IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

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 Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

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. In this lab, you will create a machine learning pipeline to predict if the first stage will land given the data from the preceding labs.
- Problems you want to find answers
  - What makes a good lauch site?
  - Predict if a launch will be successful or not
  - What features make a successful launch?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Collection using SpaceX API
  - Collection through webscrapping using Beautiful Soup
- Perform data wrangling
  - Encoded categorical variables , cleaned missing values
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Used XGBoost model to make predictions

Data Collection

- SpaceX API to CSV.
  - Data was collected using SpaceX API
  - To connect to the SpaceX API, we used the requests module in python.
  - We connected to the API and cleaned the data we needed.
  - After the data was cleaned we saved the data into a CSV file
- Web scrapping using Beautiful Soup
  - Scraped table from Wikipedia.
  - We took the table data and saved it to a pandas dataframe
  - Once we cleaned the data we saved it to a CSV file.

Data Collection – SpaceX API

- We used the get requests module from python to connect to the SpaceX API. We also cleaned the data and eventually saved in to a CSV file.

- Link to the notebook on GitHub: https://github.com/orbti/Coursera-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

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 [9]:
```
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 [10]:
```
response.status_code
```

Out[10]: 200

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

In [11]:
```
# 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 [12]:
```
# Get the head of the dataframe
data.head()
```

Out[12]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}]	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e2]	5e9

Data Collection - Scraping

- Using Beautiful Soup, we scrapped the tables from Wikipedia.
- We then parsed the information into a pandas dataframe to save as a CSV
- Link to the note book on GitHub: https://github.com/orbti/Coursera-Capstone/blob/main/jupyter-labs-webscraping.ipynb

In [4]:
```python
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [17]:
```python
# use requests.get() method with the provided static_url
# assign the response to a object
r = requests.get(static_url)
```

Create a `BeautifulSoup` object from the HTML `response`

In [20]:
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(r.text, 'html.parser')
print(soup.prettify())
```

