IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Wojciech
25.09.2021

Outline

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

Executive Summary

- Methodology:

The main purpose was to complete whole process of data collection, data transformation, EDA using pandas and SQL, modeling, creating dashboards using Dash

- Summary of results:

All of this steps were completed successfully, and the results are available on this presentation

Introduction

- Project was created for Data Science Capston project on Coursera
- The main purpose was to complete whole process of data collection, data transformation, EDA using pandas and SQL, modeling, creating dashboards using Dash

Section 1

Methodology

Methodology

Executive Summary

- Data collection:
  - API (SpaceX API)
  - Web scrapping (wikipedia.org)
- Data wrangling
  - Data was processed using pandas and BeautifulSoup
- Exploratory data analysis (EDA) using visualization and SQL
- Interactive visual analytics using Folium and Plotly Dash
- Predictive analysis using classification models
  - Classification models were prepared

Data Collection – SpaceX API

1. SpaceX API:

we got data using requests library:

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

and filtered data:

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multi
ple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Data Collection – SpaceX API

Next we filtered data to get only information about Falcon 9 launches, and we dealt with missing values in Payload mass column. First 5 rows:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.047721
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.047721
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.047721
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.047721
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857

[Link to notebook on IBM Watson]()

Data Collection - Scraping

2. Wikipedia: we got data using requests library and created BeautifulSoup object from the response:

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

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

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

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

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

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
soup.find('title').text
```

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

Data Collection - Scraping

And then we extracted information we want from it using BeautifulSoup library. First 5 rows of final DataFrame:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA (COTS)\nNRO	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

Link to notebook on IBM Watson

Data Wrangling

- Data was filtered based on date and rocket type
- Missing values were displayed and removed from some columns:

```
# Calculate the mean value of PayloadMass column
payloadmass_mean = data_falcon9.PayloadMass.mean()

# Replace the np.nan values with its mean value
data_falcon9.PayloadMass = data_falcon9.PayloadMass.replace(np.nan, payloadmass_mean)
```

```
data_falcon9.isnull().sum()
```

```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        5
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

EDA with Data Visualization

- Missing values percent in columns:

```
FlightNumber       0.000
Date               0.000
BoosterVersion     0.000
PayloadMass        0.000
Orbit              0.000
LaunchSite         0.000
Outcome            0.000
Flights            0.000
GridFins           0.000
Reused             0.000
Legs               0.000
LandingPad        40.625
Block              0.000
ReusedCount        0.000
Serial             0.000
Longitude          0.000
Latitude           0.000
dtype: float64
```

- Number of launches on sites:

```
CCAFS SLC 40     55
KSC LC 39A       22
VAFB SLC 4E      13
```

- Launch success rate:

```
In [17]: df["Class"].mean()
Out[17]: 0.6666666666666666
```

- Orbits used:

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
SO        1
HEO       1
GEO       1
ES-L1     1
```

EDA with Data Visualization

[Link to notebook on IBM Watson]

EDA with SQL

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

```
SELECT DISTINCT LAUNCH_SITE FROM SPACEXDATASET
```

Run time: **0.009 s**

Result set 1

Find

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

EDA with SQL

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

```
SELECT * FROM SPACEXDATASET WHERE LAUNCH_SITE LIKE('CCA%') LIMIT 5
```

Result set 1

DATE	TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere chee
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2

EDA with SQL

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

```
SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXDATASET WHERE CUSTOMER ='NASA (CRS)'
```

Result set 1

1
45596

```
SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXDATASET WHERE BOOSTER_VERSION = 'F9 v1.1'
```

Result set 1

1
2928

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

Hint:Use min function

```
SELECT MIN(DATE) FROM SPACEXDATASET WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

Result set 1

1
2015-12-22

EDA with SQL

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

```
SELECT DISTINCT BOOSTER_VERSION FROM SPACEXDATASET WHERE LANDING__OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
```

Result set 1

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

List the total number of successful and failure mission outcomes

```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) FROM SPACEXDATASET GROUP BY LANDING__OUTCOME
```

Result set 1

LANDING__OUTCOME	2
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	22

EDA with SQL

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

```
SELECT DISTINCT BOOSTER_VERSION FROM SPACEXDATASET WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXDATASET)
```

Result set 1

BOOSTER_VERSION
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7

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

```
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXDATASET WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND YEAR(Date) = '2015'
```

Run time: 0.005 s

Result set 1

Find

LANDING__OUTCOME	BOOSTER_VERSION	LAUNCH_SITE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

EDA with SQL

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

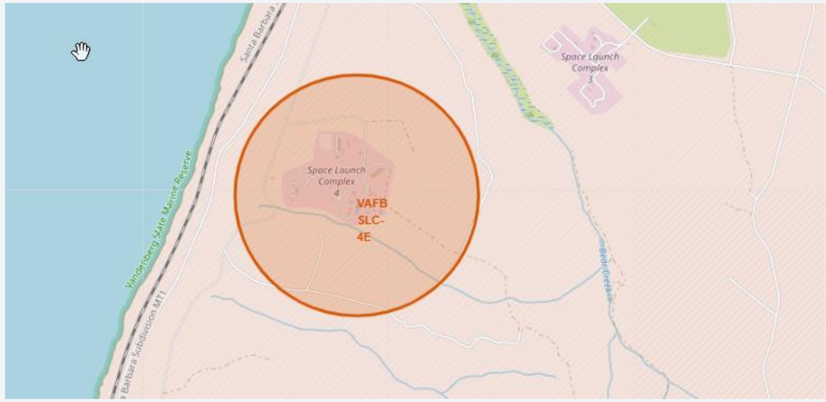
```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) FROM SPACEXDATASET
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY COUNT(LANDING__OUTCOME) DESC;
```

