SPACEX LAUNCH PREDICTION

IBM DATA SCIENCE CAPSTONE PROJECT

IBM Developer
SKILLS NETWORK

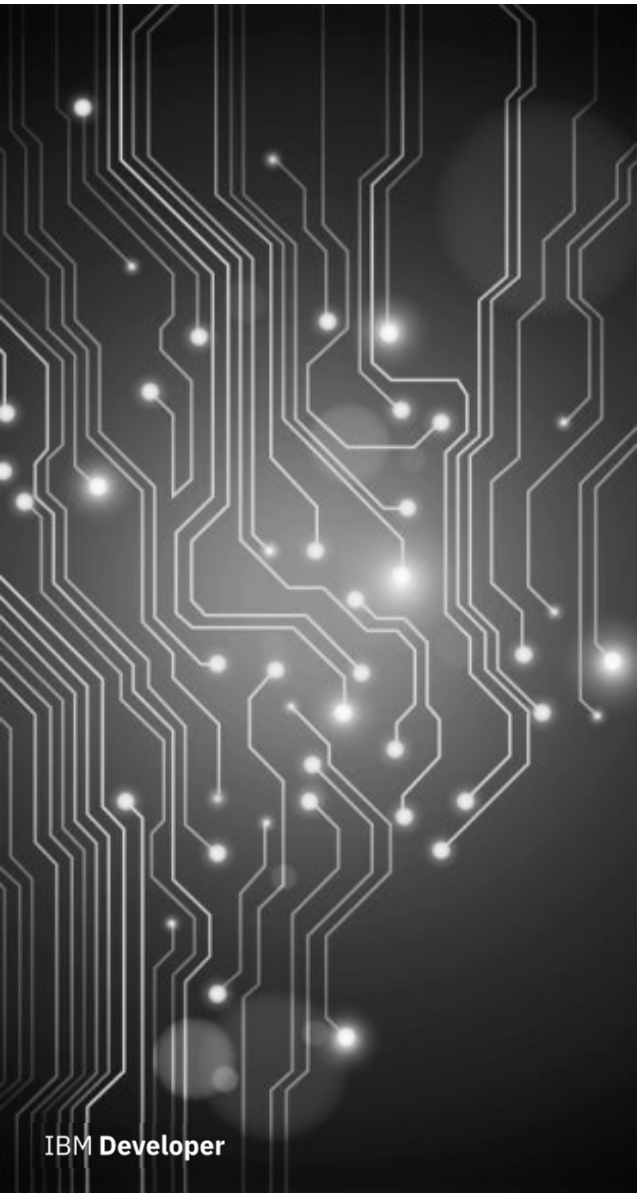
Bruno Rodrigues

October 2021

SPACEX

OUTLINE

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



EXECUTIVE SUMMARY

A QUICK BREAKDOWN

- Summary of methodologies
  - Data collection
  - Data wrangling
  - Exploratory data analysis (EDA) with visualizations
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a dashboard with Plotly Dash
  - Predictive analysis (classification)
- Summary of all results
  - EDA results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

INTRODUCTION

OUR GOALS

- **Project background and context;**
  - We predicted if the *Falcon 9* first stage will land successfully. *SpaceX* advertises *Falcon 9* launches on its website, with a cost of 62 million US dollars. Other providers cost upward of 165 million dollars each, much of the savings come from the fact that *SpaceX* can reuse the first stage of its rockets. Therefore, if we can determine whether the first stage rocket will land, we can determine the costs for future launches. This can then be used if another company would be interested in bidding against *SpaceX* for a rocket launch.
- **Problems we want to find answers to;**
  - What variables influence the landing success rate of the rockets?
  - Exploring the effect and relationships between different variables and what are their impact on the success rate of a landing.
  - What conditions does *SpaceX* have to meet in order to obtain the best results and ensure that they get the highest success rate for their rocket landings.

SECTION 1

METHODOLOGY

METHODOLOGY

LET'S DIVE IN

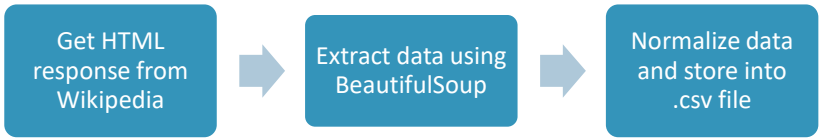
- Data collection methodology:
  - *SpaceX* Rest API
  - Web scrapping (from *Wikipedia*)
- Data wrangling - transformation of data for **machine learning** (ML);
  - One hot encoding of relevant variables.
- **Exploratory data analysis** (EDA) using visualization and **SQL**;
  - Scatter plots and bar plots to show the relationships between variables and show patterns within the data.
- Interactive visual analytics using **Folium** and **Plotly Dash**;
- Predictive analysis using **ML** classification models;
  - Building, tuning and evaluating accuracy of classification models.

DATA COLLECTION

- *SpaceX* launch data was gathered from the **SpaceX REST API** *[https://api.spacexdata.com/v4/launches/past]*;
- The **API** gives us data on launches, including information about the rocket used, payload, launch and landing specifications, and landing outcome;
- We intend to use this data in order to predict if *SpaceX* will be successful with their future launches;
- We also obtain a similar dataset by web scraping launch data directly from *Wikipedia* using **BeautifulSoup**.

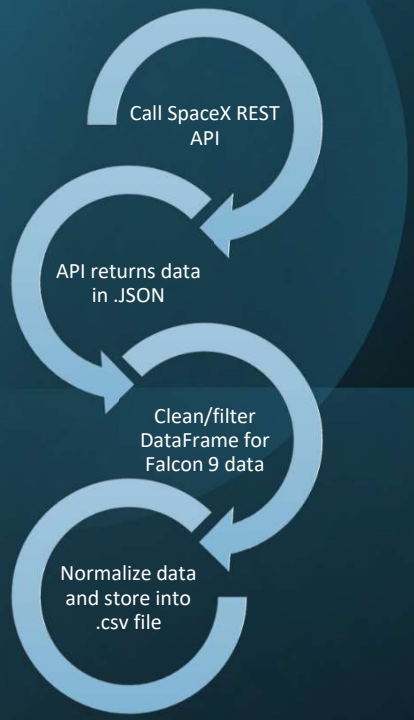
SpaceX API

Web Scraping



DATA COLLECTION

SpaceX API



IBM Developer

[GitHub Notebook]

1. Getting response from SpaceX REST API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
response.status_code
```

2. Collecting response as .JSON

```
response_static = requests.get(static_json_url)
data = pd.json_normalize(response_static.json())
```

3. Clean data (functions)

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4. Assign to DataFrame

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

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

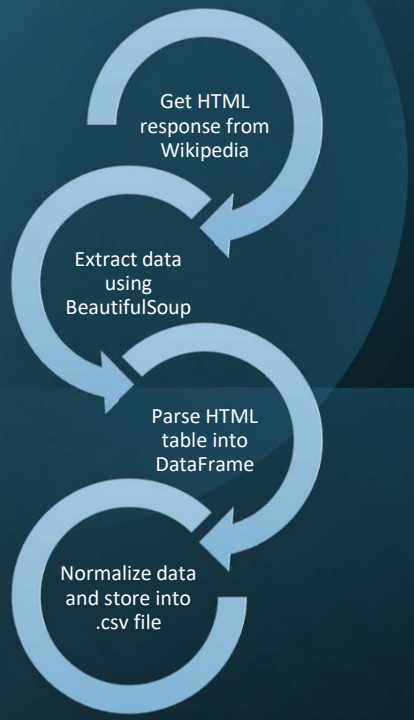
5. Filter/clean DataFrame and export to .csv file

