IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Michalakis Perikleous
11/04/2023

Outline

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

Executive Summary

Summary of Methodologies

- Collecting Data
- Data Wrangling
- Exploratory Analysis Using SQL
- Exploratory Analysis Using Pandas and Matplotlib
- Interactive Visual Analytics and Dashboard
- Predictive Analysis (Classification)

Summary of all results

- Cleaning the collecting data into a Pandas Dataframe with new labels that can be exported to .CSV file.
- Answering questions, using SQL queries, marking launching (success/fail) sites on the map.
- Creating an interactive data visualization (dashboard).
- Percentages of prediction if the first stage of the Falcon 9 lands successfully.

Introduction

Introduction

Landing the first stage of Space X's Falcon 9 rocket.

Project Background

SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage.

Problems to find answers

We have to determine if the first stage will land and after we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Get request from the SpaceX REST API.
  - Web Scraping related Wiki pages.
- Perform data wrangling
  - Wrangling Data using an API.
  - Sampling Data.
  - Dealing with Nulls.

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Standardize the data
  - Split the data into training and testing
  - Testing the data in four models (Logistic Regression, Support Vector Machine, Decision tree Classifier, K-nearest Neighbors)
  - Calculating the accuracy on the test data
  - Output the confusion matrix.

Section 1

Data Collection

Data Collection

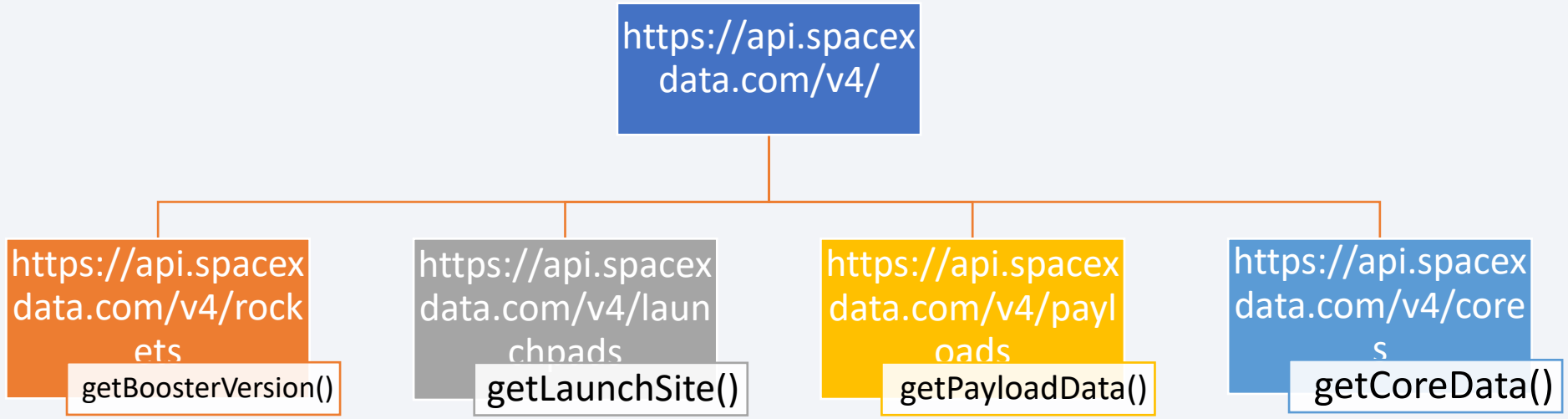
At first SpaceX launch data collected from a URL using the GET request.

```
URL='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

Then we used “`launch_data = pd.json_normalize(data)`” to convert the json result into a Dataframe.

Data Collection – SpaceX API

GitHub URL • [jupyter-labs-spacex-data-collection-api]



- We used the above functions to extract information using identification numbers in the launch data.
- We constructed our dataset using the data we have obtained. We combined the columns into a dictionary.
- We filtered the Dataframe to only include ‘Falcon 9’ launches.

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

Falcon 9 DataFrame

```
[25]
1  data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
2  data_falcon9
```

```
C:\Users\Michael\AppData\Local\Temp\ipykernel_18924\4023201451.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
5	1	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
6	2	2010-12-08	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0004	-80.577366	28.561857
7	3	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561857
8	4	2012-10-08	Falcon 9	400.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0006	-80.577366	28.561857
9	5	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561857
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
101	97	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1060	-80.603956	28.608058
102	98	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	13	B1058	-80.603956	28.608058
103	99	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1051	-80.603956	28.608058
104	100	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	12	B1060	-80.577366	28.561857
105	101	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	8	B1062	-80.577366	28.561857

101 rows × 17 columns

Data Collection - Scraping

GitHub URL • jupyter-labs-webscraping

Extracted Falcon 9 launch records HTML table from Wikipedia

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

  data = requests.get(static_url)

falcon_9_wiki = data.text

  Soup = BeautifulSoup(falcon_9_wiki, 'html.parser')

Parseed the table and convert it into a Pandas data frame

❖table_elements = soup.find_all('table')

launch_dict= dict.fromkeys(column_names)

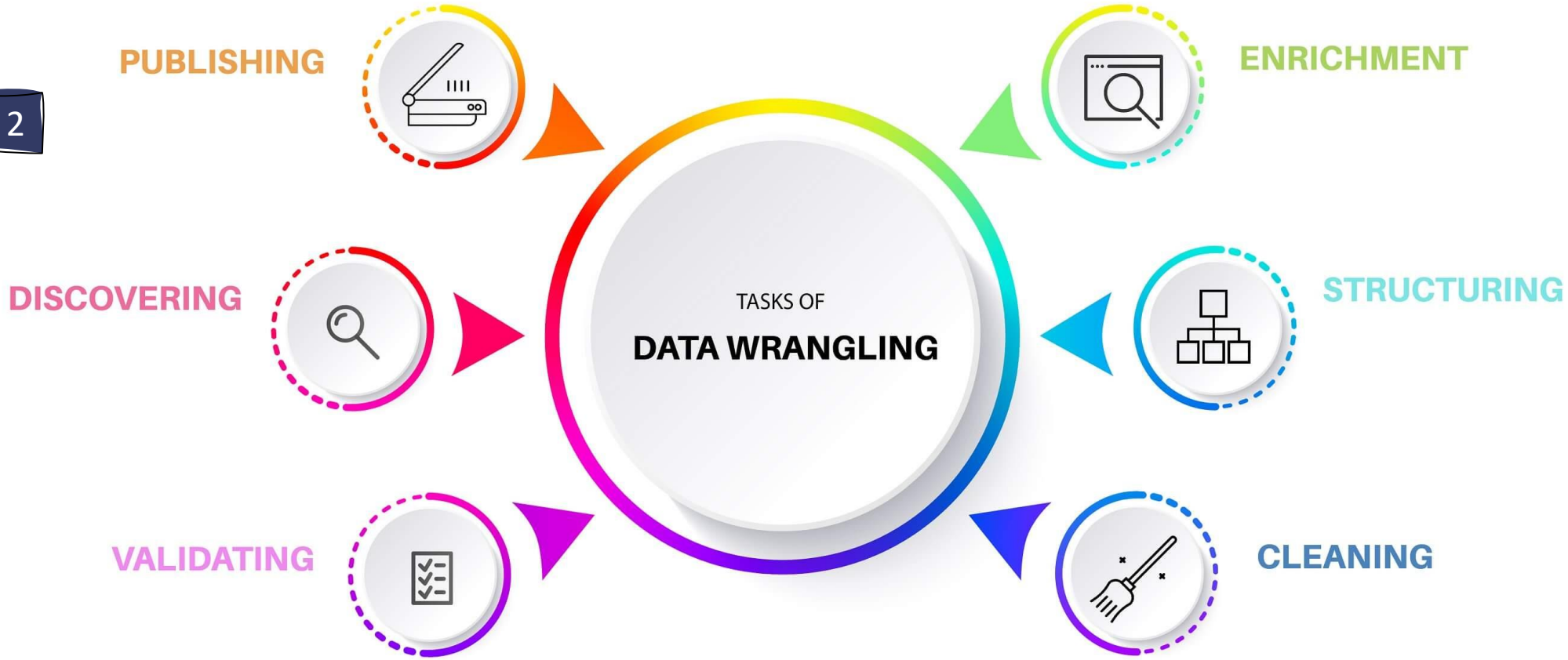
df=pd.DataFrame(launch_dict)

Falcon 9 historical records from a Wikipedia page

	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\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

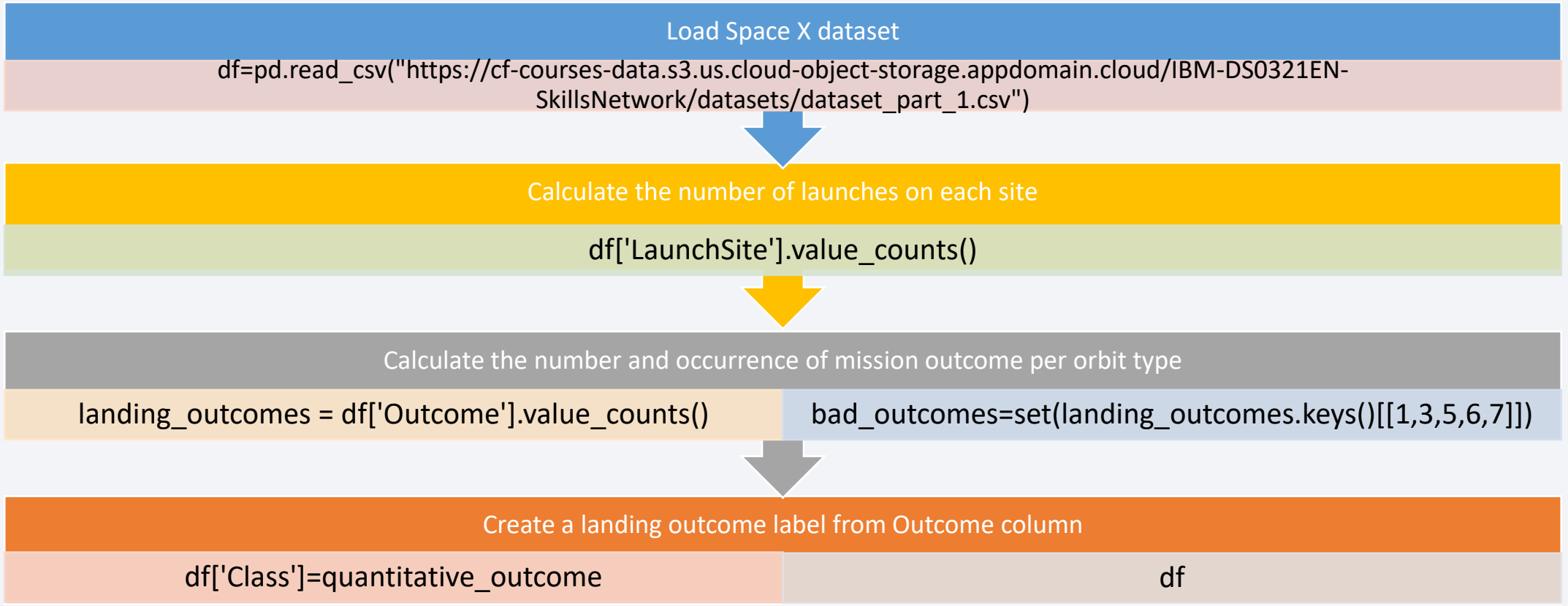
121 rows × 11 columns

Section 2


PUBLISHING
ENRICHMENT
DISCOVERING
STRUCTURING
TASKS OF
DATA WRANGLING
VALIDATING
CLEANING

Data Wrangling

GitHub URL • [labs-jupyter-spacex-Data-wrangling]