LANDING__OUTCOME	2
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

Build an Interactive Map with Folium

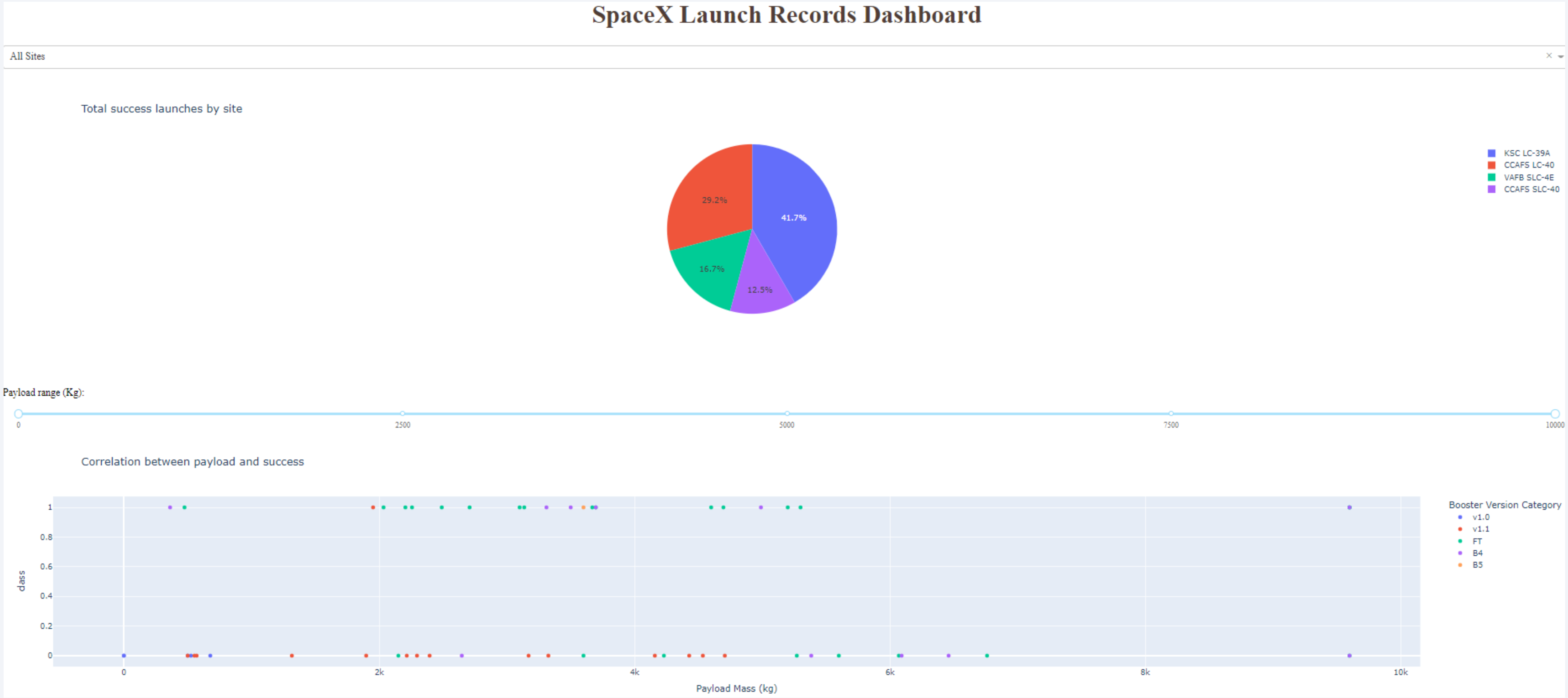
- Example map and marker creation and scrrenshot:

```python
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label
for index, row in launch_sites_df.iterrows():
    # Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
    circle = folium.Circle([row[1], row[2]], radius=1000, color='#d35400', fill=True).add_child(folium.Popup(row[0]))
    # Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
    marker = folium.map.Marker(
        [row[1], row[2]],
        # Create an icon as a text label
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % row[0],
            )
        )
    site_map.add_child(circle)
    site_map.add_child(marker)
display(site_map)
```



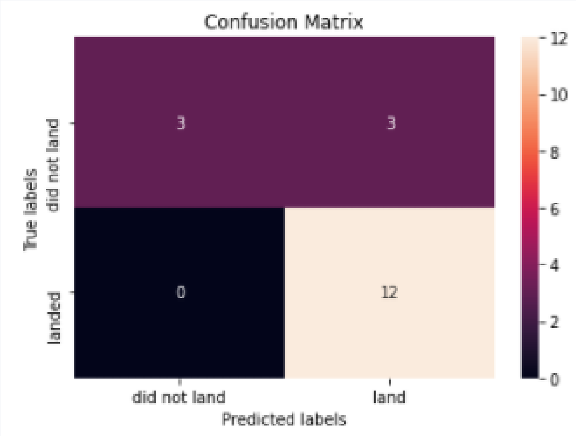
Build a Dashboard with Plotly Dash

- We added piechart, scatterplot and slider to the template:



Predictive Analysis (Classification)

- Data was prepared using StandardScaler
- GridSearchCV was used for finding best parameters for each model
- SVM performed best, with accuracy of 88%



[Link to IBM Watson Notebook]

Results

- Exploratory data analysis using pandas, and SQL helped us to understand the data and to get some insights in it, for example:

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

```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) FROM SPACEXDATASET
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY COUNT(LANDING__OUTCOME) DESC;
```

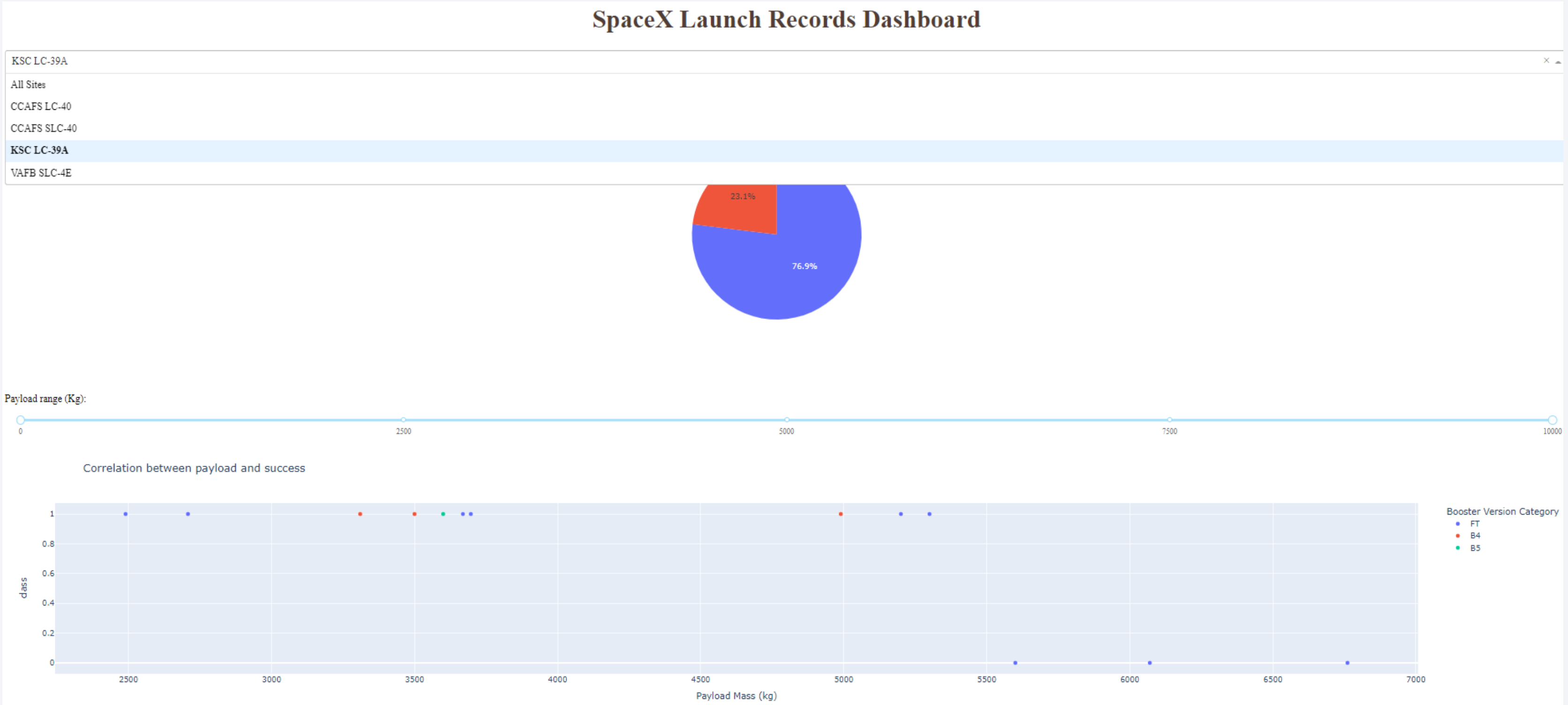
LANDING__OUTCOME	2
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