```
data_falcon9 = df.loc[(df['BoosterVersion']!='Falcon 1')]
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))

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

DATA COLLECTION

Web Scraping



IBM **Developer**

[GitHub Notebook]

1. Getting response from page

```
response = requests.get(static_url)
html_doc = response.text
```

2. Store in a BeautifulSoup object

```
soup = BeautifulSoup(html_doc, 'html.parser')
```

3. Find tables in the HTML code

```
html_tables = soup.find_all('table')
```

4. Extracting column names

```
column_names = []

rows = first_launch_table.find_all('th')

for r in range(len(rows)):

    name = extract_column_from_header(rows[r])

    if name is not None and len(name) > 0:

        column_names.append(name)
```

5. Create feature dictionary

```
launch_dict = dict.fromkeys(column_names)

# Remove an irrelevant column
del launch_dict['Date and time ( )']

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'] = []
```

6. Append data to keys

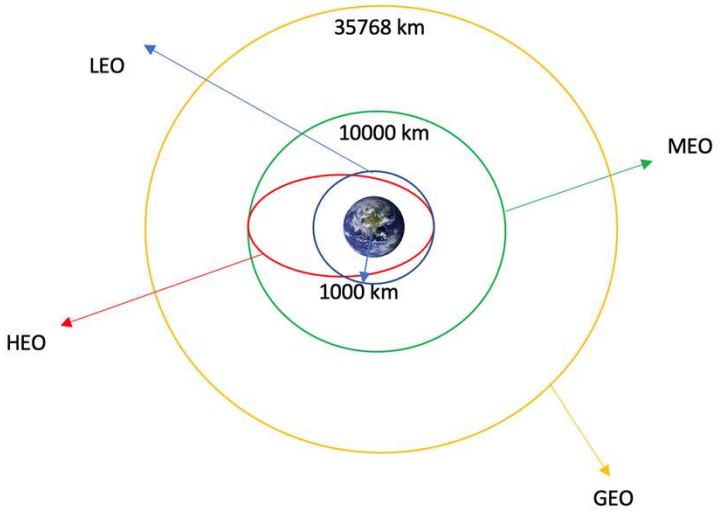
[code omitted – extract too long]

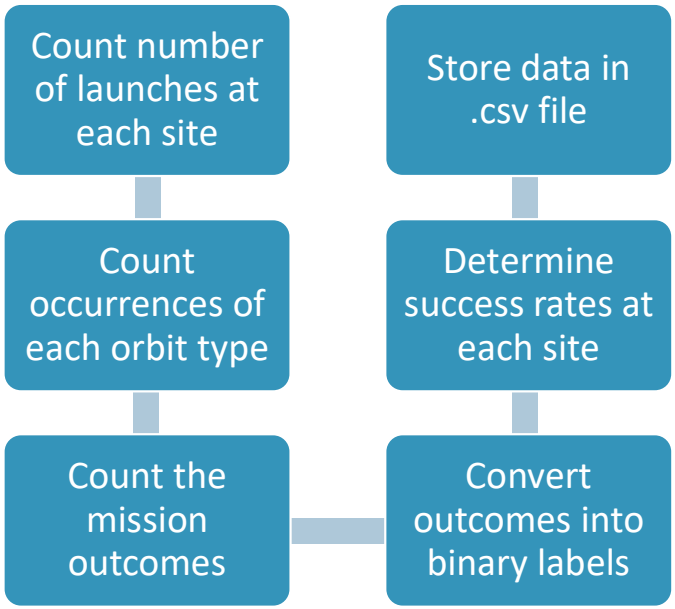
7. Filter/clean DataFrame and export to .csv file

```
df=pd.DataFrame(launch_dict)
df.to_csv('Data/spacex_web_scraped.csv', index=False)
```

DATA WRANGLING

- Ours is a simple **classification** problem, which means our data features have to be converted into a class variable with **0** (unsuccessful landing) and **1** (successful landing) as the classifiers of the model.

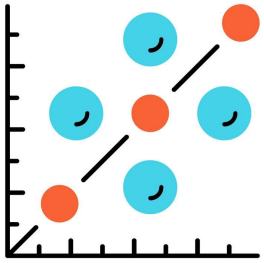




[GitHub Notebook]

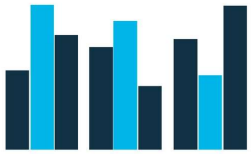
EDA WITH VISUALIZATIONS

[GitHub Notebook]



Scatter Plots

We'll make use of plenty of these in order to visualize the correlation between different variables.



Bar Plot

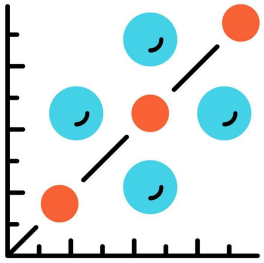
This type of plot consists of a single discrete variable being compared for multiple categories.

Line Plot

With this plot we will compare variable trends over time and make a rough prediction of how it will evolve in the future.

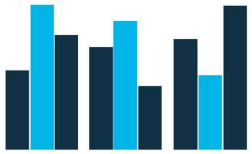
EDA WITH VISUALIZATIONS

[GitHub Notebook]



Scatter Plots

- Flight n. vs Payload mass
- Flight n. vs Launch site
- Payload mass vs Launch site
- Orbit vs Flight n.
- Payload mass vs Orbit
- Orbit vs Payload mass



Bar Plot

- Success rate vs Orbit

Line Plot

- Success rate vs Year

EDA WITH SQL

SQL QUERIES SHOWN IN THIS PRESENTATION;

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

[GitHub Notebook]

INTERACTIVE MAP WITH FOLIUM

We make use of interactive maps to visualize launch site locations and gather insights on their proximities.

- **LAUNCH OUTCOMES**
  - We clustered markers that represent each launch at each launch site;
  - Markers labelled in green and red according to outcome;
  - These represents our classifiers (successful/unsuccessful).
- **LAUNCH SITE PROXIMITIES**
  - Lines were drawn on the map to find the distance to various landmarks;

[GitHub Notebook]

INTERACTIVE DASHBOARD WITH PLOTLY DASH

An interactive dashboard was made in order to more dynamically explore the dataset.

- **PIE CHART - LAUNCHES BY SITE**
  - Displays the relative proportion between the different classes;
  - Allows for a quick understanding of outcomes at each site.
- **SCATTER PLOTS - OUTCOME VS PAYLOAD MASS**
  - Clearly shows relationships between variables;
  - We can change the payload range in order to better gain insights on the role of mass in landing outcomes;
  - Provides a clear reading of the data.

