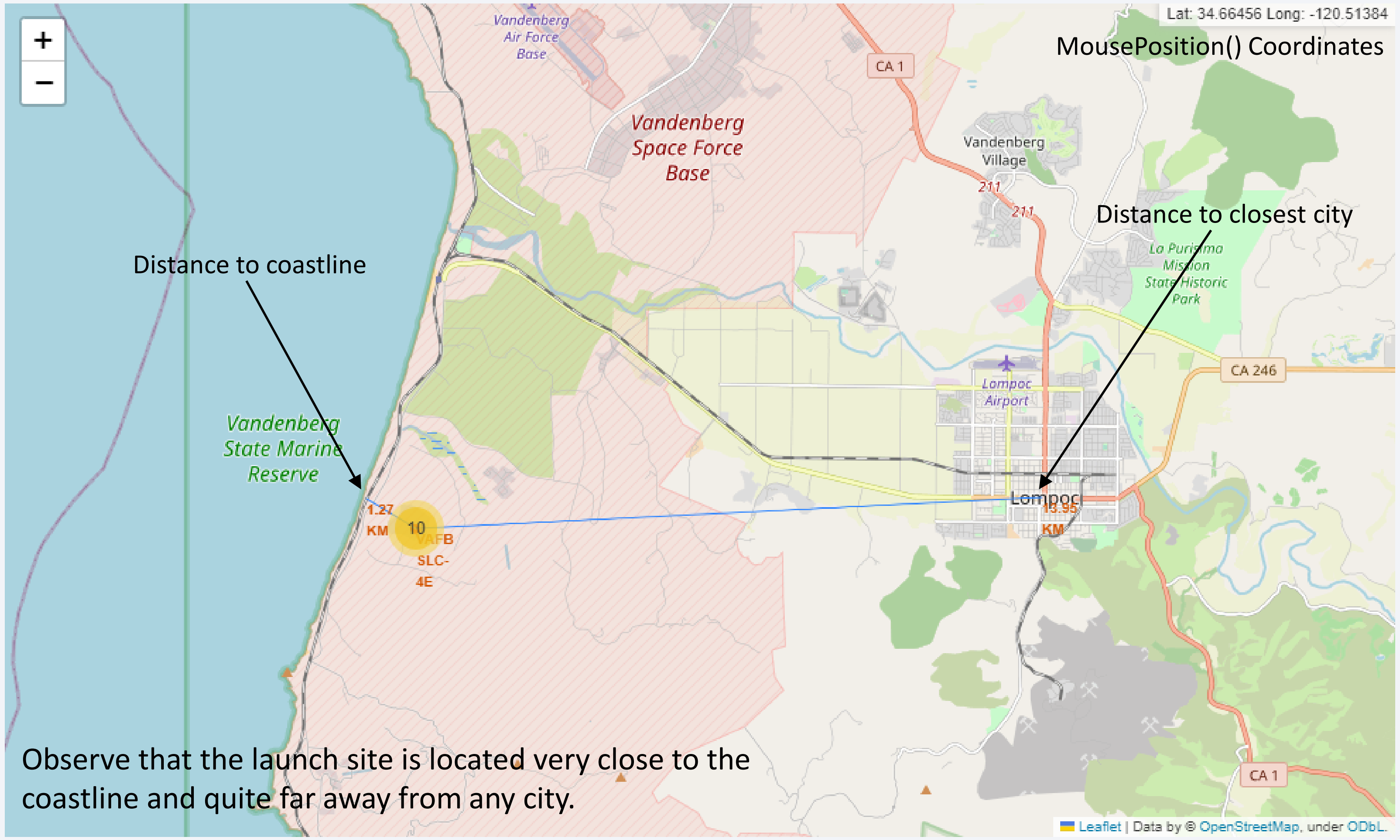
Launch Site KSC LC-39A Launch Outcomes



Green icon = successful launch

Red icon = failed launch

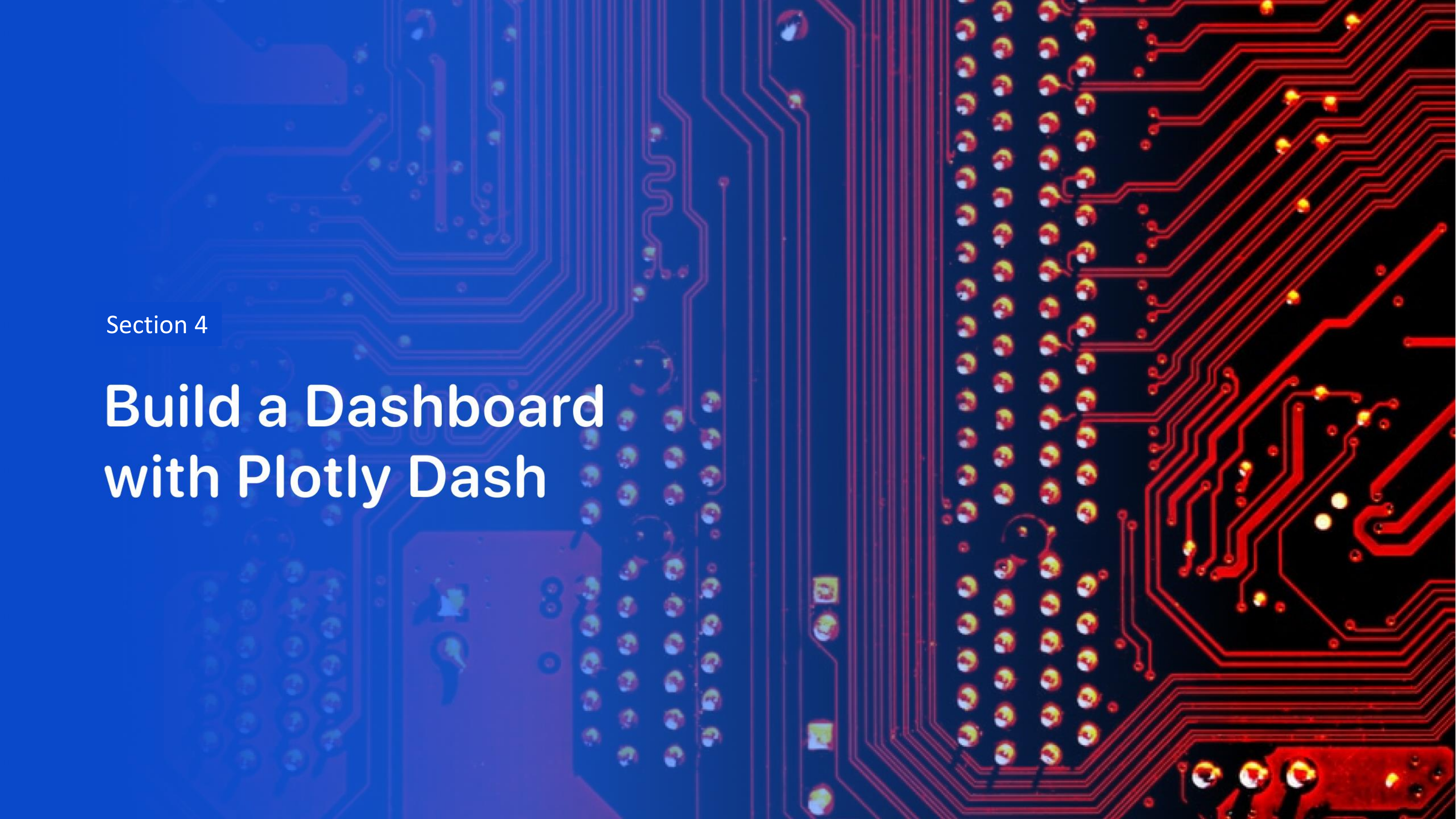
Launch Site VAFB SLC-4E Proximities


Lat: 34.66456 Long: -120.51384
MousePosition() Coordinates
Distance to closest city
Distance to coastline
Vandenberg Air Force Base
Vandenberg Space Force Base
CA 1
Vandenberg Village
211
La Purisima Mission State Historic Park
CA 246
Lompoc Airport
Vandenberg State Marine Reserve
1.27 KM
10
VAFB SLC-4E
Lompoc
13.95 KM
Leaflet | Data by © OpenStreetMap, under ODbL.

