IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jose Miguel Redondo Romero
August 28, 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection using API and Web Scraping
  - Data Wrangling using Pandas and Numpy
  - Exploratory data analysis using SQL
  - Data Visualization using Pandas, Matplolib, Seaborn, Folium and Ploty Dash
  - Machine learning prediction using Sklearn (Logisting Regression, Support Vector Machine “SVM”, Decision Tree, K Nearest Neighbor “KNN”)
- Summary of all results
  - Success rate constantly improved since 2010 reaching more than 80% in 2020
  - Orbit influences the success rate
  - With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS
  - It’s possible to use Logistic Regression, SVM, Decision Tree and KNN to predict if the landing will be successful with an accuracy of above 80%.

Introduction

Project background and context

- Let's predict if the Falcon 9 first stage will land successfully. 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 you want to find answers

- we want to determine if the first stage will land, we can determine the cost of a launch.

Section 1

Methodology

Methodology

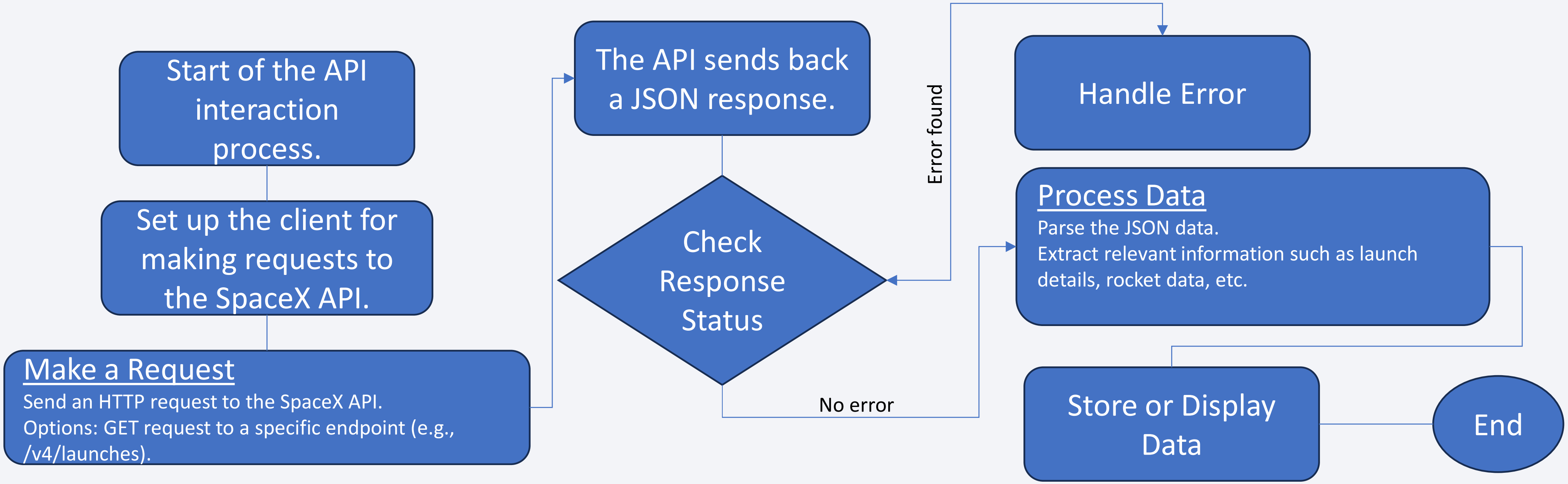
Executive Summary

- Data collection methodology:
  - API and Web Scraping used to collect the data
- Perform data wrangling
  - Pandas and Numpy used to perform data wrangling
- 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, Support Vector Machine, Decision Tree, KNN used
  - Accuracy calculated with Score method and visualized with the confusion matrix

Data Collection

- Space X data source: API_call_spacex_api.json
- Decoded the response content as a Json using .json() and turn it into a Pandas data frame .json_normalize()
- Applied getBoosterVersion function method to get the booster version
- Constructed our dataset using the data we have obtained. Combined the columns into a dictionary.
- Created the Pandas data frame from the dictionary
- Filtered the data in order to select only the Falcon 9 launches (scope for this study)
- Use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns rocket, payloads, launch pad and cores.

Data Collection – SpaceX API



- Check [this GitHub URL]of the completed SpaceX API calls notebook as an external reference and peer-review purpose

Data Collection | Scraping

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

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

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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data,'html.parser')
```

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