[GitHub Notebook]

PREDICTIVE ANALYSIS - CLASSIFICATION

- **BUILDING THE MODEL**
  - Load and transform dataset
  - Split into train/test datasets
  - Select which ML algorithms will be used
  - Set up a GridSearchCV object and its parameters
  - Train the model on our dataset
- **MODEL EVALUATION**
  - Compute model accuracy
  - Tune the hyperparameters
  - Obtain the Confusion Matrix
- **MODEL IMPROVEMENT**
  - Feature Engineering
  - Fine tuning the algorithm

[GitHub Notebook]

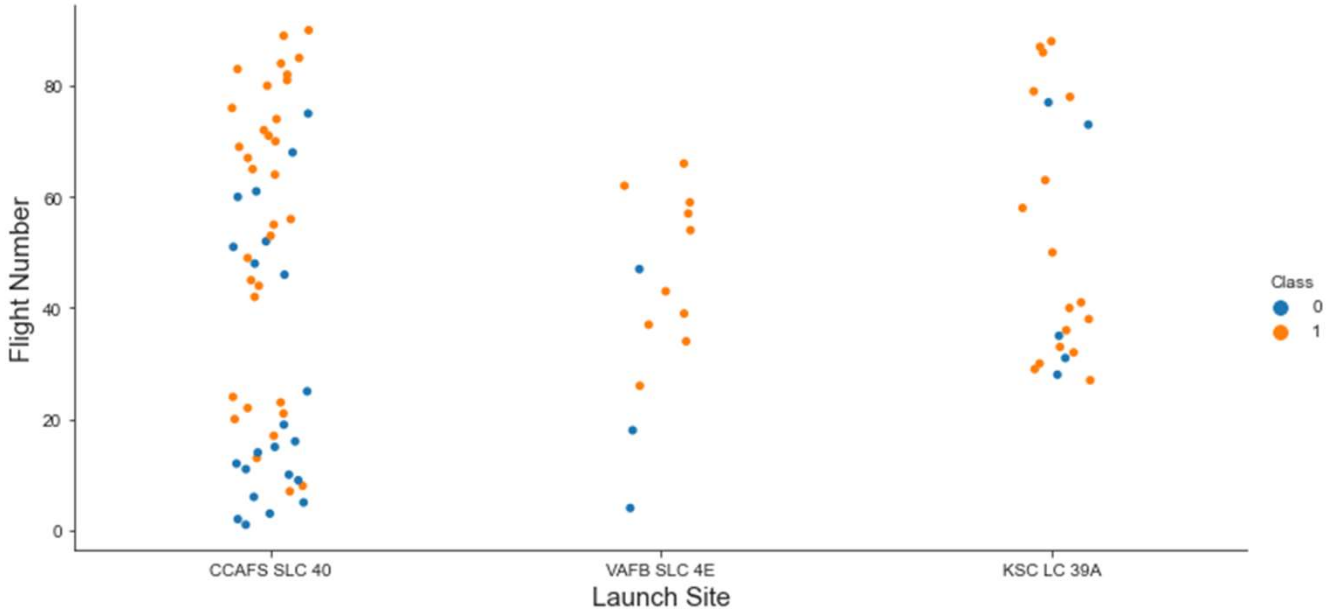
RESULTS

- Exploratory data analysis (EDA) results
- Interactive analytics in screenshots
- Predictive analysis results

SECTION 2

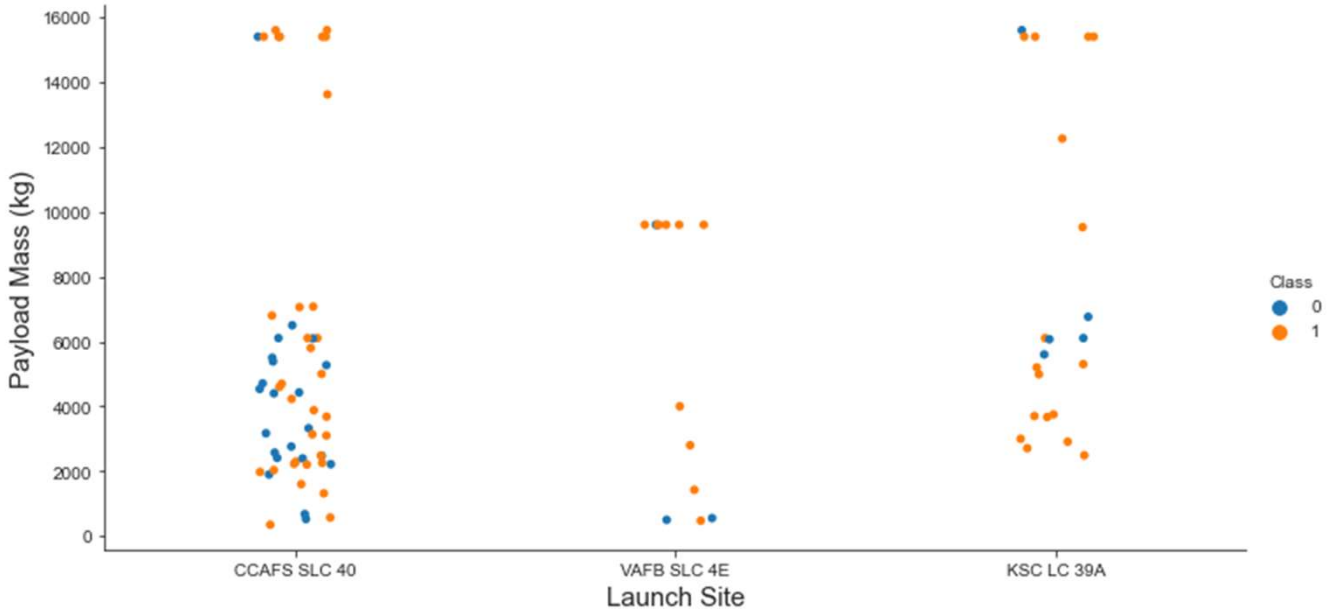
EDA WITH VISUALIZATIONS

FLIGHT NUMBER vs. LAUNCH SITE



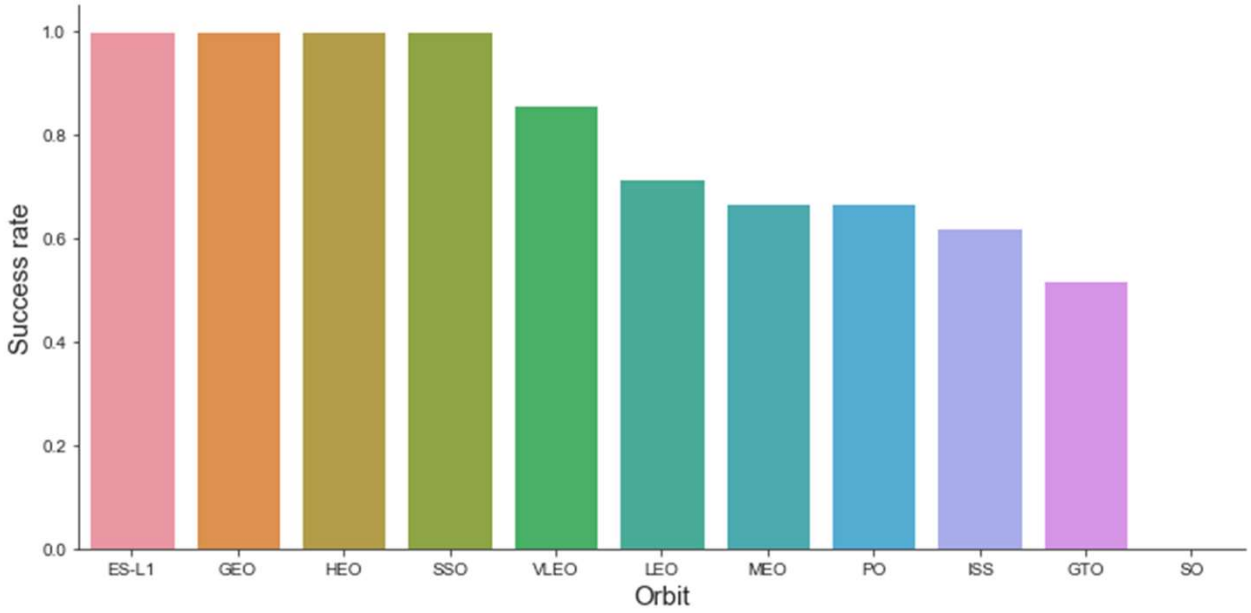
We can clearly see that for higher flight number (more recent flights) the greater the success rate of the landings.