Data wrangling

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
85	86	2020-09-03	Falcon 9	15400.000000	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	2	B1060	-80.603956	28.608058	1
86	87	2020-10-06	Falcon 9	15400.000000	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	2	B1058	-80.603956	28.608058	1
87	88	2020-10-18	Falcon 9	15400.000000	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	5	B1051	-80.603956	28.608058	1
88	89	2020-10-24	Falcon 9	15400.000000	VLEO	CCAFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	2	B1060	-80.577366	28.561857	1
89	90	2020-11-05	Falcon 9	3681.000000	MEO	CCAFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	0	B1062	-80.577366	28.561857	1

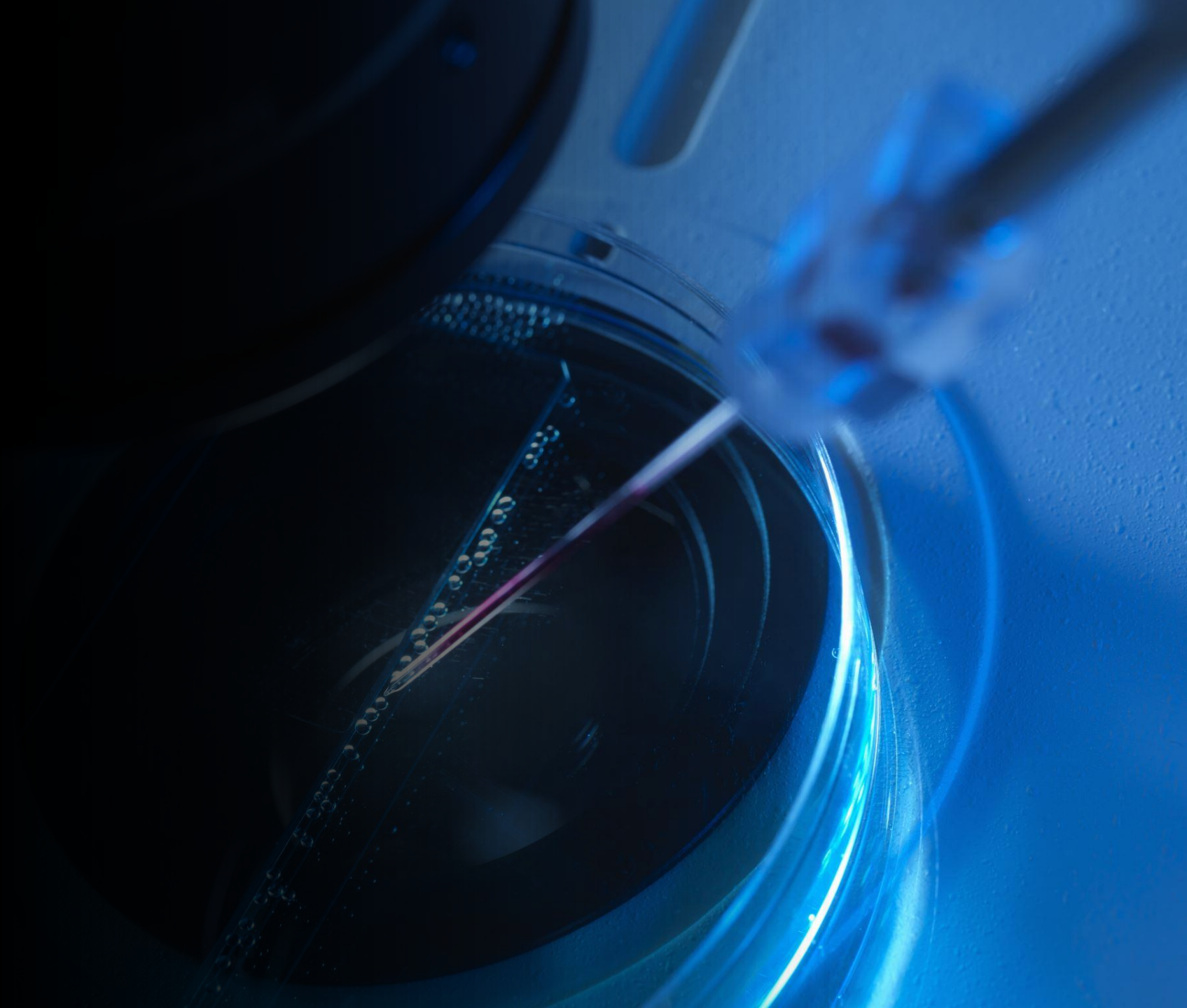
90 rows × 18 columns

Section 3a

Insights drawn from EDA

EDA and interactive visual analytics methodology

- We performed some Exploratory Data Analysis using a database. Some attributes can be used to determine if the first stage can be reused. We can then use these features with machine learning to automatically predict if the first stage can land successfully.



EDA with Data Visualization

GitHub URL • jupyter-labs-eda-dataviz

Scatter point chart

- To visualize the relationship between flight number and launch site
- To visualize the relationship between payload and launch site
- To visualize the relationship between flight number and orbit type
- To visualize the relationship between payload and orbit type

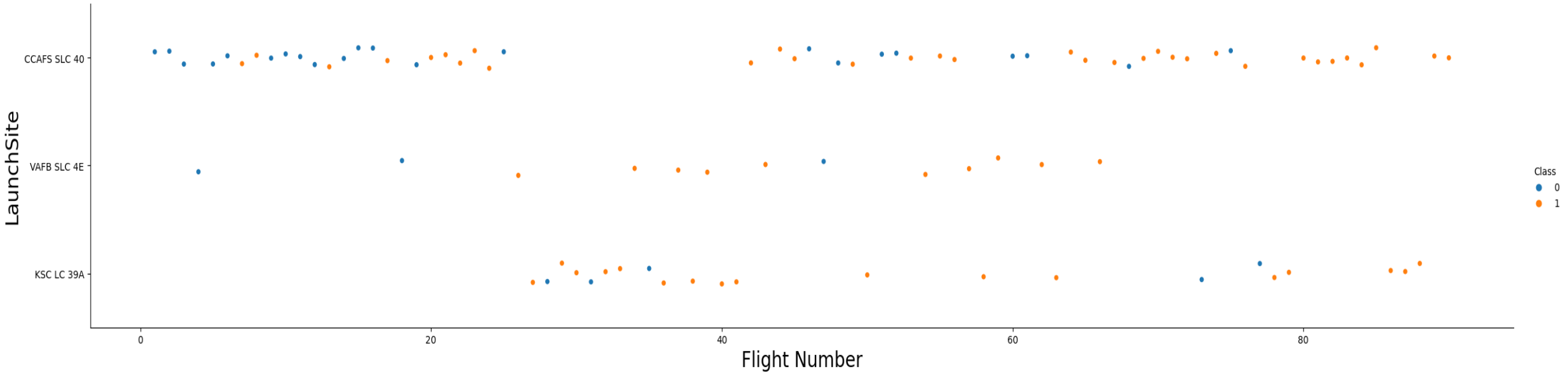
Bar chart

- To visualize the relationship between success rate of each orbit type

Line Plot

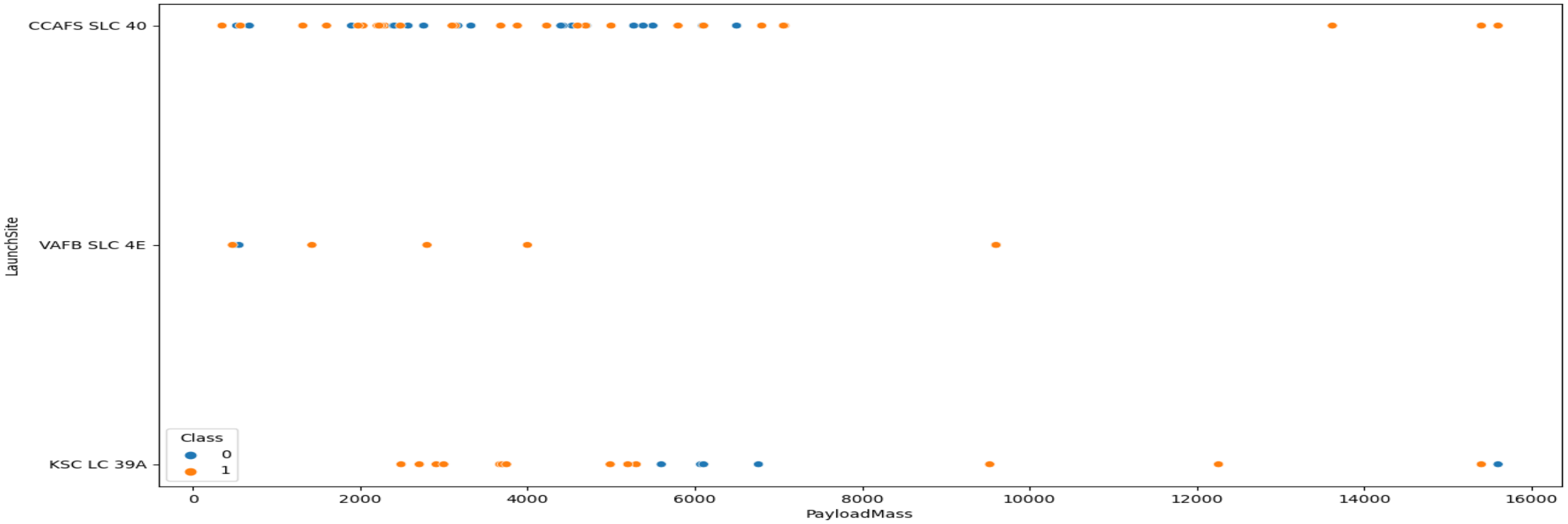
- To visualize the launch success yearly trend

