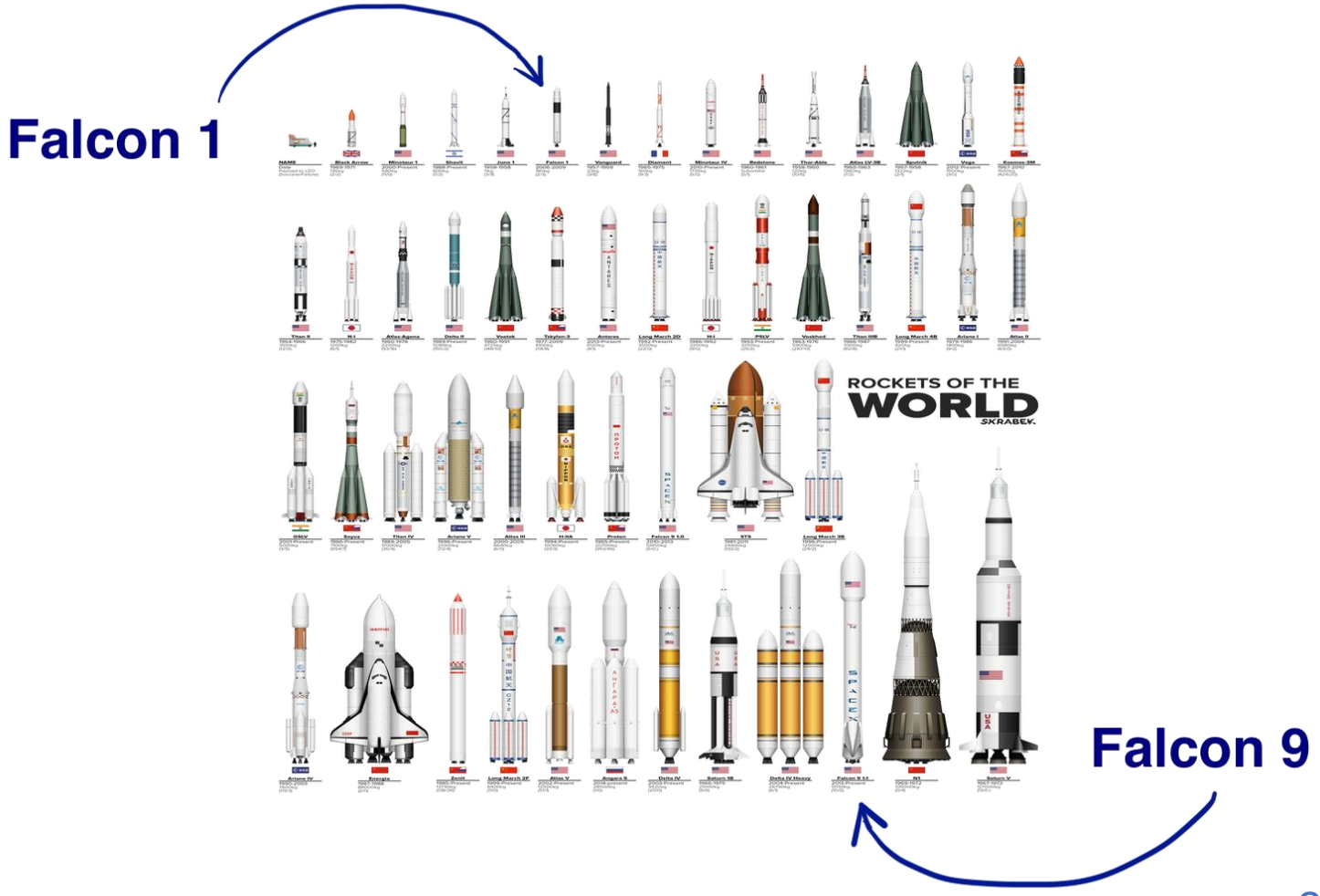
Winning Space Race with Data Science

Alvaro Diaz Falconi
May, 2022.

Outline

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



Executive Summary

- Summary of methodologies
  - Data Collection
    - With API
    - With Web Scraping
  - Data Wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building an interactive Dashboard with Plotly
  - Machine Learning Prediction
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context
  - SpaceX scored its first big headline in 2010, when it became the first private company to launch a payload into orbit and return it to Earth intact—something only government agencies like NASA or Russia's Roscosmos had done before.
  - It has lowered the cost of spaceflight through innovations such as reusable stages and fairings, saving NASA money.
  - 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 that this study find answers
  - Correlations between each rocket variables and successful landing rate.
  - Conditions to get the best results and ensure the best successful landing rate.
  - Train a machine learning model to predict successful Stage 1 recovery.

Section 1

Methodology

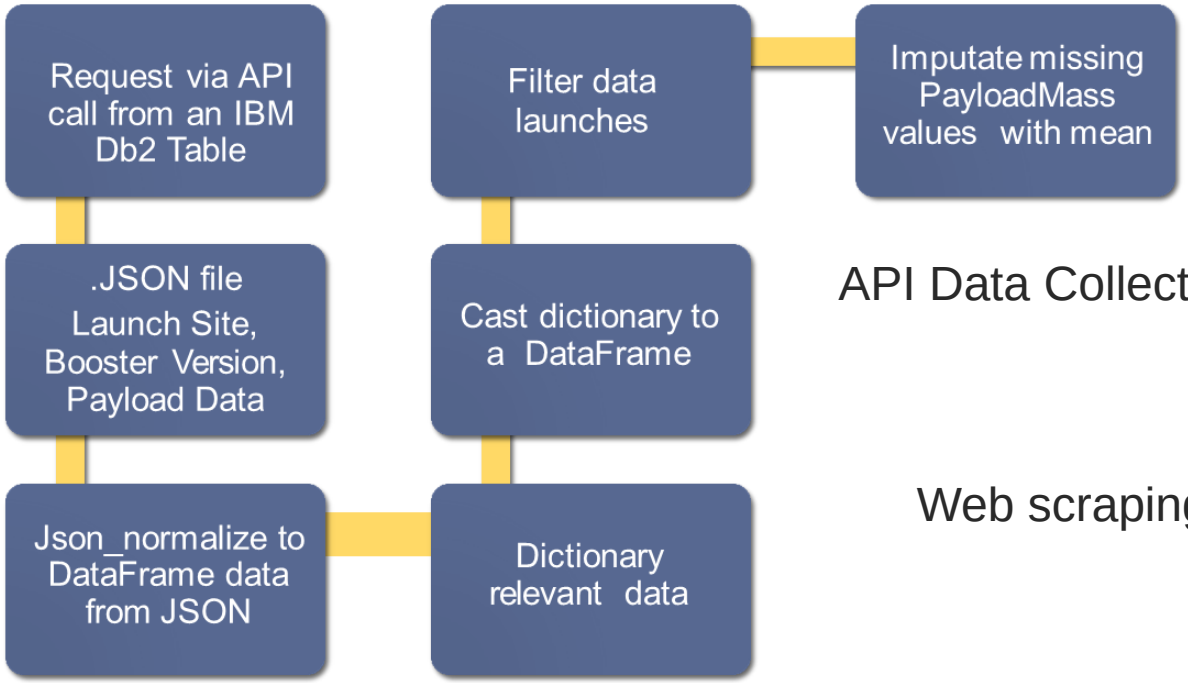
Methodology

Executive Summary

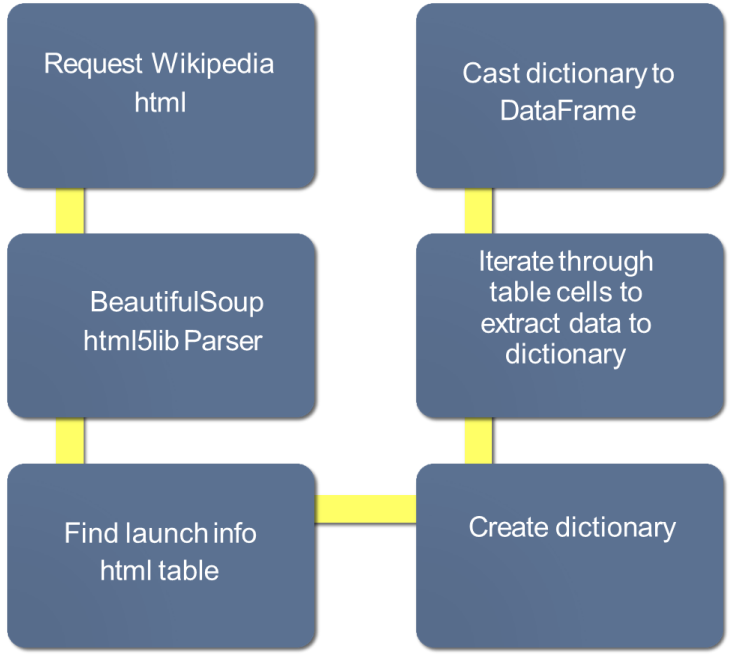
- Data collection methodology:
  - SpaceX API data
  - Web Scraping Data from tables in Wikipedia
- Perform data wrangling
  - Converting outcomes into training labels, boostering successfully/unsuccessful lands
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Tuned models using GridSearchCV, find the best Hyperparameter for SVM, Classification Trees and Logistic Regression