- Number of launches on sites:

CCAFS SLC 40	55
KSC LC 39A	22
VAFB SLC 4E	13

Results

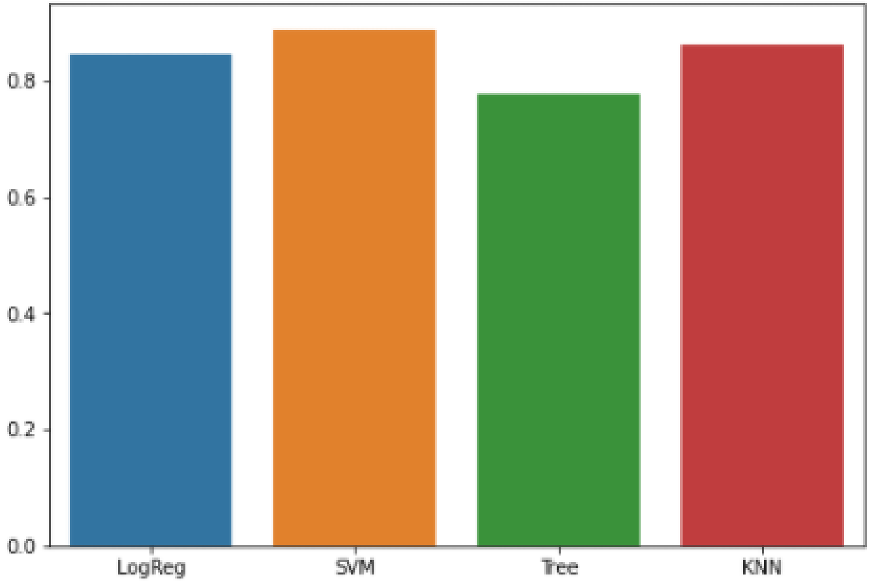
- Dash:
  - Interactive analysis to check which launchsite have highest number of success launches and to check the correlation between payload and success rate



Results

* Predictive analysis results

```
import seaborn as sns
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
method = ['LogReg', 'SVM', 'Tree', 'KNN']
score = [0.8464285714285713,0.8888888888888888,0.7777777777777778,0.8611111111111112]
ax = sns.barplot(x=method, y=score)
```

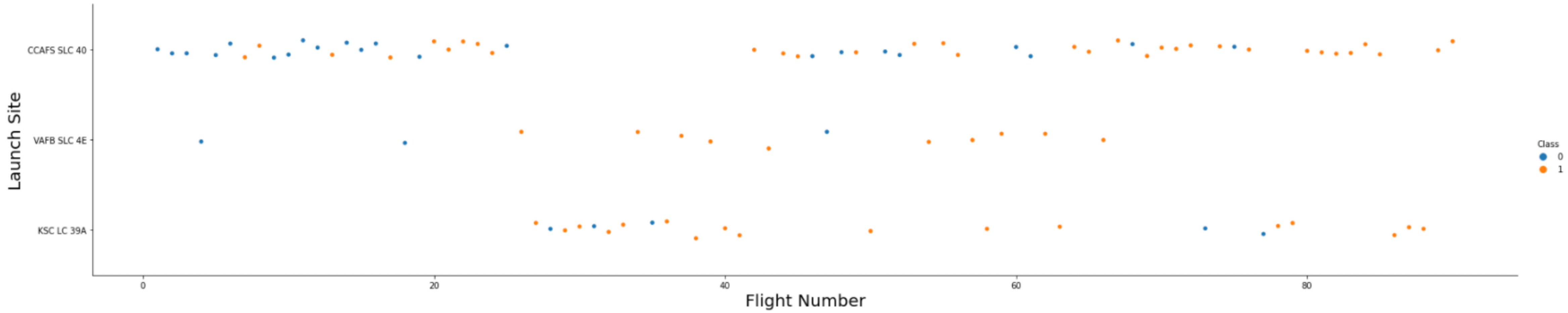


Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



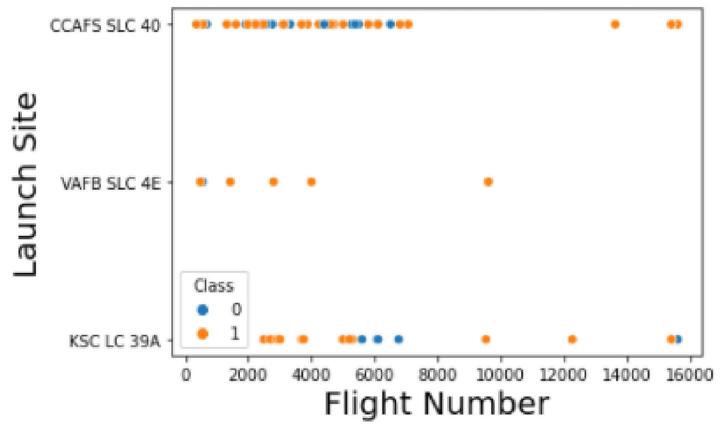
Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

At least 5 last landings on every site were successfull

The higher the flight number is, the higher chance to success is

Payload vs. Launch Site

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.scatterplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