```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`,

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

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[9]
print(first_launch_table)
```

...

You should able to see the columns names embedded in the table header elements `<th>` as follows:

- Data collected using GET function and BeatifulSoup
- GitHub Jupyter Notebook link https://github.com/Mesenho/Jupyter/blob/main/Capstone2_jupyter-labs-webscraping.ipynb

Data Wrangling

- We used .read_csv to read the csv file
- We identified the data type with the fuction .dtypes
- We used the function .value_counts() to check how many launches took place from a certain launch site or with a certain orbit or had a certain outcome.
- We identified which landing outcome could be defined as positive (1) and which one could be identified as negative (0), creating an outcome label
- We calculate the mean of landing outcome thanks to the fact that we associated 1 and 0 for a positive and negative outcome respectively.

- GitHub Jupyter Notebook link: https://github.com/Mesenho/Jupyter/blob/main/Capstone3_labs-jupyter-spacex-Data wrangling.ipynb

EDA with Data Visualization

- We use scatter plots to visualize the relationships among different couples of measures like:
  - Relationship between Flight Number and Payload Mass
  - Relationship between Flight Number and Launch Site
  - Relationship between Payload Mass and Launch Site
  - Etc. check the plots in next slides
- We use bar plots to visualize the Percentage of Success for the different orbit types
- We use bar plots to visualize the Success Rate over the years

- GitHub Jupyter Notebook link:
  https://github.com/Mesenho/Jupyter/blob/main/Capstone5_edadataviz_ExploringAndPreparingData.ipynb

EDA with SQL

- Display the names of the unique launch sites in the space mission using SELECT DISTINCT
- Display 5 records where launch sites begin with the string 'CCA' using WHERE and LIMIT 5
- Display the total payload mass carried by boosters launched by NASA (CRS) using SELECT SUM and WHERE
- List the date when the first succesful landing outcome in ground pad was achieved using SELECT MIN and WHERE
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000 using SELECT DISTINCT, WHERE and AND
- Etc. check the insights draw section for check all of these with details.

- GitHub Jupyter Notebook link: https://github.com/Mesenho/Jupyter/blob/main/Capstone4_jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Map using Folium created where it’s possible to zoom in and see the different sites, showing with colors green and red the outcome of the mission.

- GitHub Jupyter Notebook link : https://github.com/Mesenho/Jupyter/blob/main/Capstone4_jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard application
  - Adding a Launch Site dropdown menu
  - Adding call back functions
  - Adding a range slider to select the payload

Predictive Analysis (Classification)

- We split the data into train and test data
- We run different models like: Logistic Regression, Support Vector Machine SVM, Decision Tree and K-Nearest Neighbor KNN
- We calculated the accuracy of each model with Score method and visualized with the confusion matrix

- GitHub Jupyter Notebook link: https://github.com/Mesenho/Jupyter/blob/main/Capstone7_SpaceX_Machine Learning Prediction_Part_5.ipynb

Results