Data Collection

- The data collection was developed by combining:
  - requesting to the Space X API, with some basic data wrangling, formatting and cleaning of the requested data with
  - web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia



API Data Collection



Web scraping Data Collection

Data Collection – SpaceX API

- Requesting rocket launch data from SpaceX API

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

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

- Converting Response to a JSON file

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

Check the content of the response

```
print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recove
x.com/3c/0e/T8iJcSN3_o.png","large":"https://images2.imgbox.com
null,"recovery":null},"flickr":{"small":[],"original":[]},"pres
ube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spa
wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17
```

- Using custom functions to clean data

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40

- Combining the columns into a dictionary to create data frame
- Filtering df and export to a CSV

```
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
                  0
                  0
                int64
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

GitHub Link

https://github.com/Razonar/IBM-Python-Project/blob/main/01_Data_Collection_API.ipynb

Data Collection - Scraping

- Getting response from HTML

```
# use requests.get() method with the provided static_url
# assign the response to a object
F9Page = requests.get(static_url)
```

- Creating a BeautifulSoup object

```
# Use BeautifulSoup() to create a BeautifulSoup object
soup = BeautifulSoup(F9Page.text, 'html.parser')
```

- Finding all tables and assigning the result to a list

```
# Use the find_all function in the BeautifulSoup object,
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

GitHub Link

https://github.com/Razonar/IBM-Python-Project/blob/main/02_Data_Collection_With_Web_Scraping.ipynb

- Extracting column name one by one

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all(
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip
                flag=flight_number.isdigit()
        else:
```

- Creating an empty dictionary with keys
- Filling up the dict with launch records

```
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

- Creating a df and exporting it to a CSV

```
df=pd.DataFrame(launch_dict)
```

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- Cases in which the booster failed to successfully land on the dataset, attempting to land but failed because of accident.
  - True / False Ocean: the mission result has / not has successfully landed in a specific area of the ocean
  - True / False RTLS: the mission result successfully landed / not landed on the ground pad
  - True / False ASDS: the mission result has successfully landed / not landed on the drone ship

https://github.com/Razonar/IBM-Python-Project/blob/main/03_Data_Wrangling.ipynb

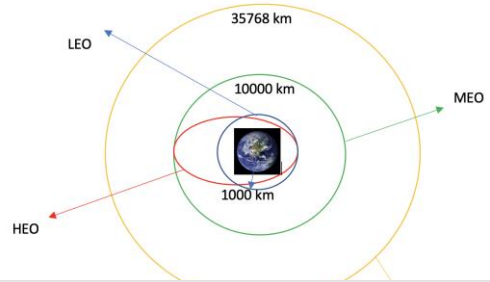
- Converting these results into training labels:
  - 1 = successful / 0 = failure

```
df["Orbit"].value_counts()
```

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

```
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
```




```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
def onehot(item):
    if item in bad_outcomes:
        return 0
    else:
        return 1
landing_class = df["Outcome"].apply(onehot)
landing_class
```

```
0     0
1     0
2     0
3     0
4     0
     ..
```

EDA with SQL

- The dataset is load into a table in a Db2 IBM cloud database, and is requested by SQL queries to answer this questions:

  - 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
  - Listing 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
  - List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - Rank the count of landing outcomes

GitHub Link

https://github.com/Razonar/IBM-Python-Project/blob/main/04_EDA_With_SQL.ipynb

EDA with Data Visualization

- Scatter charts:
  - The purpose of scatter plots is to shows how much one variable is affected by another, showing their correlation, if there is anyone.
    - FLight Number vs. Launch Site
    - Pay Load vs. Launch Site
    - FLight Number vs. Orbit Type
    - Pay Load vs. Orbit Type
- Bar chart:
  - The purpose of bar chart is to makes it easy to compare datasets between multiple groups at a glance.
    - Orbit Type vs. Success Rate
- Line chart:
  - The purpose of line charts is to shows data variables and trends clearly to helps predict the results of data that has not yet been recorded.
    - Year vs. Success Rate

https://github.com/Razonar/IBM-Python-Project/blob/main/05_EDA_With_Data_Visualization.ipynb

Build an Interactive Map with Folium

- With folium maps we can mark Launch Sites, successful and unsuccessful landings, and a proximity example to key locations: Railway, Highway, Coast, and City, allowing us to understand why launch sites may be located where they are.

- Objects created and added to a folium map:
  - Markers that show all launch sites on a map
  - Markers that show the success/failed launches for each site on the map
  - Lines that show the distances between a launch site to its proximities

Data Collection - API
Data Collection - Web Scraping
Data Wrangling
EDA - SQL
EDA - Data Visualization
Interactive Visual Analytics - Folium
Interactive Dashboard - Plotly
Machine Learning Prediction

GitHub Link

https://github.com/Razonar/IBM-Python-Project/blob/main/06_Interactive_Visual_Analytics_With_Folium.ipynb

Build a Dashboard with Plotly Dash

- Dashboard includes a pie chart and a scatter plot.

- Pie chart:
- Shows total success launches by sites, to indicate a successful landing distribution across all launch sites or to indicate the success rate of individual launch sites.

- Scatter chart:
- Shows the relationship between Outcomes and Payload mass by different boosters to determine how success depends on the launch point, payload mass, and booster version categories.

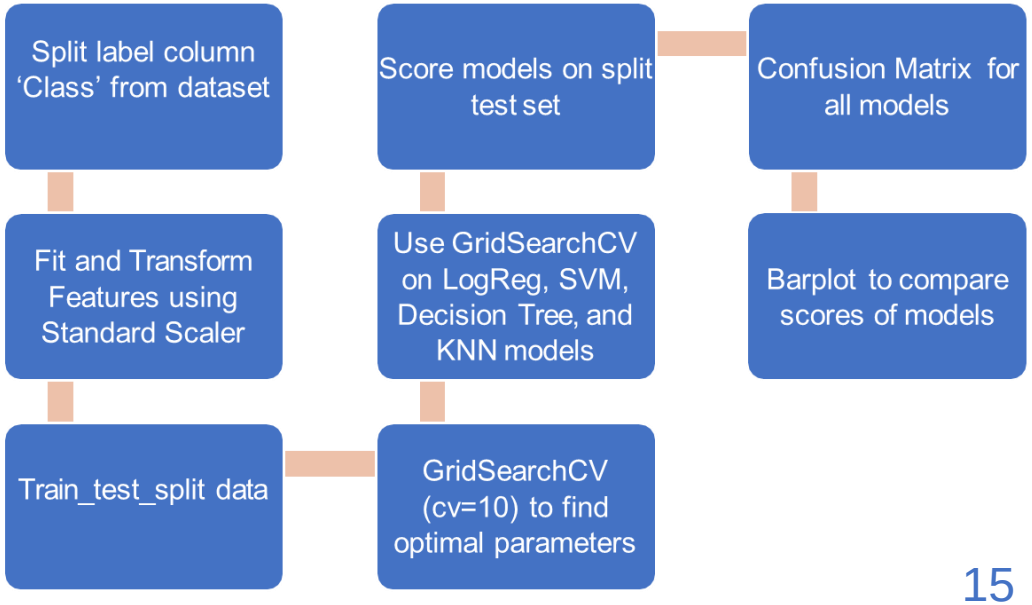
Data Collection - API
Data Collection - Web Scraping
Data Wrangling
EDA - SQL
EDA - Data Visualization
Interactive Visual Analytics - Folium
Interactive Dashboard - Plotly
Machine Learning Prediction

https://github.com/Razonar/IBM-Python-Project/blob/main/07_Interactive_Dashboard_With_Ploty.py

Predictive Analysis (Classification)

- For determine Training Labels
  - Create a column for the class
  - Standardize the data
  - Split into training data and test data
- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
  - Find the method performs best using test data

Data Collection - API
Data Collection - Web Scraping
Data Wrangling
EDA - SQL
EDA - Data Visualization
Interactive Visual Analytics - Folium
Interactive Dashboard - Plotly
Machine Learning Prediction

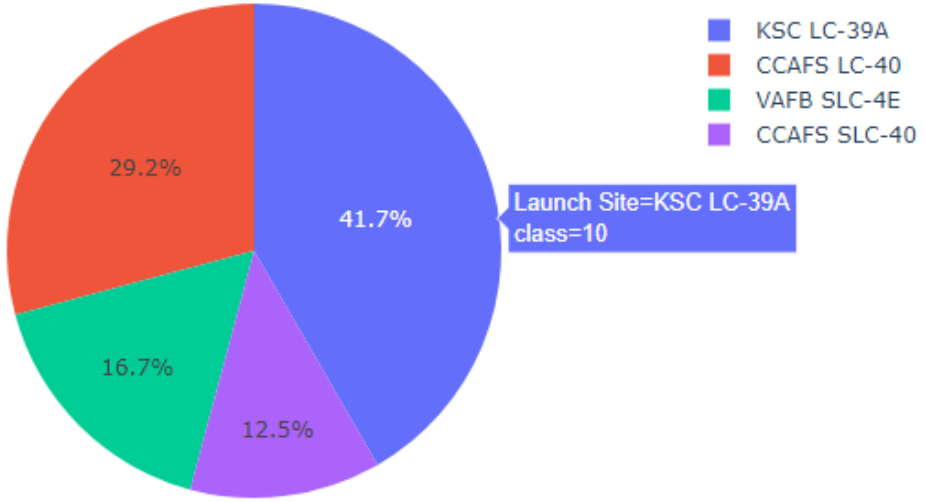


GitHub Link

https://github.com/Razonar/IBM-Python-Project/blob/main/08_Machine_Learning_Prediction.ipynb