Launch Site vs. Flight Number



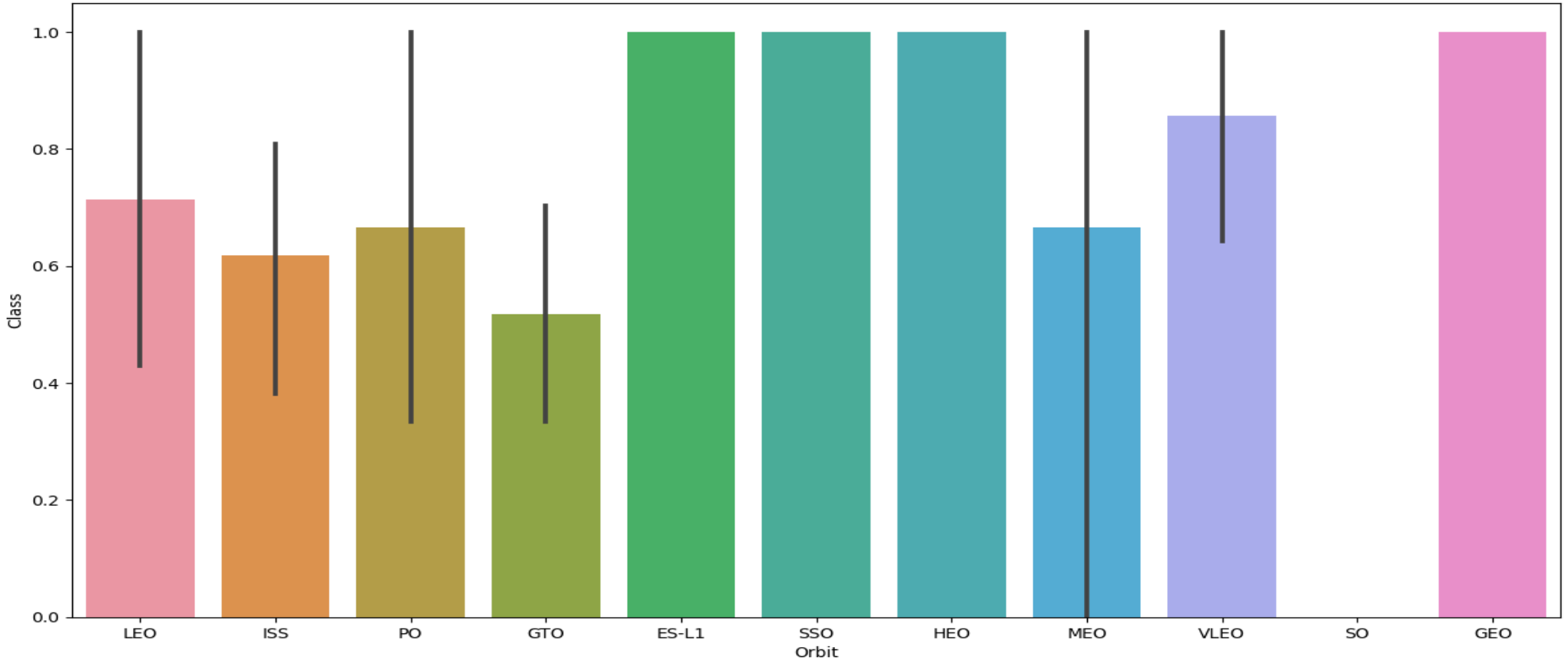
As the number of flight increases the number of successful landings also increases in all the launch sites.

Launch Site vs. Payload



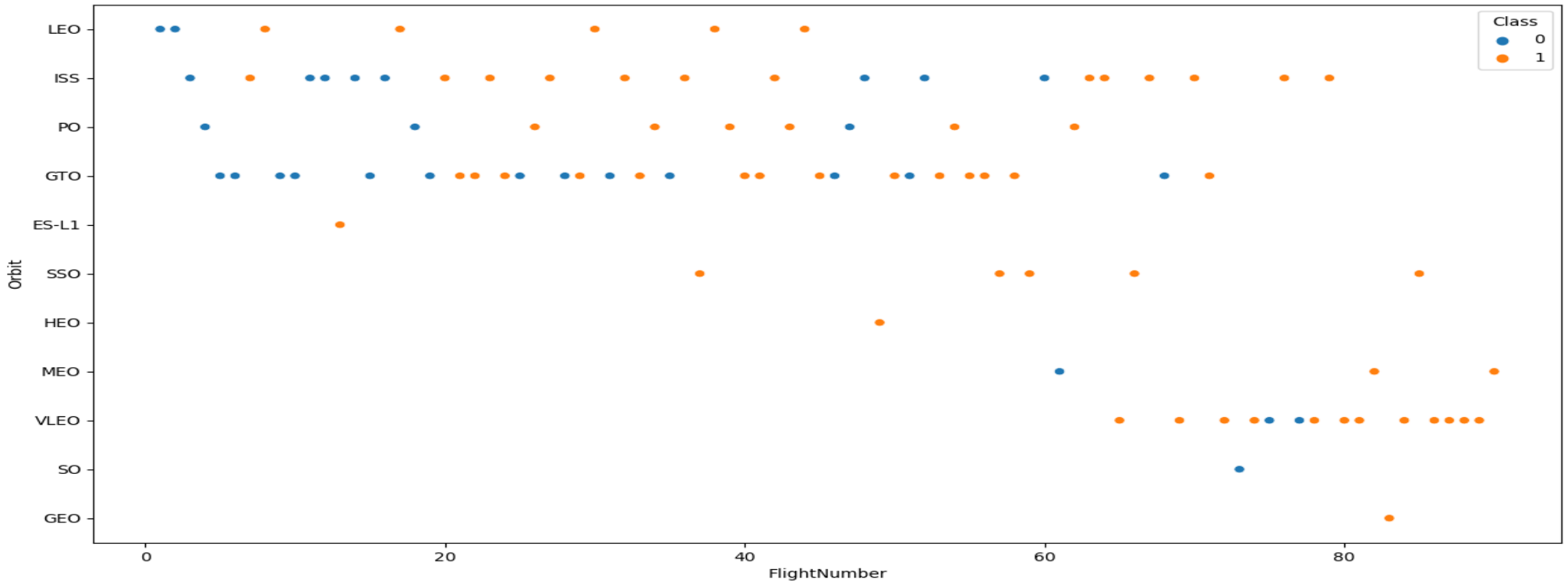
For the VAFB-SLC launch site there are no rockets launched for heavy payload mass, greater than 10000 kg.

Success Rate vs. Orbit Type



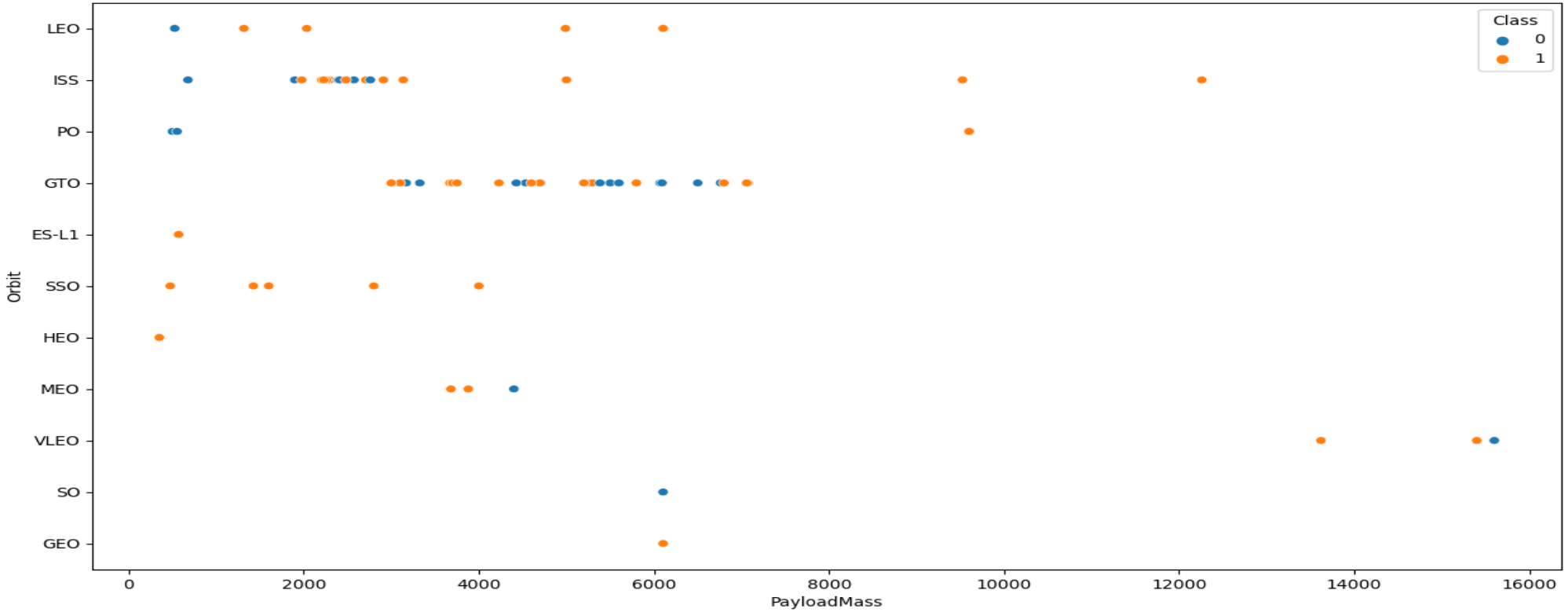
ES-L1, SSO, HEO and GEO orbits have high success rate.

Orbit Type vs. Flight Number



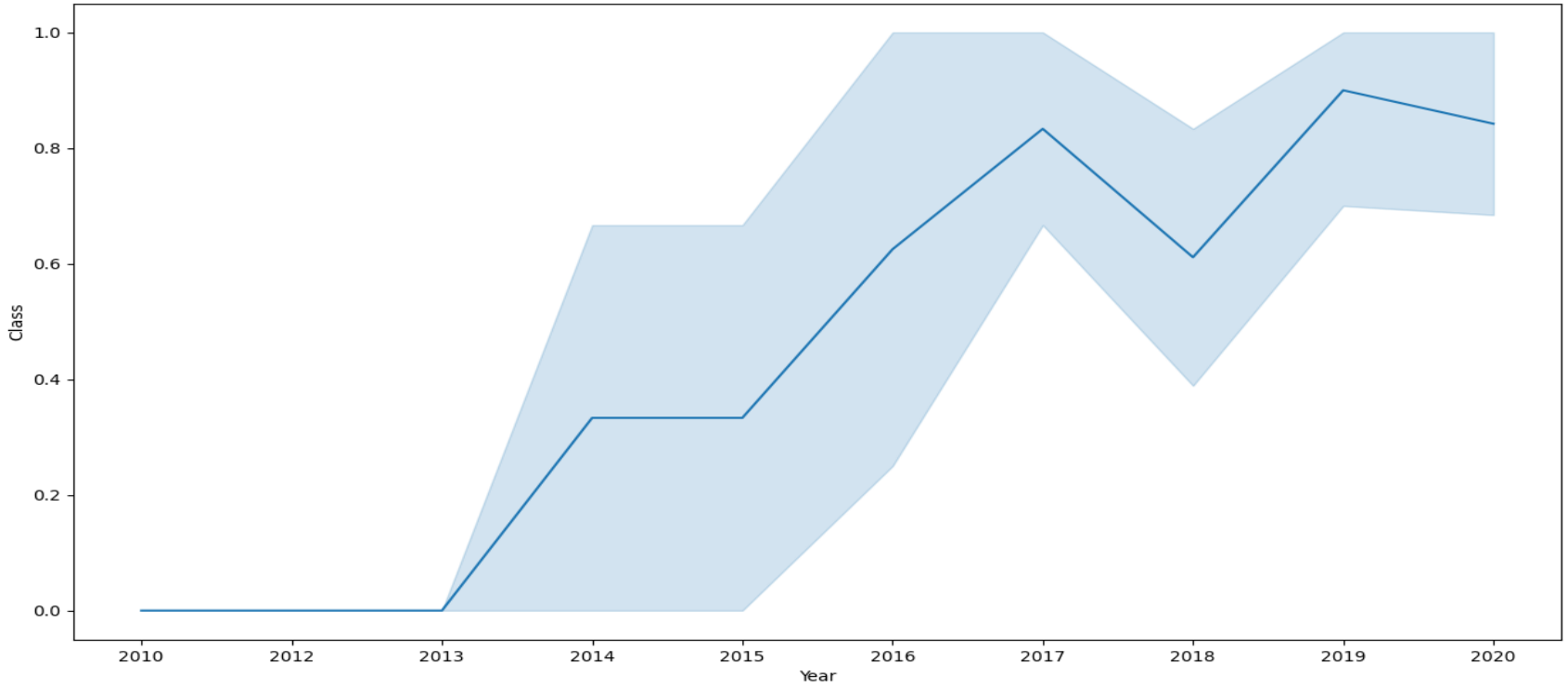
In the LEO orbit the Success appears related to the number of flights, on the other hand, there seems to be no relationship between flight number when in GTO orbit. VLEO orbit has a good number of success in higher flight numbers.