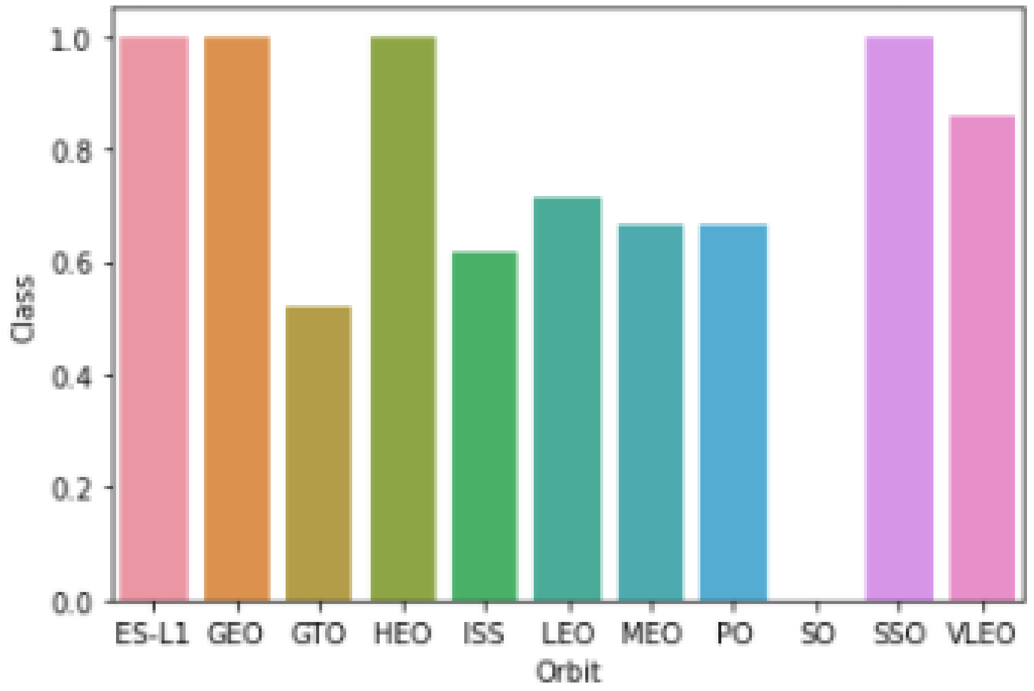
Now try to explain any patterns you found in the Payload Vs. Launch Site scatter point chart.

A lot of failures happened with payload close to 6000kg in KSC LC 39A

Low and high payloads are successful on CCAFS SLC 40

Almost all launches in VAFB SLC 4E are successful

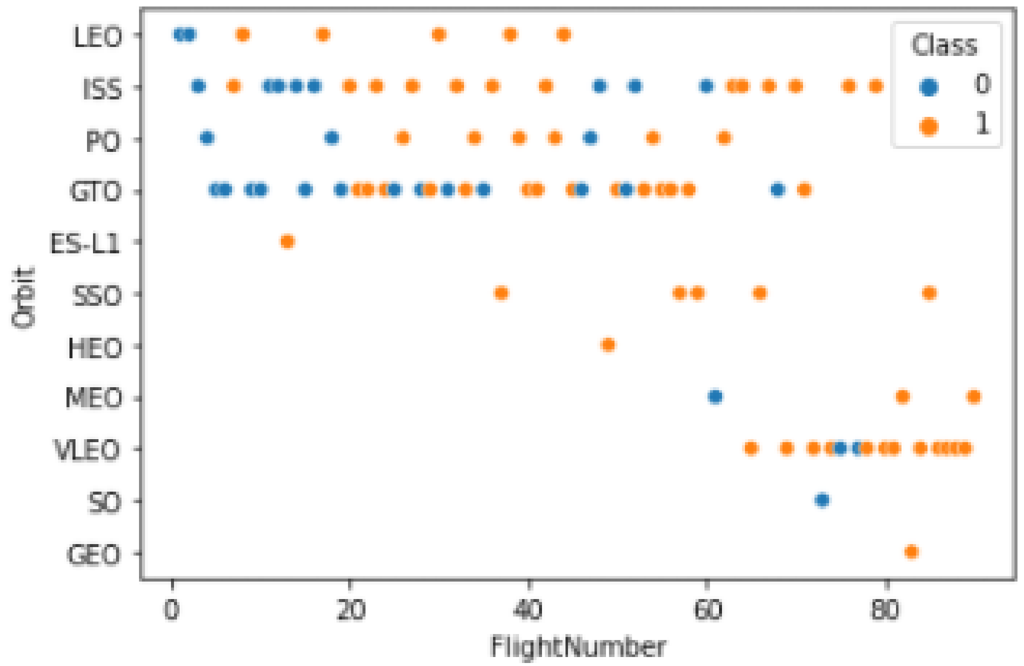
Success Rate vs. Orbit Type



Analyze the ploted bar chart try to find which orbits have high sucess rate.

ES-L1, GEO, HEO, and SSO have highest success rate

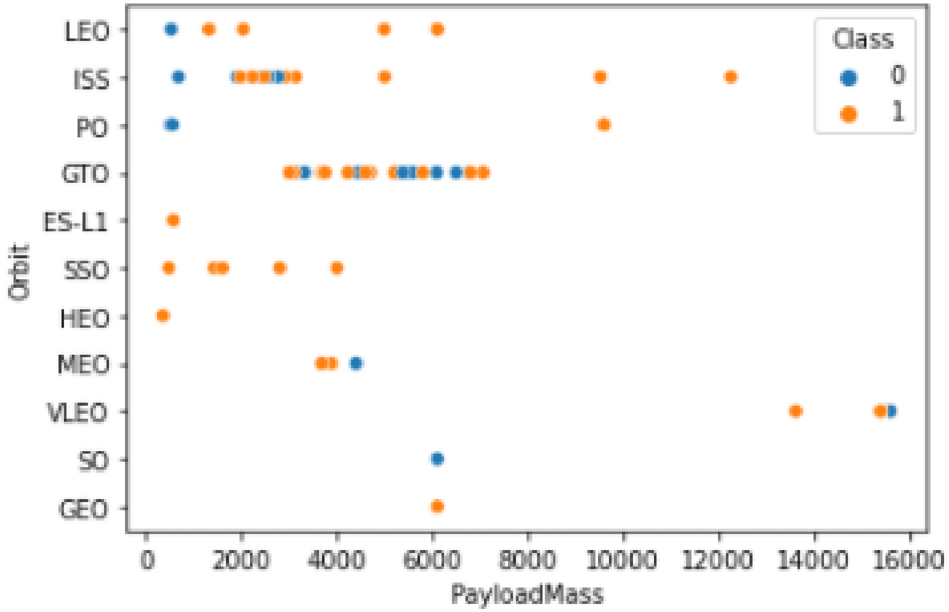
Flight Number vs. Orbit Type



You should see that in the LEO orbit the Success appears related to the number of flights;