Results

- Exploratory data analysis results



- Interactive analytics

- Predictive analysis results
  - Comparing the accuracy of the four methods, all return the same accuracy of about 83% for test data

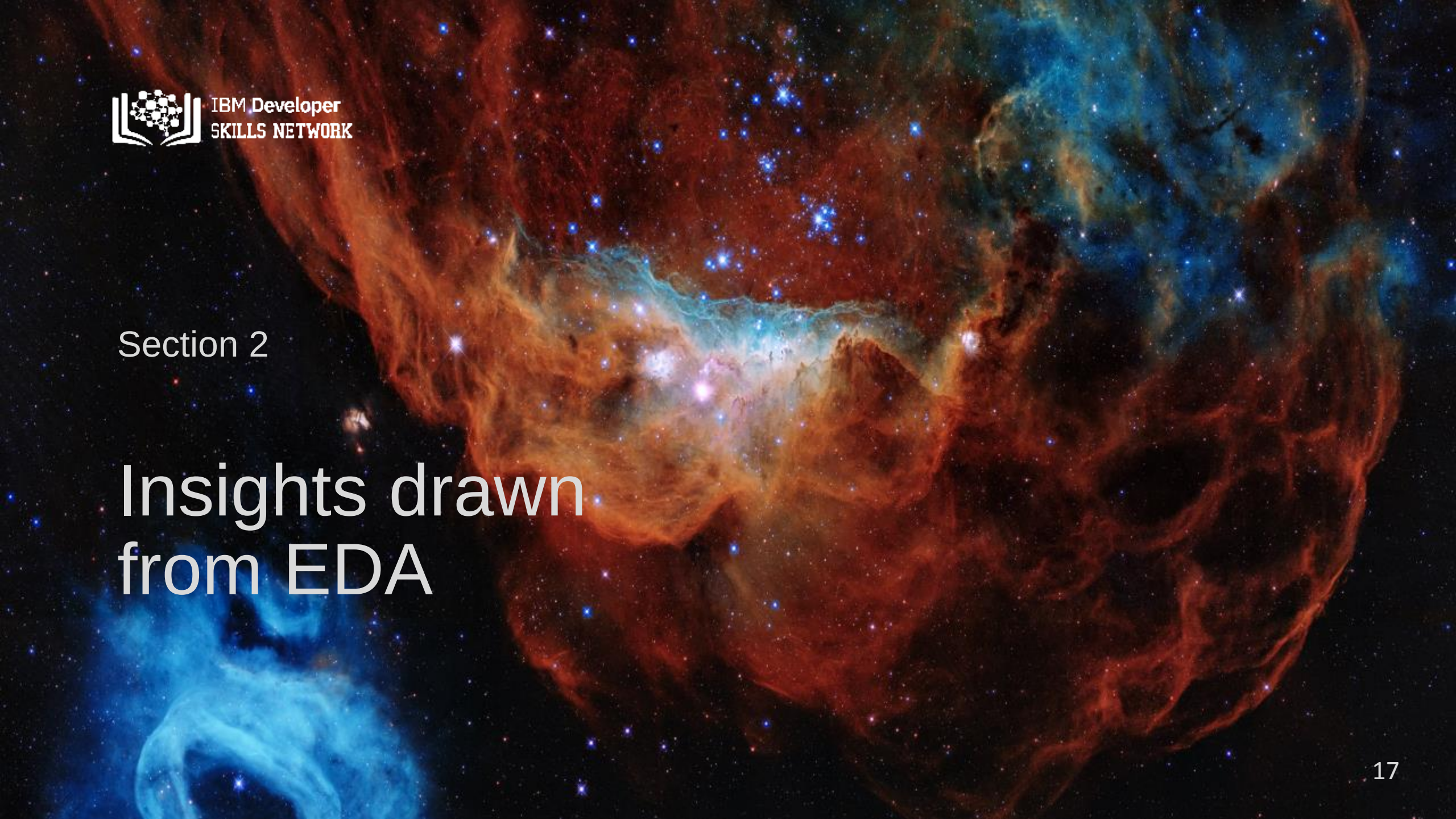
Data Collection - API
Data Collection - Web Scraping
Data Wrangling
EDA - SQL
EDA - Data Visualization
Interactive Visual Analytics - Folium
Interactive Dashboard - Plotly
Machine Learning Prediction

GitHub Link

https://github.com/Razonar/IBM-Python-Project/blob/main/08_Machine_Learning_Prediction.ipynb

```
print(methods)
print(acc)
```

```
['logistic regression', 'support vector machine', 'decision tree classifier', 'k nearest neighbors']
[0.8333333333333334, 0.8333333333333334, 0.8333333333333334, 0.8333333333333334]
```

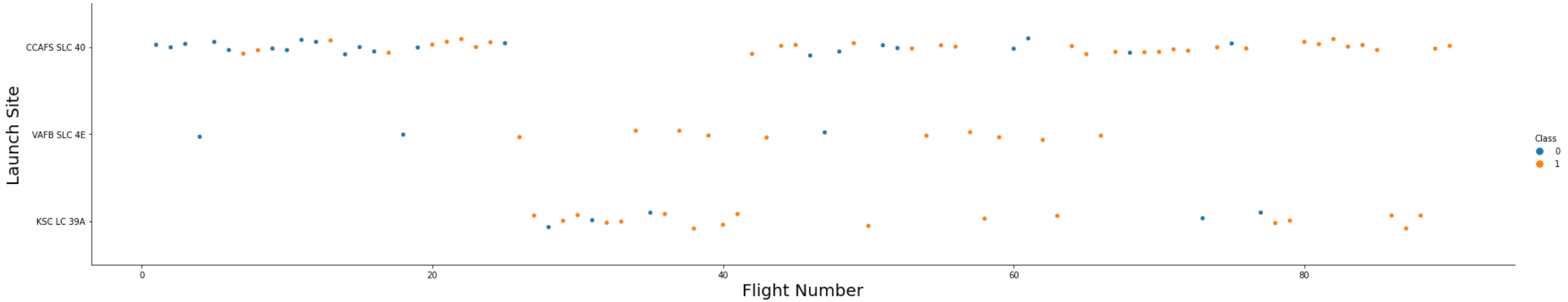


Section 2

Insights drawn from EDA

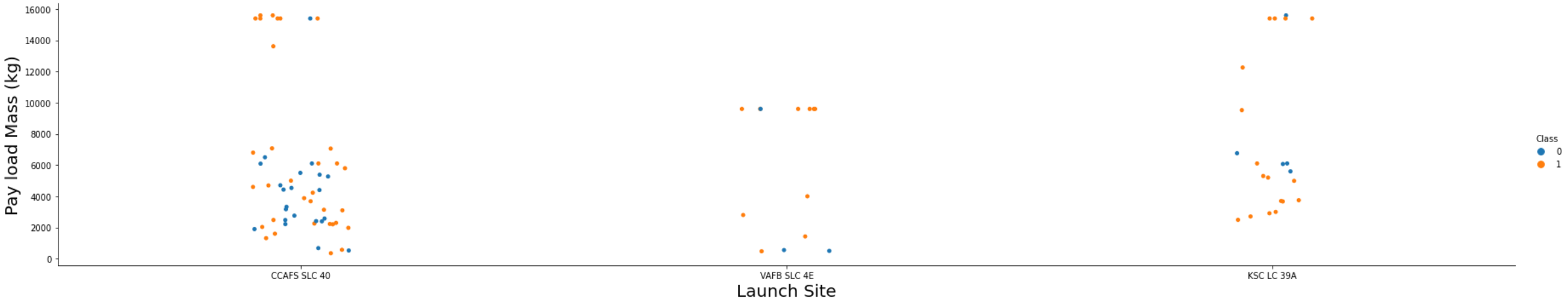
Flight Number vs. Launch Site

- Class 0 represents unsuccessful launch, Class 1 represents successful launch.
- The success rate increased as the number of flights increased.
- The success rate has increased considerably since the 20th flight



