IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Kazuhiro Nakano
Updated: 14/Nov/2021

Outline

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

Executive Summary

- **Summary of methodologies (all in Python)**
  - Data collection and Data wrangling by SpaceX REST API and Web scraping on Wikipedia page
  - Exploratory data analysis (EDA) using Matplotlib and SQL
  - Interactive visual analytics using Folium and Plotly Dash
  - Predictive analysis using scikit-learn classification models
- **Summary of all results**
  - Enough size of dataset was obtained from SpaceX API and Wikipedia page
  - EDA was done by visualizing the features of data by matplotlib + analyzing directly by SQL
  - Interactive Map and Dashboard to visualize data were created
  - The predictive models (LogReg/SVM/Tree/KNN) were built and the accuracies were confirmed

Introduction

- **Background of this project**
  - SpaceX advertises the launch of their rocket (Falcon9) costs 62M dollars, which is much cheaper than other providers (typ.165M dollars).
  - It is because SpaceX can reuse the first stage of their rocket. Therefore, the actual cost would depend on whether its first stage lands successfully and be reusable, or not.
- **Problems to be answered in this project**
  - “Will the first stage of Falcon9 rocket land successfully and be reusable?”
  - It is answered by machine learning model, whose INPUT is “Conditions of a rocket launch” and OUTPUT is binary “success or failure”

➔ Depending on this answer, a competitor company (SpaceY) will be able to estimate the actual cost of SpaceX’s Falcon9 launch and investigate the competitiveness.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API + Web scraping on Wikipedia page
- Perform data wrangling
  - Sampling valid data (Falcon9) ➔ Handling nulls (replace nan to the mean of its column)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic regression/SVM/Decision tree/K-Neighbors models built by scikit-learn
  - Hyperparameters of each model were tuned by GridsearchCV class in scikit-learn

Data Collection

- Data were collected in two different ways:
  - SpaceX REST API
  - Web scraping on SpaceX's Wiki page

➔ Key phrases and flowcharts of these Data Collection processes are described in following pages

Data Collection – SpaceX API

- Request SpaceX launch data using GET request to “SpaceX REST API”
- Convert the json data in the response into Pandas Dataframe
- Take a subset of the dataframe keeping only the necessary information
- GitHub URL: https://github.com/kazu-nak/cousera-capstone/blob/master/Week1_Data%20Collection%20API.ipynb

Request launch data to SpaceX REST API

response = requests.get(‘https://api.spacexdata.com/v4/launches/past’)

↓

Convert the obtained json data into Pandas dataframe

data = pd.json_normalize(response.json())

↓

Take the subset of the dataframe keeping only necessary info

data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

↓

Convert the format of some data columns to make it easy to handle

(from string format to datetime format, from 1-length list object to a value, …)

Data Collection – Scraping

- 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/kazu-nak/cousera-capstone/blob/master/Week1_Data%20Collection%20with%20Web%20Scraping.ipynb

Request the Falcon9 Launch Wiki page from its URL

response = requests.get(URL*) ➔ soup = BeautifulSoup(response.text)

Extract all column/variable names from the HTML table header

html_tables = soup.find_all('table’)

Create a data frame by parsing the launch HTML tables by “for loops”

for table in soup.find_all('table',"wikitable plainrowheaders collapsible"): ➔ Extract each table

for rows in table.find_all("tr"): ➔ Get each row

*https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork26802033-2021-01-01

Data Wrangling

- Request and parse the SpaceX launch data using the GET request
- Filter the dataframe to only include Falcon 9 launches
- Dealing with Missing Values
- GitHub URL: https://github.com/kazu-nak/cousera-capstone/blob/master/Week1_Data%20Collection%20API.ipynb

Convert “IDs” in the dataframe into “actual data” by using APIs

For example, "https://api.spacexdata.com/v4/rockets/XXX" gives us the rocket’s name from ID (XXX)

↓

Sampling necessary data (keep only Falcon9’s data)

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

↓

Replace nulls with the mean value of its columns

```
PM_mean = data_falcon9['PayloadMass'].mean()
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PM_mean)
```

EDA with Data Visualization

- **Scatter charts (overlaying the outcome of the launch)** to observe the relationships of:
  - Flight number vs. Payload mass
  - Flight number vs Launch site
  - Flight number vs Orbit type
  - Payload vs Orbit type
  - Launch sites vs Payload mass
- **Bar chart** to compare the success rate per orbit types
- **Line chart** to observe the yearly trend of the success rate
- GitHub URL:
  https://github.com/kazu-nak/cousera-capstone/commit/e939c88e919ad45de20675d98648b910875591be

EDA with SQL

- SQL queries performed:
  - SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL
  - SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
  - SELECT SUM(PAYLOAD_MASS__kg_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
  - SELECT AVG(PAYLOAD_MASS__kg_) FROM SPACEXTBL WHERE BOOSTER_VERSION LIKE 'F9 v1.1%'
  - SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING__OUTCOME='Success (ground pad)'
  - SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE (LANDING__OUTCOME='Success (drone ship)') AND (PAYLOAD_MASS__kg_ BETWEEN 4000 AND 6000)
  - SELECT COUNT(MISSION_OUTCOME) FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%'
  - SELECT COUNT(MISSION_OUTCOME) FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Failure%'
  - SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_=(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
  - SELECT DATE, LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE (LANDING__OUTCOME='Failure (drone ship)') AND (DATE BETWEEN '2015-01-01' AND '2015-12-31')
  - SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS count FROM SPACEXTBL WHERE (DATE BETWEEN '2010-06-04' AND '2017-03-20') GROUP BY LANDING__OUTCOME ORDER BY count DESC
- GitHub URL: https://github.com/kazu-nak/cousera-capstone/blob/master/Week2_EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

Map objects added to the folium map:

- **Markers** with text icons
  - To indicate the name of launch sites on the map
- **Circles** whose centers are launch sites
  - To visualize the locations of launch sites clearly on the map
- **Lines** between a launch site and some proximity landmarks (coastline, city, railway, highway)
  - To visualize and investigate the actual distances of landmarks from a launch site
- GitHub URL: https://github.com/kazu-nak/cousera-capstone/blob/master/Week3_Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

Plots/graphs and interactions in the dashboard:

- **Pie chart**
    - To visualize the Total success launches
    - Interact with the Dropdown list to change the launch site to be plotted
- **Scatter chart**
    - To visualize the relationship between Payload Mass (kg) and outcome of the launch (Success or Failure)
    - Interact with the Dropdown list to change the launche site to be plotted
    - Interact with RangeSlider to change the x-axis range of graph

- GitHub URL:
  https://github.com/kazu-nak/cousera-capstone/blob/master/Week3_Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