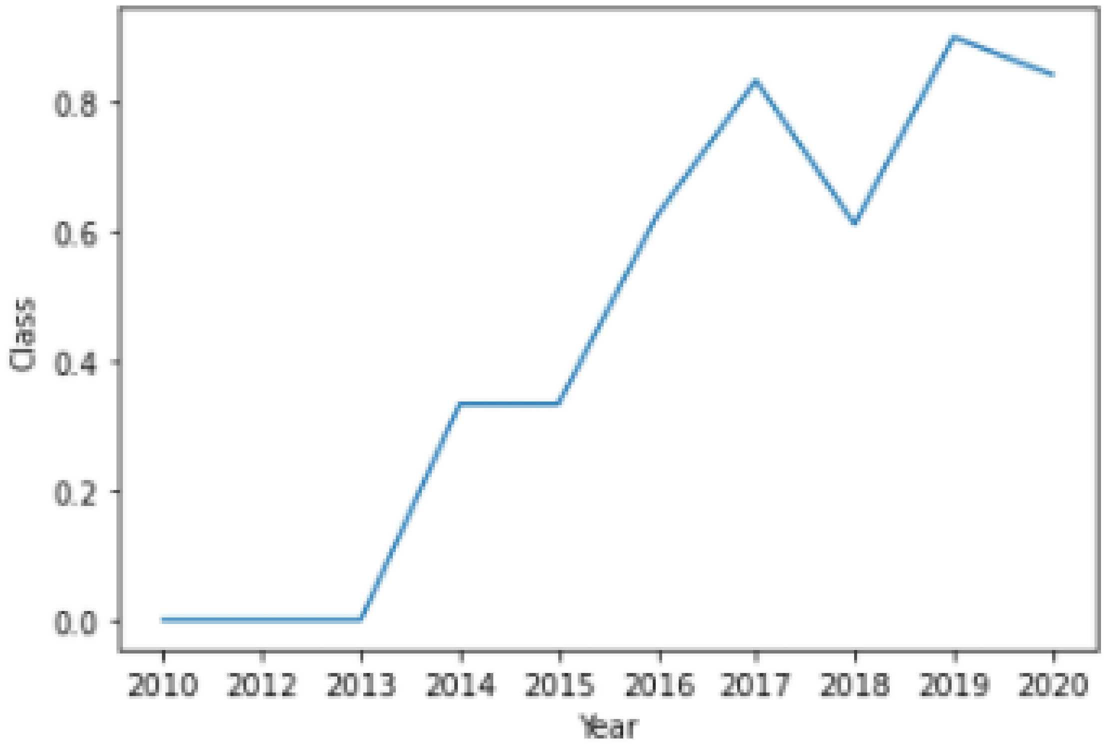
on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

Launch Success Yearly Trend



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

All Launch Site Names

```sql
SELECT DISTINCT LAUNCH_SITE FROM SPACEXDATASET
```

Result set 1

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

SELECT * FROM SPACEXDATASET WHERE LAUNCH_SITE LIKE('CCA%') LIMIT 5 Run time: 0.026 s

Result set 1

Find

DATE	TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere chee
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

```
SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXDATASET WHERE CUSTOMER ='NASA (CRS)'
```

Result set 1

1
45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

```
SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXDATASET WHERE BOOSTER_VERSION = 'F9 v1.1'
```

Result set 1

1
2928

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
SELECT MIN(DATE) FROM SPACEXDATASET WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