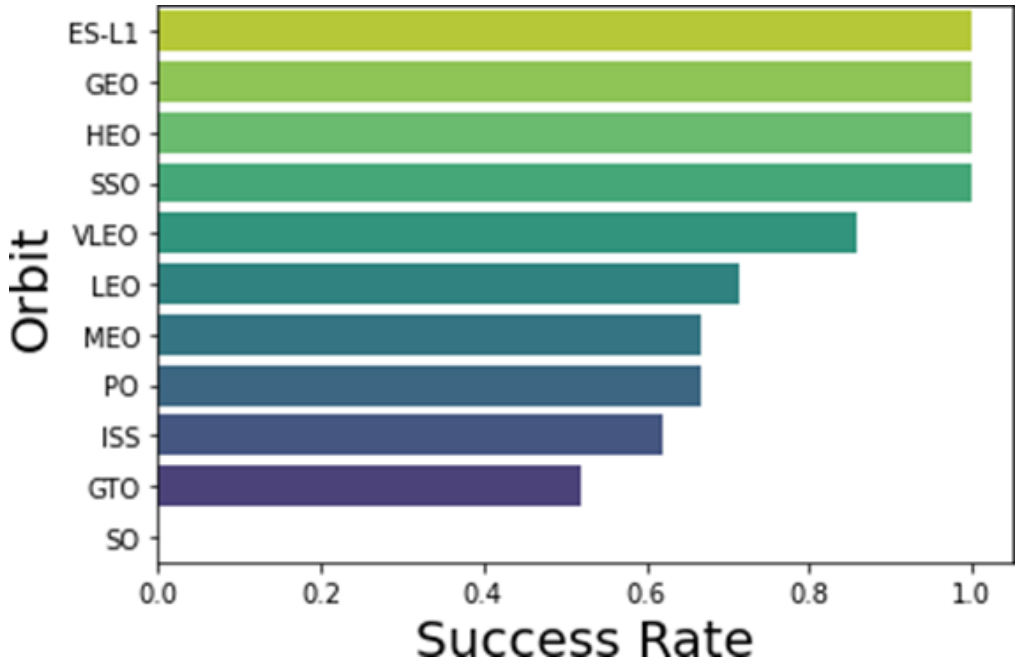
Payload vs. Launch Site

- Class 0 represents unsuccessful launch, Class 1 represents successful launch.
- It is difficult to make decisions based on this analysis because there are no clear pattern between successful launch and Pay Load Mass.



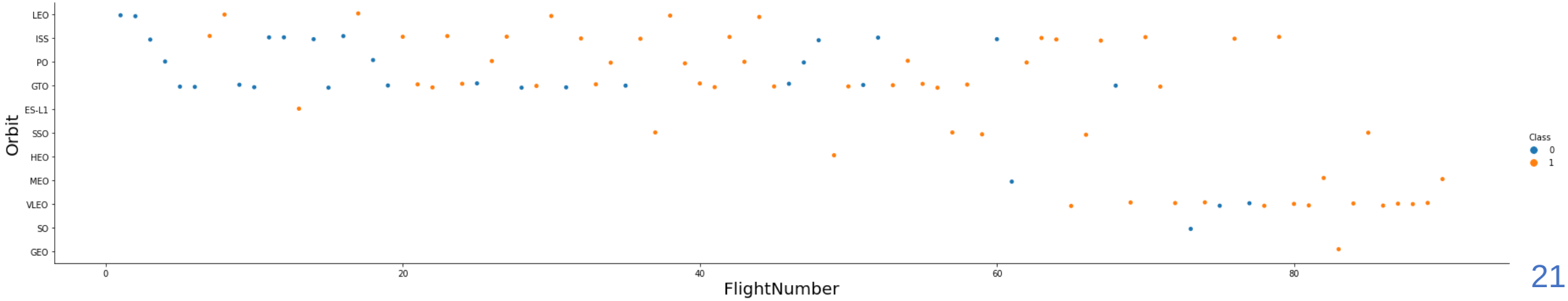
Success Rate vs. Orbit Type

- Orbit types SSO, HEO, GEO, and ES-L1 have success rates of 100%.
- The success rate of orbit type GTO is only 50%, and type SO with only one attempt have 0%



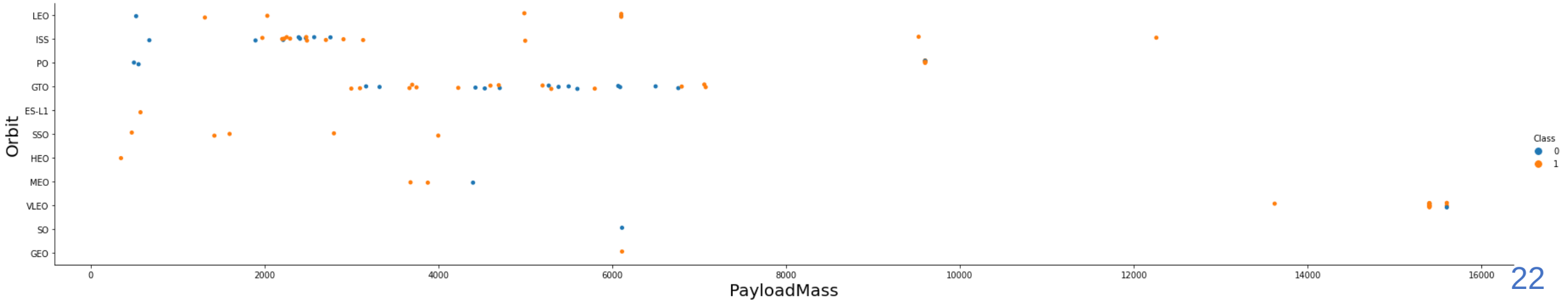
Flight Number vs. Orbit Type

- Class 0 represents unsuccessful launch, Class 1 represents successful launch.
- Starting with LEO with a moderate success rate, then with VLEO gets a high success rate, used in the more recent launches



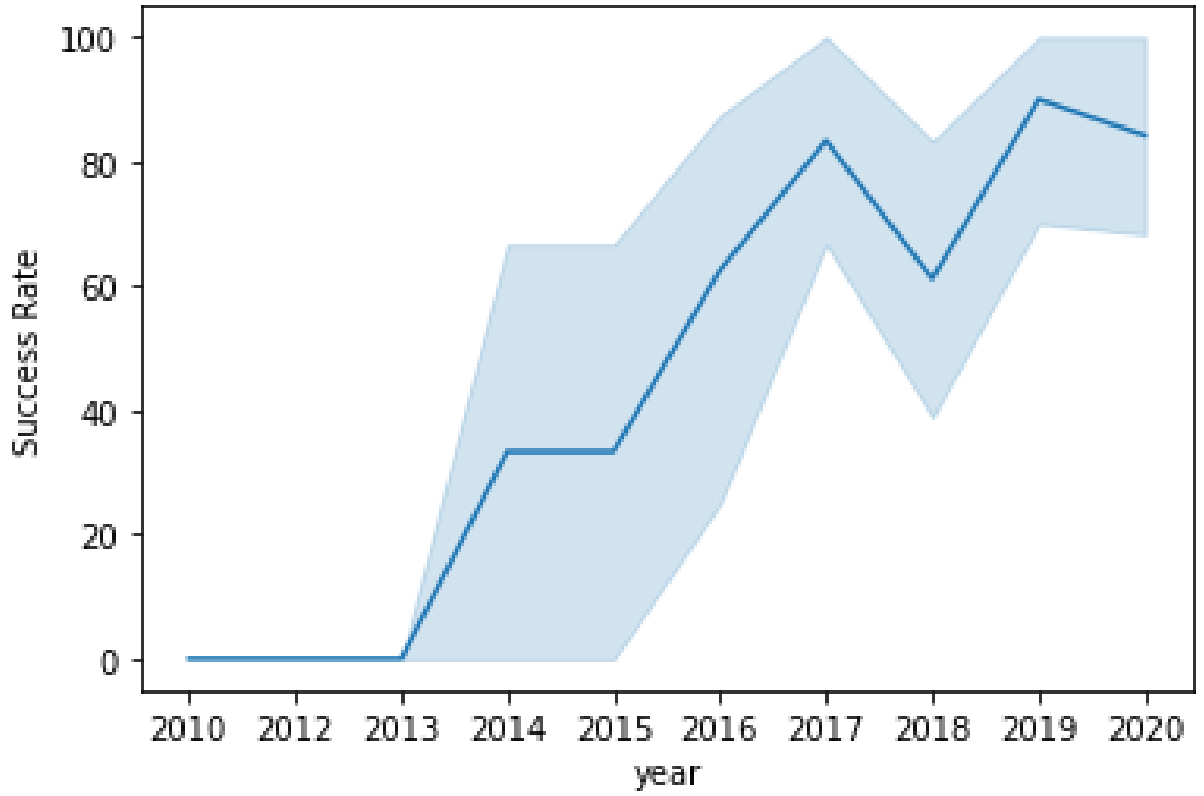
Payload vs. Orbit Type

- With heavy payloads the successful landing rate is bigger for LEO and ISS.



Launch Success Yearly Trend

- The success rate has continued to increase in the period 2013 - 2017.
- The rate decreased slightly in 2018.
- The last rate is about 80%.