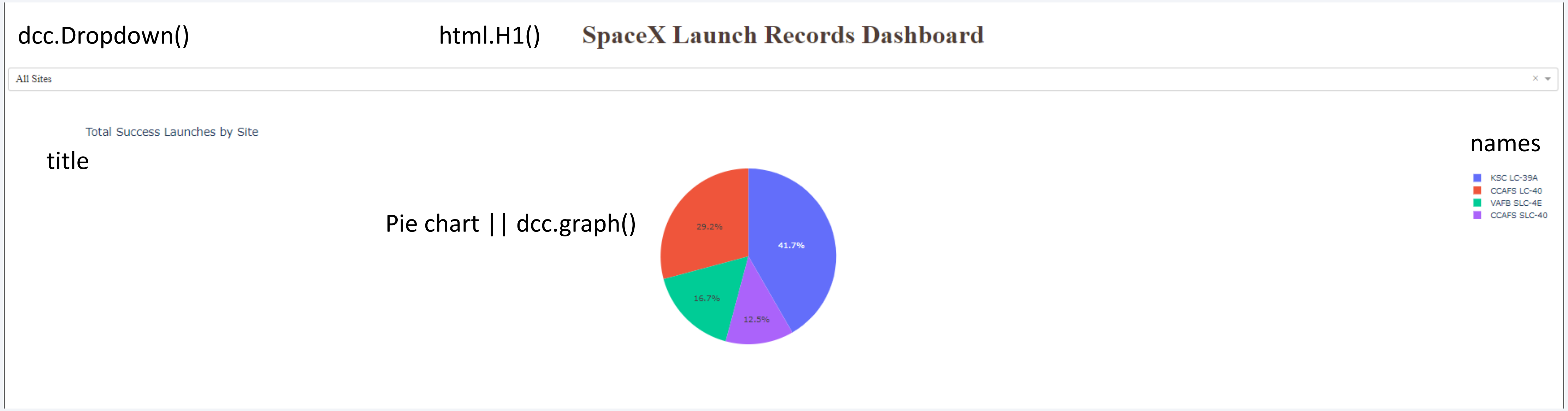
Observe that the launch site is located very close to the coastline and quite far away from any city.

Section 4

Build a Dashboard with Plotly Dash

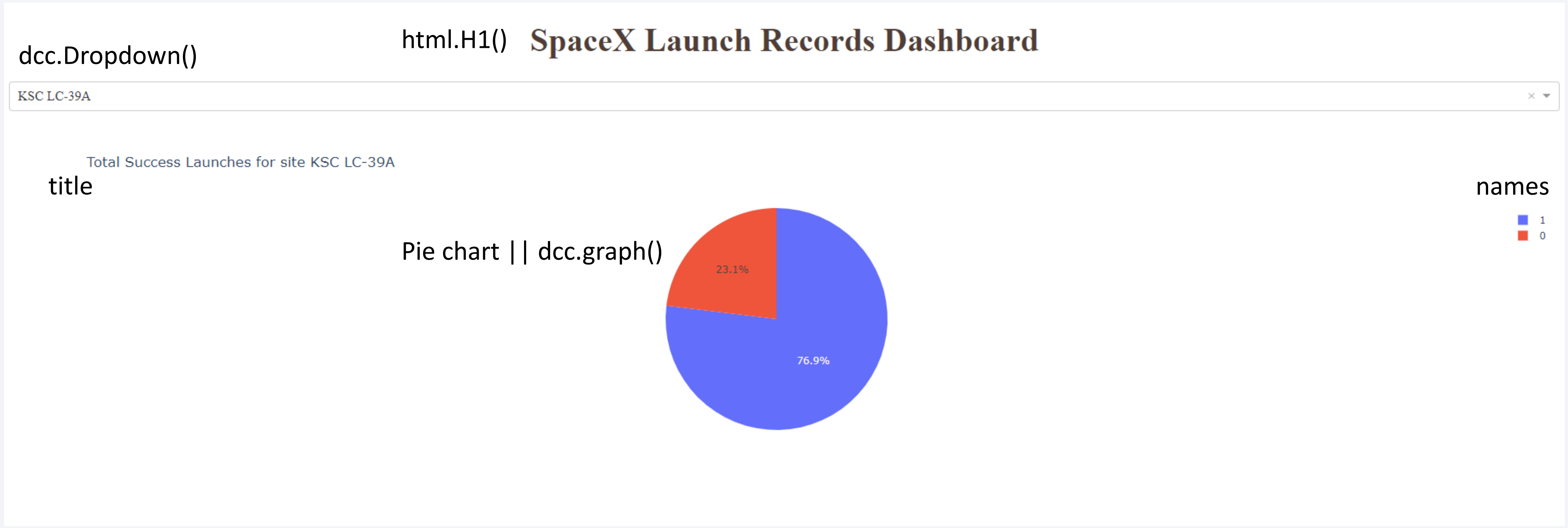


SpaceX Launch Sites Success Rates



- 41.7% of the total successful launches were from the KSC LC-39A launch site while CCAFS SLC-40 only accounts for 12.5% of the successful launches