- Success rate constantly improved since 2010 reaching more tan 80% in 2020
- Orbit influences the success rate
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS
- It’s posible to use Logistic Regresion, Support Vector Machine, Decision Tree and KNN to predict if the landing Will be successful with an accuracy of above 80%

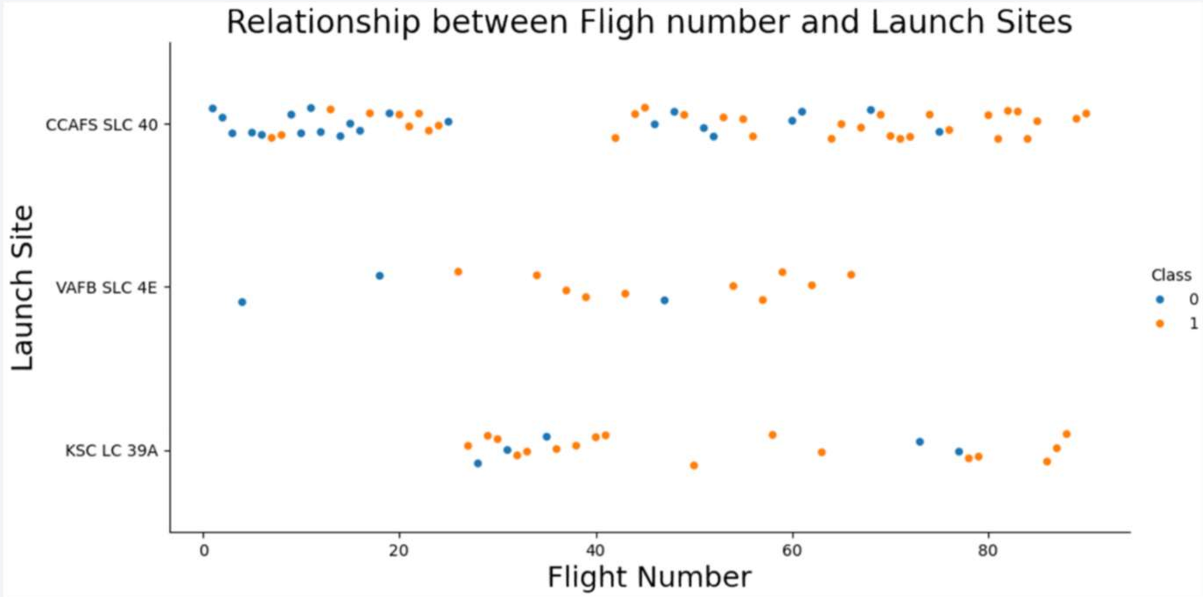
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

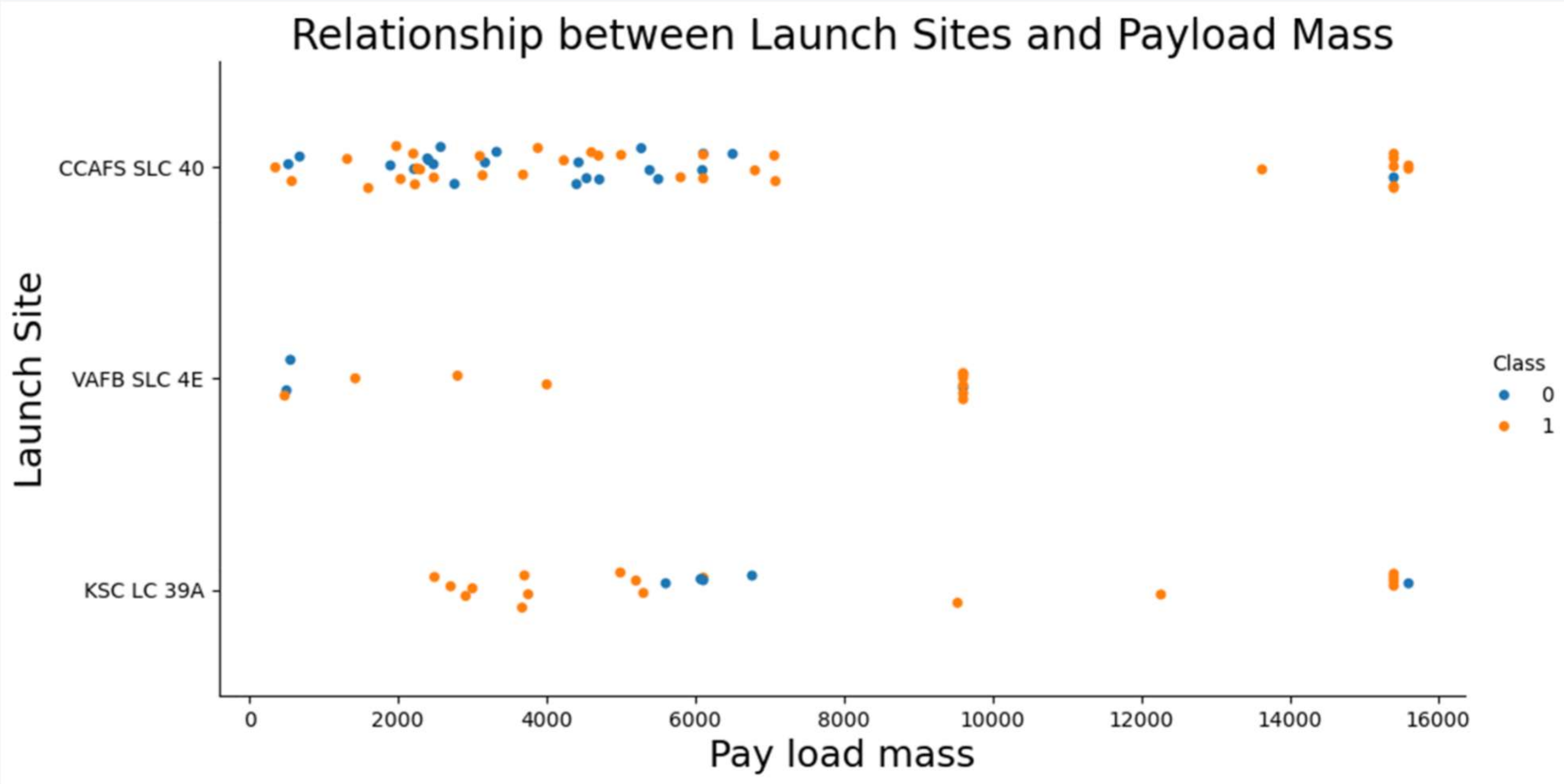
"Class" meaning:
1 = Success
0 = Failed

As we can see most of the releases have been read listed from the CCAFS SLC40 location as shown in the graph.



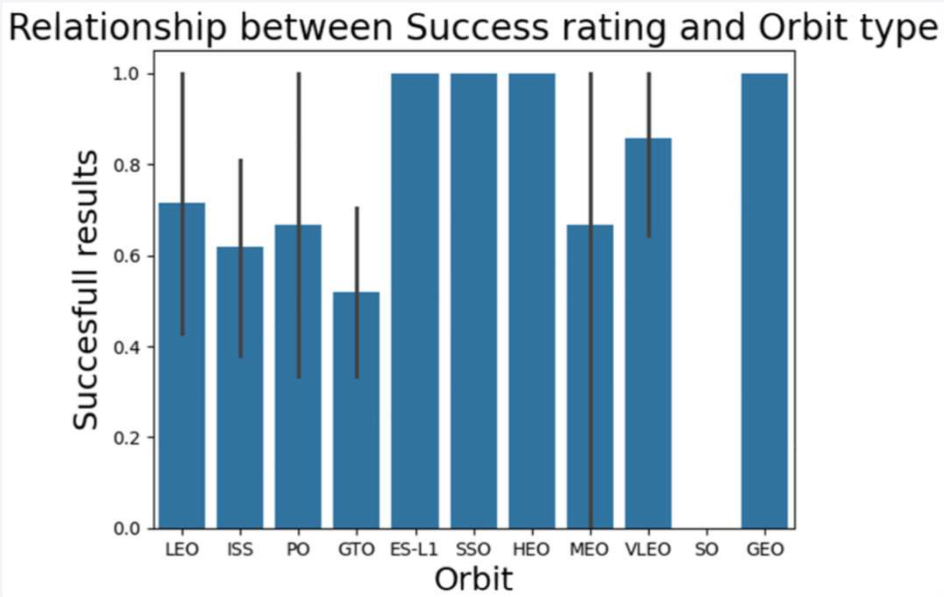
Payload vs. Launch Site

“Class” meaning:
1 = Success
0 = Failed



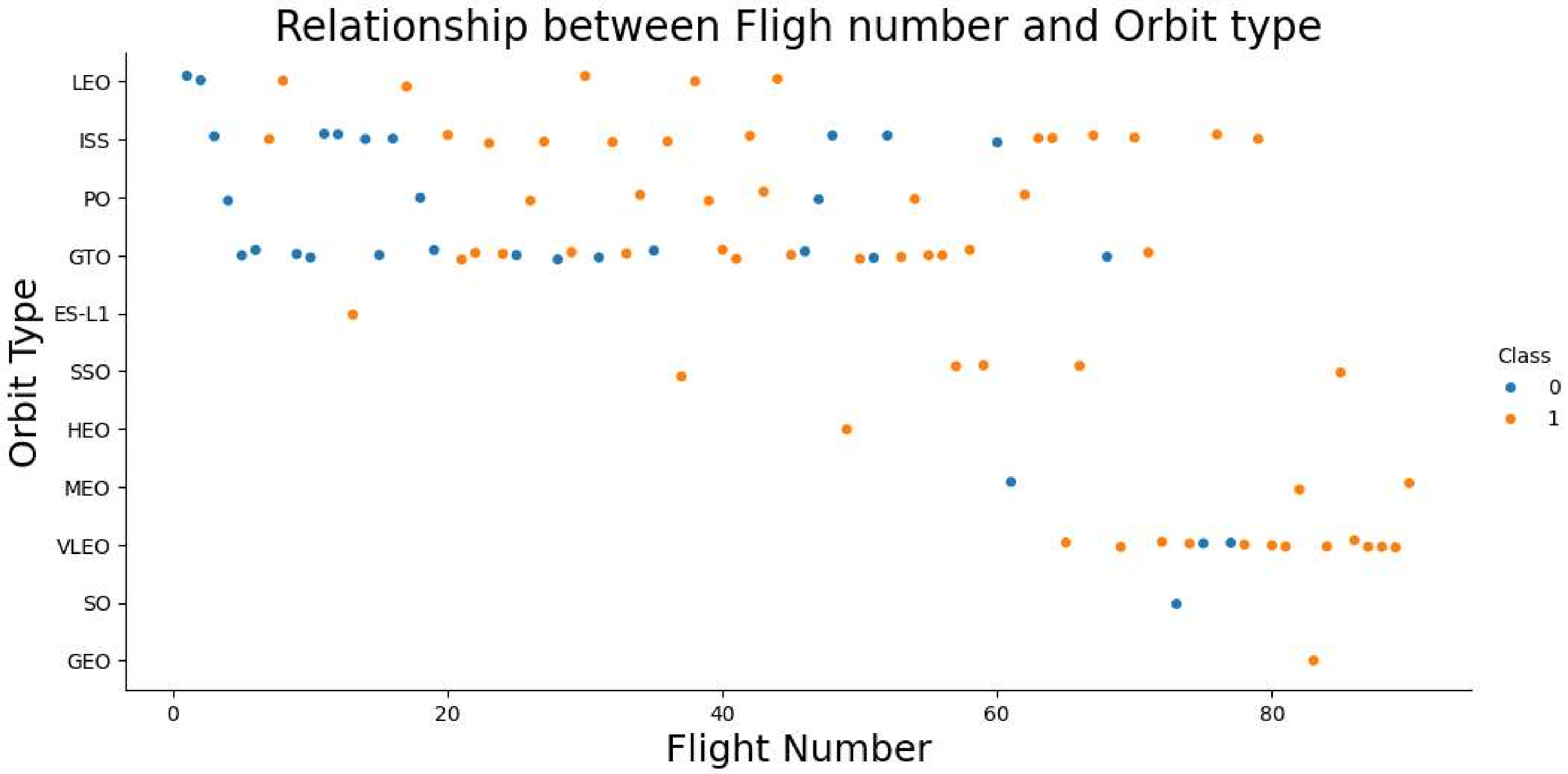
Success Rate vs. Orbit Type

We can see that there are four orbits with a 100% success rate which are the most recurrent as we can see in the next slide.



Flight Number vs. Orbit Type

"Class" meaning:
1 = Success
0 = Failed

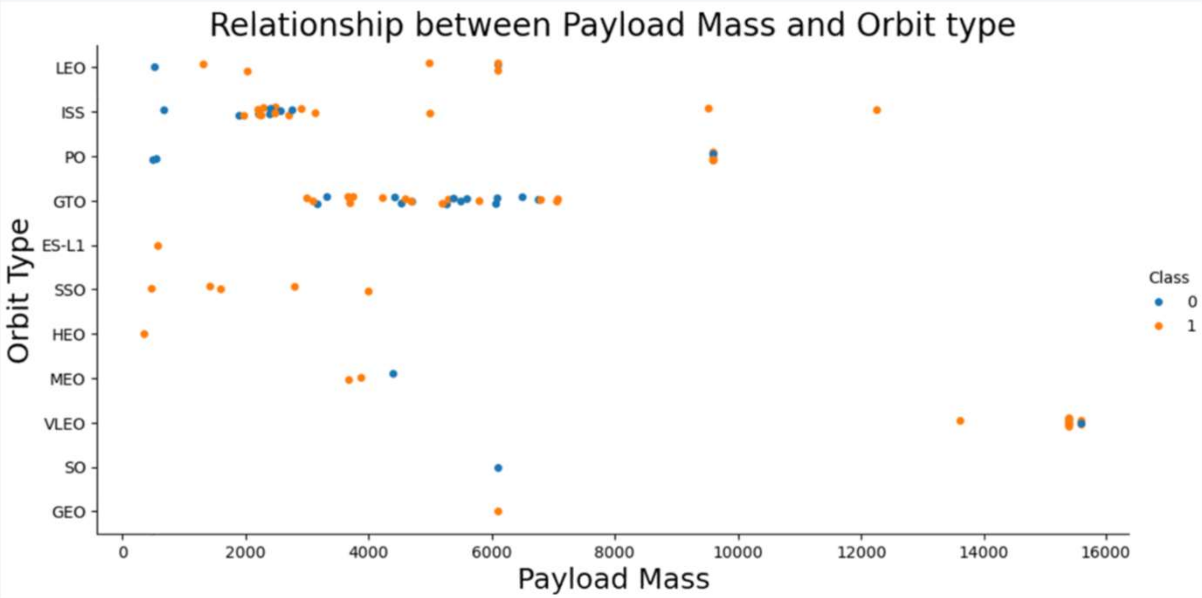


Payload vs. Orbit Type

"Class" meaning:
1 = Success
0 = Failed

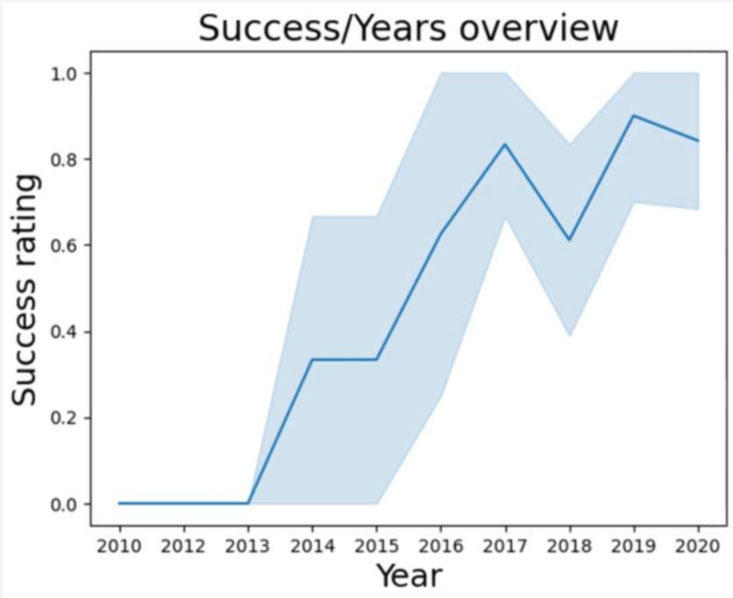
Mostly the maximum payload as shown in the graph has been up to 8000kg with a few cases with higher payloads.

In any case, with higher payloads than 8K we see that the failing cases are only two.



Launch Success Yearly Trend

Clearly, we can see that from 2013 onwards, the success rate is increasing year after year with the exception of 2018.



All Launch Site Names

- Check beside the result of different launch sites filtered accordingly
- The snipping shows the SQL sentence from the LAB used and which is the result

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