All Launch Site Names

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

```
 * ibm_db_sa://mbq78074:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

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

%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL

Launch Site Names Begin with 'CCA'

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

```
 * ibm_db_sa://mbq78074:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	cus
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	S
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)	
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	

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

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass_kg FROM SPACEXTBL \
```

```
 * ibm_db_sa://mbq78074:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08k
qb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

total_payload_mass_kg
45596

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

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS avg_payload_mass_kg FROM SPACEXTBL WHI
```

```
 * ibm_db_sa://mbq78074:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08k
qb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

avg_payload_mass_kg
2928

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

First Successful Ground Landing Date

```
%sql SELECT MIN(DATE) AS first_successful_landing_date FROM SPACEXTBL WHERE
```

```
 * ibm_db_sa://mbq78074:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08k
qb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

first_successful_landing_date
2015-12-22

```
%sql SELECT MIN(DATE) AS first_successful_landing_date FROM
SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Succes
```

```
 * ibm_db_sa://mbq78074:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08k
qb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

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

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (drone
ship)' AND (PAYLOAD_MASS__KG_ BETWEEN 4000
AND 6000)
```

Total Number of Successful and Failure Mission Outcomes

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

```
 * ibm_db_sa://mbq78074:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08k
qb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

mission_outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

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

Boosters Carried Maximum Payload

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

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

2015 Launch Records

```
%sql SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WH
```

```
 * ibm_db_sa://mbq78074:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08k
qb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

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