KSC LC-39A Success Ratio


dcc.Dropdown()

html.H1() **SpaceX Launch Records Dashboard**

KSC LC-39A

Total Success Launches for site KSC LC-39A

title

names

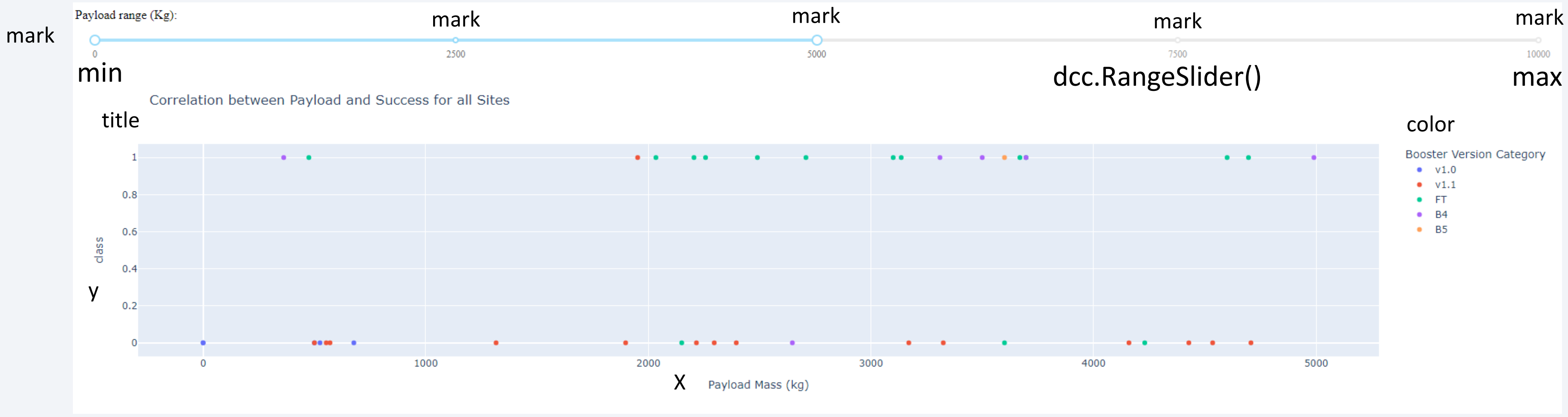
🟦 1

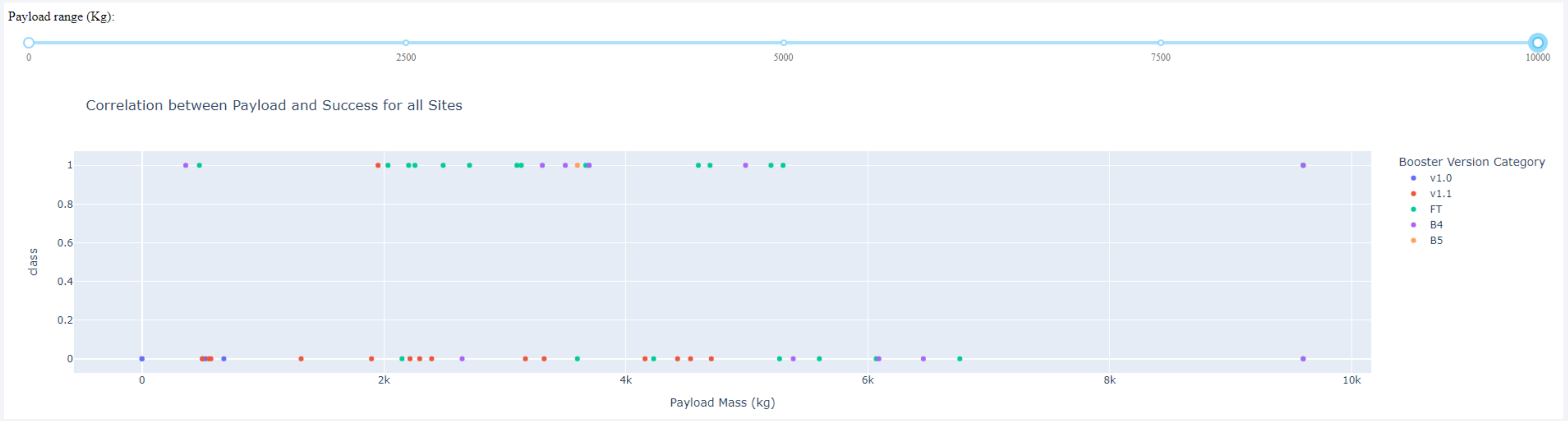
🟥 0

Pie chart || dcc.graph()