```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;
```

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

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

Launch Site Names Begin with 'CCA'

- The snipping shows the SQL sentence from the LAB used according to the instructions

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

```
%sql SELECT * FROM SPACEXTABLE 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
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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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 snipping shows the SQL sentence from the LAB used according to the instructions

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

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM "SPACEXTABLE" WHERE Customer="NASA (CRS)";
```

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

SUM("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- The snipping shows the SQL sentence from the LAB used according to the instructions

Display average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM "SPACEXTABLE" WHERE Booster_Version like "F9 v1.1%";
```

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

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

First Successful Ground Landing Date

- The snipping shows the SQL sentence from the LAB used according to the instructions

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

Hint:Use min function

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE "Landing_Outcome" like "Success%";
```

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

MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The snipping shows the SQL sentence from the LAB used according to the instructions

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

```
%sql SELECT Booster_Version FROM SPACEXTABLE 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

- The snipping shows the SQL sentence from the LAB used according to the instructions

List the total number of successful and failure mission outcomes

```
%sql SELECT COUNT(CASE WHEN Mission_Outcome like "Success%" THEN 1 END) AS total_success, COUNT(CASE WHEN Mission_Outcome like "Failure%" THEN 1 END) AS total_failure FROM SPACEXTABLE;
```

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

total_success	total_failure
100	1

Boosters Carried Maximum Payload

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

```
%sql SELECT Date, Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE);
```