```
%sql SELECT LANDING__OUTCOME, BOOSTER_VERSION,
LAUNCH_SITE FROM SPACEXTBL WHERE LANDING__OUTCOME =
'Failure (drone ship)' AND YEAR(DATE) = '2015'
```

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

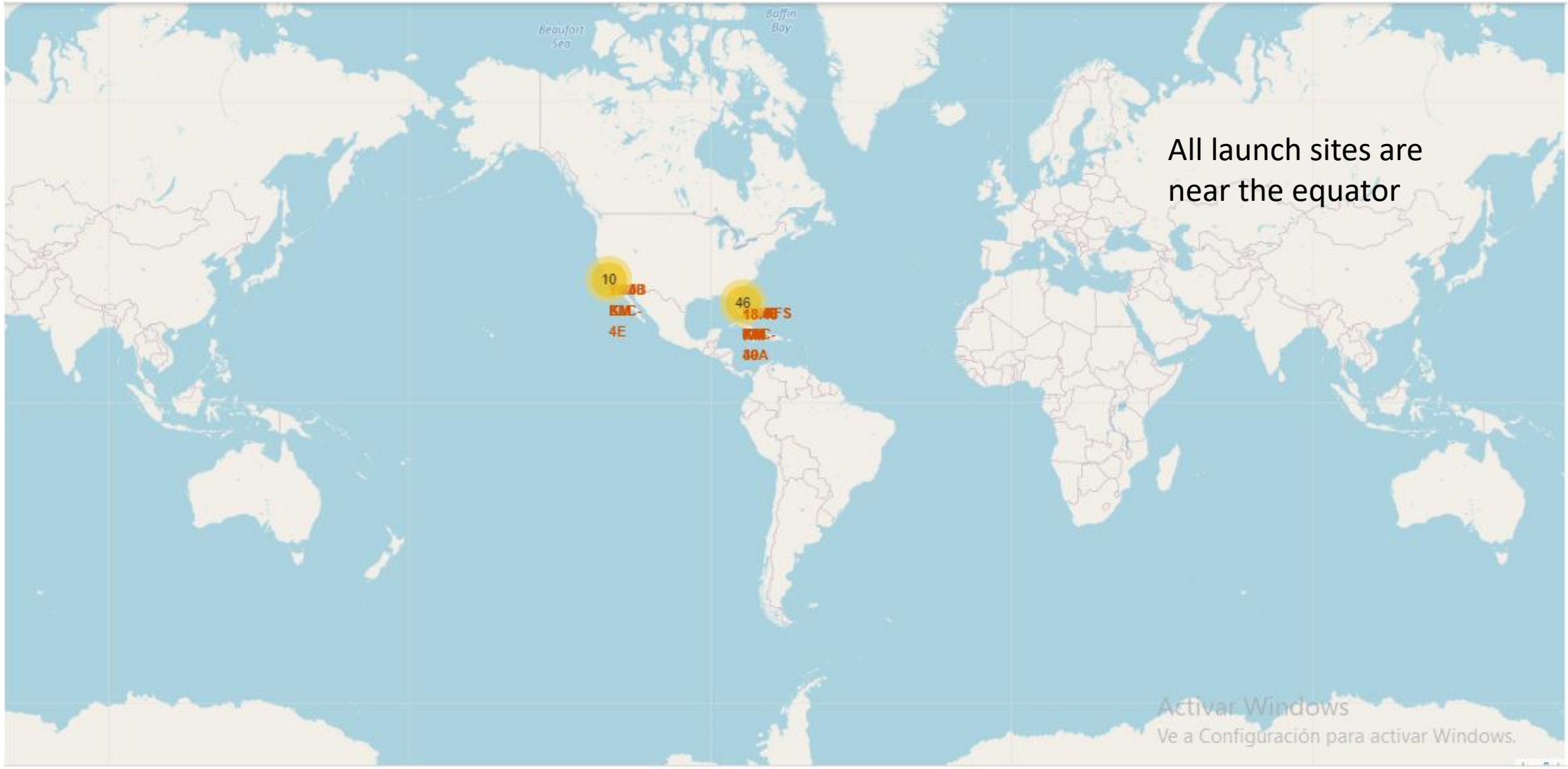
landing__outcome	total_number
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

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

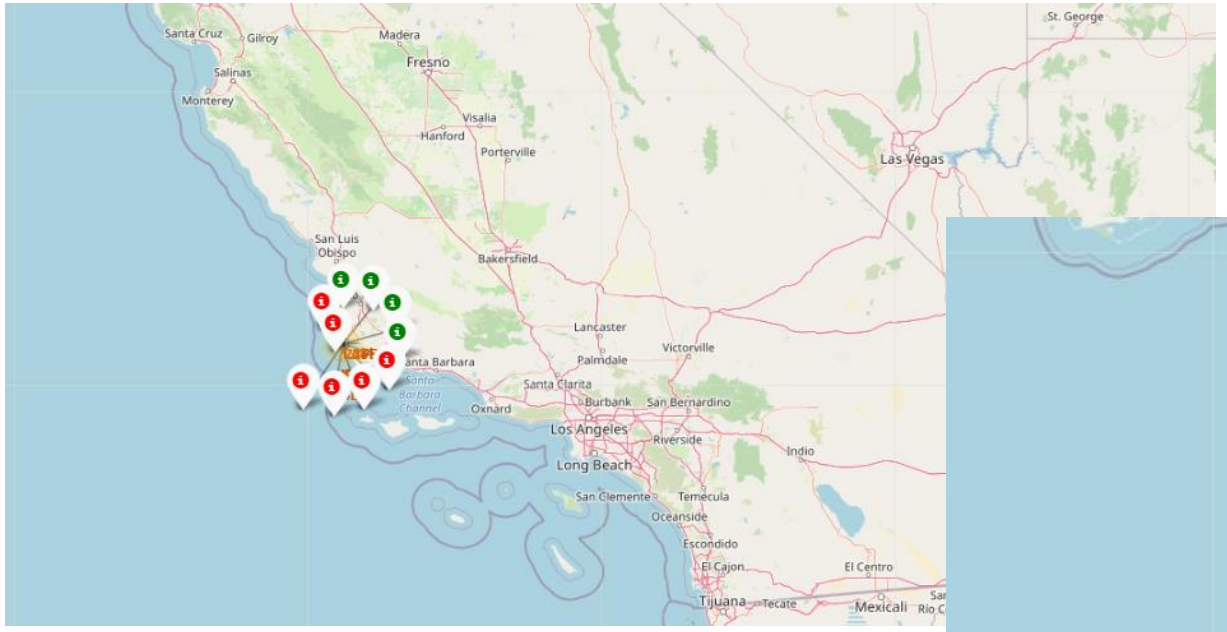
Section 3

Launch Sites Proximities Analysis

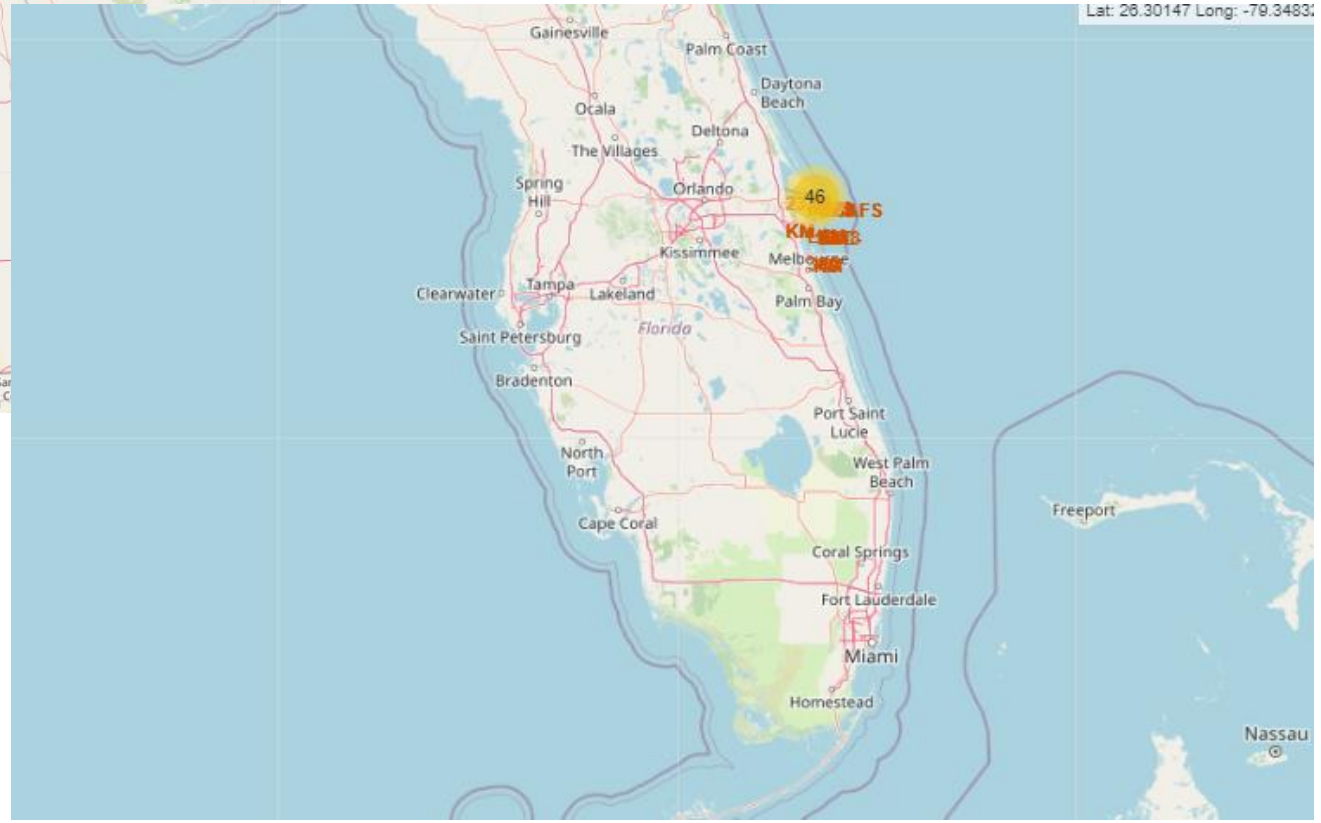
All Launch Sites Locations



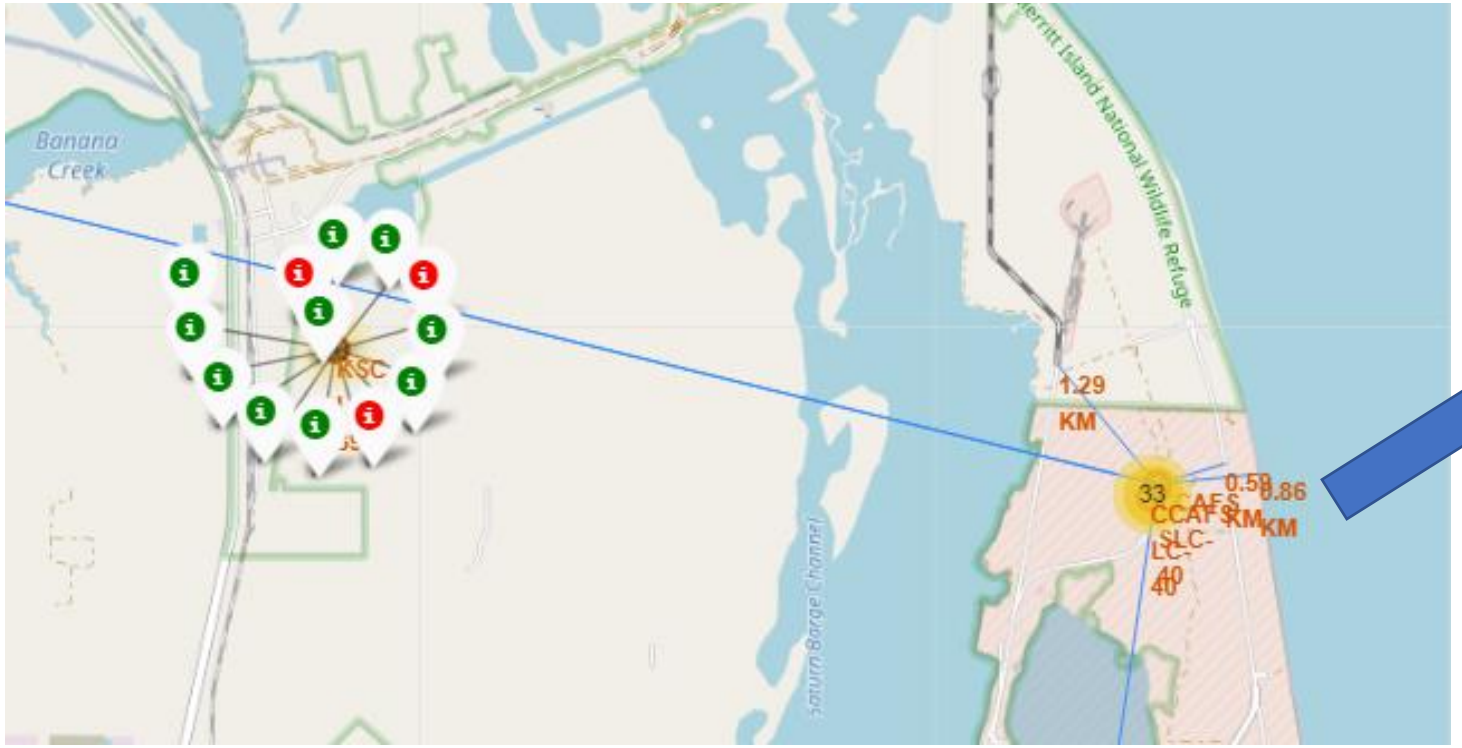
Labels of Launch Outcomes

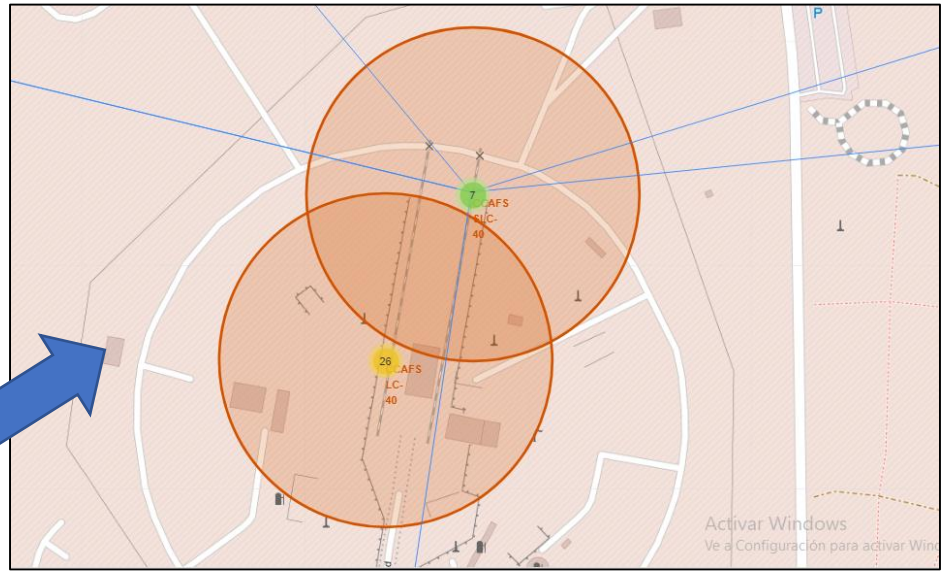


All launch sites are near the coast



Proximities of Launch Sites





The launch sites are close to railways, highways, and coastline, but far from cities.

Section 4

Build a Dashboard with Plotly Dash

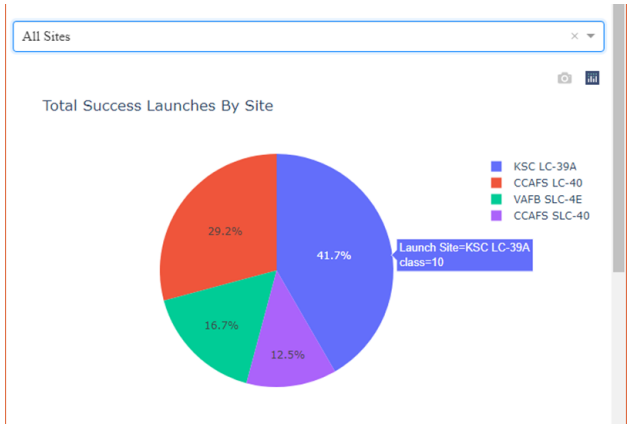
Plotly Dash Plots

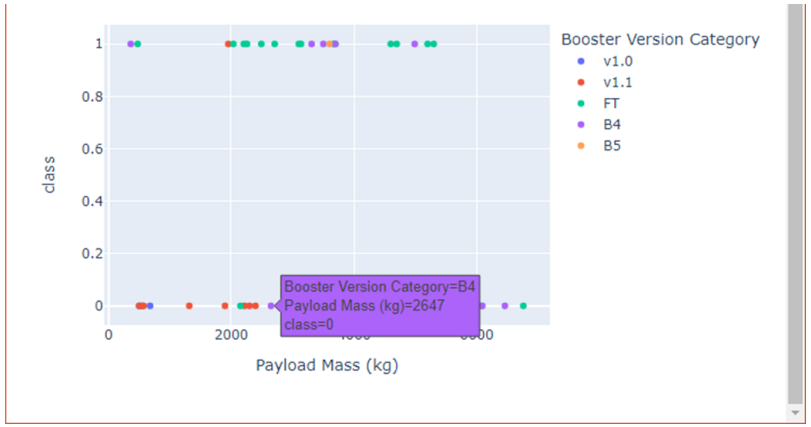