PAYLOAD MASS vs. LAUNCH SITE



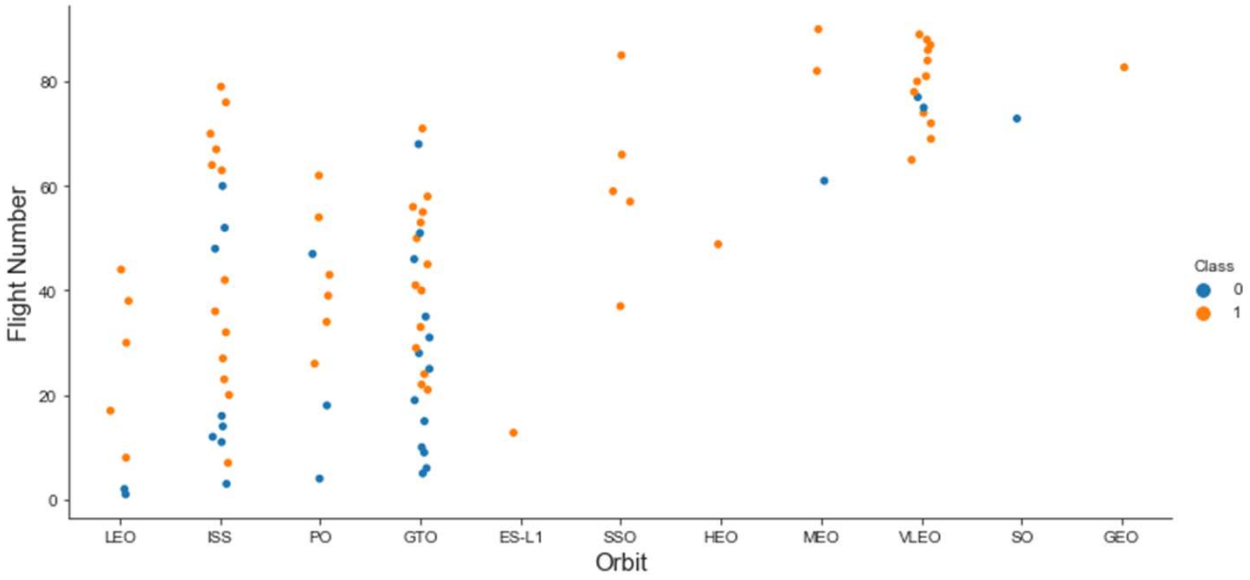
There is no clear pattern to be found on this one, although there is some indication that higher payloads may have better landing outcomes.

SUCCESS RATE vs. ORBIT TYPE



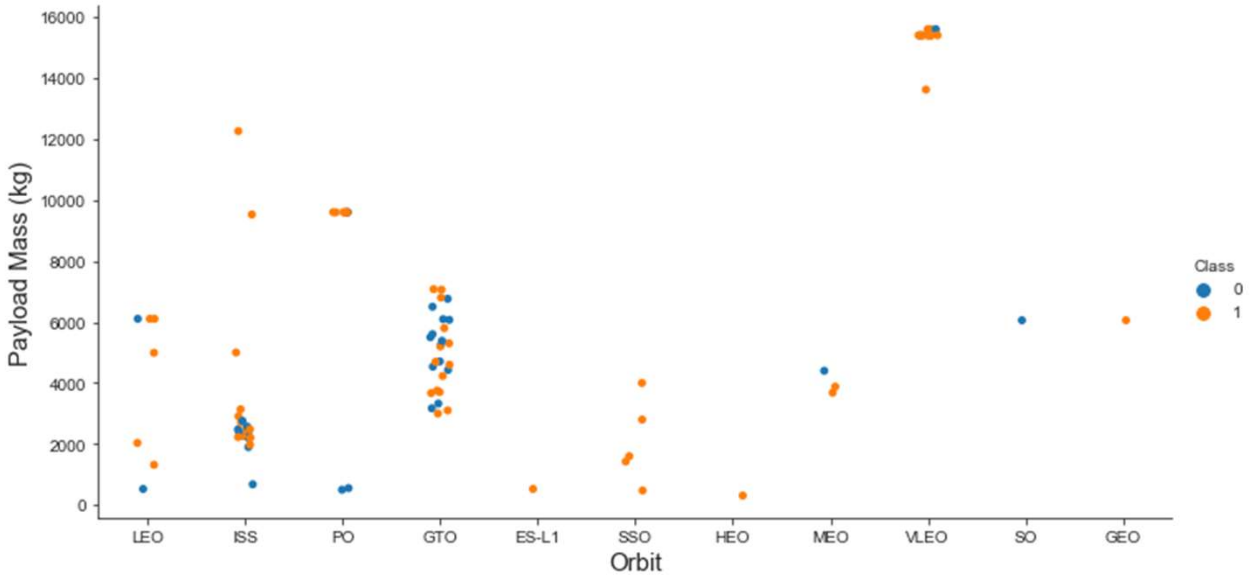
Launches whose payloads had their orbits as ES-L1, GEO, HEO and SSO have a solid landing success rate. Successive orbits rank lower, but still above 50%. Only launches with SO has their orbit have not registered any successful landings.

FLIGHT NUMBER vs. ORBIT TYPE



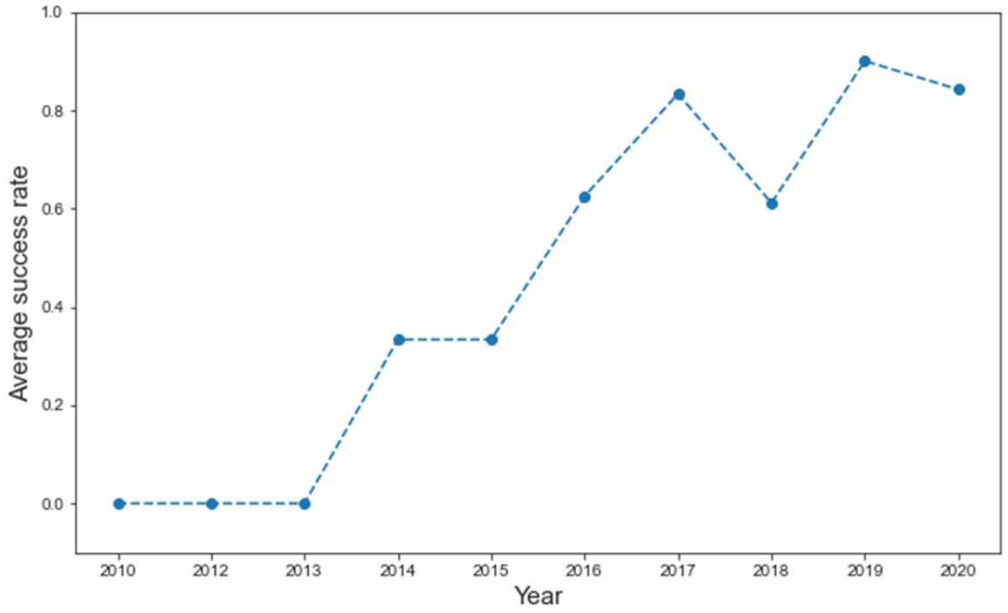
Recall that higher flight number indicates more recent flights, so over time we see an increased success rate. We also see how *SpaceX* opens to new orbits on later flights, with increasingly higher success rate as time goes on.