Orbit Type vs. Payload



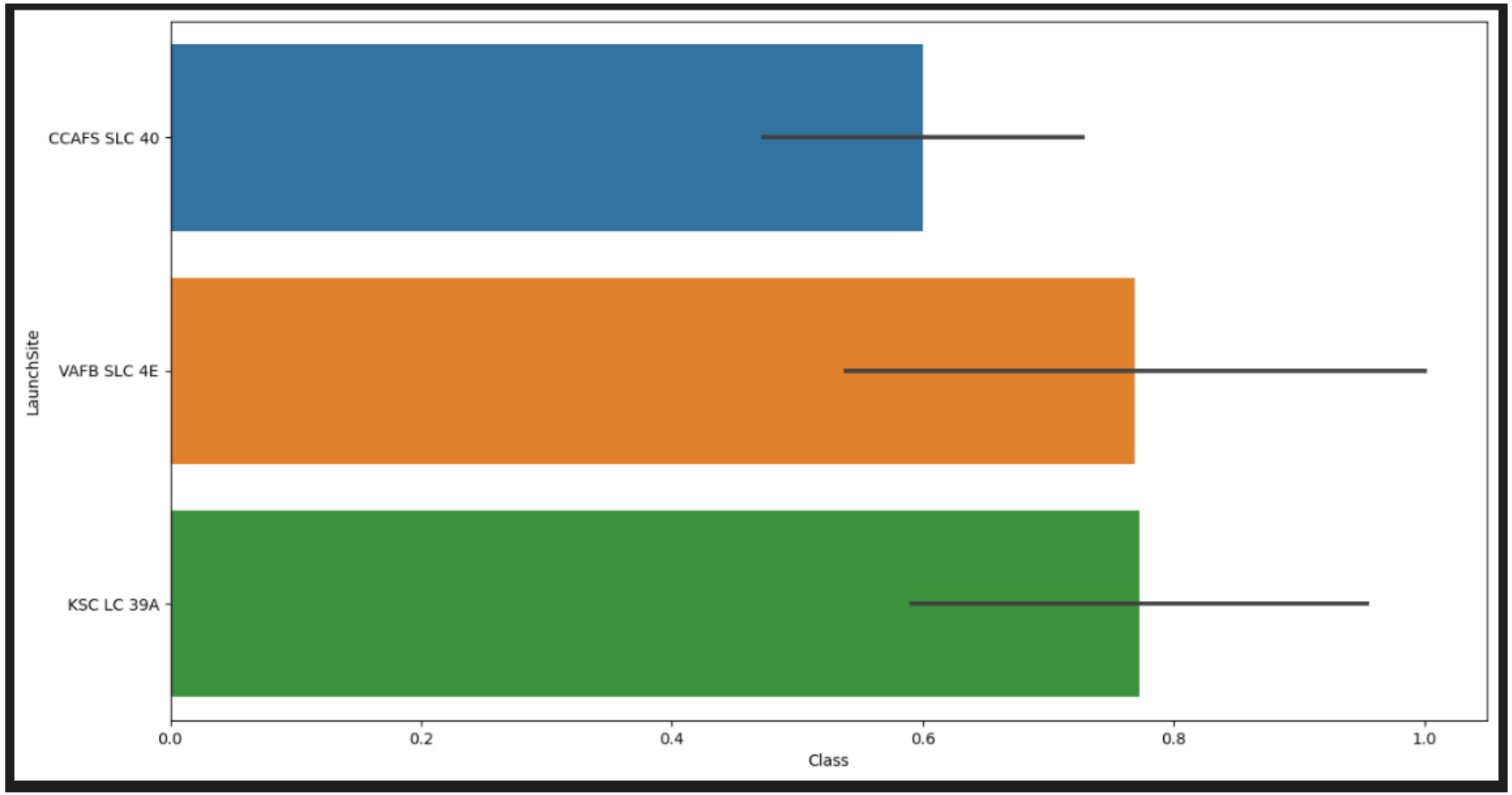
With heavy payloads the successful landing or positive landing rate are more for LEO and ISS. However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here. SSO has only few positive landings in lower payloads.

Launch Success Yearly Trend



From the year 2013 and up the success rate kept increasing.

Launch Sites Vs. Success Rate



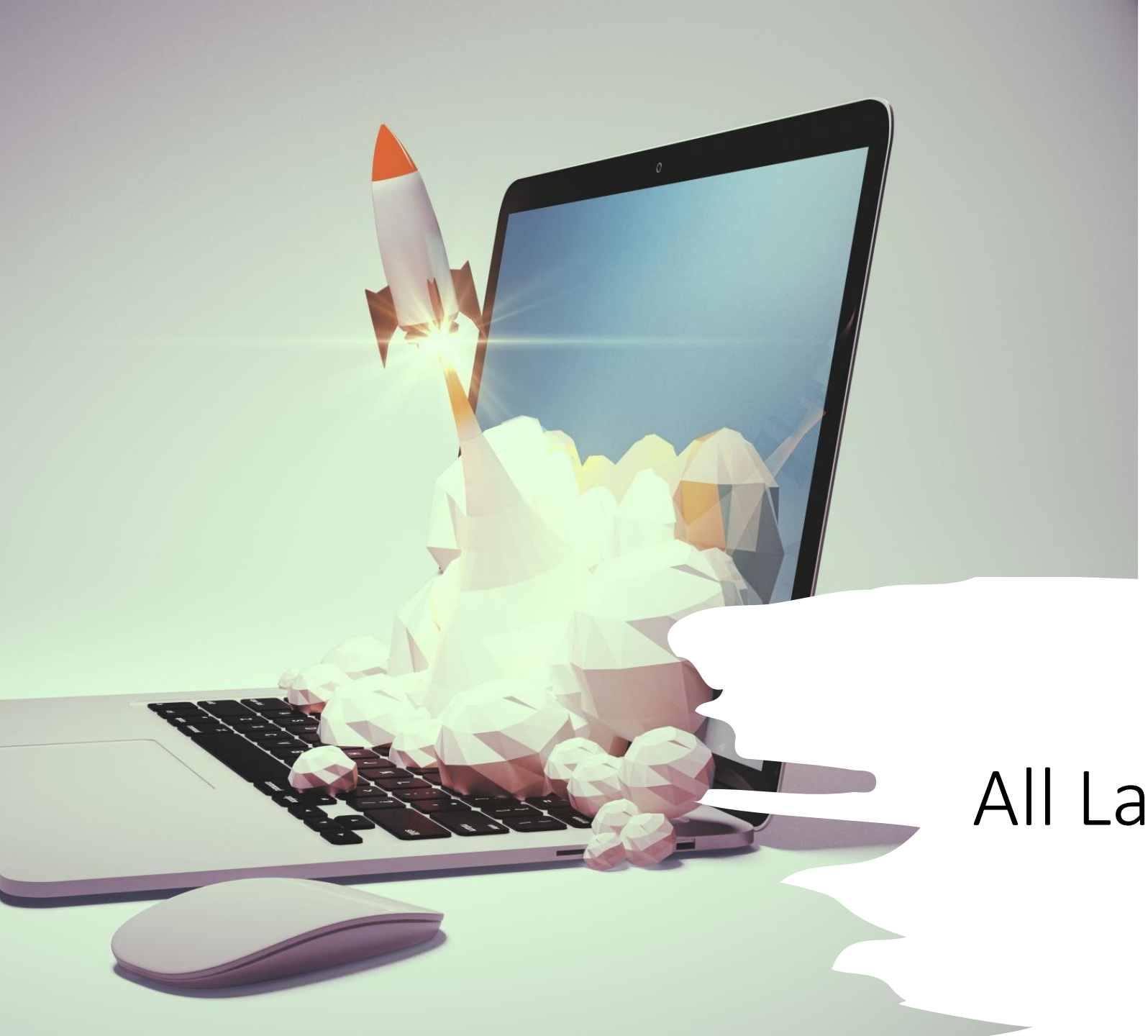
- Section 3b
- EDA using SQL

EDA with SQL

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order

GitHub URL

- jupyter-labs-eda-sql-coursera_sqllite



```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL
```

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

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

All Launch Site Names

Launch Site Names Begin with 'CCA'

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

```
 * 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

Total Payload Mass

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass_kg
FROM SPACEXTBL
WHERE CUSTOMER = 'NASA (CRS)'
```

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

total_payload_mass_kg
45596

Average Payload Mass by F9 v1.1

```
1  %%sql
2  SELECT AVG(PAYLOAD_MASS__KG_) AS avg_payload_mass_kg
3  FROM SPACEXTBL
4  WHERE BOOSTER_VERSION = 'F9 v1.1'
```

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

avg_payload_mass_kg
2928.4

First Successful Ground Landing Date

```
1  %%sql
2  SELECT min(date) as FirstSuccessfullLanding from SPACEXTBL where  "Landing _Outcome"='Success (ground pad)'
```

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

FirstSuccessfullLanding
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

```
1  %%sql
2  SELECT BOOSTER_VERSION
3  FROM SPACEXTBL
4  WHERE "Landing _outcome" = 'Success (drone ship)' and (PAYLOAD_MASS__KG_ between 4000 and 6000)
```

```
 * 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

```
%%sql
SELECT MISSION_OUTCOME, COUNT(*) AS total_number
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME
```

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

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

Boosters Carried Maximum Payload

```
%%sql
SELECT DISTINCT BOOSTER_VERSION, PAYLOAD_MASS__KG_
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL);
```

```
 * 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

```
1  %%sql
2  SELECT substr(Date, 4, 2) as month,booster_version,"Landing _Outcome"
3  from SPACEXTBL where "Landing _Outcome"='Failure (drone ship)' and substr(Date,7,4)='2015'
```

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

month	Booster_Version	Landing _Outcome
01	F9 v1.1 B1012	Failure (drone ship)
04	F9 v1.1 B1015	Failure (drone ship)

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

```
1  %%sql
2  SELECT "Landing _Outcome", count("Landing _Outcome") as LANDING_OUTCOME_COUNT,DATE
3  from SPACEXTBL where Date between  '04-06-2010' and '20-03-2017' group by "Landing _Outcome"
```

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

Landing _Outcome	LANDING_OUTCOME_COUNT	Date
Controlled (ocean)	3	18-04-2014
Failure	3	05-12-2018
Failure (drone ship)	4	10-01-2015
Failure (parachute)	2	04-06-2010
No attempt	10	08-10-2012
No attempt	1	06-08-2019
Success	20	07-08-2018
Success (drone ship)	8	08-04-2016
Success (ground pad)	6	18-07-2016


Section 4
Launch Sites
Proximities Analysis
Folium

Build an Interactive Map with Folium

Folium Map Object

- To create a base map.

Folium Circle Object and Folium Marker Object

- To add a highlighted circle area with a text label on a specific coordinate.

Folium map.FeatureGroup()

- To add a circle launch site in Dataframe.

Folium MarkerCluster() Object

- To add all marker in it and then add it to the map.