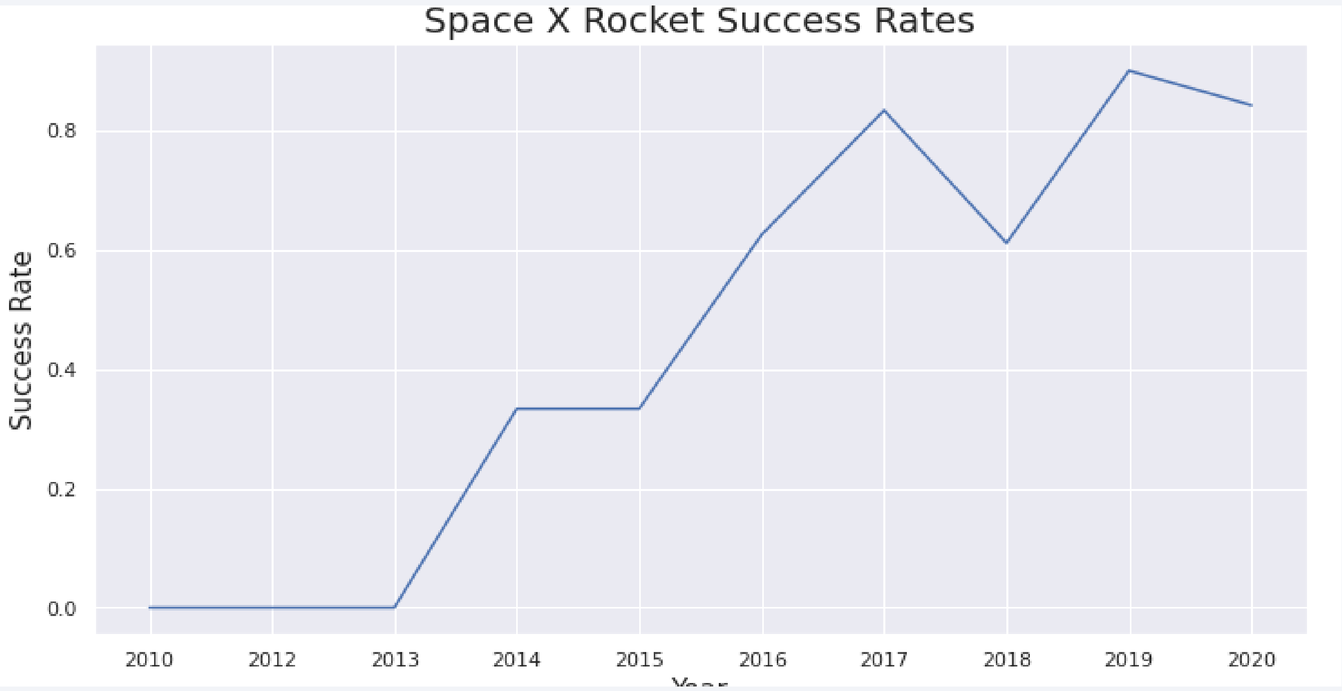
```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

Data Wrangling

- We used categorical encoding.
- We cleaned out all the columns we will not use.
- We filled in missing data.
- Link to the notebook on GitHub: https://github.com/orbti/Coursera-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We explored the data to look at the success rates for each year of launches.
- https://github.com/farishelmi17/Applied-Data-Science-Capstone-SpaceX/blob/main/notebook:Exploratory_Data_Analysis_with_Visualisation_Lab_jJkKVG6F1.ipynb



EDA with SQL

- We loaded the SpaceX data into a sqlite database.
- Items we looked at in SQL:
  - Names of unique launch sites
  - Total payload mass carried by boosters launched by NASA (CRS)
  - Average payload mass carried by booster version F9 v1.1
- Rank the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order.
- Link to notebook on GitHub: https://github.com/farishelmi17/Applied-Data-Science-Capstone-SpaceX/blob/main/notebook:Exploratory_Data_Analysis_with_SQL__eqznon1EA.ipynb

Build an Interactive Map with Folium

- Marked all launch sites on a mpa.
- Mark the success/failed launches for each site on the map.
- Calculate the distance between a launch site to its proximities.
- Link to notebook on GitHub: https://github.com/orbti/Coursera-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We created a interactive dashboard with Plotly and Dash
- We plotted a pie chart showing the total launches by a site.
- We plotted a scatter plot of the relationship between class and payload size in mass (kg).