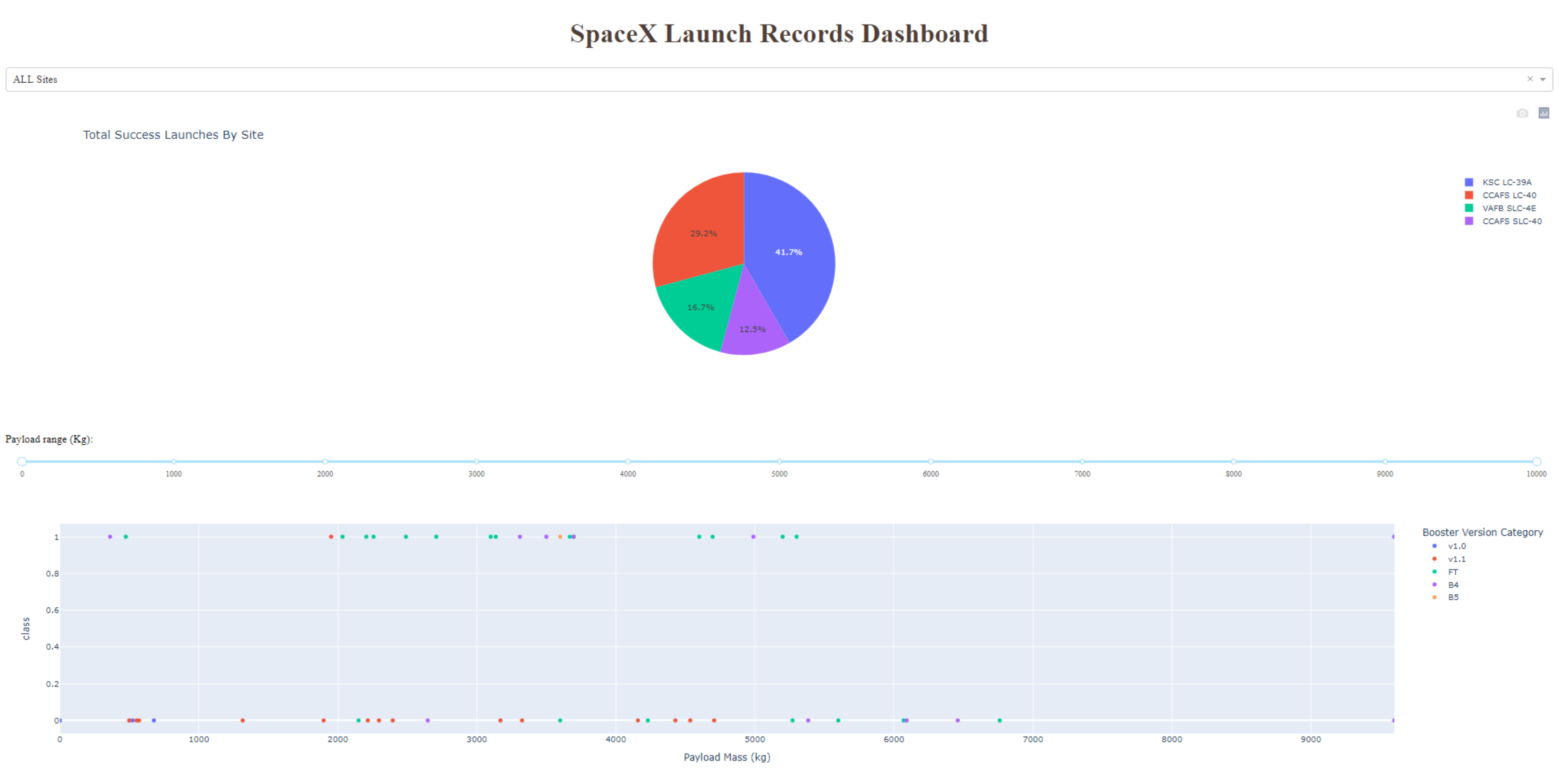
Predictive Analysis (Classification)

- **Build four types of models by using sklearn**
  - Logistic Regression classification model
  - Support Vector Machine classification model
  - Decision Tree classification model
  - K Nearest Neighbors classification model
- **Split the dataset into train and test by using sklearn (with size of test data 20%)**
- **Fit models by training data, changing the hyperparameters iteratively**
  - using sklearn's GridsearchCV class
- **Predict the accuracies on test data for every model**
- **Choose the model which has the best accuracy**
- GitHub URL: https://github.com/kazu-nak/cousera-capstone/blob/master/Week4_Machine%20Learning%20Prediction.ipynb

Results

- Exploratory data analysis results
  - Orbit types affected to success rate clearly
  - Yearly trend of success rate is increasing
  - Payload mass (kg) has correlation with success rates
- Interactive analytics demo in screenshots
  - Shown in the next slide
- Predictive analysis results
  - The outcome of the launch (success or failure) can be predicted by machine learning model (Logistic Regression, SVM, KNN) with the accuracy 0.83
  - Those models were trained by 72 data, and tested on 18 data

Results



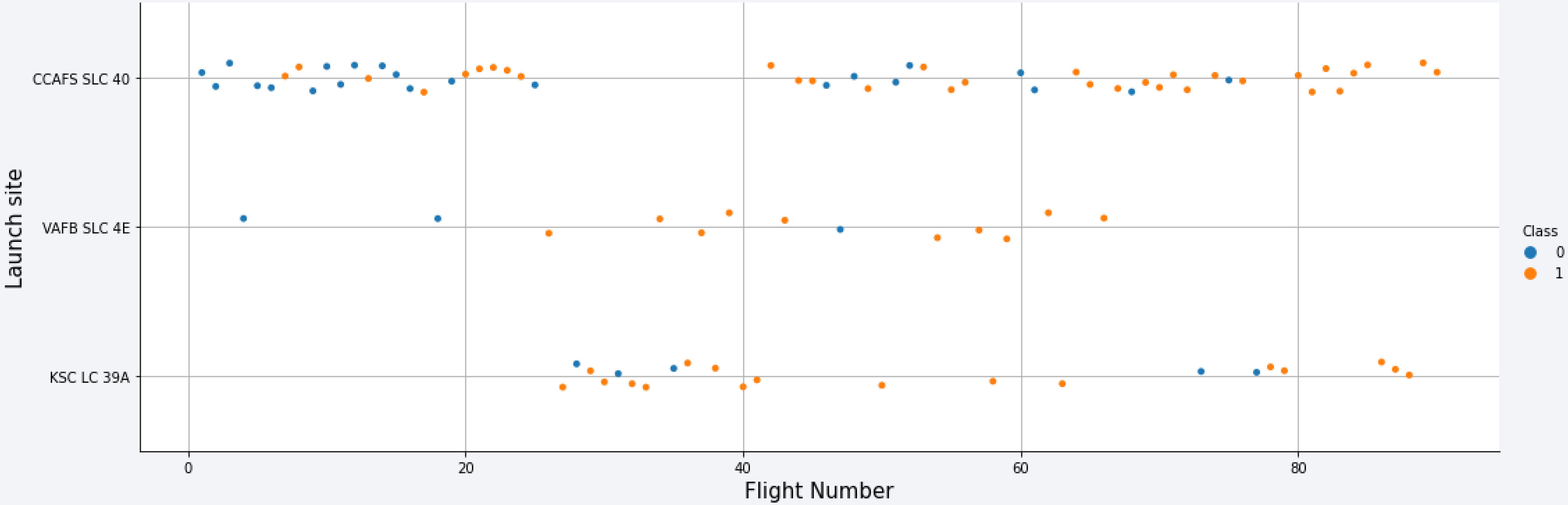
SpaceX Launch Records Dashboard
ALL Sites
Total Success Launches By Site
KSC LC-39A
CCAFS LC-40
VAFB SLC-4E
CCAFS SLC-40
41.7%
29.2%
16.7%
12.5%
Payload range (Kg):
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
Booster Version Category
v1.0
v1.1
FT
B4
B5
class
Payload Mass (kg)