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

Date	Booster_Version	PAYLOAD_MASS__KG_
2019-11-11	F9 B5 B1048.4	15600
2020-01-07	F9 B5 B1049.4	15600
2020-01-29	F9 B5 B1051.3	15600
2020-02-17	F9 B5 B1056.4	15600
2020-03-18	F9 B5 B1048.5	15600
2020-04-22	F9 B5 B1051.4	15600
2020-06-04	F9 B5 B1049.5	15600
2020-09-03	F9 B5 B1060.2	15600
2020-10-06	F9 B5 B1058.3	15600
2020-10-18	F9 B5 B1051.6	15600
2020-10-24	F9 B5 B1060.3	15600
2020-11-25	F9 B5 B1049.7	15600

- The snipping shows the SQL sentence from the LAB used according to the instructions

2015 Launch Records | failure landing_outcomes

- The snipping shows the SQL sentence from the LAB used according to the instructions

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql SELECT CASE substr(Date, 6, 2) WHEN '01' THEN 'January' WHEN '02' THEN 'February' WHEN '03' THEN 'March' WHEN '04' THEN 'April' WHEN '05' THEN 'May' WHEN '06' THEN 'June' WHEN '07' THEN 'Jul
```

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

Month_Name	Booster_Version	Launch_Site
January	F9 v1.1 B1012	CCAFS LC-40
April	F9 v1.1 B1015	CCAFS LC-40

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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%sql SELECT Landing_Outcome, COUNT(*) AS outcome_count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY outcome_count DESC;
```

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

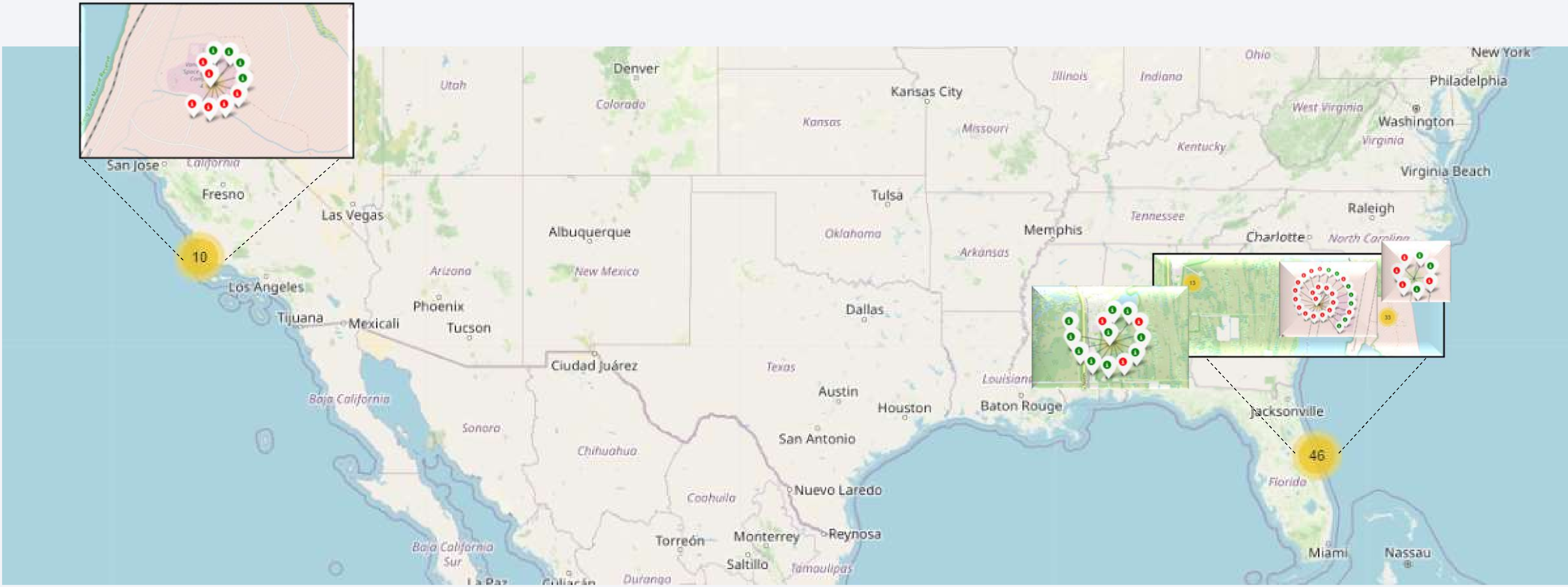
Landing_Outcome	outcome_count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

- The snipping shows the SQL sentence from the LAB used according to the instructions

Section 3

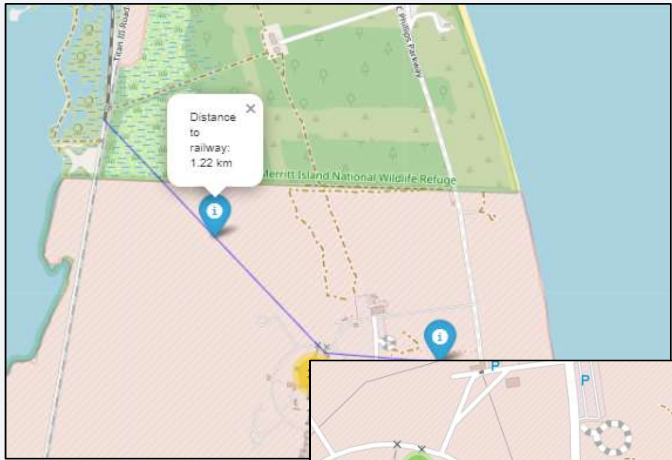
Launch Sites Proximities Analysis

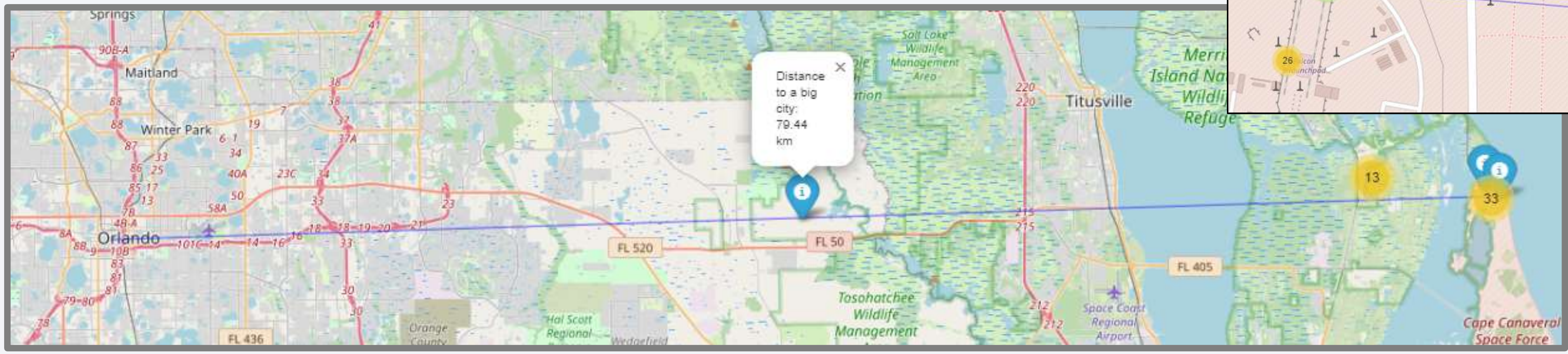
Launch sites overview | success vs failed



Distances between a launch site to its proximities

Coast line (0,8Km), **railway** (1,22Km) and **big city** (79,44Km) distance taken for analycing the key reasons (pros and cons.)