Pie chart:

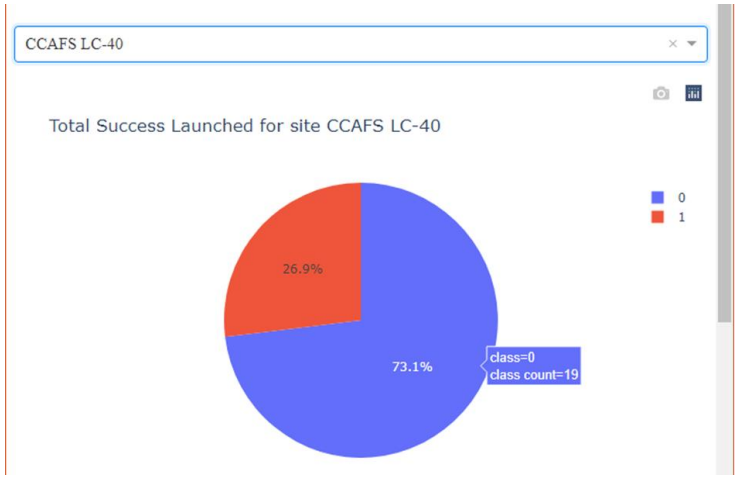
Shows total success launches by sites, to indicate a successful landing distribution across all launch sites or to indicate the success rate of individual launch sites.

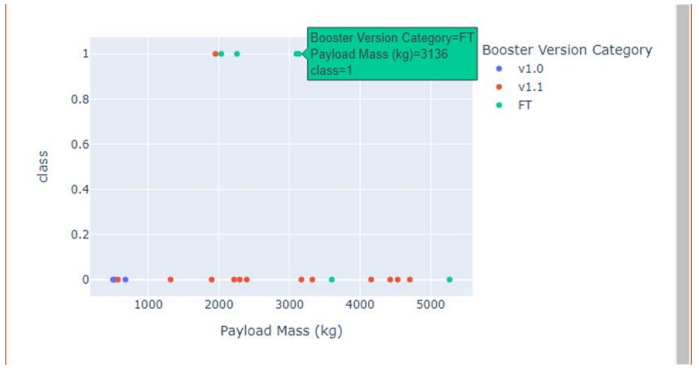
Scatter chart:

Shows the relationship between Outcomes and Payload mass by different boosters to determine how success depends on the launch point, payload mass,
and booster version categories

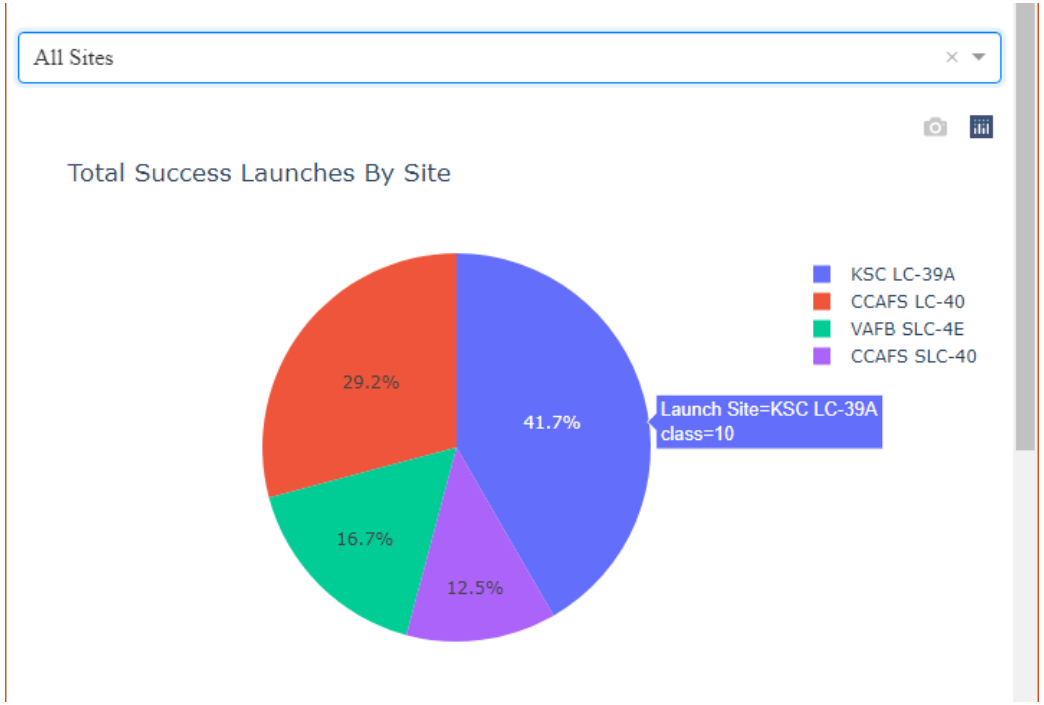


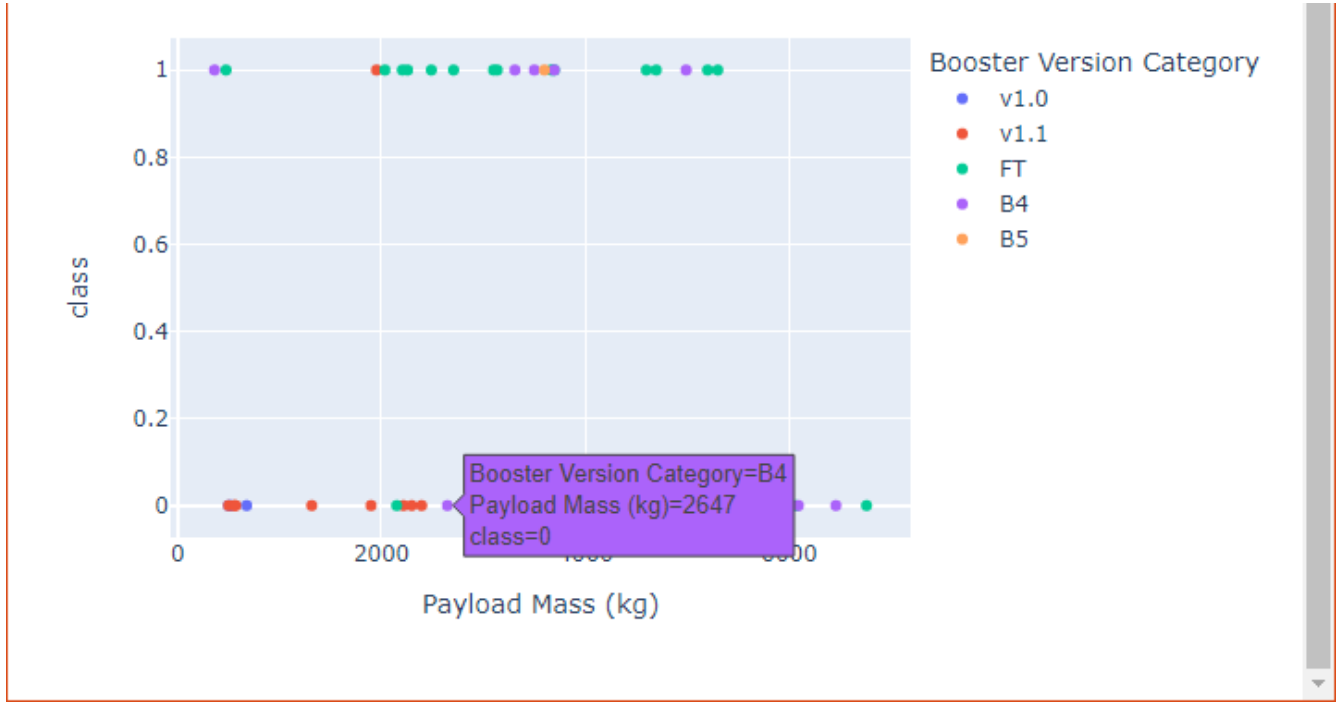




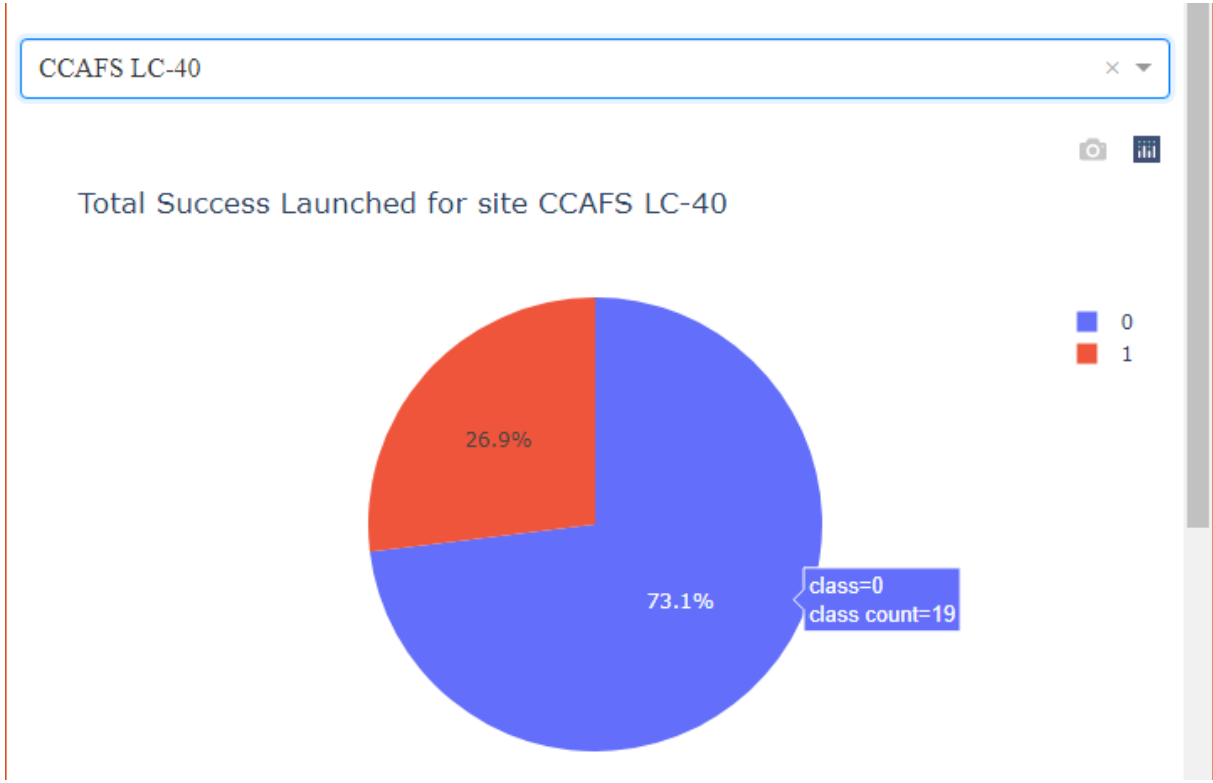


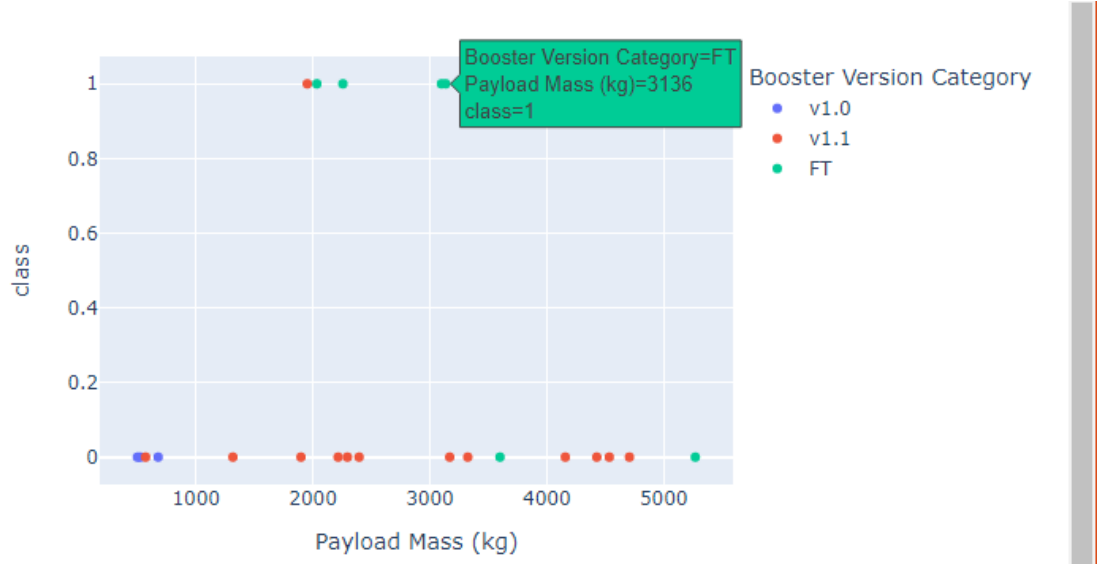
Total Success Launches By Site





CCAFS LC-40 Success Launches By Site





Section 5

Predictive Analysis (Classification)

Classification Accuracy

- In the test set, the accuracy of all models is more or less the same at 83.33%.
- Because the test size was small at 18, more data is needed to determine the optimal model.