Folium MousePosition() Object

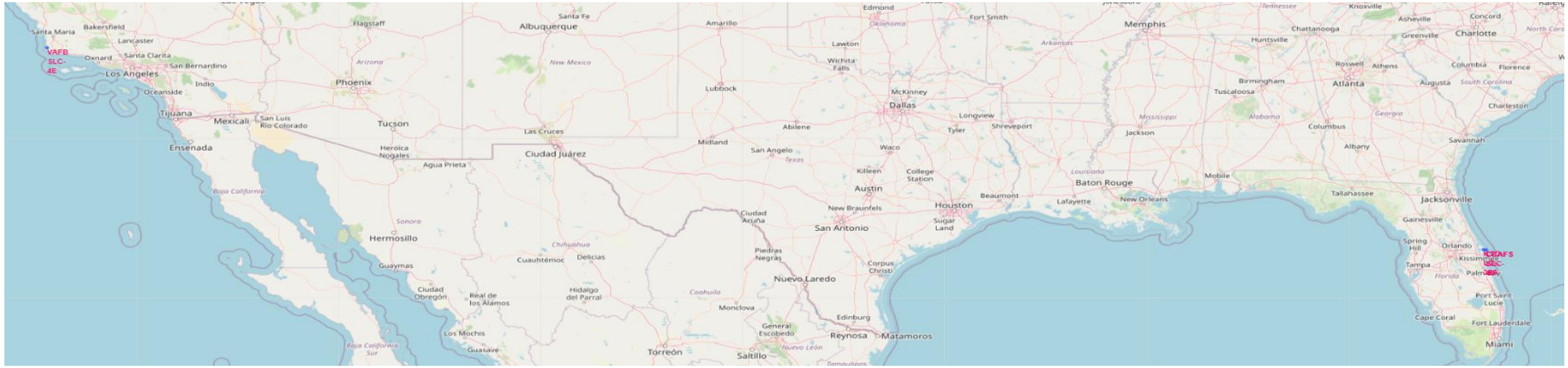
- To get coordinate for a mouse over a point on the map.

Folium Polyline()

- To draw a line between two points

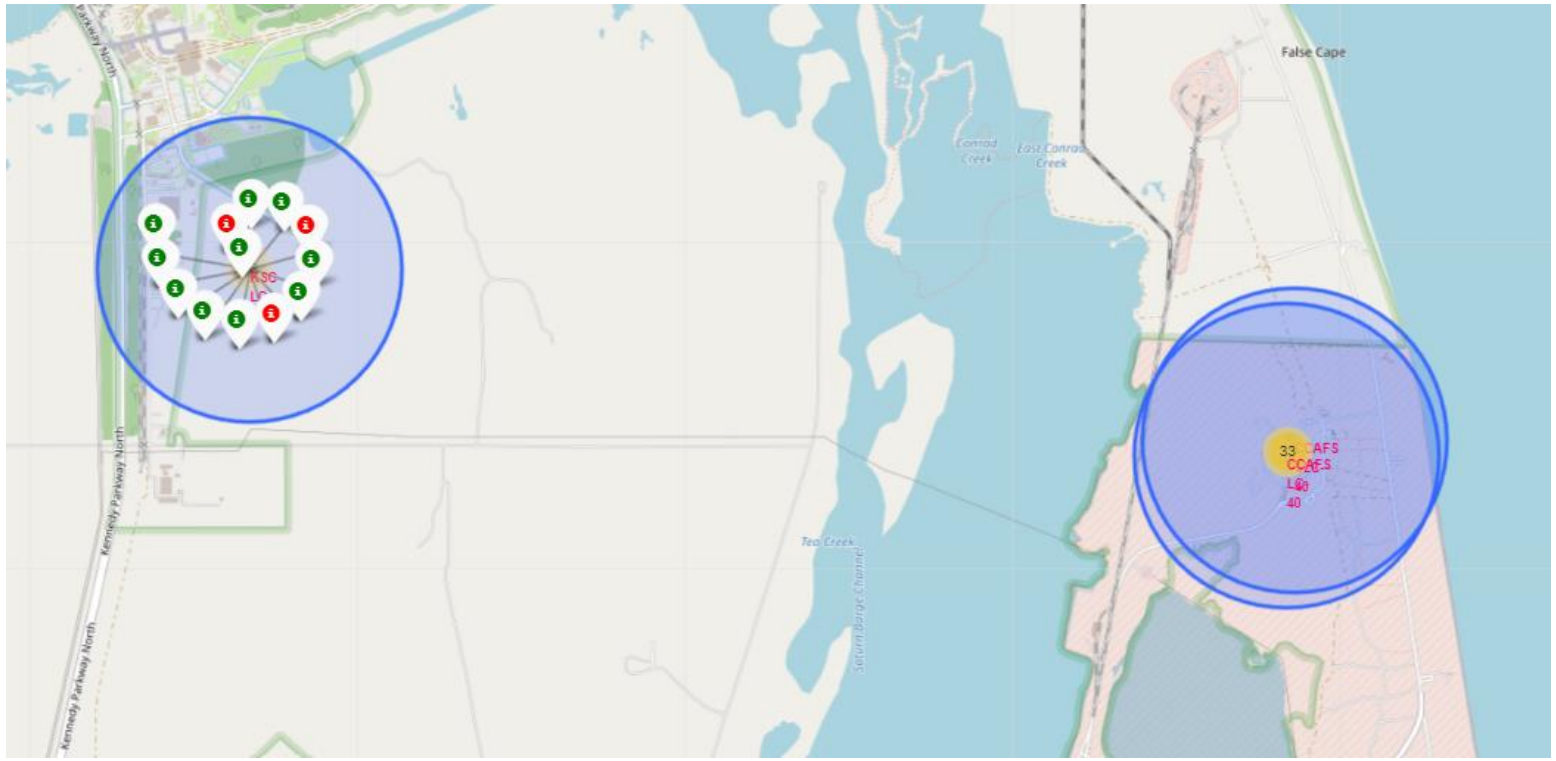
GitHub URL • lab_jupyter_launch_site_location

SpaceX Launch Sites



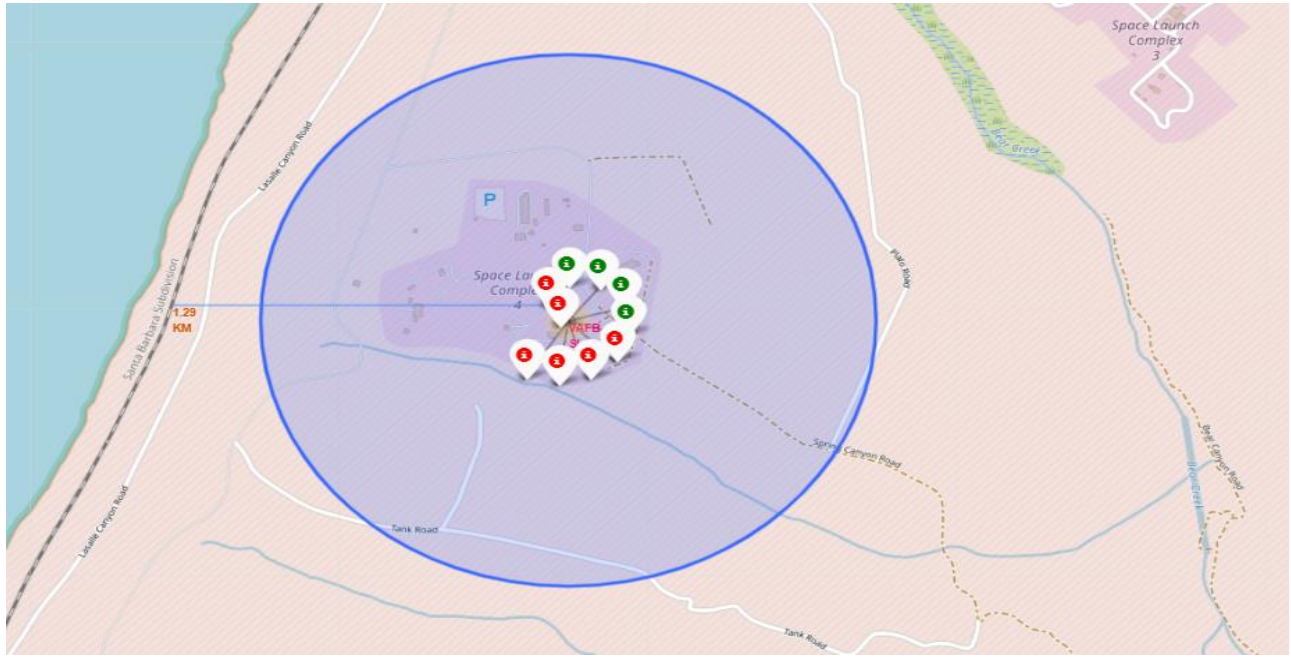
- All launch sites are in proximity to the Equator line.
- Most of them are situated to the east coast because it gives an additional boost due to the rotational speed of Earth.
- All launch sites are in very close proximity to the coast if something goes wrong during the ascent, the debris will fall into an ocean instead of a densely populated area.

Success/failed launches for each site on the map

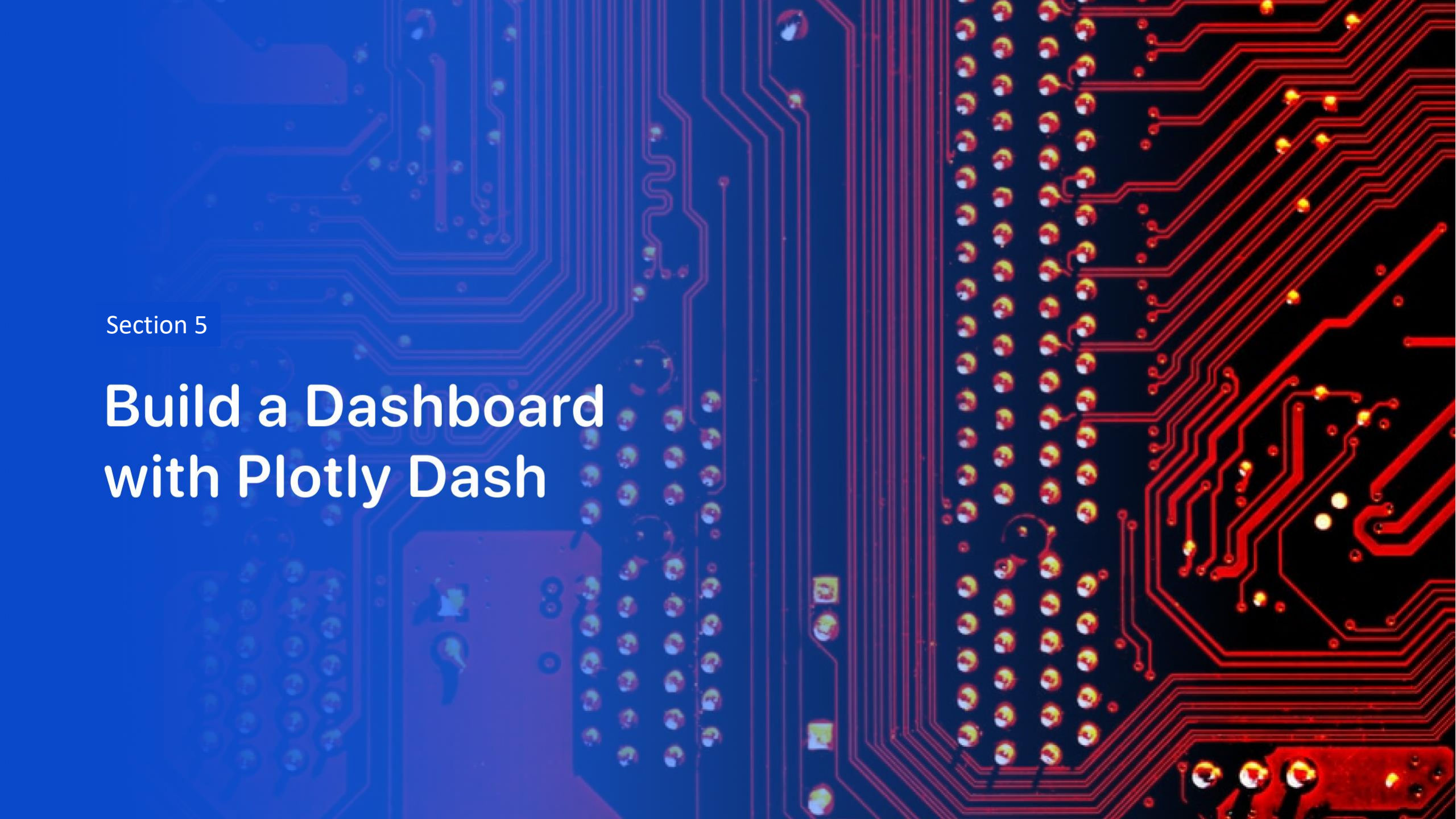


From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates. Green color equals success and red color failure.