Section 2

Insights drawn from EDA

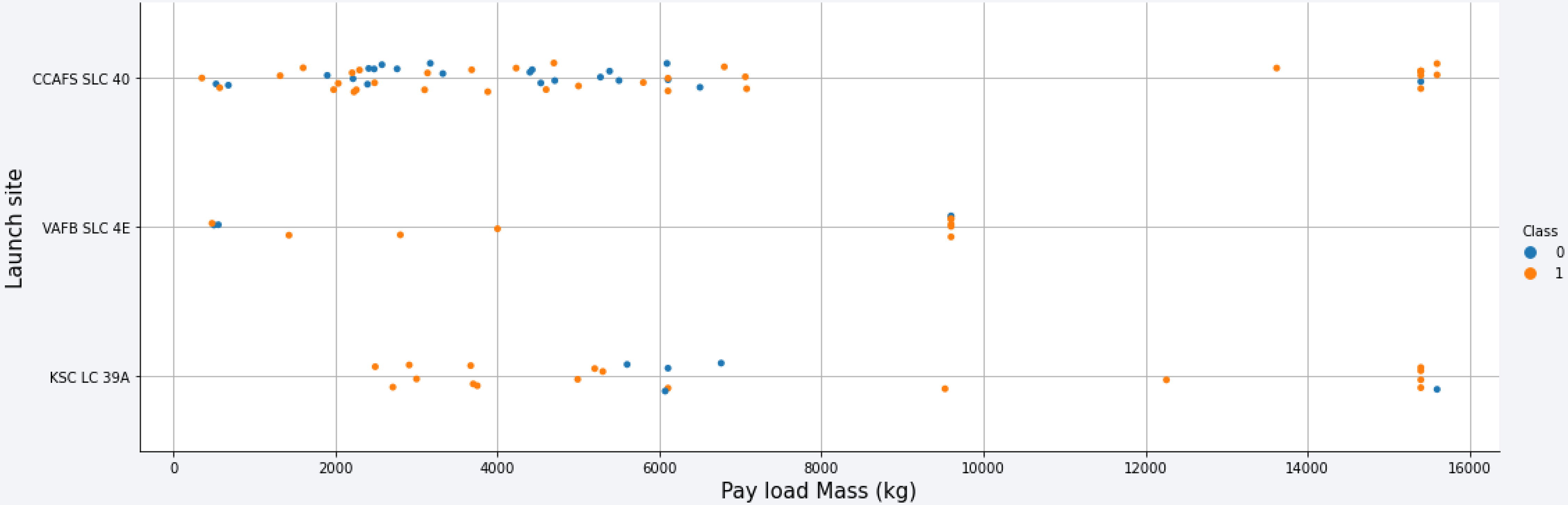
Flight Number vs. Launch Site

- From Flight number 1 to 20, the successful rate is very low.
  Most of those first 20 flights were launched from **CCAFS SLC 40**
- On the other hand, **KSC LC 39A** started to be used after 26th flight.



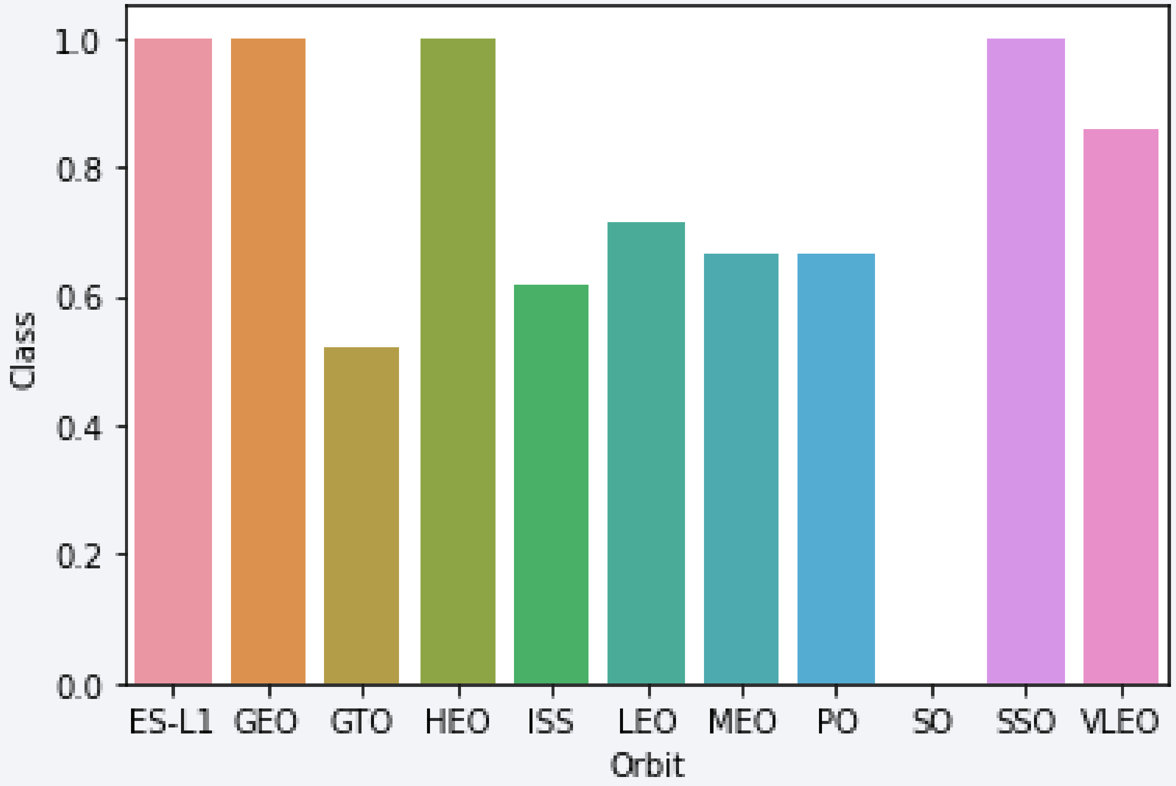
Payload vs. Launch Site

- Most of the launches were with payload mass under 8,000 kg
- For **VAFB-SLC,** there are no rockets launched for heavy payload mass greater than 10,000 kg