- Explain why you added those plots and interactions
- Link to notebook on GitHub: https://github.com/orbti/Coursera-Capstone/blob/main/dash/spacex_dash_app.py

Predictive Analysis (Classification)

- I loaded in our data using pandas. Then split the data into a train and test dataset.
- I used a Pipeline to standardize and impute the data.
- I used GridSearchCV to select the best hyperparameters for the model I was looking at.
- Link to notebook on GitHub: https://github.com/orbti/Coursera-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

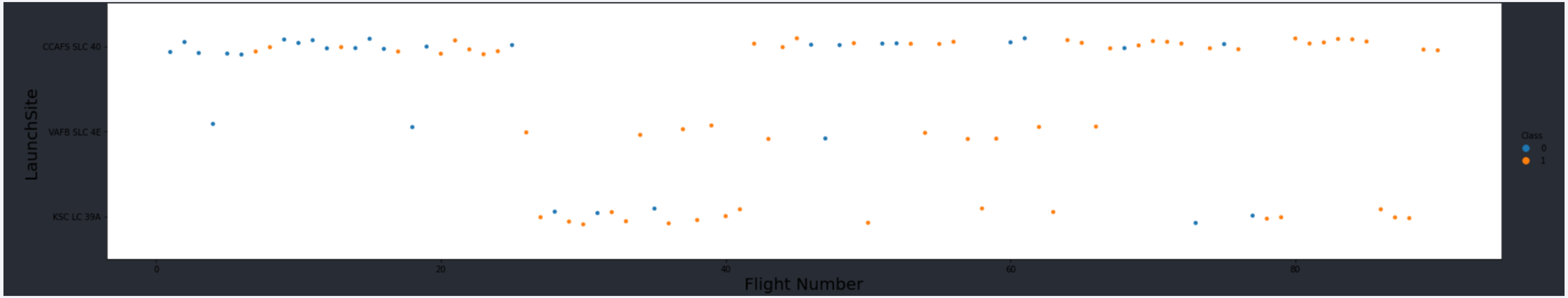
Results

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

Section 2

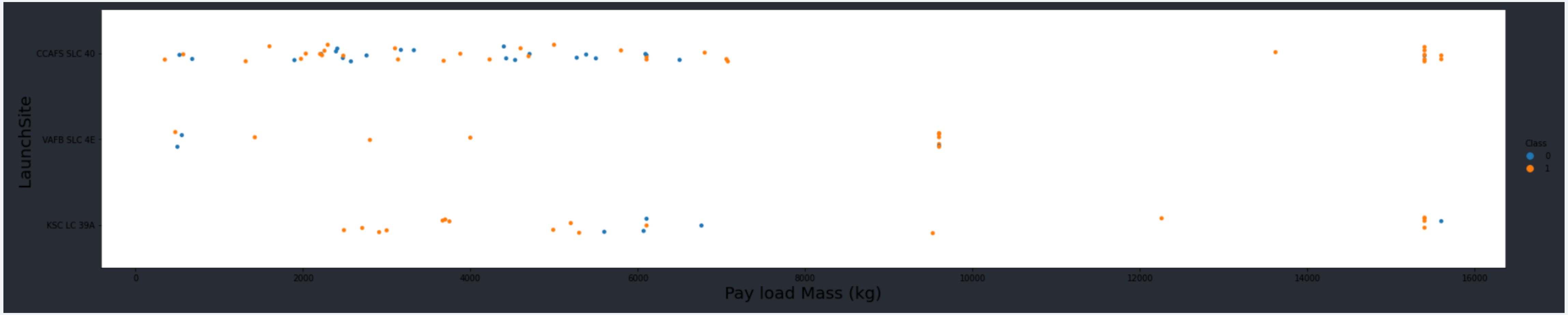
Insights drawn from EDA

Flight Number vs. Launch Site



- The sites with a greater number of flights have more successful launches.

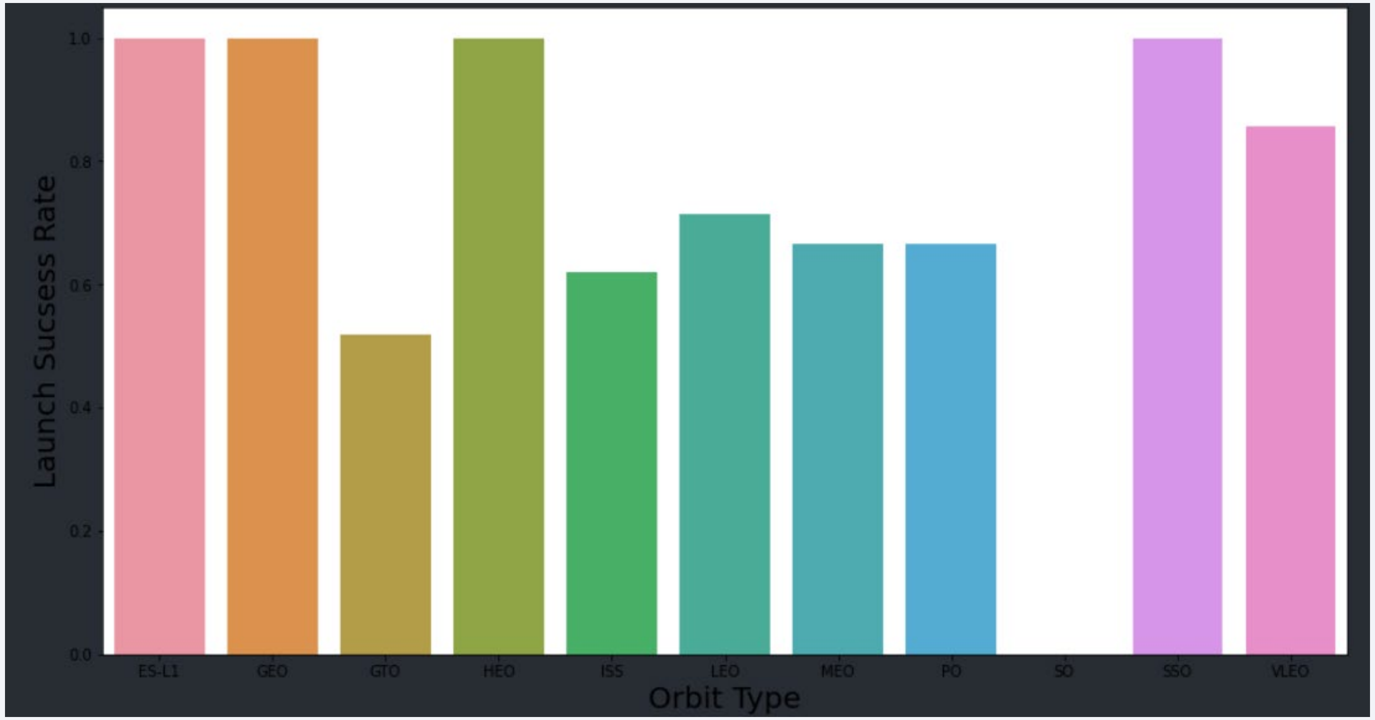
Payload vs. Launch Site



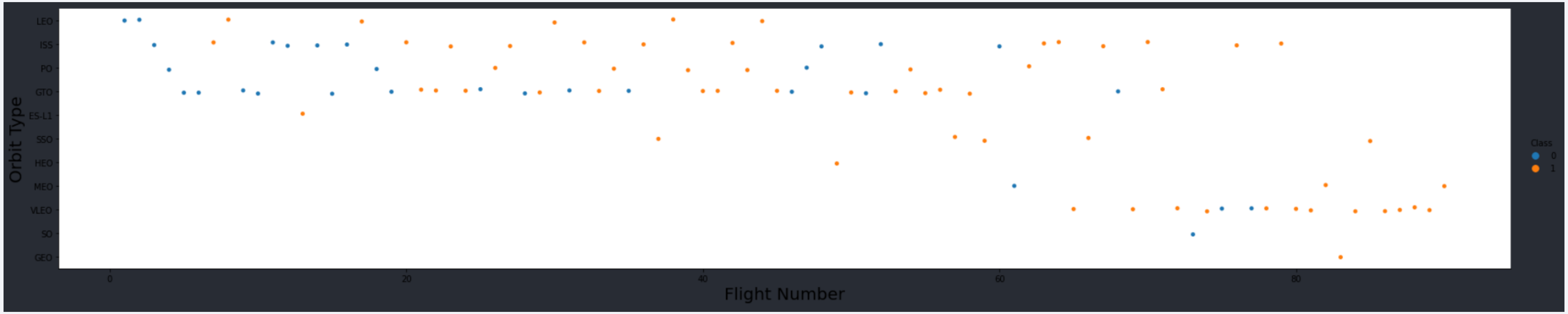
- Larger the payload the more successful the launch.

Success Rate vs. Orbit Type

- Launch with orbit type ES-L1, GEO, HEO AND SSO had the most successful launches.

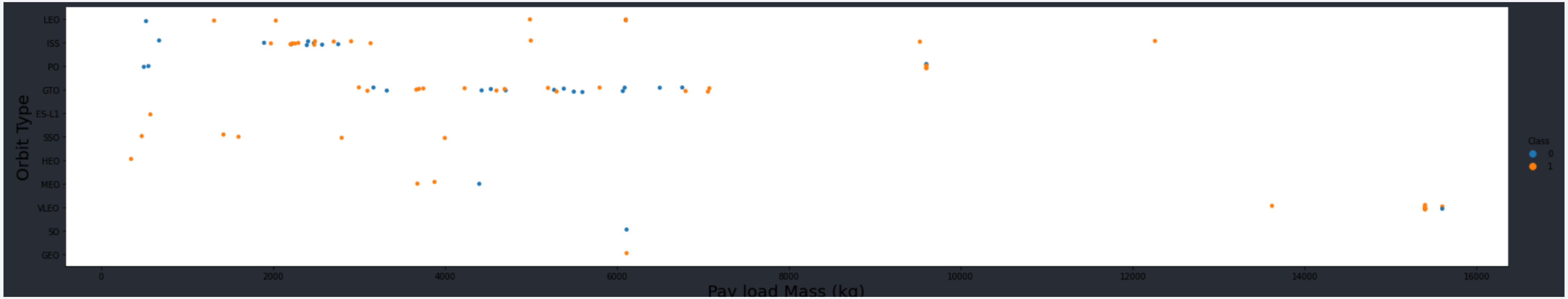


Flight Number vs. Orbit Type



- 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.

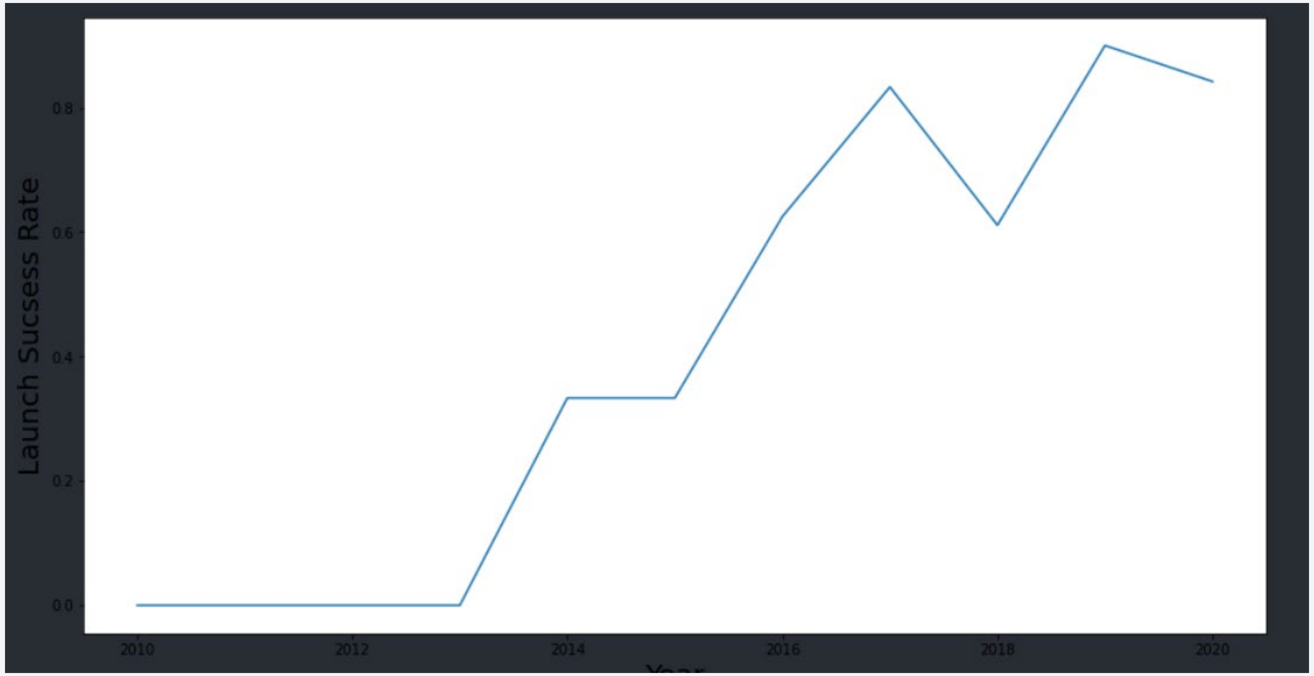
Payload vs. Orbit Type



- 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

- you can observe that the success rate since 2013 kept increasing till 2020



All Launch Site Names

- We selected all the launch sites and grouped them.