Launch Sites Location



- Launch sites are in close proximity to railways and highways in order to be easily accessed.
- Launch sites are in close proximity to coastline and far from cities, so that debris will fall into an ocean instead of a densely populated area.


Section 5
Build a Dashboard with Plotly Dash

Build a Dashboard with Plotly Dash

Pie chart

- To show the total successful launches count for all sides.

Scatter chart

- To show then correlation between payload and launch success.

Dropdown list

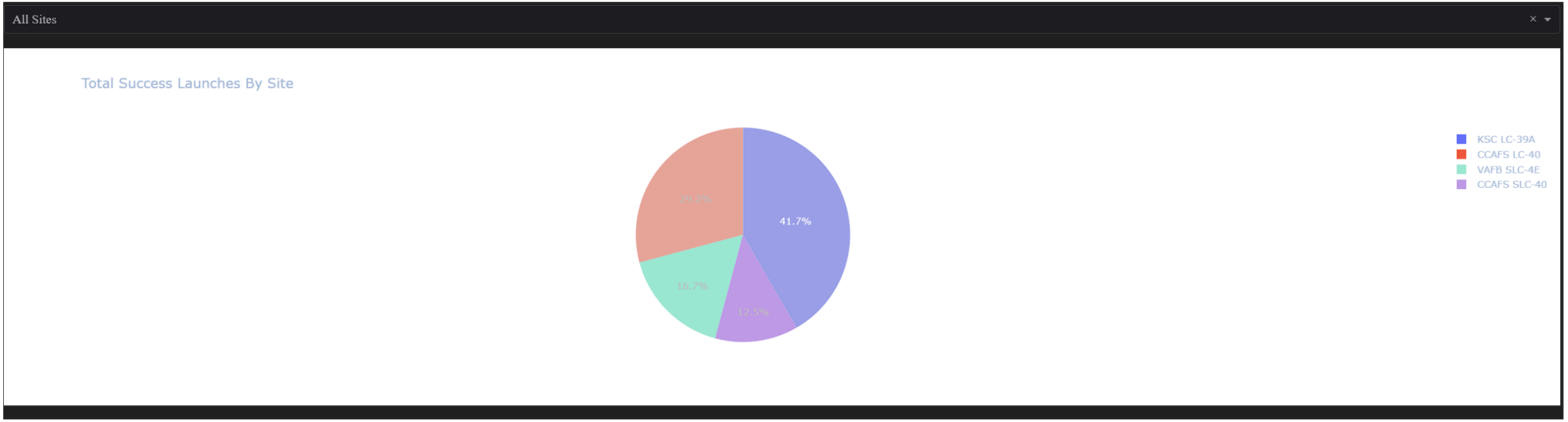
- To enable Launch Site selection.

Slider

- To select payload range.

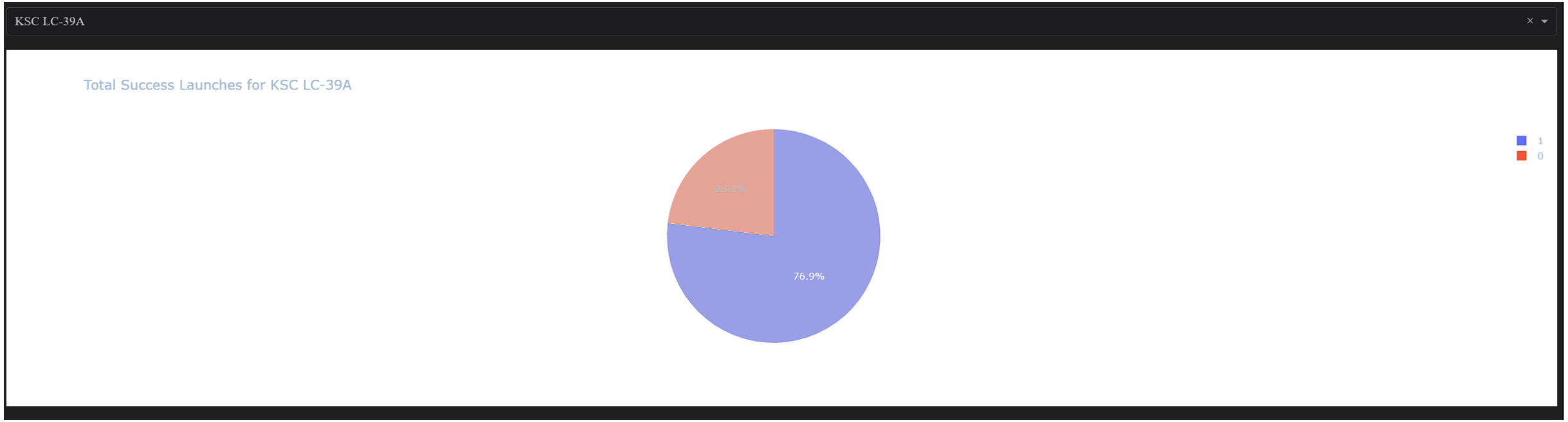
GitHub URL • spacex_dash_app

Total Success Launches By Site – All Sites



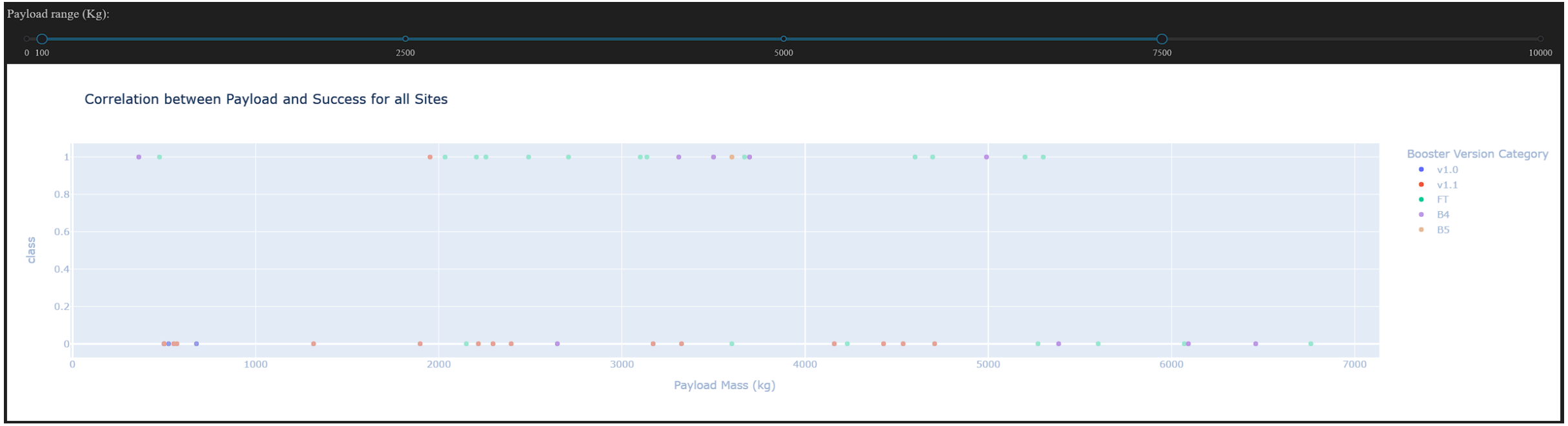
KSC LC-39A and CCAFS LC-40 launch sites have combined 70.9% success ratio.

Total Success Launches for KSC LC-39A



KSC LC-39A launch site has 76.9% success ratio.

Correlation Between Success and Payload mass (kg)



For 5300 kg < payload mass < 7000 kg we don’t observe a success launch.

Section 5

Predictive Analysis (Classification)

Predictive Analysis (Classification)

Standardize our data

- Train Data
- Test Data

Train model – Grid Search

- Find the best performing hyperparameters
- Determine the model with best accuracy