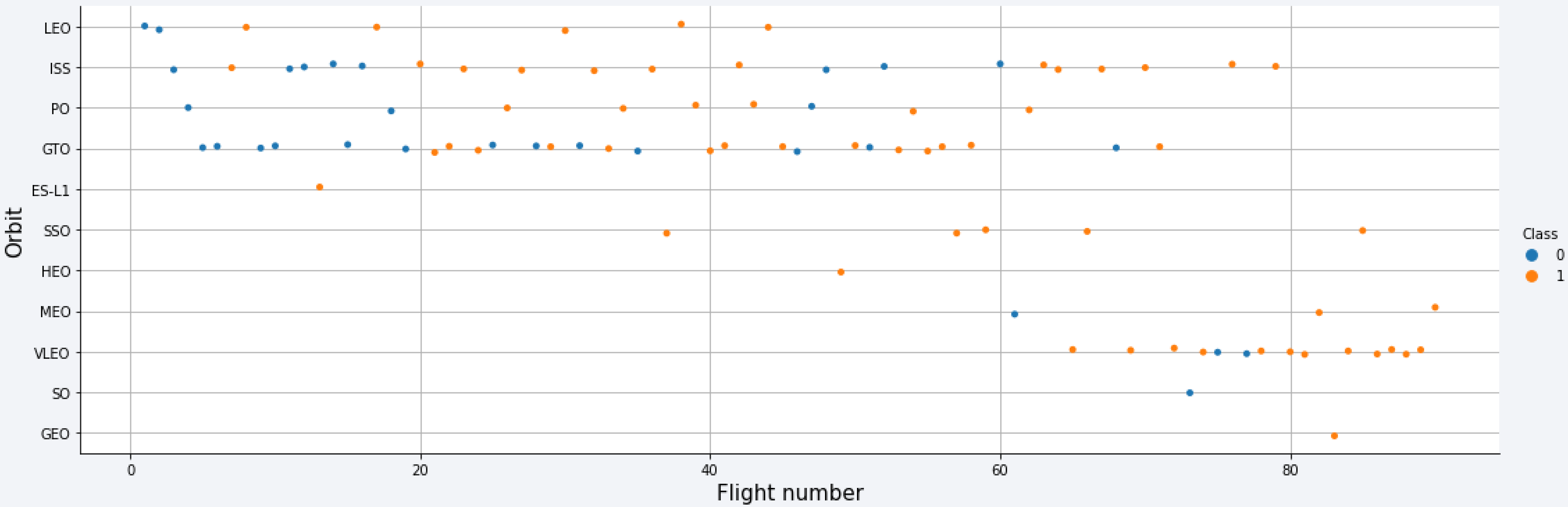
Success Rate vs. Orbit Type

- Success Rate strongly depends on the Orbit Type
- Orbit Type **SO** looks the most difficult (success rate=0%) whereas **ES-L1/GEO/HEO/SSO**’s success rate=100%



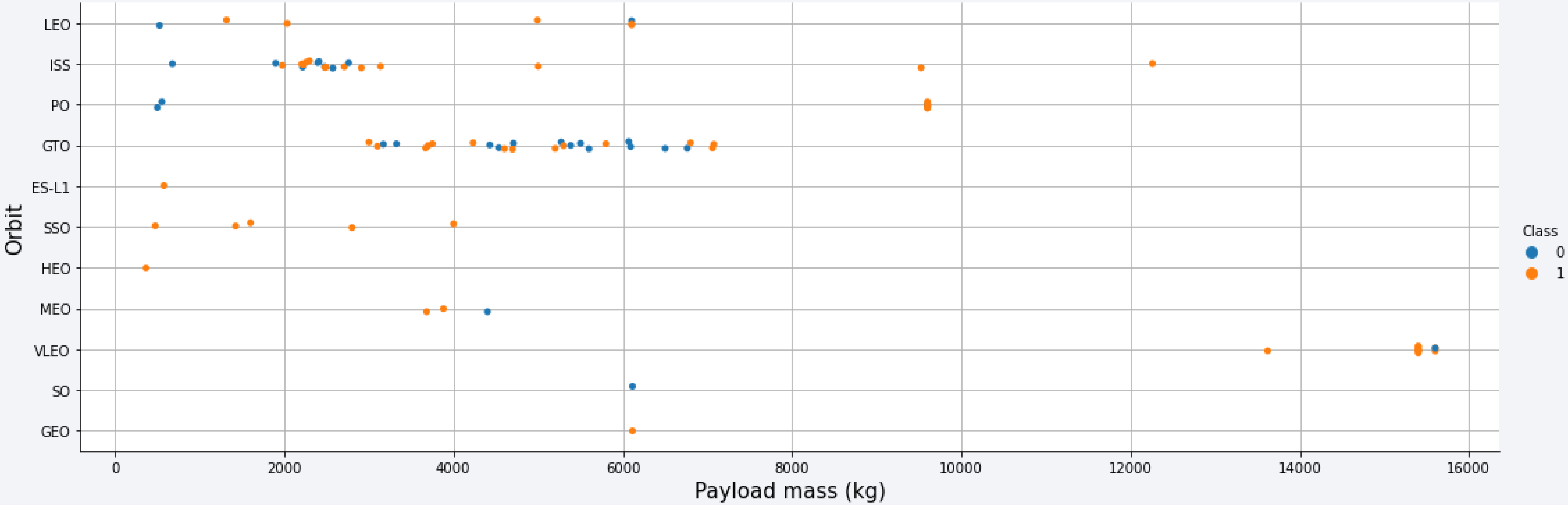
Flight Number vs. Orbit Type

- Total number of flights differs by orbit type very much.
- Orbit Type **ES-L1/SO/GEO** happened just once whereas **LEO/ISS/PO/GTO/VLEO** happened a lot of times.



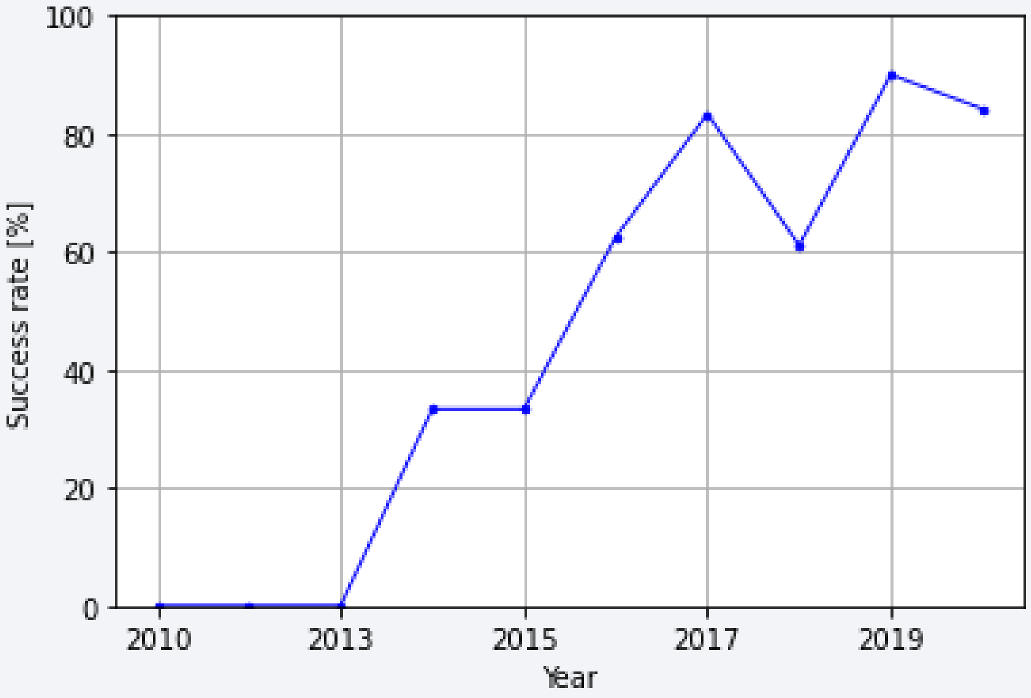
Payload vs. Orbit Type

- Most of the payloads ranged less than 8,000 kg
- Orbit type **ES-L1** carried only very light payload (less than 1,000 kg)
- On the other hand, Orbit type **VLEO** carried only heavier payload than 13,000kg



Launch Success Yearly Trend

- Success rate tends to be increased year by year.
- From 0% (2010) to more than 80% (2020)
- In 2018, unexpected decrease of success rate compared to 2017.