```
%%sql
    select Launch_Site
    from spacextbl
    group by Launch_Site
```

[7] ✓ 0.3s

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

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

Launch Site Names Begin with 'CCA'

```
%%sql
    select *
    from spacextbl
    where launch_site like 'CCA%'
    limit 5
```

[8] ✓ 0.6s

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Selected all columns where launch site contains CCA at the start and limited it to 5 results.

Total Payload Mass

```
%%sql
    select Customer, sum(PAYLOAD_MASS__KG_) total_payload_mass__kg_
    from SPACEXTBL
    where Customer = 'NASA (CRS)'
    group by Customer
```

[9] ✓ 0.4s

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

Customer	total_payload_mass__kg_
NASA (CRS)	45596

- Selected customer and sum of payload mass.
- Filtered results to only included customers from ‘NASA (CRS)’ when grouped by customer.

Average Payload Mass by F9 v1.1

- Selected the average payload of each booster with version ‘F9 v1.1’

```
%%sql
SELECT Booster_Version, AVG(PAYLOAD_MASS__KG_) total_payload_mass__kg_, COUNT(*)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.1%'
GROUP BY Booster_Version
```

✓ 0.4s

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

Booster_Version	total_payload_mass__kg_	COUNT(*)
F9 v1.1	2928.4	5
F9 v1.1 B1003	500.0	1
F9 v1.1 B1010	2216.0	1
F9 v1.1 B1011	4428.0	1
F9 v1.1 B1012	2395.0	1
F9 v1.1 B1013	570.0	1
F9 v1.1 B1014	4159.0	1
F9 v1.1 B1015	1898.0	1
F9 v1.1 B1016	4707.0	1
F9 v1.1 B1017	553.0	1
F9 v1.1 B1018	1952.0	1

First Successful Ground Landing Date

```
%%sql
    SELECT *
    FROM SPACEXTBL
    WHERE Date = (
        SELECT MIN(Date)
        FROM SPACEXTBL
        WHERE `Landing _Outcome` LIKE 'Success (ground pad)'
    )
```

1] ✓ 0.4s

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
01-05-2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)

- January 05, 2017 was the first successful launch date.

Successful Drone Ship Landing with Payload between 4000 and 6000

- We filtered using where and between.

```
%%sql
SELECT Booster_Version
FROM SPACEXTBL
WHERE `Landing _Outcome` = 'Success (drone ship)'
    AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
```

✓ 0.3s

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

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

Total Number of Successful and Failure Mission Outcomes

- Counted all rows that are grouped by mission outcome.

```
%%sql
SELECT Mission_Outcome, COUNT(Mission_Outcome)
FROM SPACEXTBL
GROUP BY Mission_Outcome
```

0.3s

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

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

Boosters Carried Maximum Payload

- Used a subquery to filter each booster versions max payload.