- 76.9% of rockets launched from the KSC LC-39A site were successful
- 23.1% of rockets launched from the KSC LC-39A site were unsuccessful

Global (All Sites) Payload and Success Rate Correlation





Global (All Sites) Payload and Success Rate Correlation

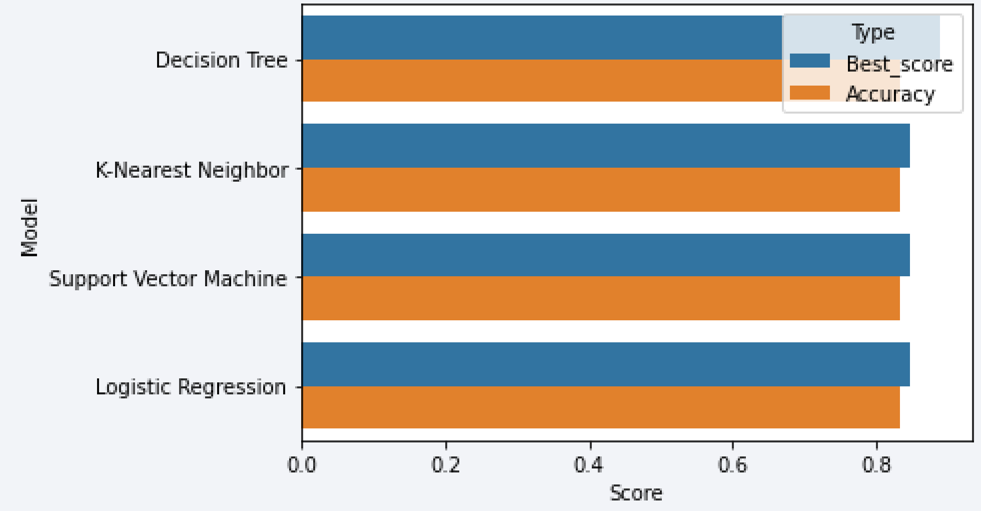
- Most of the successful launches are concentrated in the 2000-4000kg Payload mass range
- SpaceX utilized the ‘FT’ booster version in the majority of its Falcon 9 successful launches
- Booster version ‘V1.1’ was utilized in most of the failed launches
- There is one failed launch that had no payload. It used the ‘v1.0’ booster

Section 5

Predictive Analysis (Classification)

Classification Accuracy

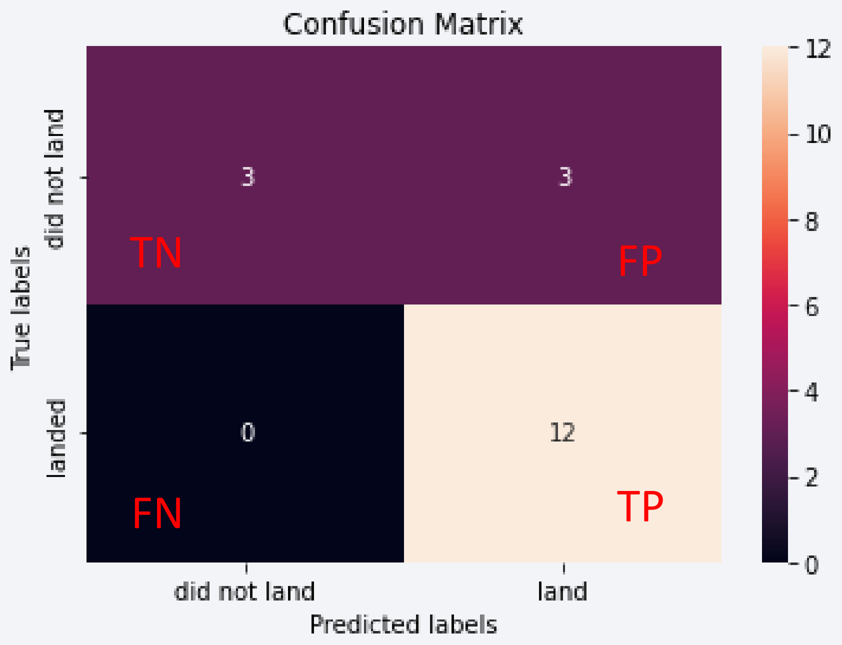
- The Decision Tree had the highest classification accuracy of 0.875 on the training data
- The models had identical accuracy on the test data