All Launch Site Names

- There are four unique launch sites
  ➔ CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, VAFB SLC-4E
- DISTINCT statement was used extract unique elements

```
In [7]: %%sql
SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;

 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

Out[7]:

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

Launch Site Names Begin with 'CCA'

- 5 records are displayed in the screenshot below
- Wildcard character '%' and LIKE operator were used to specify the launch sites beginning with 'CCA'

In [11]:

```
%%sql
SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

Out[11]:

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 45,596 kg
- SUM statement and WHERE clause were used to calculate the sum of payloads from NASA (CRS)

```
In [36]: %%sql
SELECT SUM(PAYLOAD_MASS__kg_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'

 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.

Out[36]:
```

1
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 was 2,534 kg
- AVG statement and WHERE clause were used to calculate the average of payload mass by F9 v1.1
- Here, to take all the variants of F9 v1.1 into account, (for example, F9 v1.1 B1012), LIKE operator was used.

```
In [40]: %%sql
SELECT AVG(PAYLOAD_MASS__kg_) FROM SPACEXTBL WHERE BOOSTER_VERSION LIKE 'F9 v1.1%'

 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

Out[40]:

1
2534

First Successful Ground Landing Date

- The dates of the first successful landing outcome on ground pad was 2015-12-22
- MIN statement was used for DATE columns in order to pick up the first date in the table

In [45]:
```
%%sql
SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING__OUTCOME='Success (ground pad)'
```

```
 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

Out[45]:

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List of the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
  ➔ F9 FT B1022 / F9 FT B1026 / F9 FT B1021.2 / F9 FT B1031.2
- BETWEEN operator was used to specify the range of the payload

```
In [51]: %%sql
SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE (LANDING__OUTCOME='Success (drone ship)') AND (PAYLOAD_MASS__kg_ BETWEEN 4000 AND 6000)

 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.

Out[51]:
```

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

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failure mission outcomes
  ➔ 100 successes and 1 failure in total 101 missions
- LIKE operator was used to take all Success variants in the MISSION_OUTCOME into account

```
In [72]: %%sql
SELECT COUNT(MISSION_OUTCOME) FROM SPACEXTBL;

 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.

Out[72]: 1
         101
```

```
In [73]: %%sql
SELECT COUNT(MISSION_OUTCOME) FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%';

 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.

Out[73]: 1
         100
```

```
In [74]: %%sql
SELECT COUNT(MISSION_OUTCOME) FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Failure%';

 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.

Out[74]: 1
         1
```

Boosters Carried Maximum Payload

- List of the names of the booster which have carried the maximum payload mass
  ➔ Shown in the screenshot below
- Subquery was used for this complex querying as described in the screenshot below

```
In [80]: %%sql
SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_=(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)

 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

Out[80]:

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 of the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  ➔ Shown in the screenshot below

- BETWEEN operator was used with datetime format to limit the range of dates

In [94]:
```
%%sql
SELECT DATE, LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE (LANDING__OUTCOME='Failure (drone ship)') AND (DATE BETWEEN '2015-01-01' AND '2015-12-31')
```

```
 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

Out[94]:

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

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

- Rank of the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order
  ➔ Shown in the screenshot below
- GROUP BY statement was used to analyze the data per Landing Outcomes