```
%%sql
SELECT
    Booster_Version,
    PAYLOAD_MASS__KG_
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL
)
```

[14] 0.6s

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

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

- We used a substr function to filter all years to 2015 and landing outcome is failure with a d.rone

```
%%sql
SELECT
    SUBSTR(Date,4,2) AS month,
    `Landing _Outcome`,
    Booster_Version,
    Launch_Site
FROM SPACEXTBL
WHERE SUBSTR(Date,7,4) = '2015'
    AND `Landing _Outcome` = 'Failure (drone ship)'
```

0.3s

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

month	Landing _Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- 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(*) count
    FROM SPACEXTBL
    WHERE Date BETWEEN '04-06-2010' AND '20-03-2017'
    GROUP BY `Landing _Outcome`
    ORDER BY count DESC
```

[37] ✓ 0.2s

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

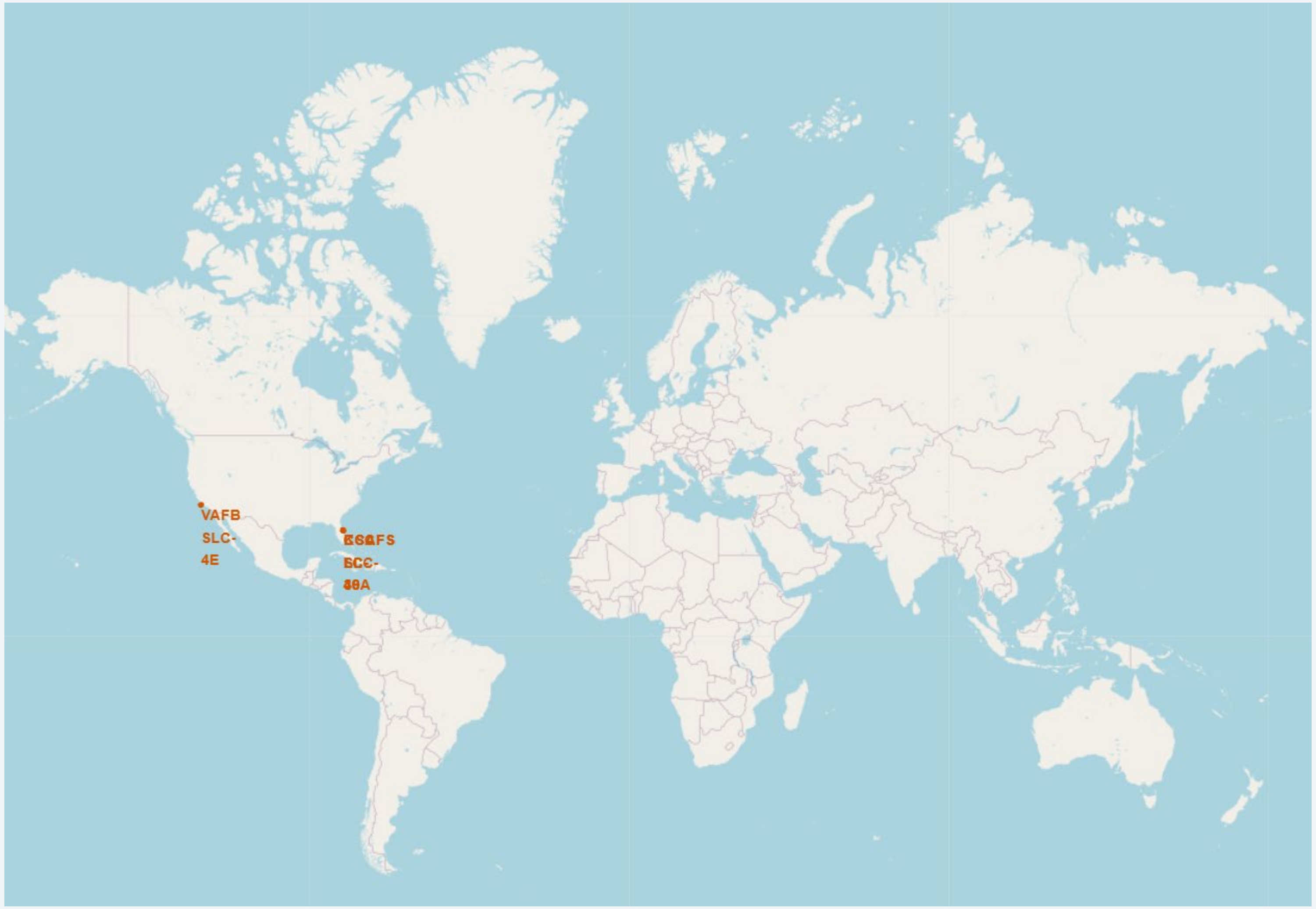
Landing _Outcome	count
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

Section 3

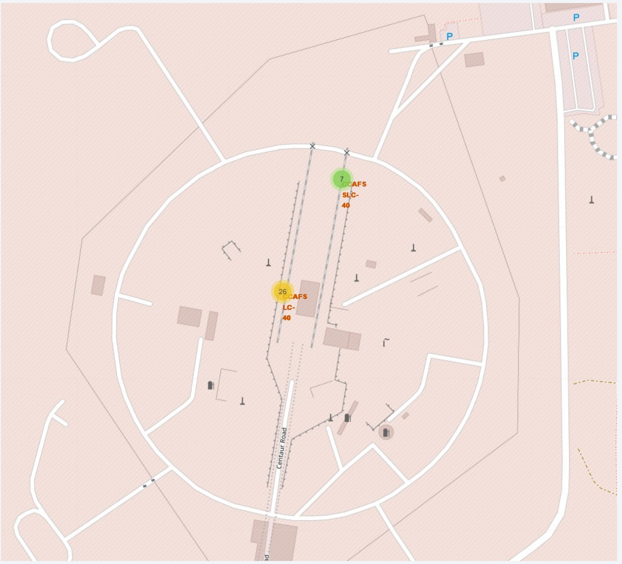
Launch Sites Proximities Analysis

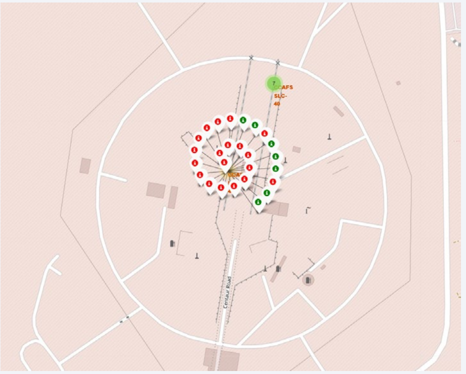
All Launch sites Globally

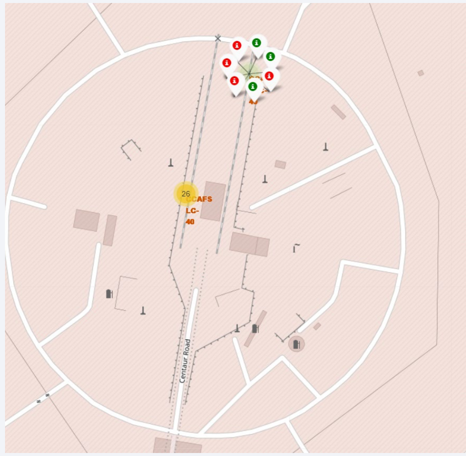
- All launch sites are located in the United States.