Model	Accuracy	Best_score
Decision Tree	0.833333	0.885714
K-Nearest Neighbor	0.833333	0.848214
Support Vector Machine	0.833333	0.848214
Logistic Regression	0.833333	0.846429

Confusion Matrix

- The problem here are the 3 false positives. In other words for 3 launches the decision tree model predicted that they were not going to land, but they did not land.

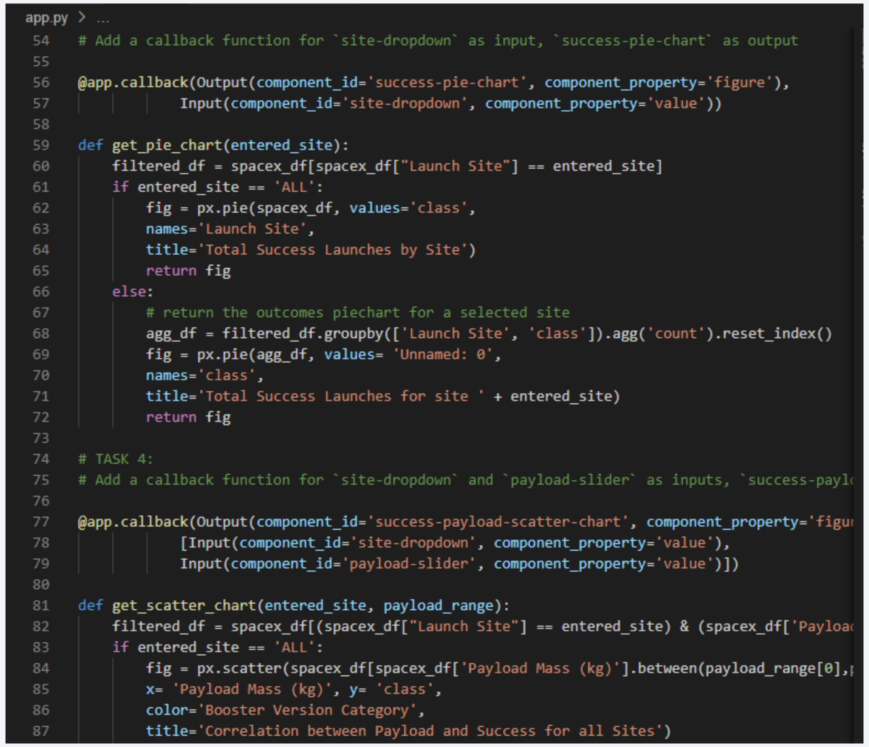


Conclusions

- SpaceX is a fierce competitor who benefits of extensive industry know-how gathered after so many flights.
- Flights seem to have a higher success rate in orbits: ES-L1, GEO, HEO and SSO
- Most of the successful launches are concentrated in the 2000-4000kg Payload mass range
- Our best performing model can predict if the stage 1 of a SpaceX launch will land given we know details like the launch site, booster version used and orbit with an accuracy of ~87.5%.