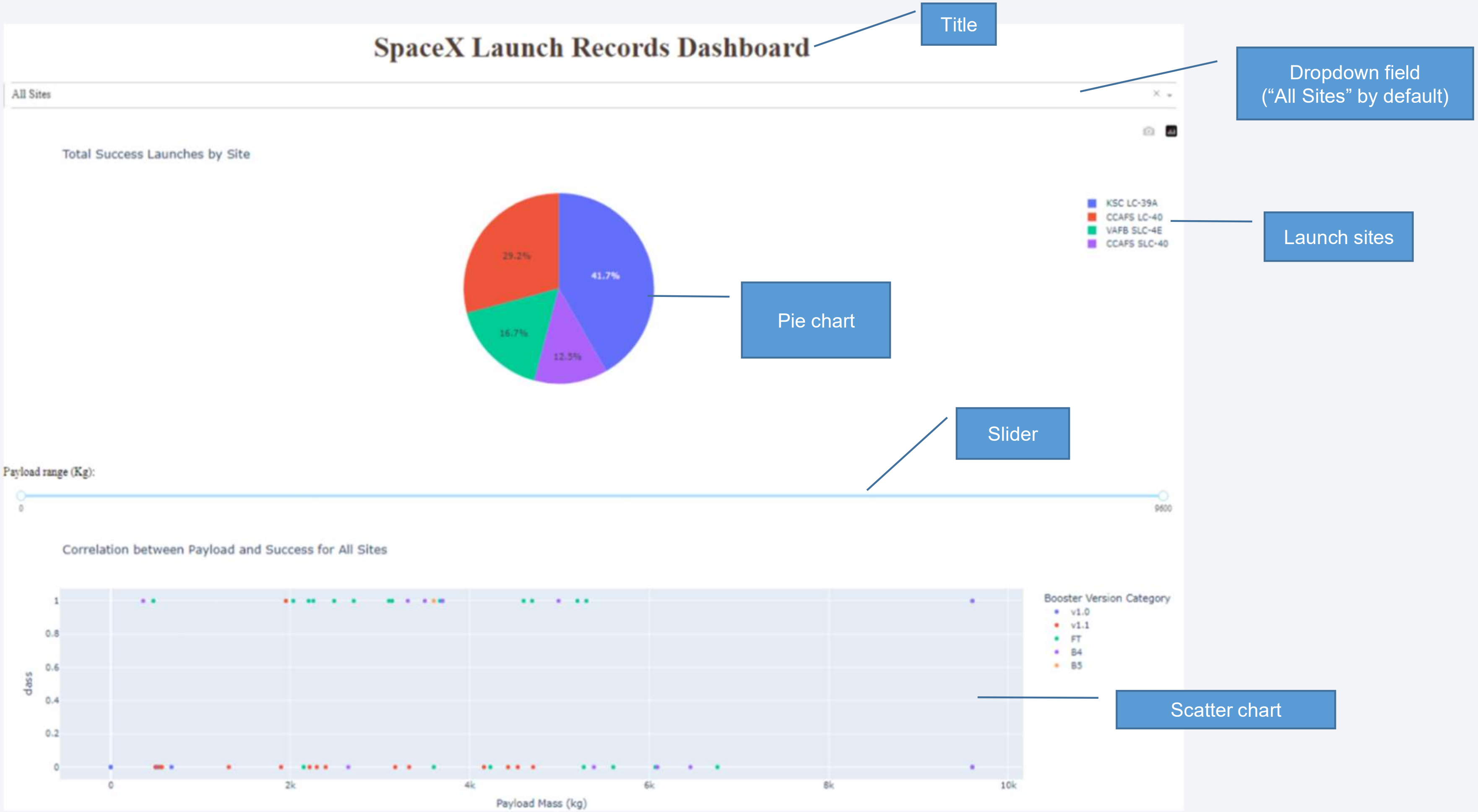




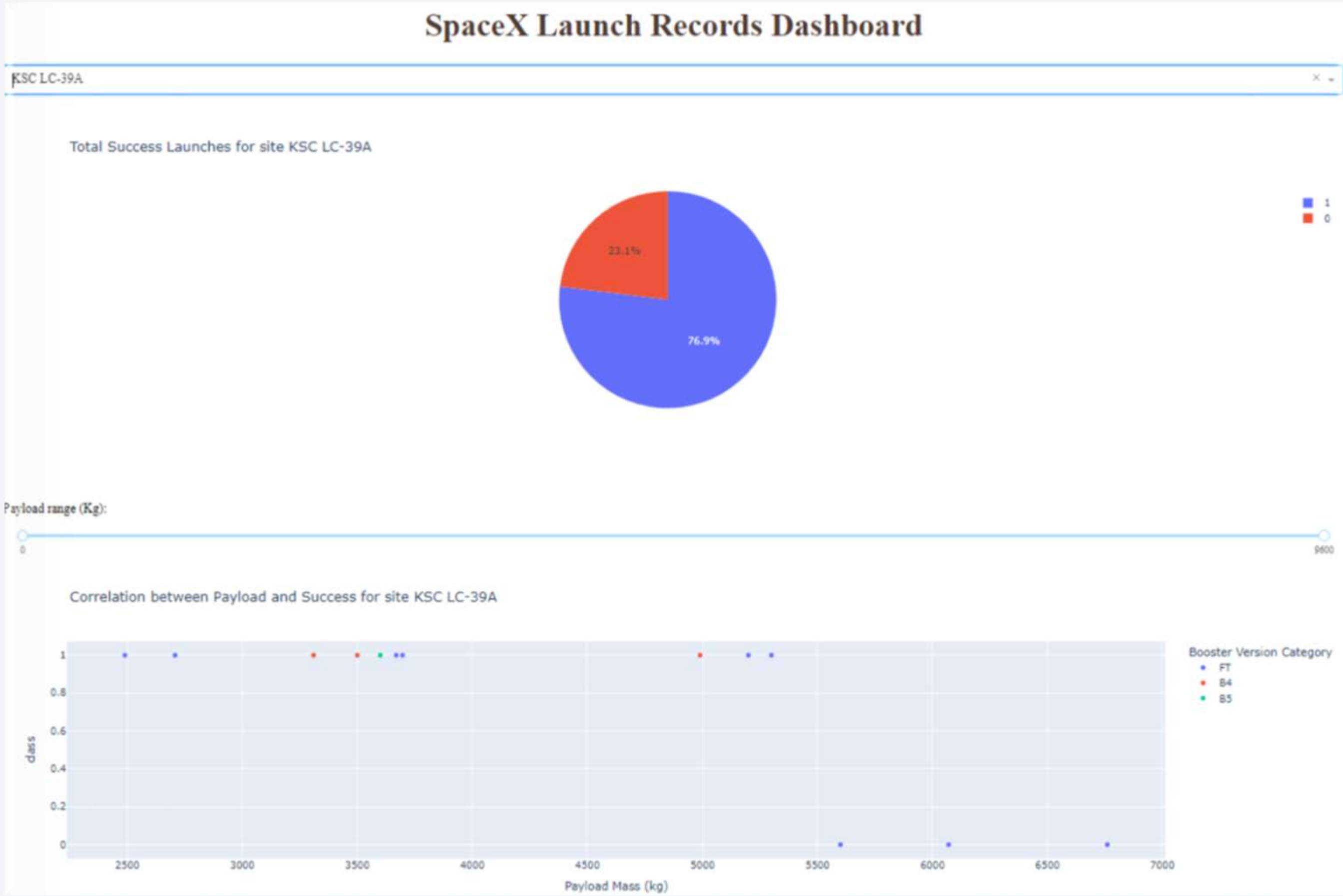
Section 4

Build a Dashboard with Plotly Dash

Dashboard main page (defaulted values)



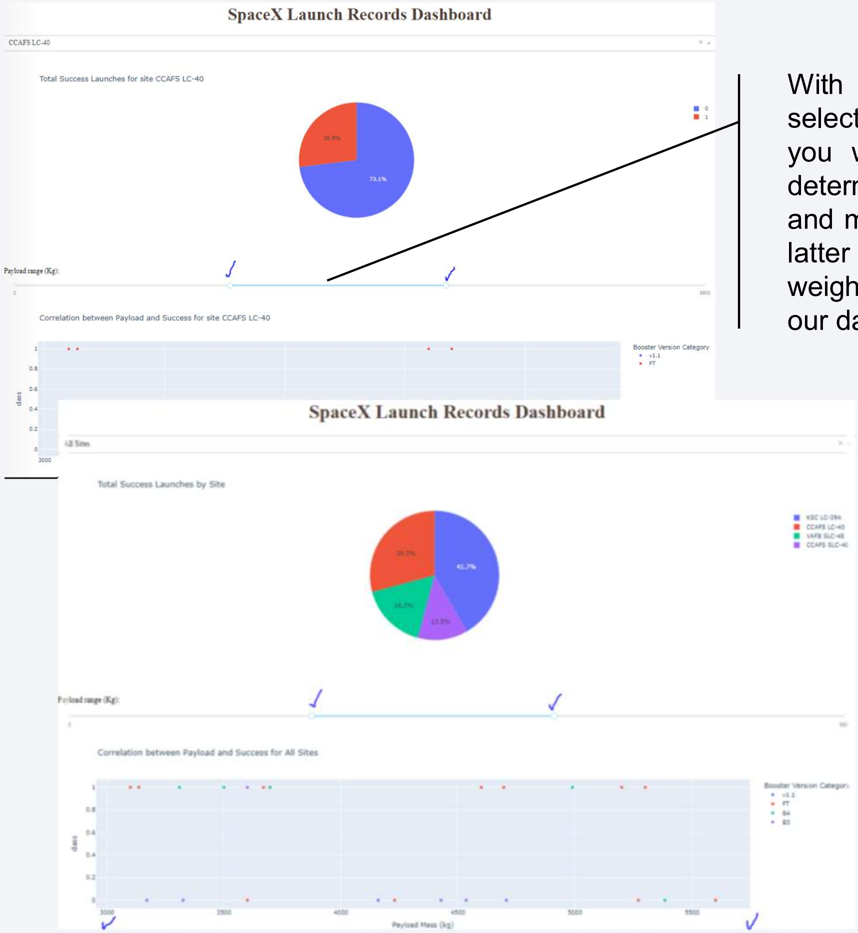
Highest launch success ratio dashboard



As showed in the dashboard, the launch site **KSC LC-39A** is the highest one from the "successful rating" (**76,9%**) perspective.

Let’s note that all launched until 5500 kg payload are success meanwhile the three launches over this weight were unsuccessful.