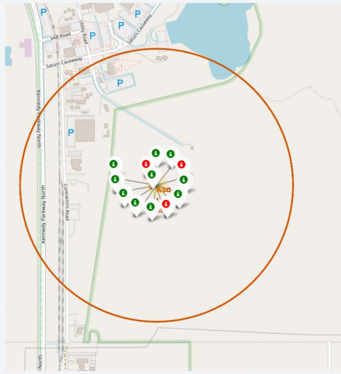


Successful/Failed Markers for Launch Sites





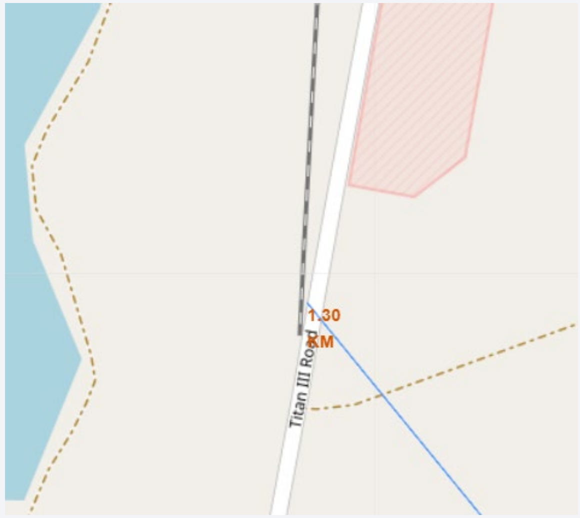




- Florida and California sites
- Red means failed launches
- Green means successful launches

Launch Sites Distance to Landmarks

- Are launch sites near railways? No
- Are launch sites near highways? No
- Are launch sites near coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