Appendix

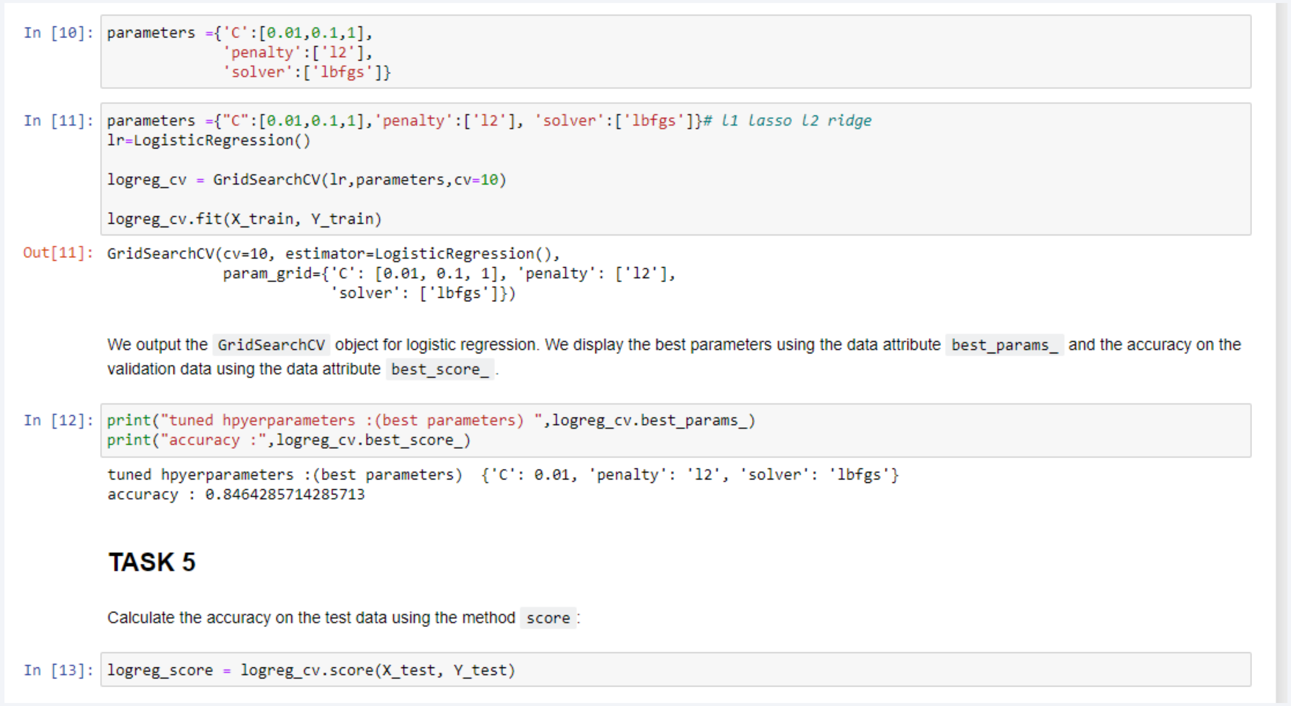
- Dashboard code snippet
    - Showing callback functions



app.py > ...

```
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output

@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
            Input(component_id='site-dropdown', component_property='value'))

def get_pie_chart(entered_site):
    filtered_df = spacex_df[spacex_df["Launch Site"] == entered_site]
    if entered_site == 'ALL':
        fig = px.pie(spacex_df, values='class',
        names='Launch Site',
        title='Total Success Launches by Site')
        return fig
    else:
        # return the outcomes piechart for a selected site
        agg_df = filtered_df.groupby(['Launch Site', 'class']).agg('count').reset_index()
        fig = px.pie(agg_df, values= 'Unnamed: 0',
        names='class',
        title='Total Success Launches for site ' + entered_site)
        return fig

# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-paylo

@app.callback(Output(component_id='success-payload-scatter-chart', component_property='figu
            [Input(component_id='site-dropdown', component_property='value'),
            Input(component_id='payload-slider', component_property='value')])

def get_scatter_chart(entered_site, payload_range):
    filtered_df = spacex_df[(spacex_df["Launch Site"] == entered_site) & (spacex_df['Payloa
    if entered_site == 'ALL':
        fig = px.scatter(spacex_df[spacex_df['Payload Mass (kg)'].between(payload_range[0],
        x= 'Payload Mass (kg)', y= 'class',
        color='Booster Version Category',
        title='Correlation between Payload and Success for all Sites')
```

- Example of how we created, trained and evaluated models



```
In [10]: parameters ={'C':[0.01,0.1,1],
                      'penalty':['l2'],
                      'solver':['lbfgs']}

In [11]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
         lr=LogisticRegression()

         logreg_cv = GridSearchCV(lr,parameters,cv=10)

         logreg_cv.fit(X_train, Y_train)

Out[11]: GridSearchCV(cv=10, estimator=LogisticRegression(),
                      param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                                  'solver': ['lbfgs']})
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
In [12]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
         print("accuracy :",logreg_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
         accuracy : 0.8464285714285713
```

TASK 5

Calculate the accuracy on the test data using the method `score` :

```
In [13]: logreg_score = logreg_cv.score(X_test, Y_test)
```

Thank you!