In [107]:
```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS count FROM SPACEXTBL WHERE (DATE BETWEEN '2010-06-04' AND '2017-03-20') GROUP BY LANDING__OUTCOME ORDER BY count DESC
```

```
 * ibm_db_sa://wfr03162:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

Out[107]:

landing__outcome	COUNT
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

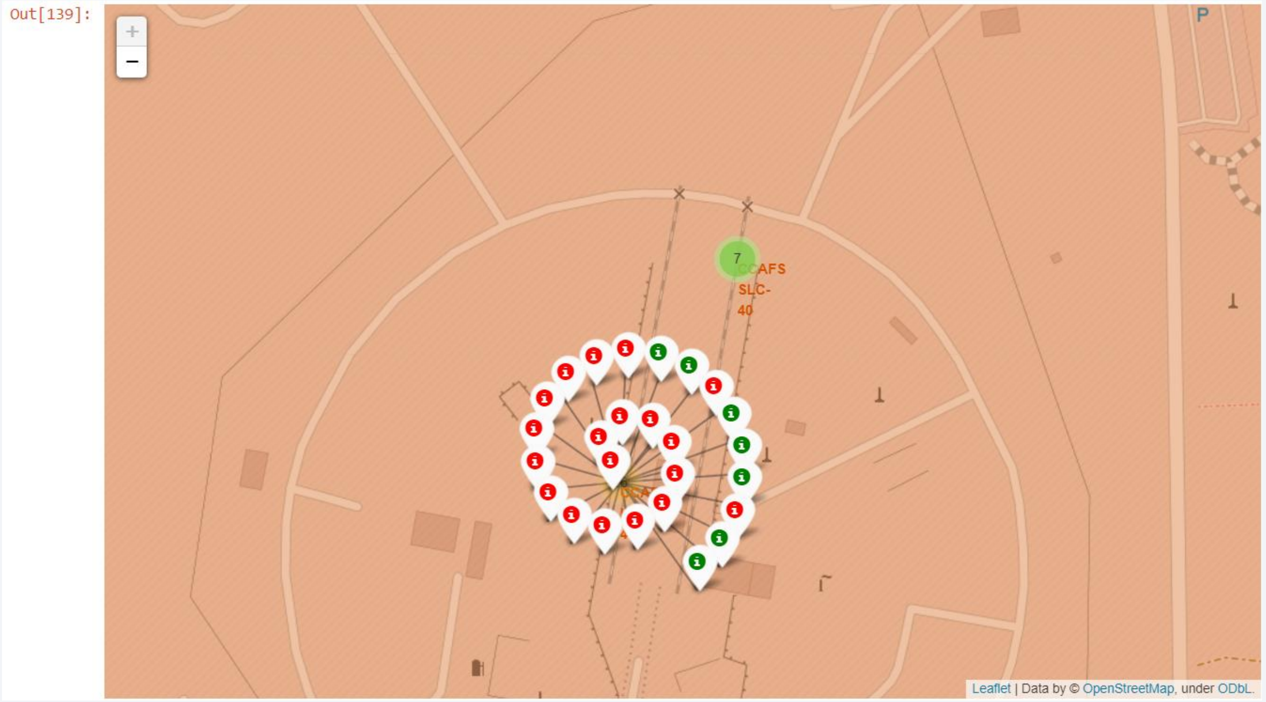
Launch Sites Proximities Analysis

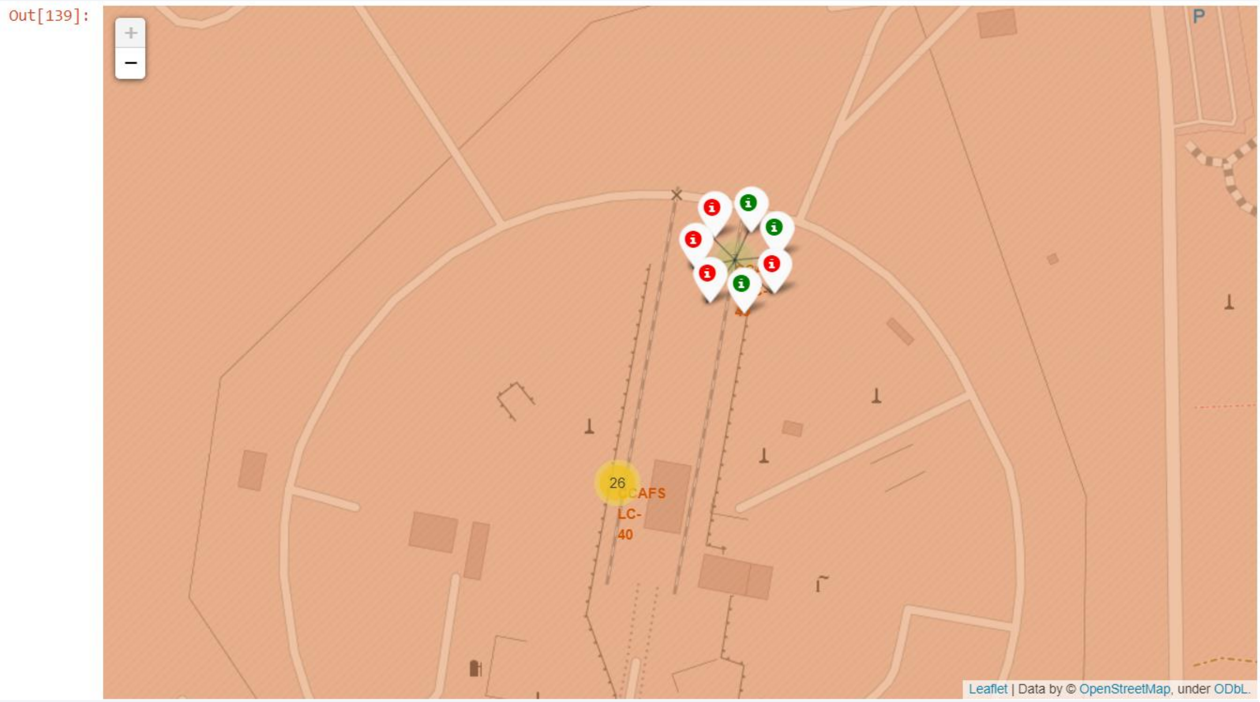
All launch sites' location markers on a global map

- All launch sites locate close to coastlines
- VAFB SLC-4E is on west coast, whereas other launch sites are on east coast

Color-labeled launch outcomes on the map

- Launch outcome can be clearly shown by icon colors at its exact location on the map
- Total number of flights, success rate of each launch site can be understood by clicking its location on the map

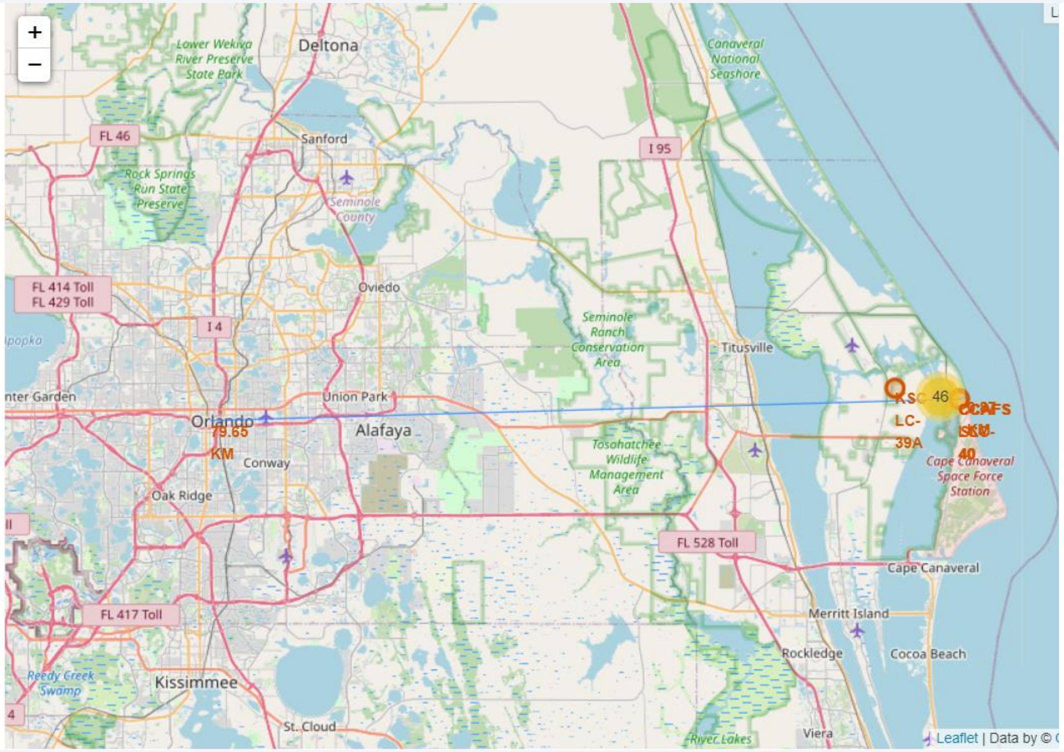




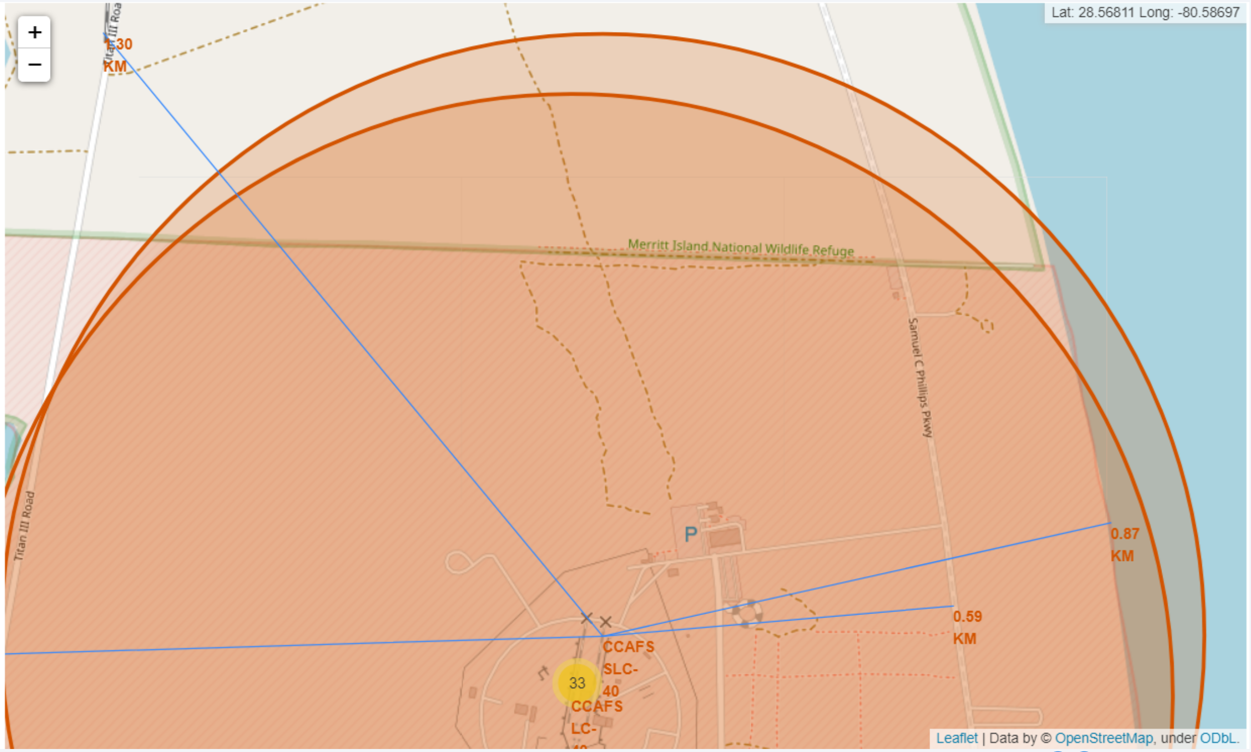
Distance from launch site to proximities

- Launch site is distanced <u>more than 50km from a big city zone</u>, whereas <u>some other proximities (coastline, railway, highway) are located closely (almost within 1km)</u>