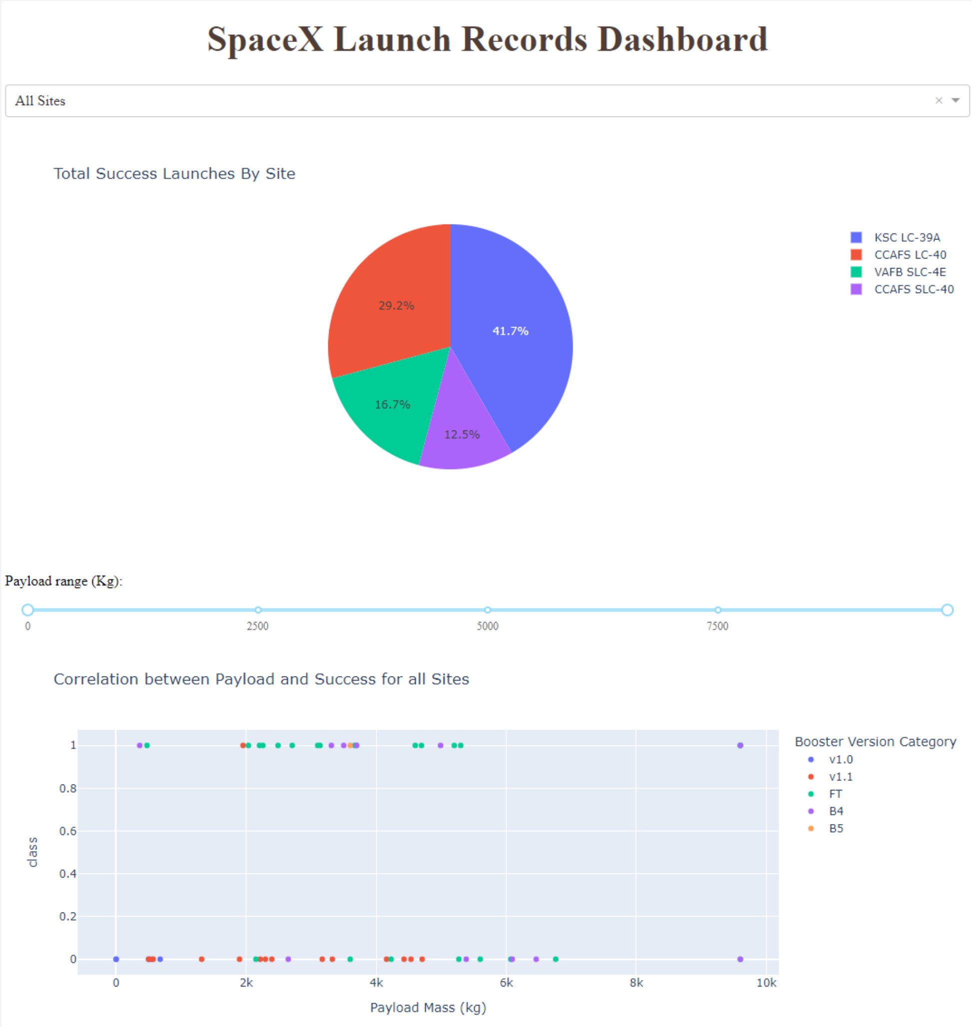


Section 4

Build a Dashboard with Plotly Dash

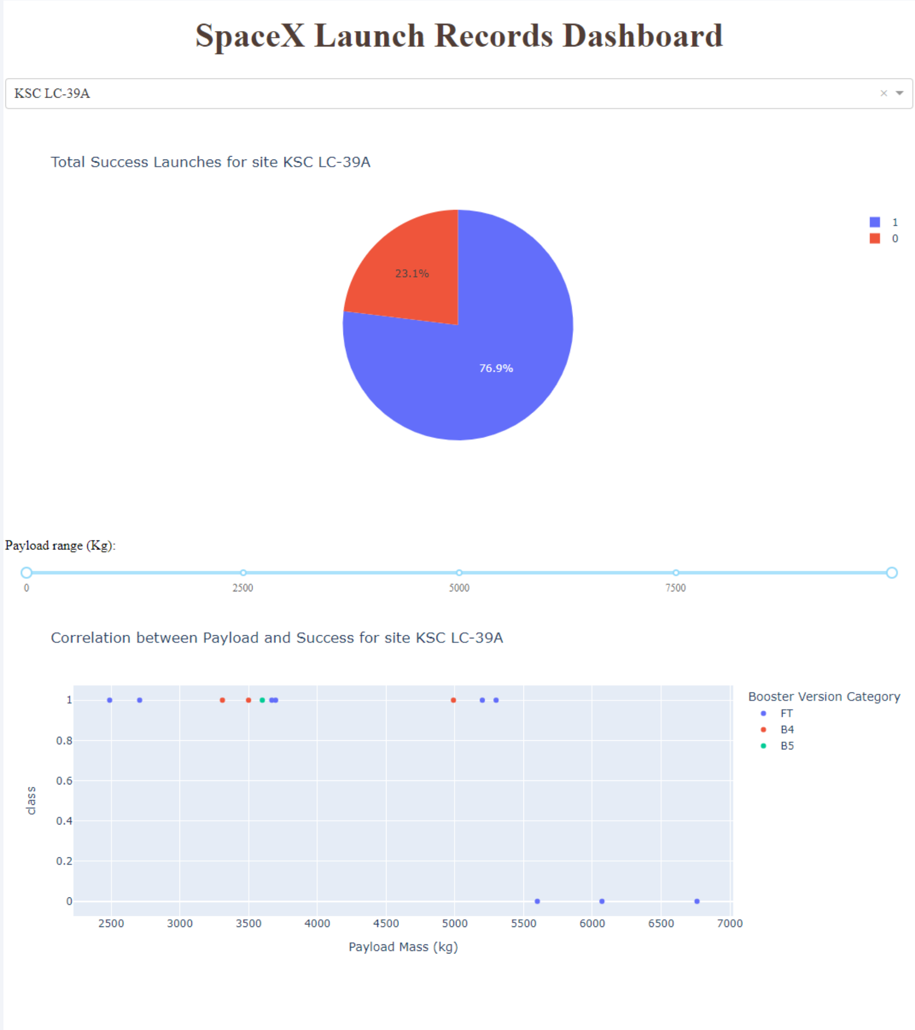
Dashboard showing all sites

- Most successful launch site is KSC LC-39A



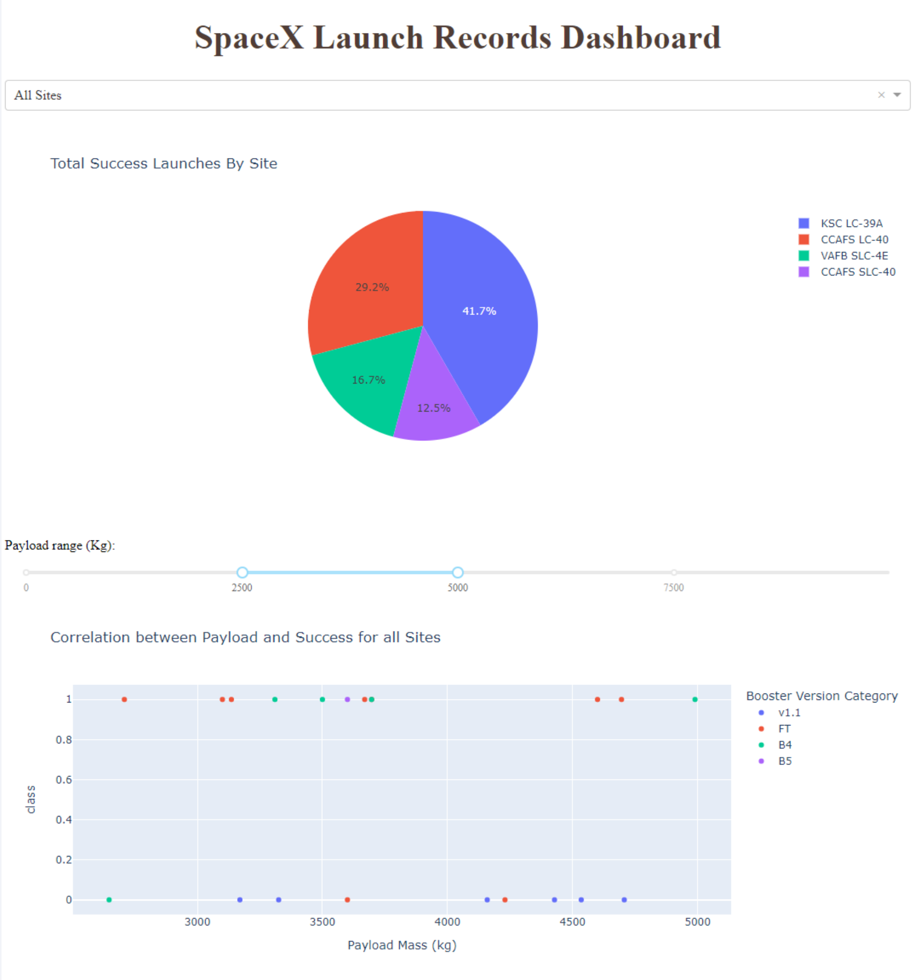
Dashboard showing site KSC LC-39A

- Heavy payloads did not do well in this site.