PAYLOAD MASS vs. ORBIT TYPE



In general, there seems to be a positive correlation between orbits and higher payload mass. A notable exception being GTO, where the data points don't seem to indicate any clear relationship.

LAUNCH SUCCESS - YEARLY TREND



SpaceX has had a steady increase in their average success rate over the years, with a notable dip in 2018 (it's possible that the company had been experimenting with other variables in their launches).

SECTION 3

EDA WITH SQL

UNIQUE LAUNCH SITES

```
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL
```

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

QUERY EXPLANATION

Rather straightforward. By using the command **DISTINCT** we specify that we only want unique entries found in the **LAUNCH_SITE** column.

LAUNCH SITE NAMES BEGINNING WITH 'CCA'

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

DATE	TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40

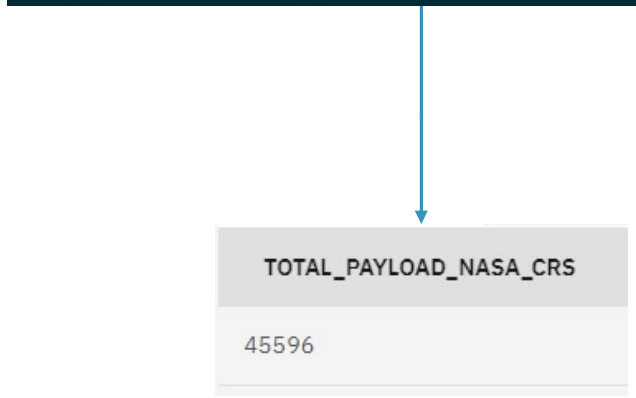
QUERY EXPLANATION

We use the **LIKE** command to indicate a search over a character pattern ('CCA%'): we want the name of the launch site to start with 'CCA'.

The command **LIMIT** is set to 5 in order to display the first 5 results only.

TOTAL PAYLOAD MASS WITH CUSTOMER 'NASA (CRS)'

```
SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_NASA_CRS
FROM SPACEXTBL
WHERE CUSTOMER = 'NASA (CRS)'
```



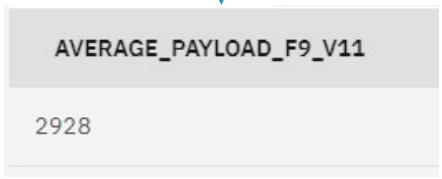
TOTAL_PAYLOAD_NASA_CRS
45596

QUERY EXPLANATION

By making use of the **SUM** function, we get the total from the **PAYLOAD_MASS__KG_** column. We then specify the customer by making use of the command **WHERE**.

AVERAGE PAYLOAD MASS WITH BOOSTER VERSION F9 v1.1

```
SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_F9_v11
FROM SPACEXTBL
WHERE BOOSTER_VERSION = 'F9 v1.1'
```



AVERAGE_PAYLOAD_F9_V11
2928

QUERY EXPLANATION

By making use of the **AVG** function, we get the average out of the **PAYLOAD_MASS__KG_** column. We then specify the booster version by making use of the command **WHERE**.

FIRST SUCCESSFUL GROUND LANDING

```
SELECT DATE
FROM SPACEXTBL
WHERE LANDING__OUTCOME LIKE '%ground pad%'
    AND MISSION_OUTCOME = 'Success'
ORDER BY DATE ASC
LIMIT 1
```

DATE
2015-12-22

```
SELECT MIN(DATE)
FROM SPACEXTBL
WHERE LANDING__OUTCOME LIKE '%ground pad%'
    AND MISSION_OUTCOME = 'Success'
```

QUERY EXPLANATION

We use **WHERE** and **LIKE** to specify the type of landing from the **LANDING__OUTCOME** column to include '%ground pad%'.

Then we order the results by ascending **DATE**, so that the first row is the earliest date, and use **LIMIT** to only get one result.

Alternatively, we can use the function **MIN** to find the lowest (earliest) date.

SUCCESSFUL DRONE SHIP LANDING PAYLOAD BETWEEN 4000-6000

```
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE LANDING__OUTCOME LIKE '%drone ship%'
    AND MISSION_OUTCOME = 'Success'
    AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
```

BOOSTER_VERSION
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

QUERY EXPLANATION

We display only the **BOOSTER_VERSION**. We use **WHERE** and **LIKE** to select the **LANDING__OUTCOME**(s) that meet the required text string (‘%drone ship%’).

With the **AND** clauses we specify the **MISSION_OUTCOME** to be successful and we set a condition for **PAYLOAD_MASS__KG_** to be **BETWEEN 4000 AND 6000**.

TOTAL NUMBER OF SUCCESSFUL AND FAILED MISSIONS

```
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME
```

MISSION_OUTCOME	TOTAL
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

QUERY EXPLANATION

This is a simple use of the function **COUNT** on **MISSION_OUTCOME** followed by **GROUP BY**.

Note that a subquery can be used here to group the successful outcomes together and avoid the break that comes from the one isolated data entry.

BOOSTERS CARRIED MAX PAYLOAD

```
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL) = PAYLOAD_MASS__KG_
```

BOOSTER_VERSION
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5

QUERY EXPLANATION

We display only the **BOOSTER_VERSION**. Here we set a subquery where we select the maximum **PAYLOAD_MASS__KG_** from the table such that it returns only the boosters that carried that set maximum payload.

2015 LAUNCH RECORDS - FAILED LANDINGS

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

BOOSTER_VERSION	LAUNCH_SITE
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

QUERY EXPLANATION

Pretty straightforward. We display **BOOSTER_VERSION** and **LAUNCH_SITE**. We use **WHERE** to set the **LANDING_OUTCOME** to be only include failed landings on drone ships. We also set a filter so that **DATE** is only for the year 2015.