Result set 1

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
SELECT DISTINCT BOOSTER_VERSION FROM SPACEXDATASET WHERE LANDING__OUTCOME = 'Success (drone ship)' ...
```

Result set 1

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) FROM SPACEXDATASET GROUP BY LANDING__OUTCO..

Result set 1

LANDING__OUTCOME	2
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	22

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

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

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

SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXDATASET WHERE LANDING__OUTC... Run time: 0.010 s

Result set 1

Find

LANDING__OUTCOME	BOOSTER_VERSION	LAUNCH_SITE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
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

LANDING__OUTCOME	2
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 4

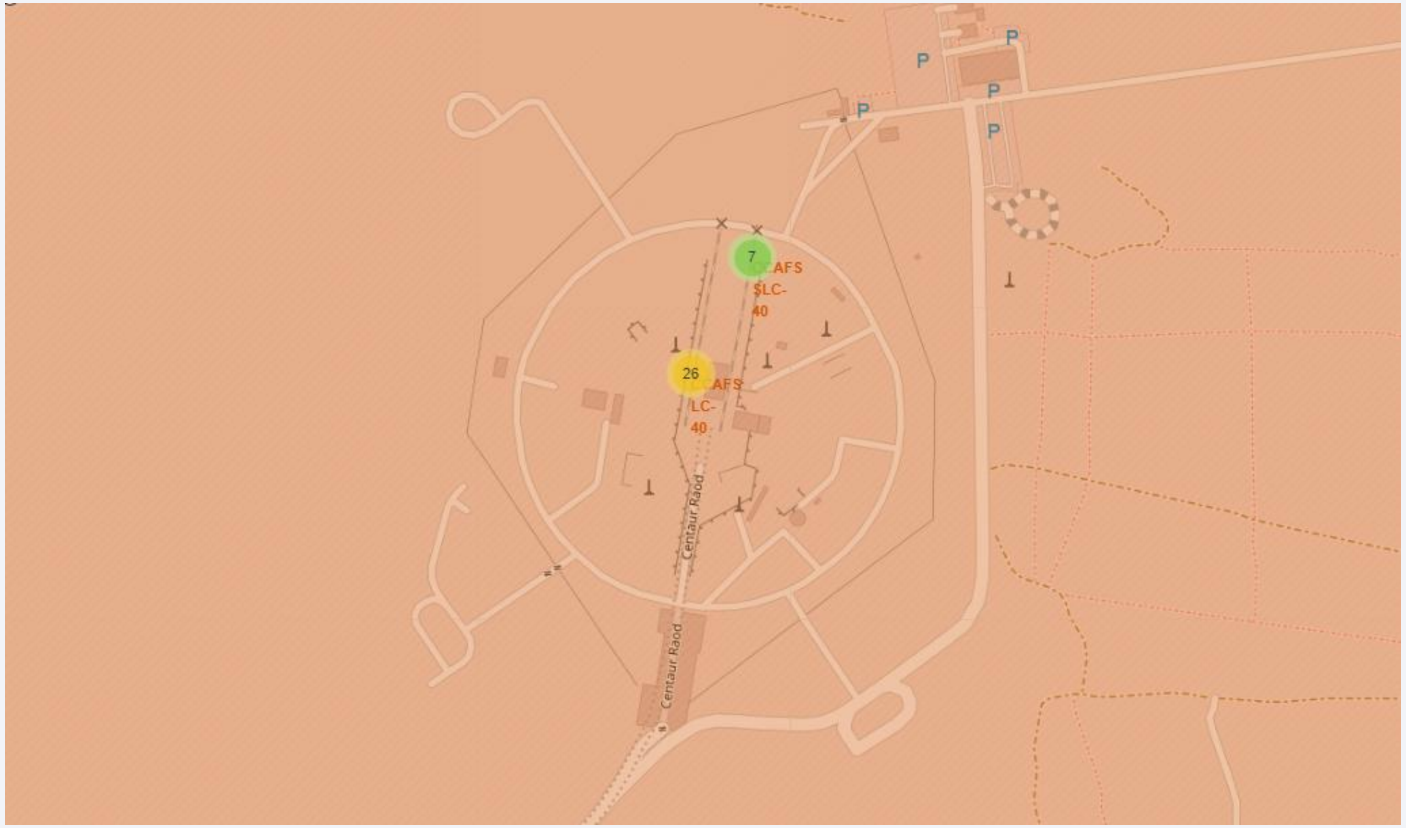
Launch Sites Proximities Analysis

Folium Launchsites map

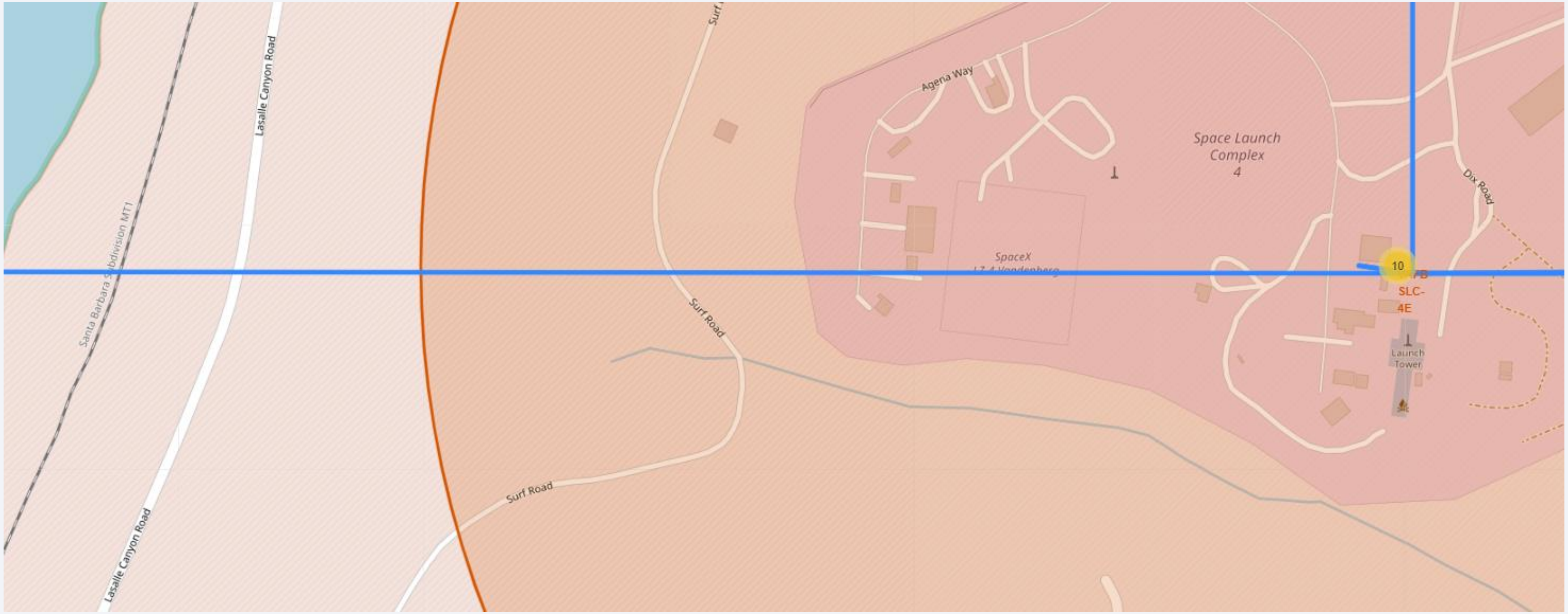
- Launchsites are based on coastline

Folium failed and successful launches marked

- Success vs failed launches on example launchpad



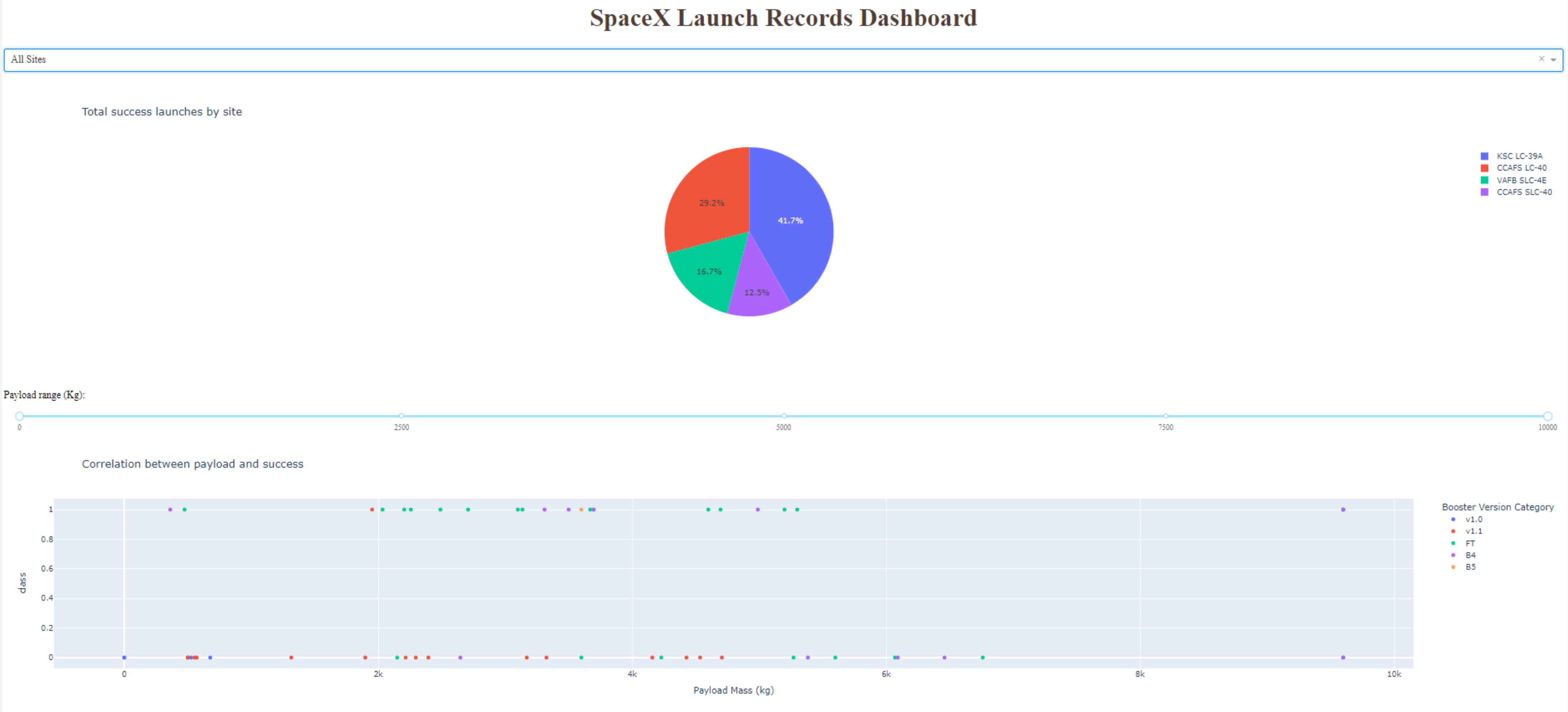
Folium – drawing lines



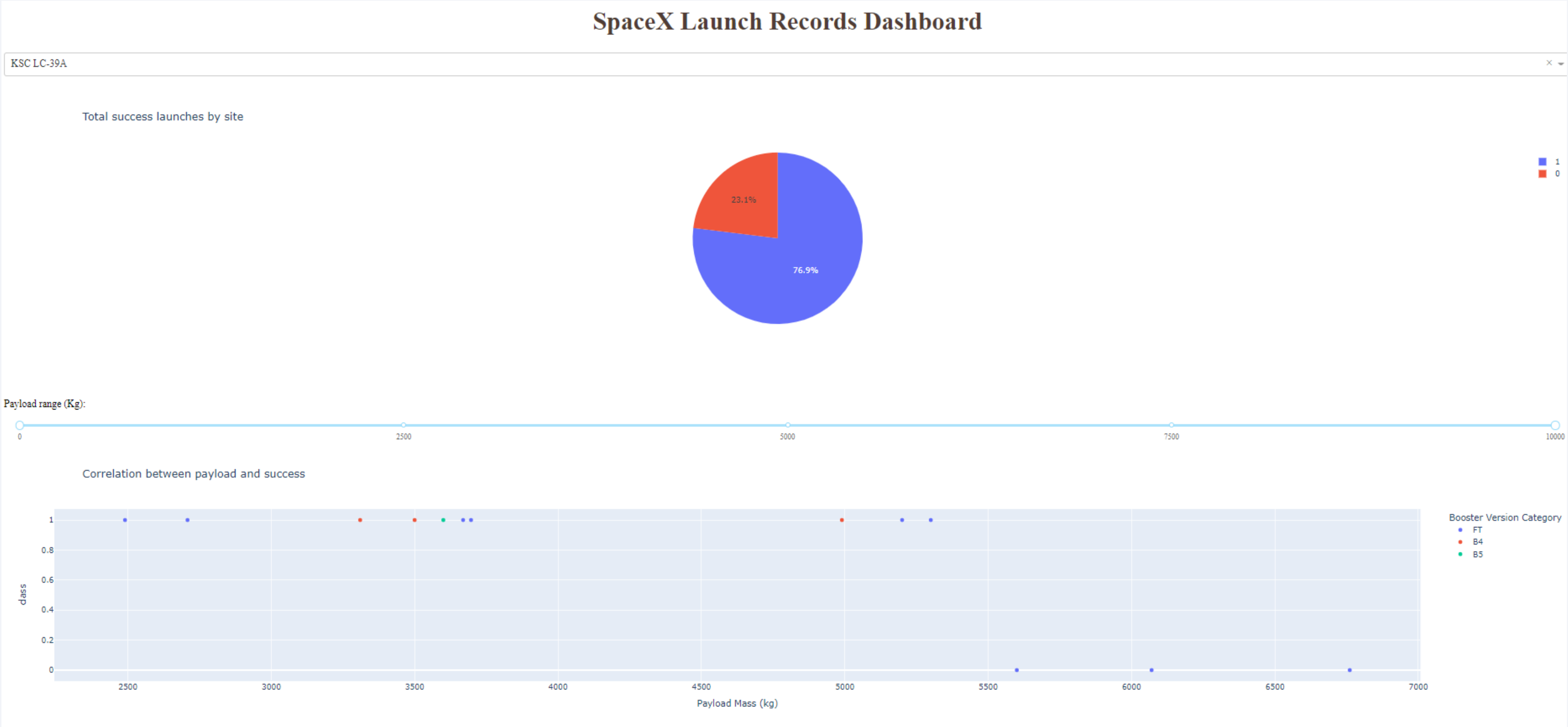
Section 5

Build a Dashboard with Plotly Dash

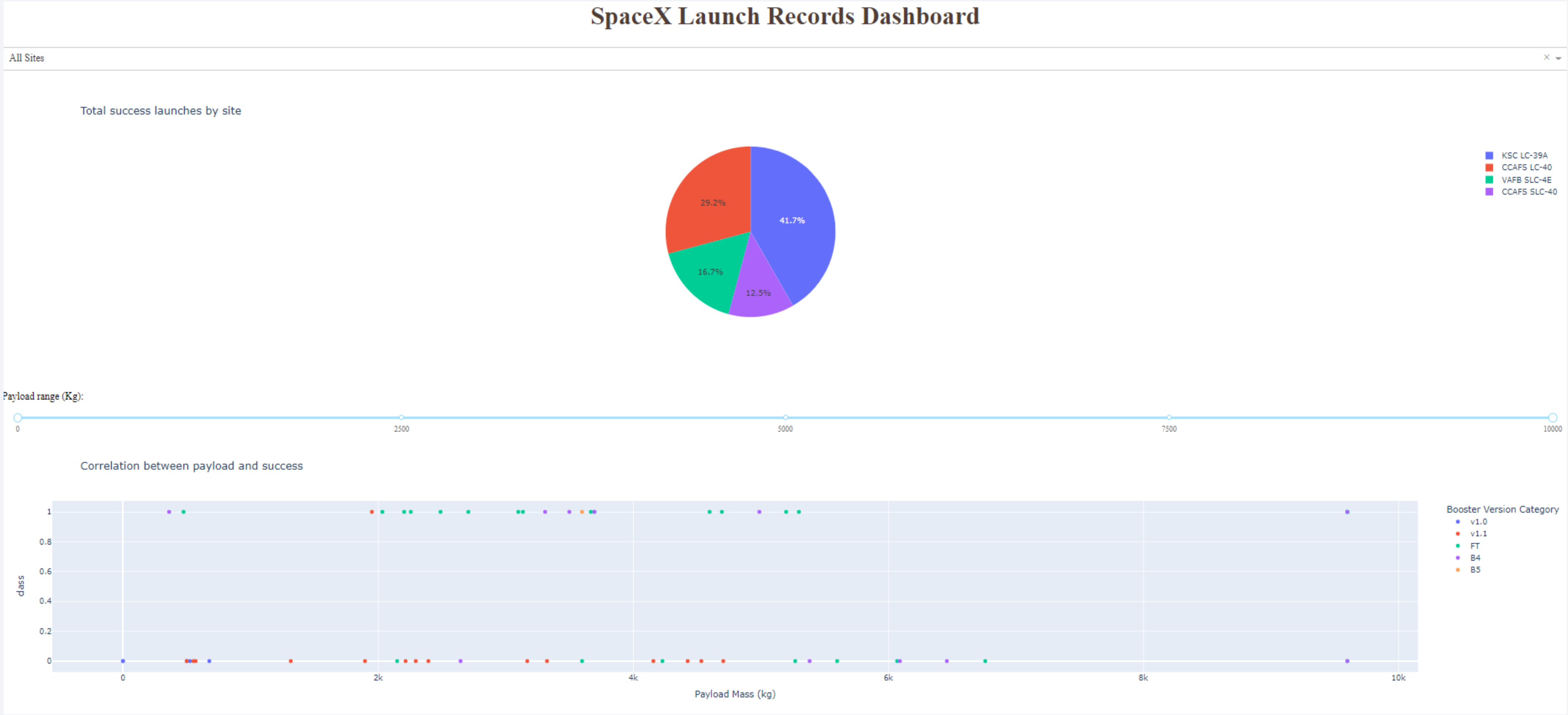
Dash – dashboard showing overall succes launches



Dash – most success launches site insights



Dash – payload vs launch outcome



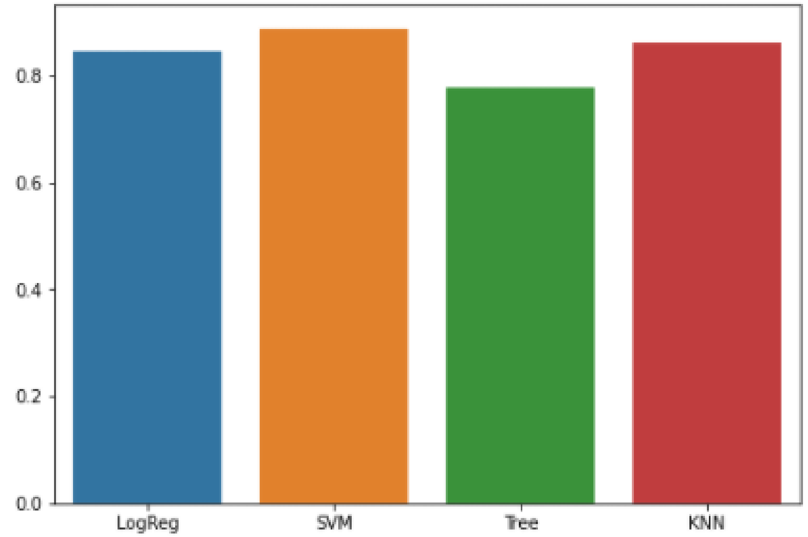
Section 6