Dashboard with payload set to 2500-5000

- With payloads between 2500 and 5000, booster version FT and B4 were very successful.



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- XGBoost had the best accuracy with 83% on the test data.

Find the method performs best:

```python
from xgboost import XGBClassifier

pipe = Pipeline([
    # ('scaler', preprocessing.StandardScaler()),
    ('model', XGBClassifier())
])

parameters = {
    'model__colsample_bytree': [0.5, 0.75, 1],
    'model__max_depth': range(2, 10, 1),
    'model__n_estimators': range(150, 180, 5),
    'model__learning_rate': [1, 0.1, 0.01, 0.05],
    'model__min_child_weight': range(0, 15, 5)
}

grid = GridSearchCV(
    estimator=pipe,
    param_grid=parameters,
    scoring='accuracy',
    n_jobs=10,
    verbose=2,
    cv=10
)

xgboost_model = grid.fit(X_train, Y_train)
```

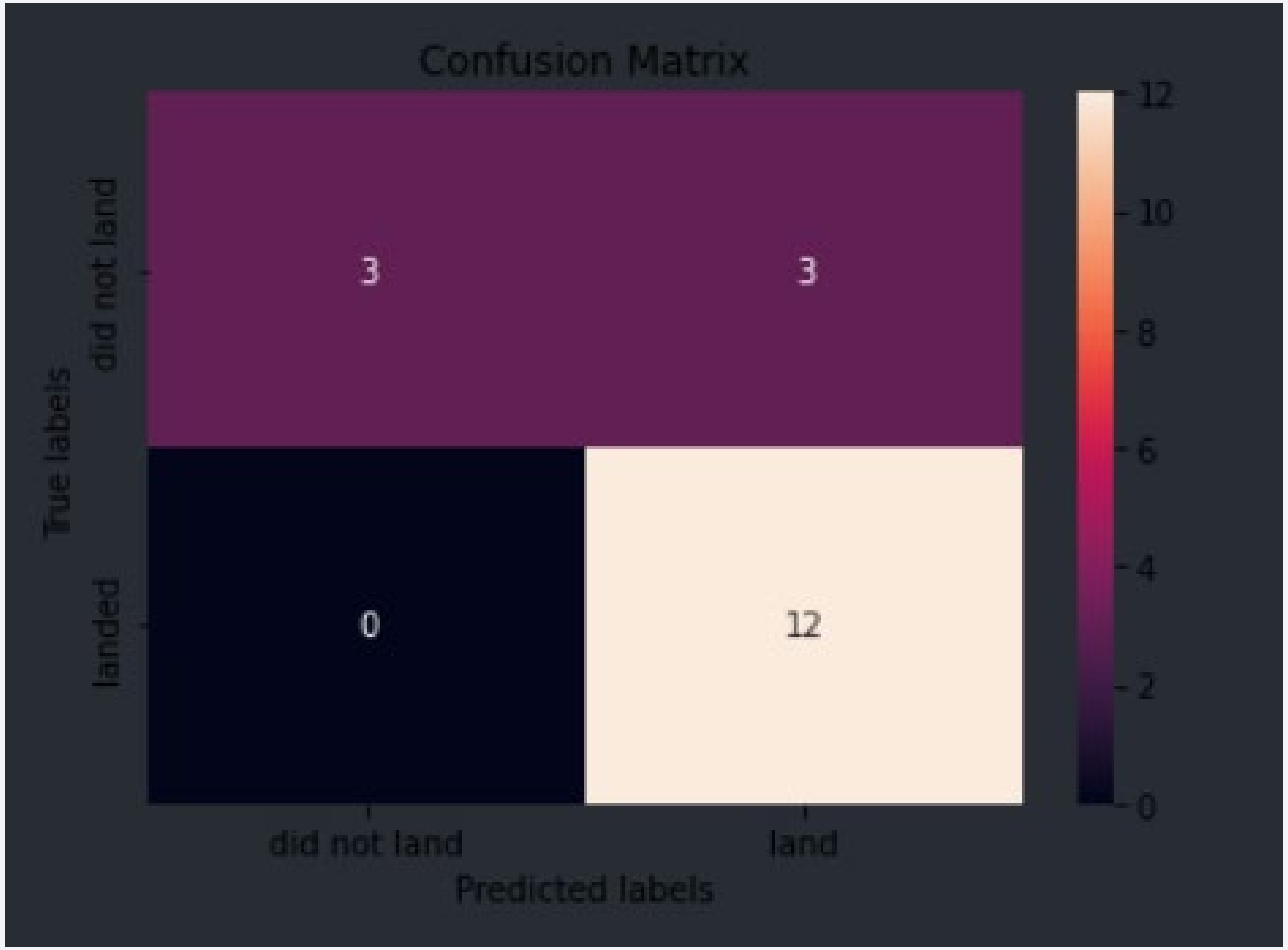
```
Fitting 10 folds for each of 1728 candidates, totalling 17280 fits
```

```python
print("tuned hpyerparameters :(best parameters) ",xgboost_model.best_params_)
print("accuracy :",xgboost_model.best_score_)

print(f'Test accuracy: {xgboost_model.score(X_test, Y_test)}')

yhat = xgboost_model.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

```
tuned hpyerparameters :(best parameters)  {'model__colsample_bytree': 0.75, 'model__learning_rate': 0.01, 'model__max_depth': 2, 'model__min_child_weight': 0, 'model__n_estimators': 150}
accuracy : 0.8607142857142855
Test accuracy: 0.8333333333333334
```

Confusion Matrix

- The confusion matrix for XGBoost shows that the model is good at predicting successful landings. It still needs to be optimized to predict failed landings.



Conclusions

- Larger the payload the more successful the launch site is
- Launch success rate started to increase after year 2013 and is still trending up.
- XGBoost was the best model to predict launch success.

Thank you!