2015 LAUNCH RECORDS - FAILED LANDINGS

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

LANDING__OUTCOME	TOTAL
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3

QUERY EXPLANATION

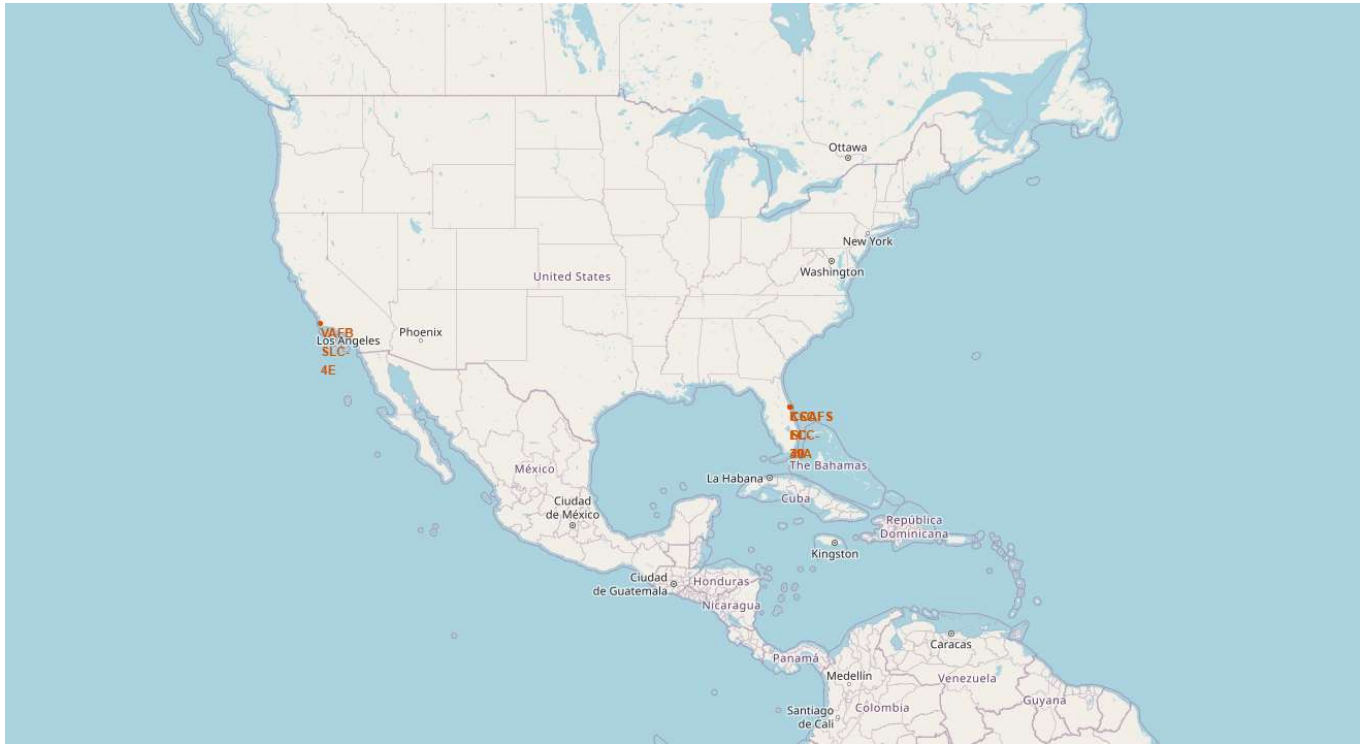
This query displays all **LANDING_OUTCOME**(s) (ungrouped) and their respective total by using the function **COUNT**. We use **WHERE** to specify the **DATE** and we also order the total count from highest to lowest using **ORDER BY TOTAL DESC**.

SECTION 4

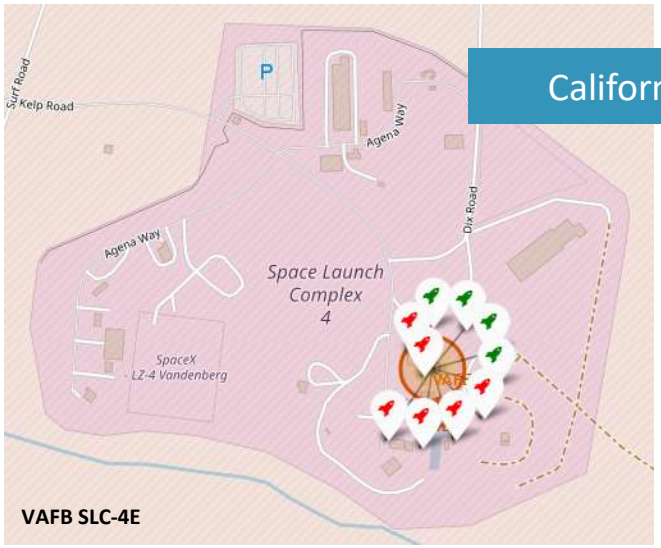
MAPPING LAUNCH SITES WITH FOLIUM

LAUNCH SITE LOCATIONS

- All launch sites are shown in the map to the right.
- SpaceX uses a total of 4 facilities;
  - **California:**
    - VAFB SLC-4E
  - **Florida:**
    - KSC LC-39A
    - CCAFS LC-40
    - CCAFS SLC-40