Predictive Analysis (Classification)

Classification Accuracy

- The SVM model have highest accuracy

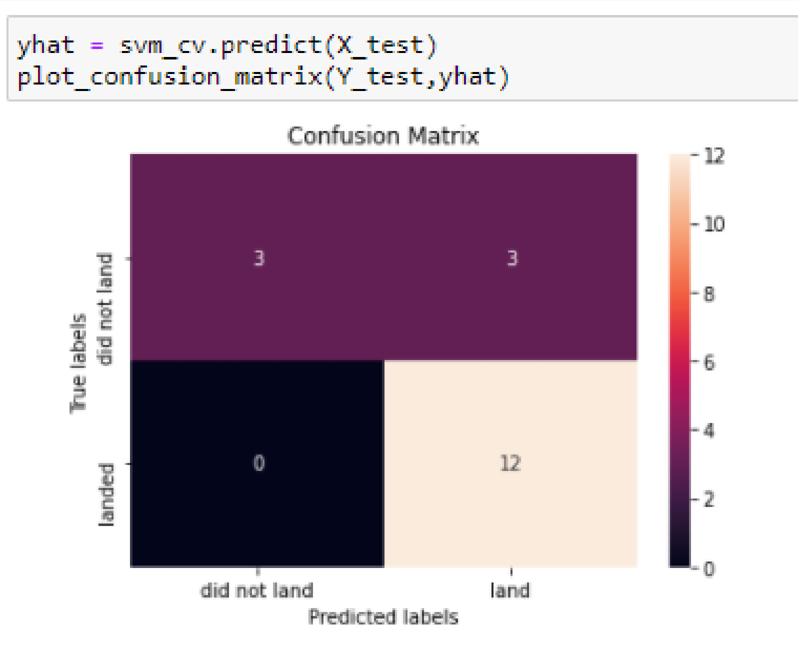
```python
import seaborn as sns
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
method = ['LogReg', 'SVM', 'Tree', 'KNN']
score = [0.8464285714285713,0.8888888888888888,0.7777777777777778,0.8611111111111112]
ax = sns.barplot(x=method, y=score)
```



Confusion Matrix

- Confusion matrix ofr SVM model:

We see that only 3 datapoints are not correctly classified



Conclusions

- Each model performed very good
- SVM have highest accuracy: 88%
- Decision tree classifier performed worst, with 0.77% accuracy
- GridSearchCV is a great tool for finding the best model parameters

Thank you!

Appendix

- All of the code is available on Github