Dashboard options



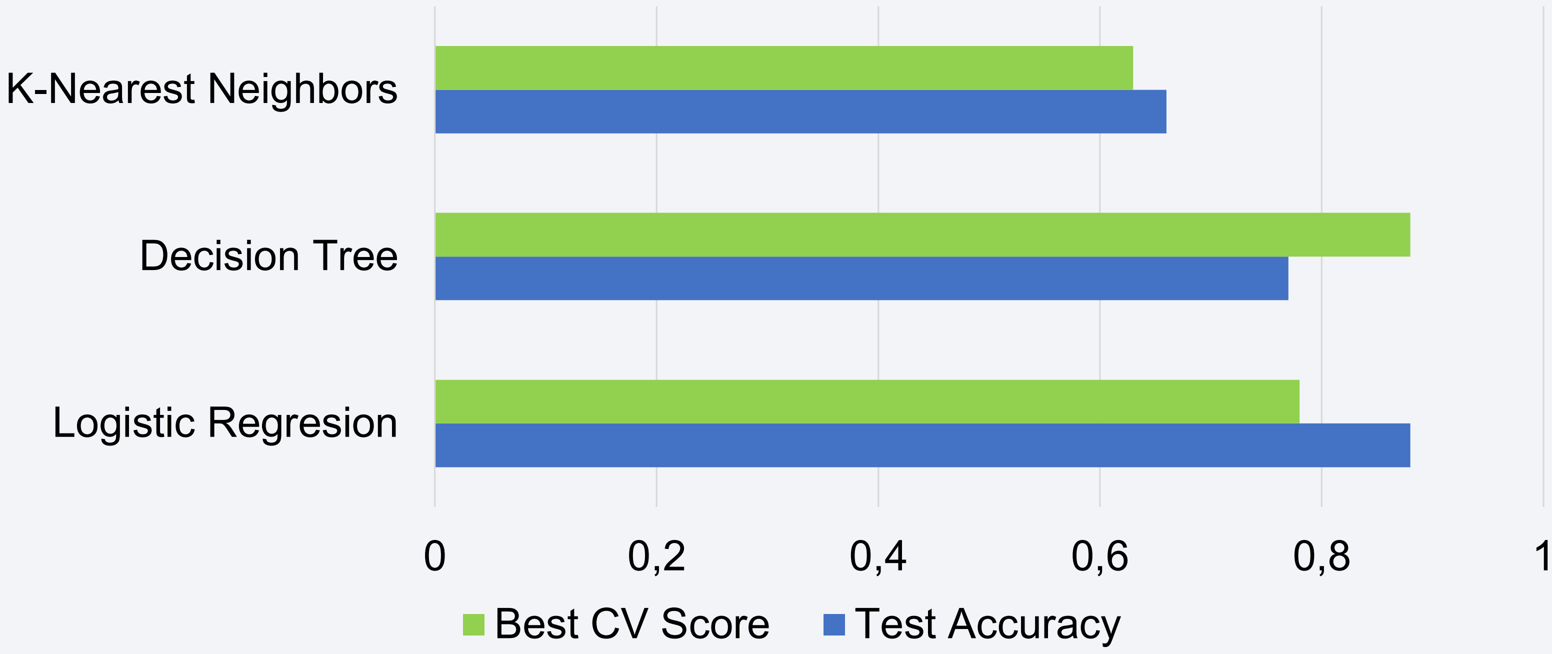
With ***the slider*** you can select the weights range you want to examine by determining the minimum and maximum weights, the latter being the maximum weight of all the records in our data set.

With the dropdown menu, you can select any launch site to analyze one by one its results, as well as all using the default option “All Sites”.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



Confusion Matrix

- Let’s see the confusion matrixes as follows:

Note: for the SVM (only) the CV is 2 instead of 10 because the laptop performance issues got during the execution.

TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters` .

```
parameters = {'kernel':('linear','rbf','sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

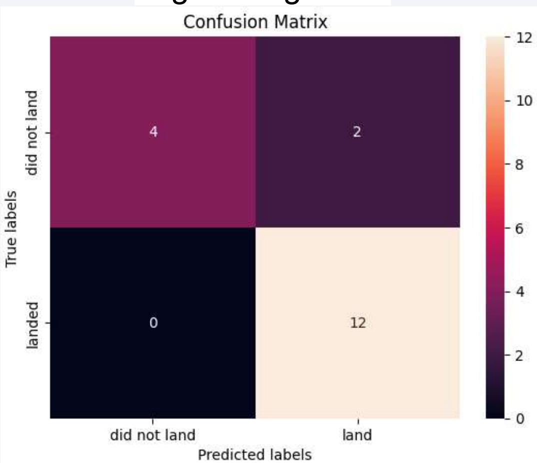
```
svm_cv = GridSearchCV(estimator=svm, param_grid=parameters, scoring='accuracy', cv=2, n_jobs=-1, verbose=1)
svm_cv.fit(X_train, Y_train)
```

```
Fitting 2 folds for each of 75 candidates, totalling 150 fits
▸ GridSearchCV
 ▸ estimator: SVC
   ▸ SVC
```

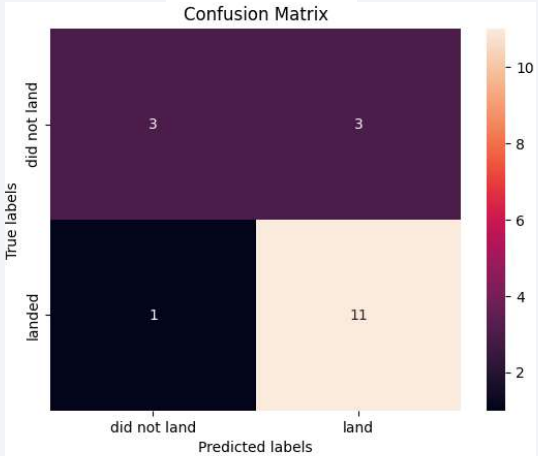
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.03162277660168379, 'gamma': 0.001, 'kernel': 'linear'}
accuracy : 0.8194444444444444
```

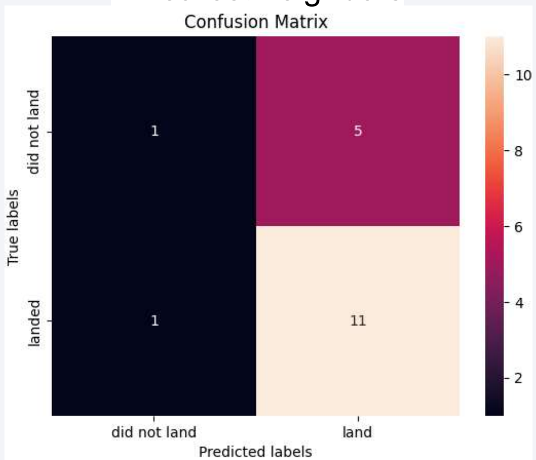
Logistic regresion



Decision tree



K nearest neighbors



Conclusions

- Logistic Regression Test Accuracy: 0.8888888888888888
- Decision Tree Test Accuracy: 0.7777777777777778
- K-Nearest Neighbors Test Accuracy: 0.6666666666666666
- Logistic Regression Best CV Score: 0.788888888888889
- Decision Tree Best CV Score: 0.888888888888889
- K-Nearest Neighbors Best CV Score: 0.6333333333333334

✓The ***best method based on test accuracy*** is: Logistic Regression

✓The ***best method based on cross-validation score*** is: Decision Tree

Appendix

- The summarized files used as a result of this work are available in this GitHub repository
    - Data Collection Jupyter Notebook (JN link)
    - Web scraping JN link
    - Data wrangling JN link
    - EDA SQL JN link
    - EDA Data Vision JN link
    - Launch site locations JN link
    - Machine Learning Prediction JN link

My gratitude to:

- IBM Skills Build
- *and* FUNDAE.

Thank you!