COLOR LABELLED LAUNCHES



California Site

Florida Sites

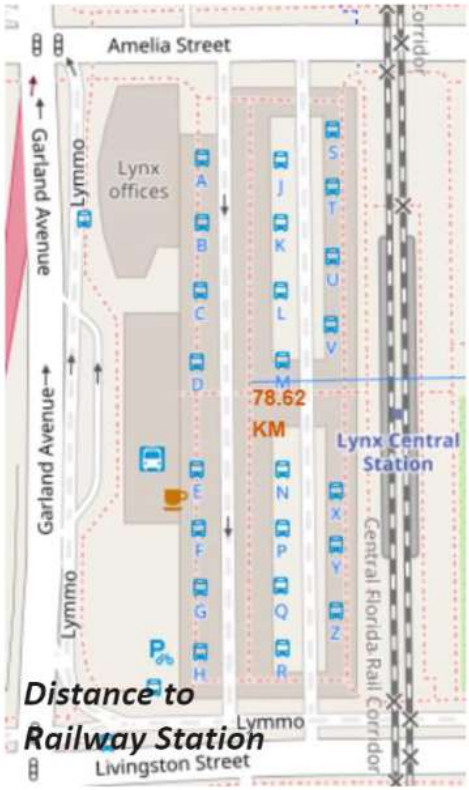
GREEN MARKERS

Successful landings

RED MARKERS

Unsuccessful landings

LAUNCH SITE PROXIMITIES



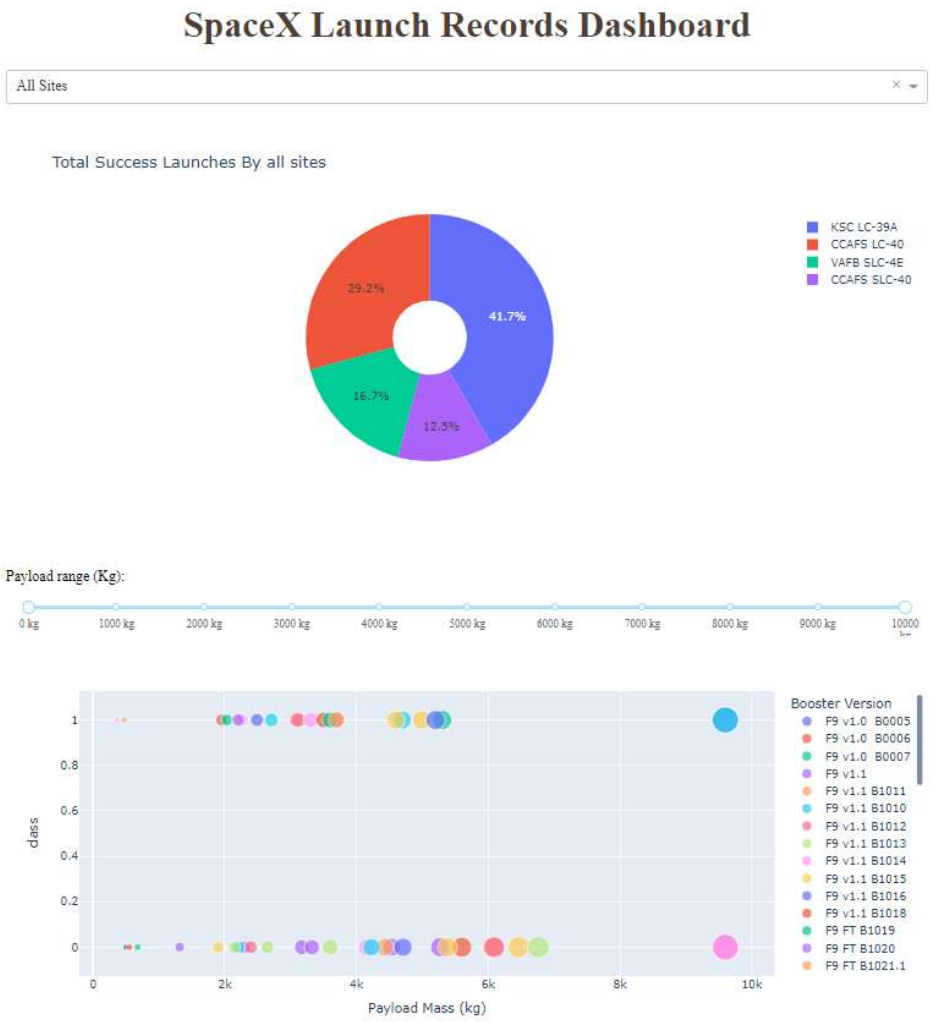
- **Close to railways? No.**
- **Close to highways? No.**
- **Close to coastline? Yes.**
- **Close to cities? No.**

SECTION 5

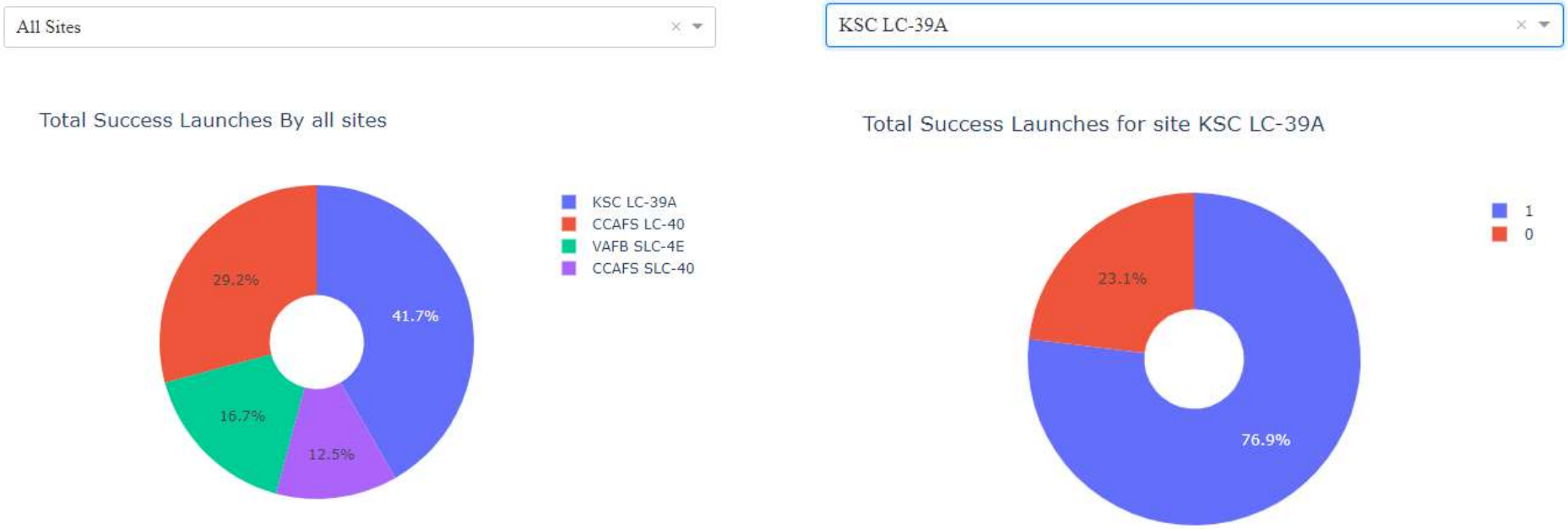
DASHBOARD WITH PLOTLY DASH

OVERVIEW

- Dashboard made with *Plotly Dash*.
- Pie chart displays the success rate by site (drop down menu allows for launch site selection).
- A scatterplot of the launches with their respective payload is show underneath, also divided by their class (successful or unsuccessful landing).
- It's possible to change the payload range by making use of the slider.