```
acc=[]
methods=[]
acc.append(logreg_cv.score(X_test,Y_test))
methods.append('logistic regression')
logreg_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

```
acc.append(svm_cv.score(X_test,Y_test))
methods.append('support vector machine')
svm_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

```
acc.append(tree_cv.score(X_test,Y_test))
methods.append('decision tree classifier')
tree_cv.score(X_test,Y_test)
```

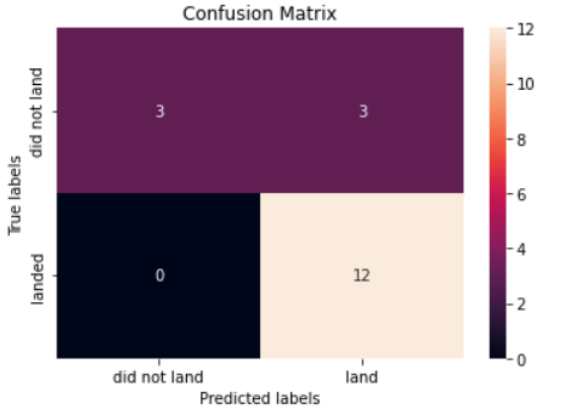
```
0.8333333333333334
```

```
acc.append(knn_cv.score(X_test,Y_test))
methods.append('k nearest neighbors')
knn_cv.score(X_test,Y_test)
```

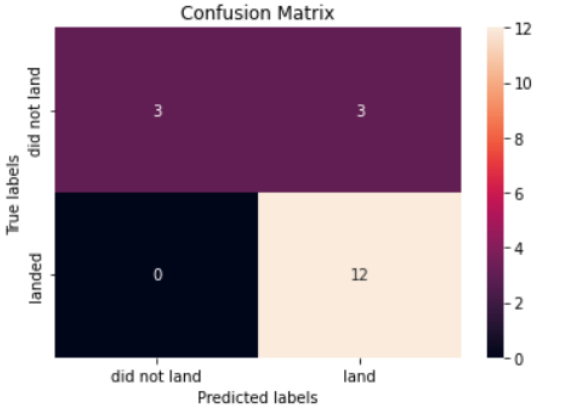
```
0.8333333333333334
```

Confusion Matrix

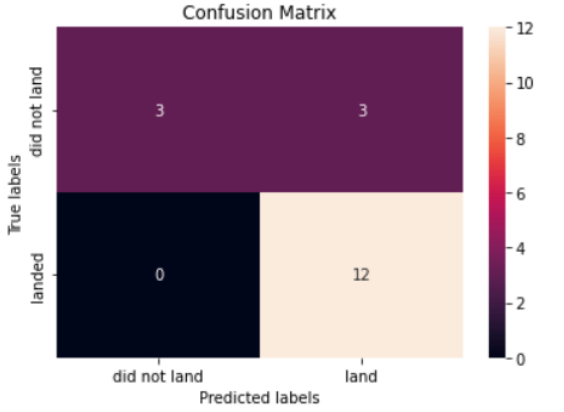
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



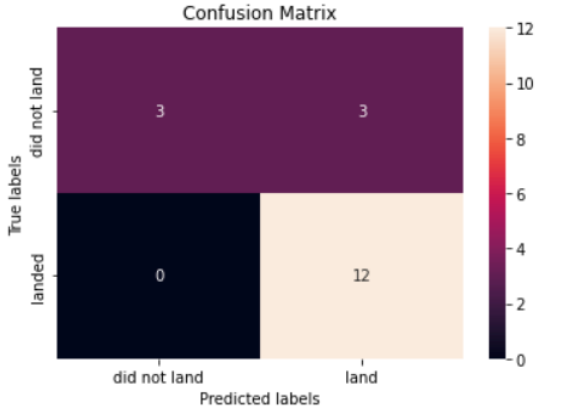

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```




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




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



- The confusion matrix is the same for all models because they performed the same for the test set.
- These models predict successful landings.

Conclusions

- The success rate grows from the first years, rising to a vale that exceeded 80%.
- Orbital types SSO, HEO, GEO, and ES-L1 have a success rate of 100%.
- The launch sites are close to railways, highways, and coastline, but far from cities.
- KSLC-39A has the highest number of launch successes and the highest success rate among all sites.
- All models have more or less the same accuracy of 83.33%.

```
print(methods)
print(acc)
```

```
['logistic regression', 'support vector machine', 'decision tree classifier', 'k nearest neighbors']
[0.8333333333333334, 0.8333333333333334, 0.8333333333333334, 0.8333333333333334]
```

Appendix

- 01_Data_Collection_API.ipynb
- 02_Data_Collection_With_Web_Scraping.ipynb
- 03_Data_Wrangling.ipynb
- 04_EDA_With_SQL.ipynb
- 05_EDA_With_Data_Visualization.ipynb
- 06_Interactive_Visual_Analytics_With_Folium.ipynb
- 07_Interactive_Dashboard_With_Ploty.py
- 08_Machine_Learning_Prediction.ipynb


IBM Developer
SKILLS NETWORK
Thank you!