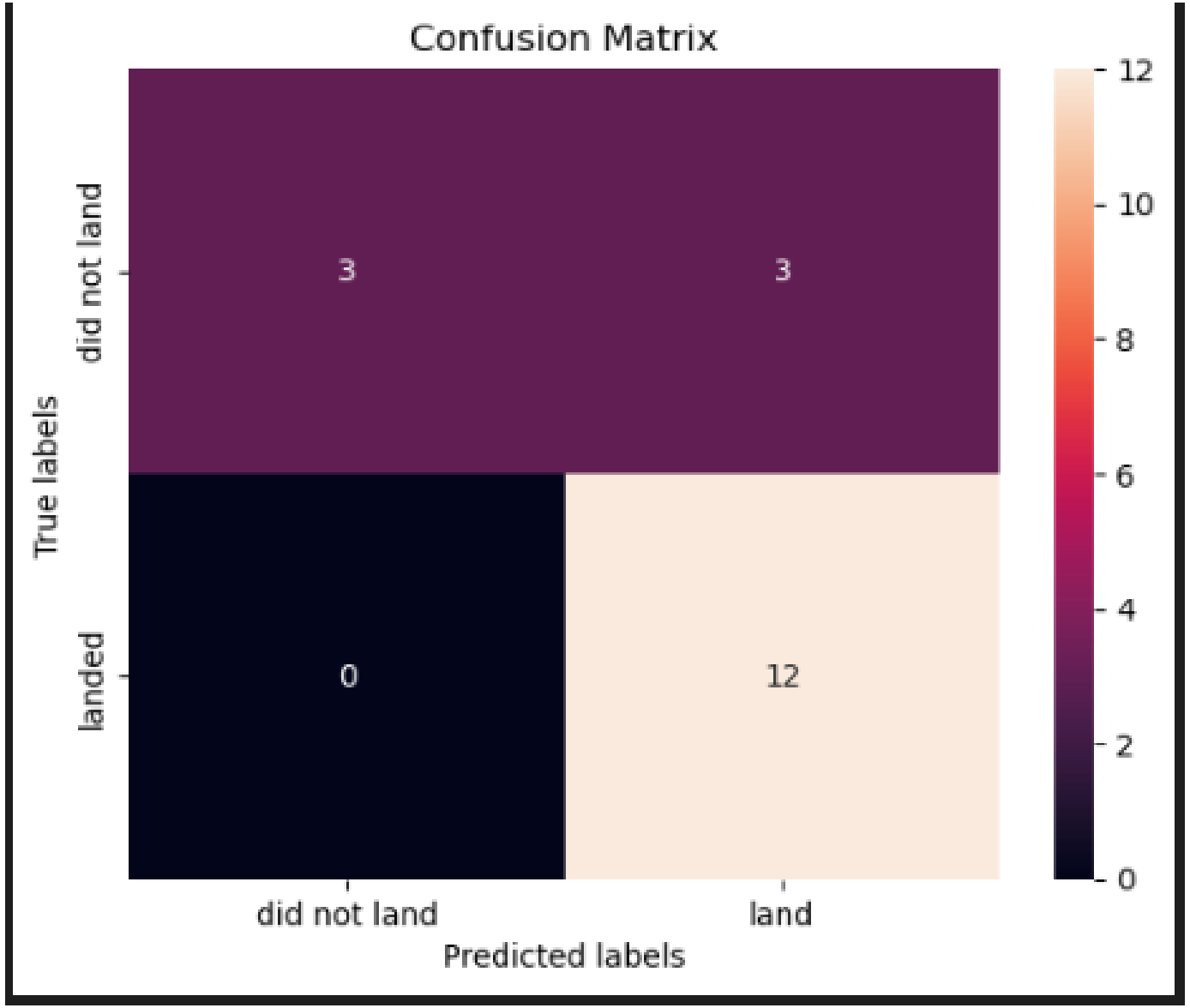
Model

- Logistic Regression
- Support Vector machine
- Decision Tree Classifier
- K-nearest Neighbors