- Example Launch site in the screenshots below = **CCAFS SLC-40**

Out[145]:



Out[147]:

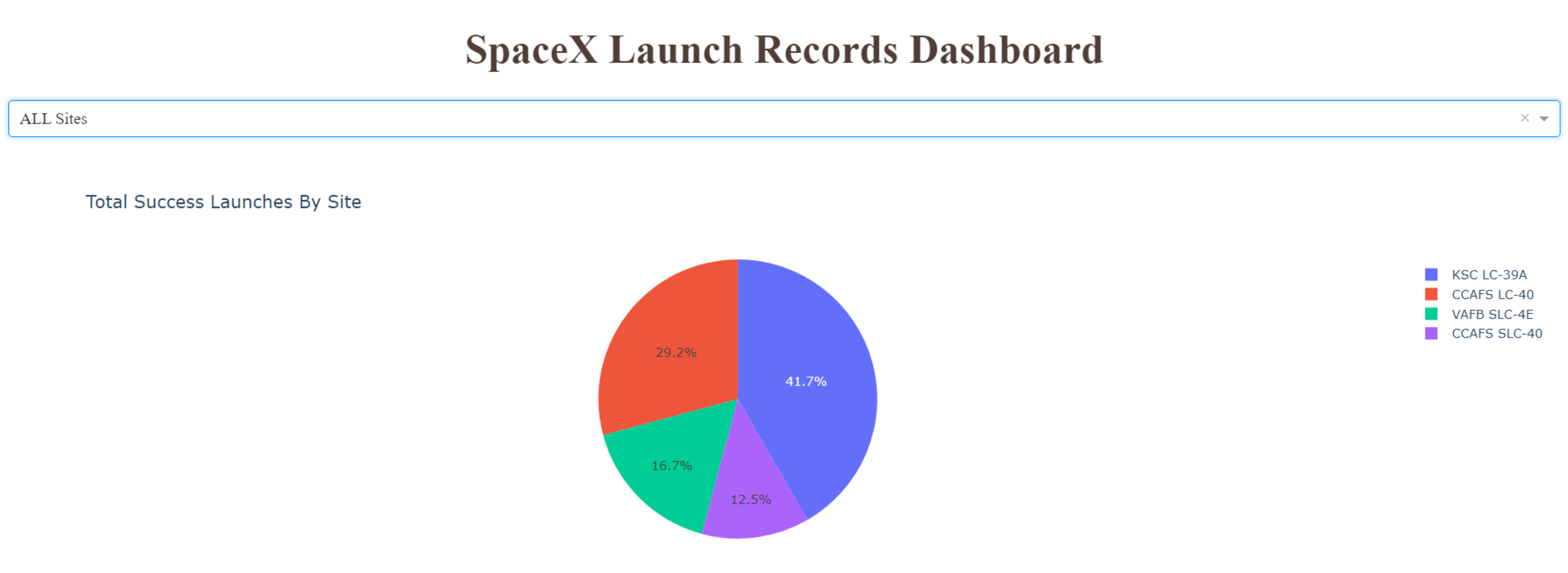


Section 5

Build a Dashboard with Plotly Dash

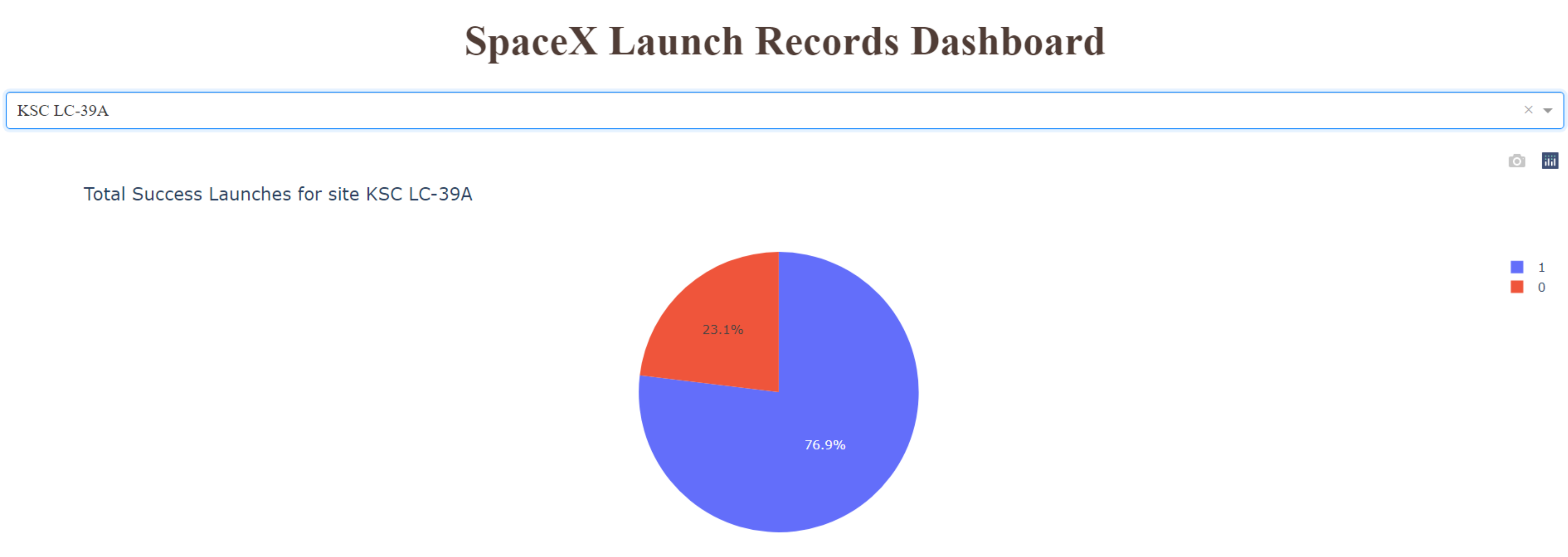
Pie chart - Launch success count for all sites

- More than 40% of the successful launch were from **KSC LC-39A**
- **CCAFS SLC-40** has the least numbers of the successful launches



Pie chart – The Launch site with highest launch success ratio

- **KSC LC-39A** has the highest launch success ratio
- The success ratio was **76.9%**

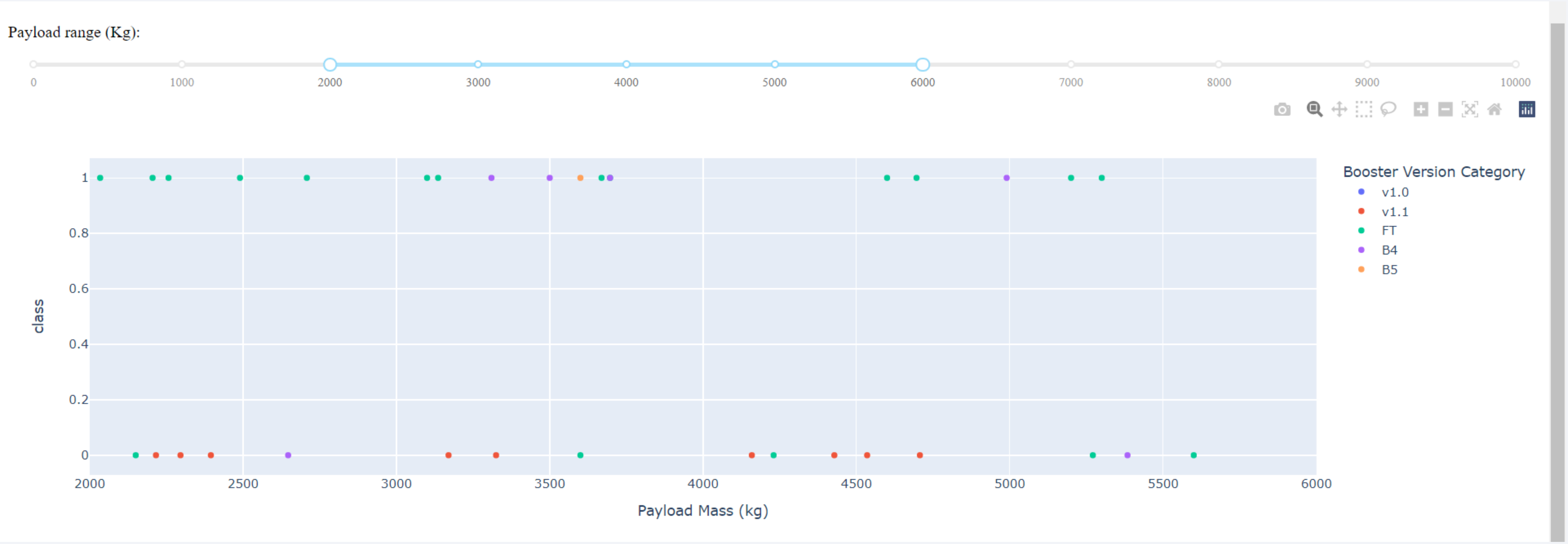


Scatter plot - Payload vs. Launch Outcome scatter plot for all sites

- For Payload mass **over 6,000kg**, the success launch ratio tends to be quite low
- Successful launches are mainly mapped in Payload's **range 2,000 - 6,000 kg**

Scatter plot - Payload vs. Launch Outcome scatter plot for all sites

- For Payload mass **over 6,000kg**, the success launch ratio tends to be quite low
- Successful launches are mainly mapped in Payload's **range 2,000 - 6,000 kg**



Section 6

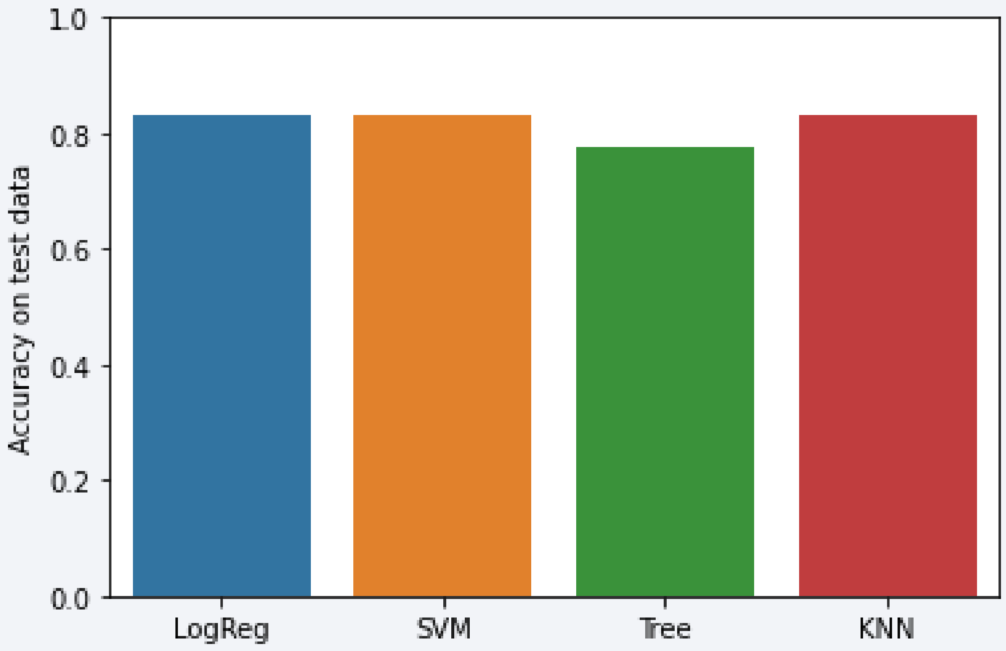