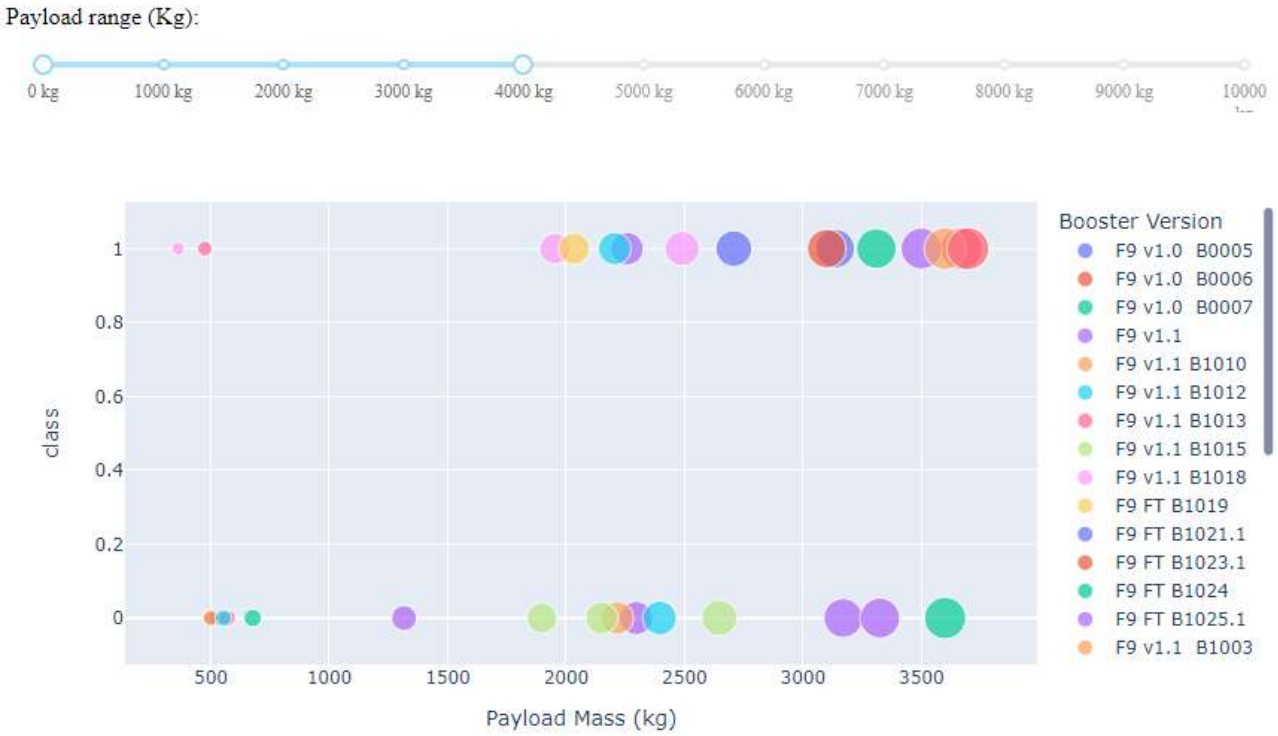


DASHBOARD – PIE CHARTS



We can see that **KSC LC-39A** is the launch site with the highest success rate, where 76.9% of its landings were successful. When we select that launch site we can see a further breakdown in the pie chart, which now gives us the percentage slices of each class.

DASHBOARD – SCATTER PLOT (0KG-4000KG)



- Here we display the lower end of the payload range.
- We can see that launches with lower payloads don't usually end in with a successful landing.
- Most successful landings are in the 2000-4000Kg range.

DASHBOARD – SCATTER PLOT (4000KG-10000KG)

- Here we display the higher end of the payload range.
- Most launches in this range end up with unsuccessful landings.
- When compared to the previous payload range, the 4000-10000Kg range has a significantly lower success rate.

SECTION 6

PREDICTIVE ANALYSIS

CLASSIFICATION

CLASSIFICATION ACCURACY

We compute the accuracy of each model in order to find the highest performing one.

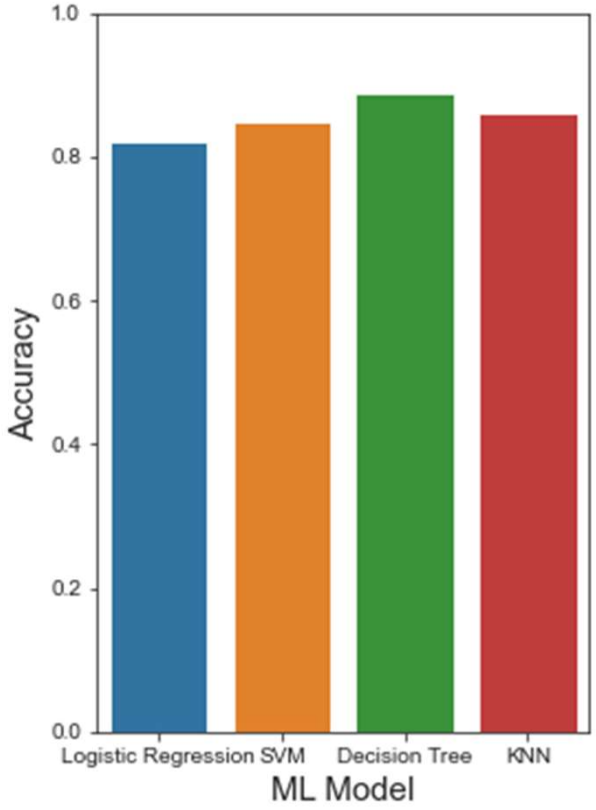
```
LogReg accuracy: 0.8214285714285714 | SVM accuracy: 0.85 | Decision Tree accuracy: 0.8875000000000002 | KNN accuracy: 0.8607142857142855
LogReg score: 0.8333333333333334 | SVM score: 0.8333333333333334 | Decision Tree score: 0.9444444444444444 | KNN score: 0.8333333333333334

Best algorithm = Decision Tree
```

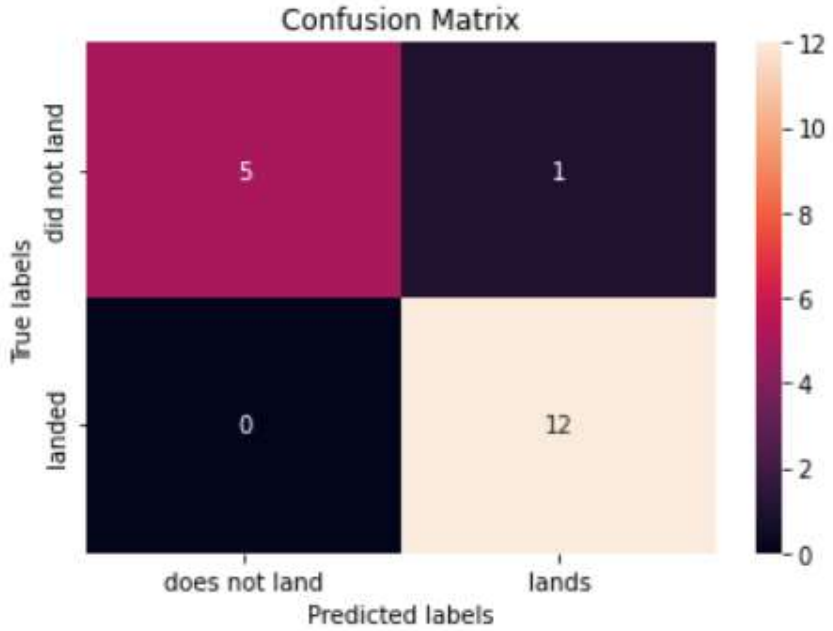
Despite it being quite close, we can see that the Decision Tree algorithm is the best at getting matches with the test dataset.

The model hyperparameters that lead to this high accuracy are shown bellow:

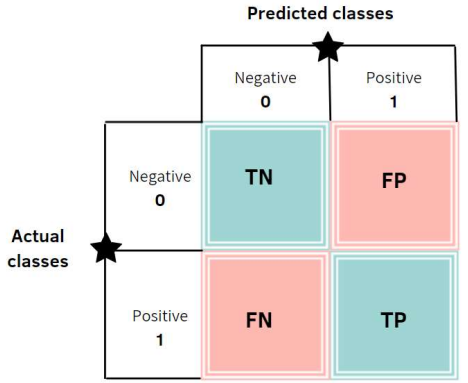
```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 14, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2,
'splitter': 'best'}
accuracy : 0.8875000000000002
```



CONFUSION MATRIX



The confusion matrix for the Decision Tree model shows us just how good our predictive model is. When the model is compared to the test data, it only gets one of the data points incorrect (as a false positive). All other data points have their outcomes correspond to the model prediction.



CONCLUSION

- The Decision Tree algorithm is the best pick for our predictive model using this dataset.
- Payloads within the mid-range (~4000Kg) have in general a higher landing success rate.
- *SpaceX* has come a long way, as their landing success rates have steadily increased over the years and show a rising trend going forward.
- KSC LC-39A in Florida, is the site with the most successful landing outcomes overall.
- Orbits GEO, HEO, SSO, ES-L1 have the best success rate.

THANK YOU

BRUNO RODRIGUES

https://github.com/kouniam

brunomfr.ua@gmail.com

SPACEX