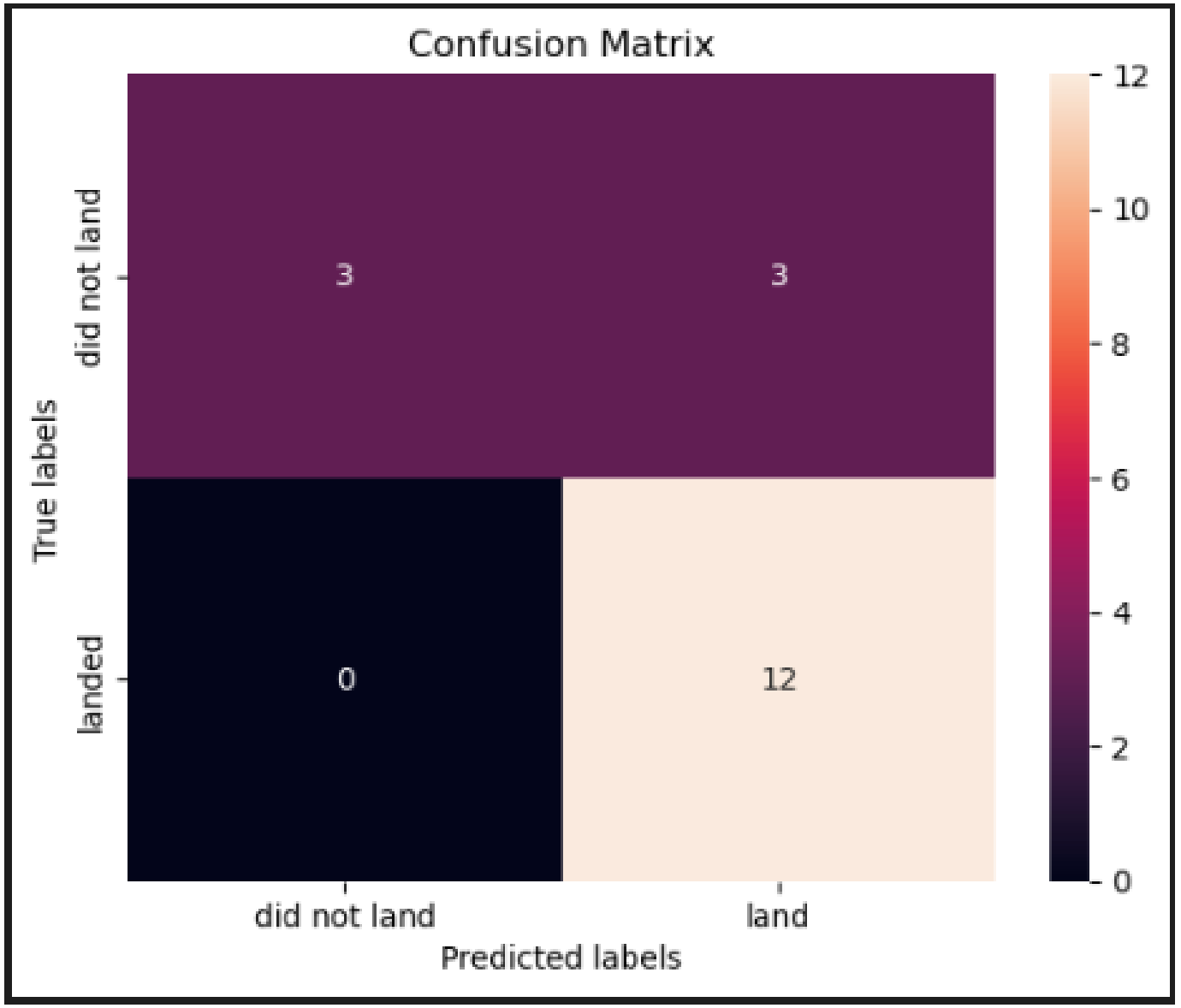
GitHub URL

- SpaceX_Machine_Learning_Prediction.ipynb

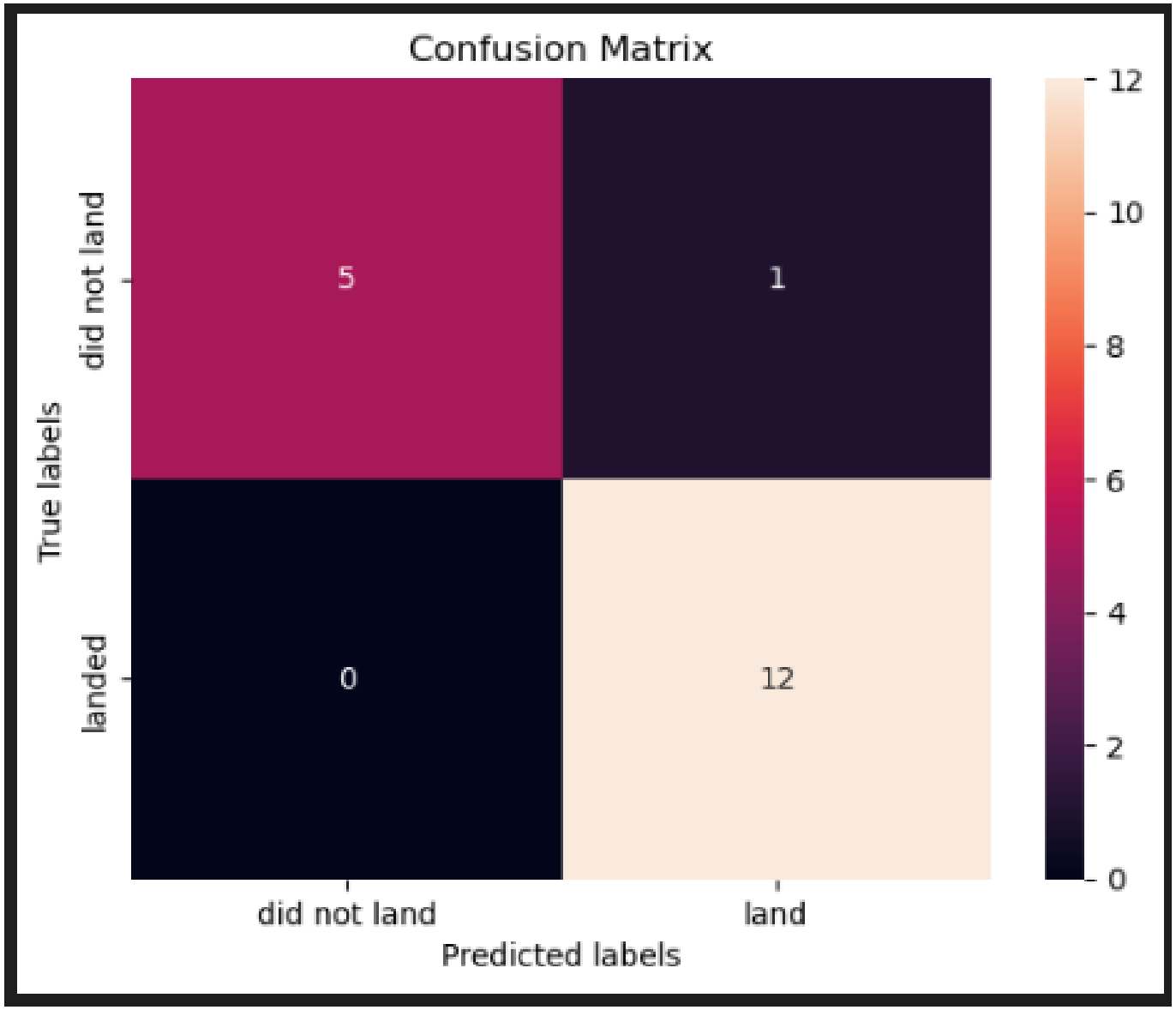
Logistic regression



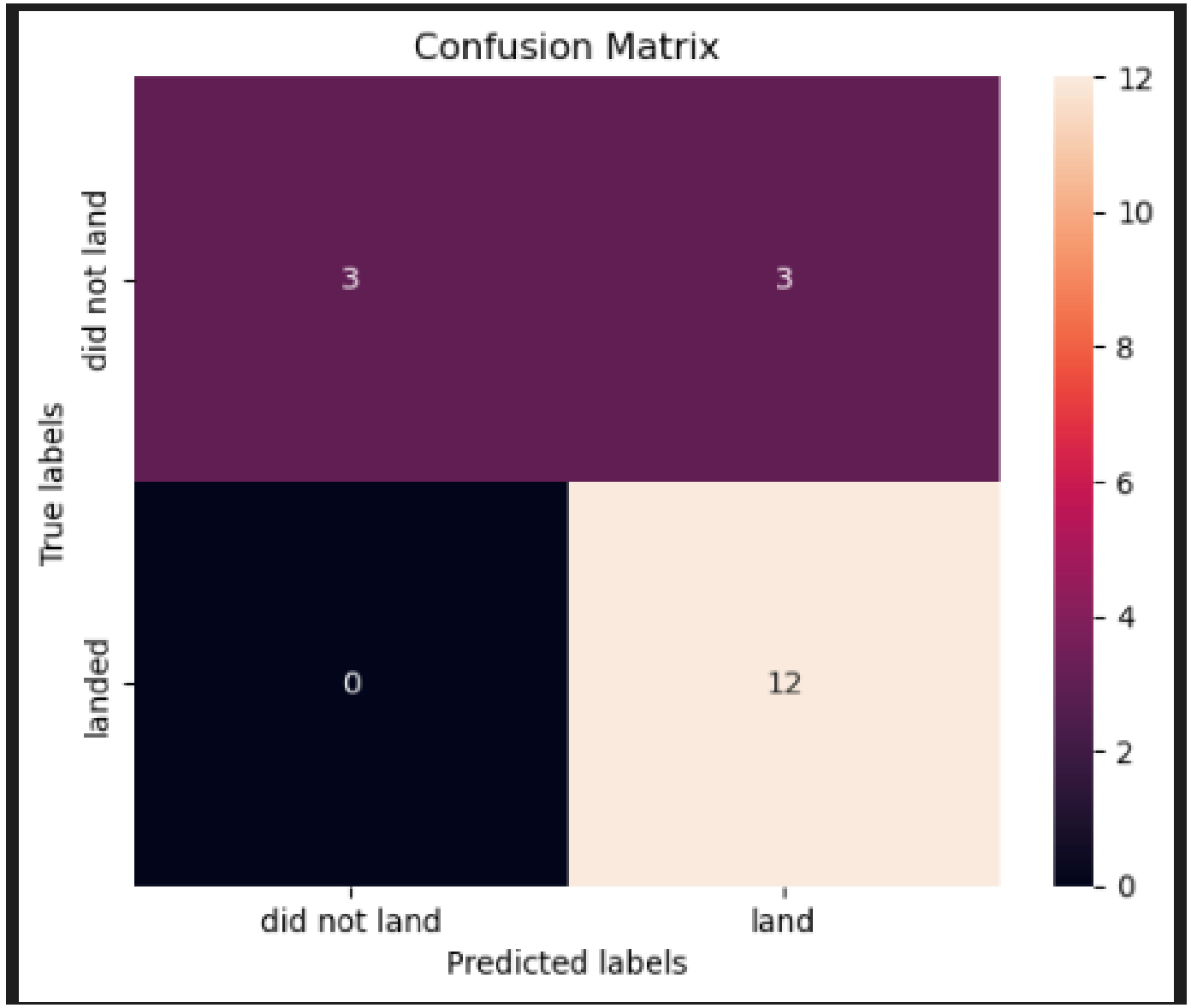
Support Vector Machine



Decision Tree Classifier



K-nearest neighbors

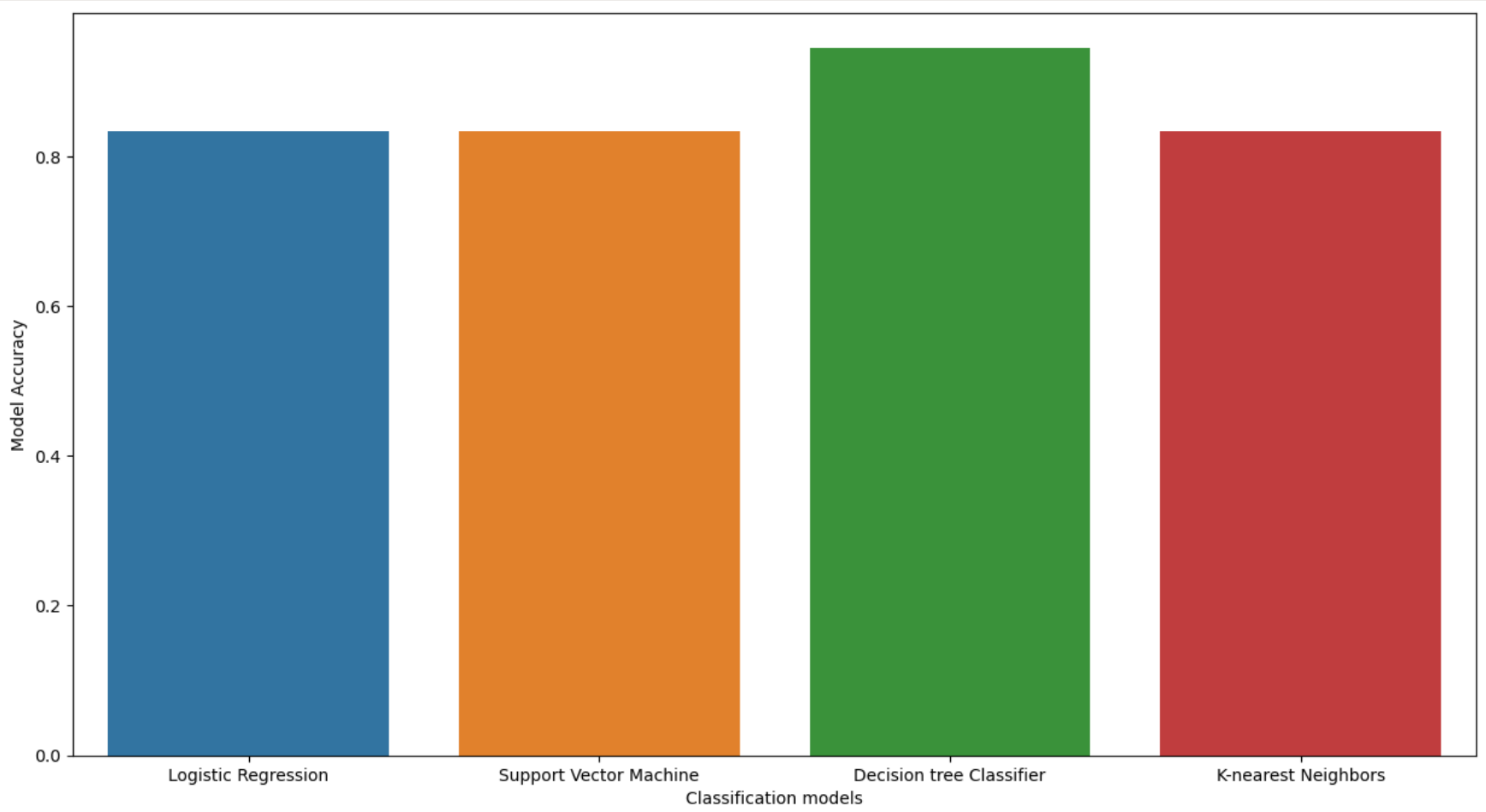


Method performs best

```
prediction_method = {'logreg_cv.score(X,Y)': logreg_cv.score(X_test,Y_test),
                     'svm_cv.score(X,Y)': svm_cv.score(X_test,Y_test),
                     'tree_cv.score(X,Y)': tree_cv.score(X_test,Y_test),
                     'knn_cv.score(X,Y)': knn_cv.score(X_test,Y_test)
}

for key, value in prediction_method.items():
    print(key, '=', value)


best_perform_method = max(prediction_method.values())


print('Tree model method is the one that performs best')
```

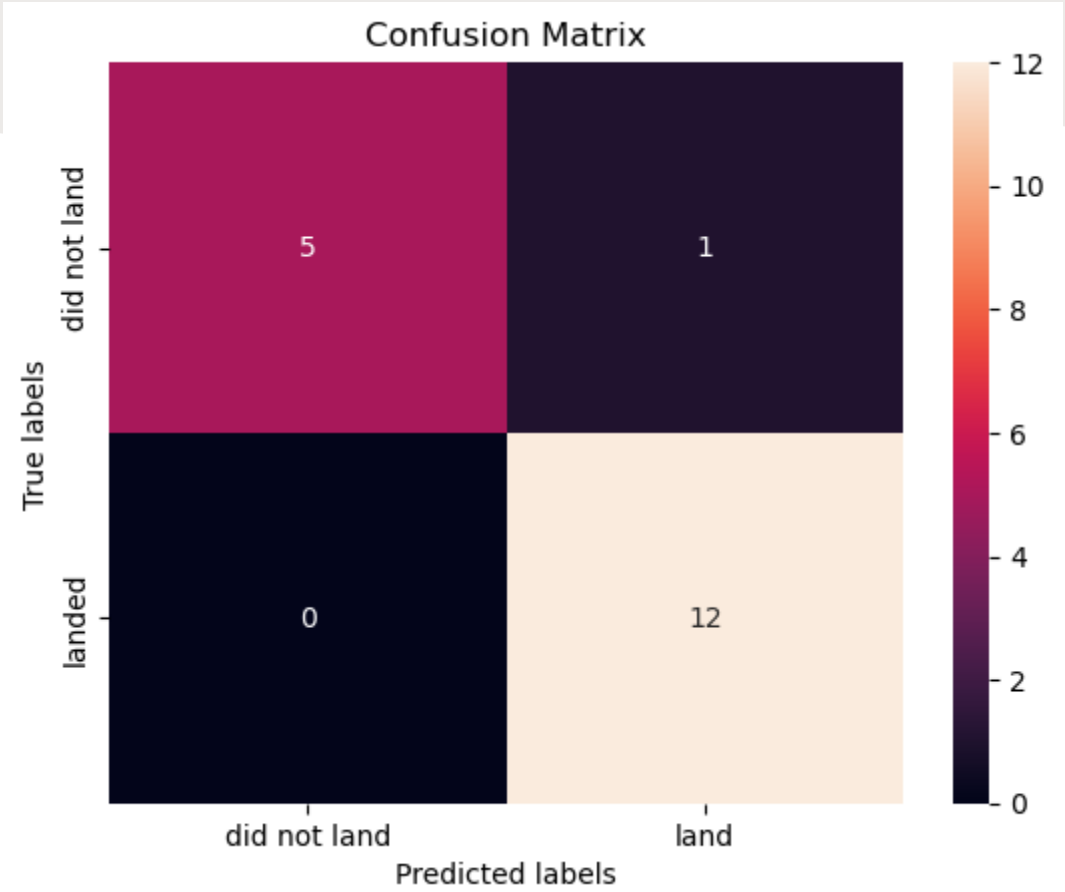
```
logreg_cv.score(X,Y) = 0.8333333333333334
svm_cv.score(X,Y) = 0.8333333333333334
tree_cv.score(X,Y) = 0.9444444444444444
knn_cv.score(X,Y) = 0.8333333333333334
Tree model method is the one that performs best
```

Classification Accuracy



Decision tree Classifier has the highest accuracy

Confusion Matrix



The accuracy of the tree model is (TP+TN)/total = (12+5)/18 = 0.94. Comparing this model with the others we see that TP and TN values are higher than the others, which makes this kind of model classification better.

Conclusions

Data visualization is more effective and attractive. It helps scientists to explore data, communicate with data clearly, share unbiased representation of data and use them to support recommendations to different stakeholders.

All launch sites are in proximity to the Equator line and in very close proximity to the coast.

According to the Decision Tree Classifier model the SpaceX first stage rocket has a success ratio 67% that will land again. This is a relative good result if someone wants to launch a rocket collaborating with SpaceX.

Appendix

```
data = {'Classification models':['Logistic Regression', 'Support Vector Machine', 'Decision tree Classifier', 'K-nearest Neighbors'], 'Model Accuracy':[logreg_cv.score(X_test,Y_test), svm_cv.score(X_test,Y_test), tree_cv.score(X_test,Y_test), knn_cv.score(X_test,Y_test)]}
models_class = pd.DataFrame(data)
models_class
```

```
c:\Users\Micha\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the defa
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

	Classification models	Model Accuracy
0	Logistic Regression	0.833333
1	Support Vector Machine	0.833333
2	Decision tree Classifier	0.944444
3	K-nearest Neighbors	0.833333

```
plt.figure(figsize=(15,8))
sns.barplot(data = models_class, x='Classification models', y='Model Accuracy')
```

```
orbit = df[['LaunchSite', 'Class']].groupby('LaunchSite')
orbit.mean()
plt.figure(figsize=(15,8))
sns.barplot(data = df, y='LaunchSite', x='Class')
```

✓ 0.2s

Thank you!