Predictive Analysis (Classification)

Classification Accuracy

- The accuracies for all built classification models in bar chart

➔Logistic Regression, SVM and KNN models showed the same accuracy **0.83**

➔Whereas Decision Tree model showed less accuracy **0.78**



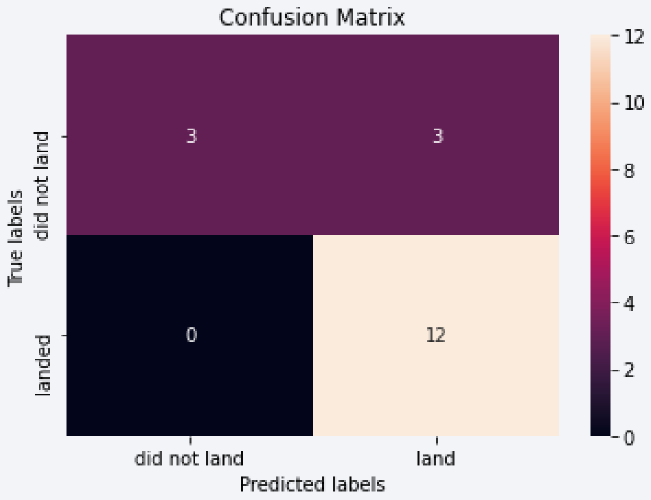
Confusion Matrix

- Logistic Regression, SVM and KNN models resulted the same confusion matrix.
- Three "false positive" cases were observed in every model.
- No "false negative" was observed.

Logistic regression

SVM

KNN



Conclusions

- After the hyperparameter tuning in each model, Logistic Regression, SVM and KNN models showed the same accuracy on the test data set.
(The accuracy was 0.83 on the test data set)
- The confusion matrix were also same among those three models
(Three "false positive", no "false negative")
- **For this problem, any of those three models are acceptable.**
For now, Logistic Regression model would be chosen because of the simplicity and easiness to understand the model structure and algorithm.

- As the next trial, further hyperparameter tuning on each model could be tried.

Thank you!