Data and SpaceX: Insights on Rockets and their Launches

Ryan Wm. Greene

03/15/2024

Outline

Executive Summary

- Methodologies for…
  - Data Collection
  - Data Wrangling
  - Data Analysis
  - Folium & Plotly Visualizations
  - ML Predictions and Insights
- Results
- Screenshots
- Final Insights and Takeaways

Introduction & Goal

Instead of using rocket science to determine if the first stage will land successfully, you will train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

The commercial space age is here, companies are making space travel affordable for everyone. Virgin Galactic is providing suborbital spaceflights. Rocket Lab is a small satellite provider. Blue Origin manufactures sub-orbital and orbital reusable rockets. Perhaps the most successful is SpaceX. SpaceX’s accomplishments include:

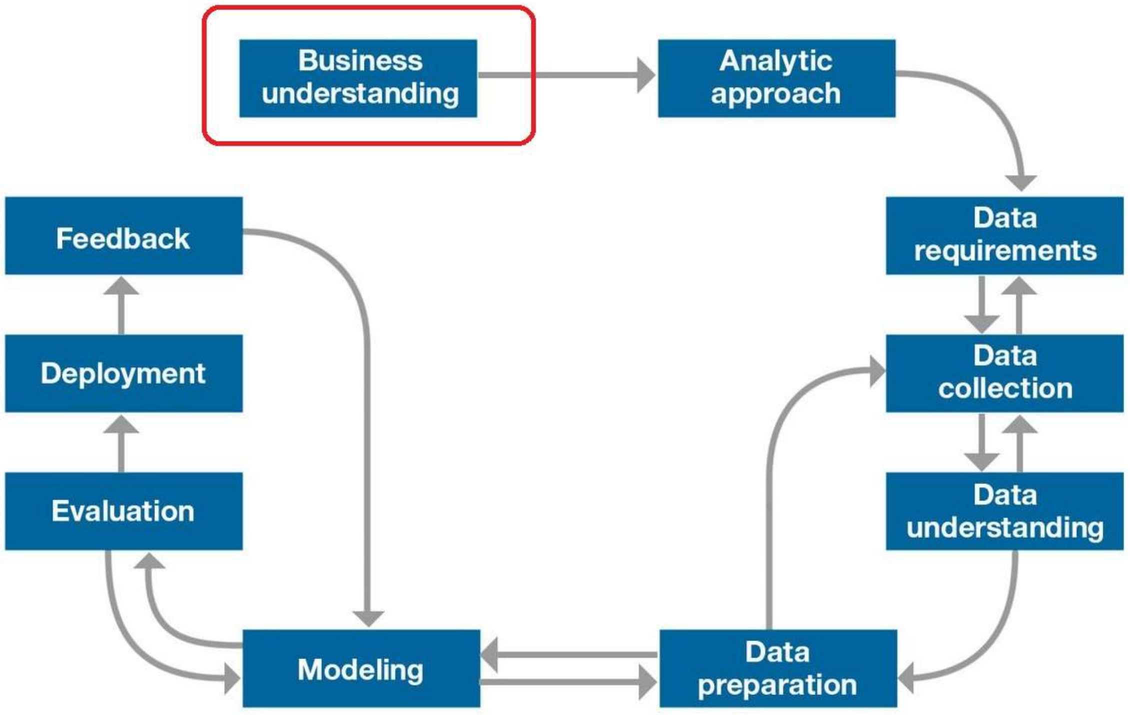
Sending spacecraft to the International Space Station.

Starlink, a satellite internet constellation providing satellite Internet access.

Sending manned missions to Space.

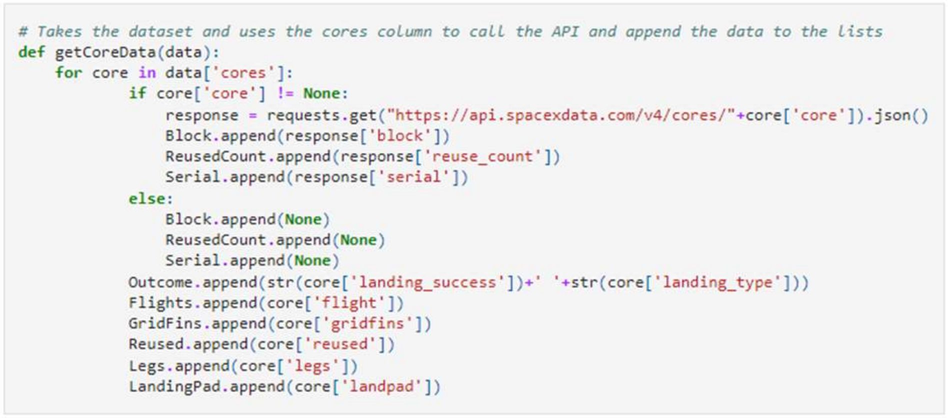
One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch.

Methodologies



Data Collection – SpaceX Api

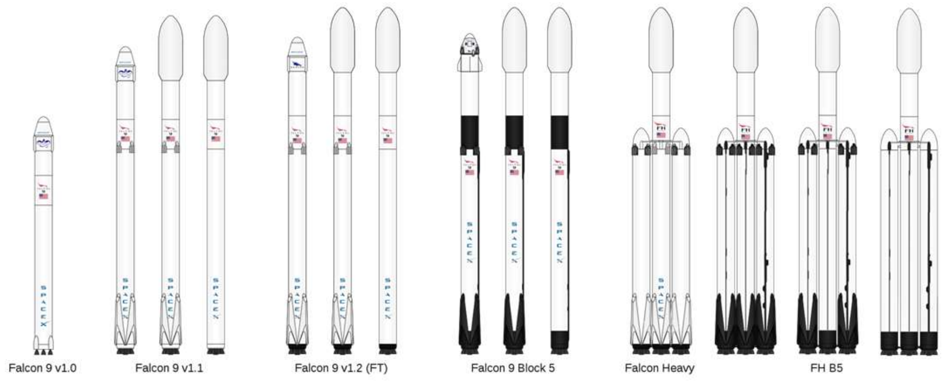
- By utilizing Space X’s API we can call & response past rocket launches and get a huge JSON of data. We’re able to then parse through, append relevant information to our data frame (ie. launch pads, booster versions) and use average values to fill in for empty values (namely in the case of payload mass)



```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

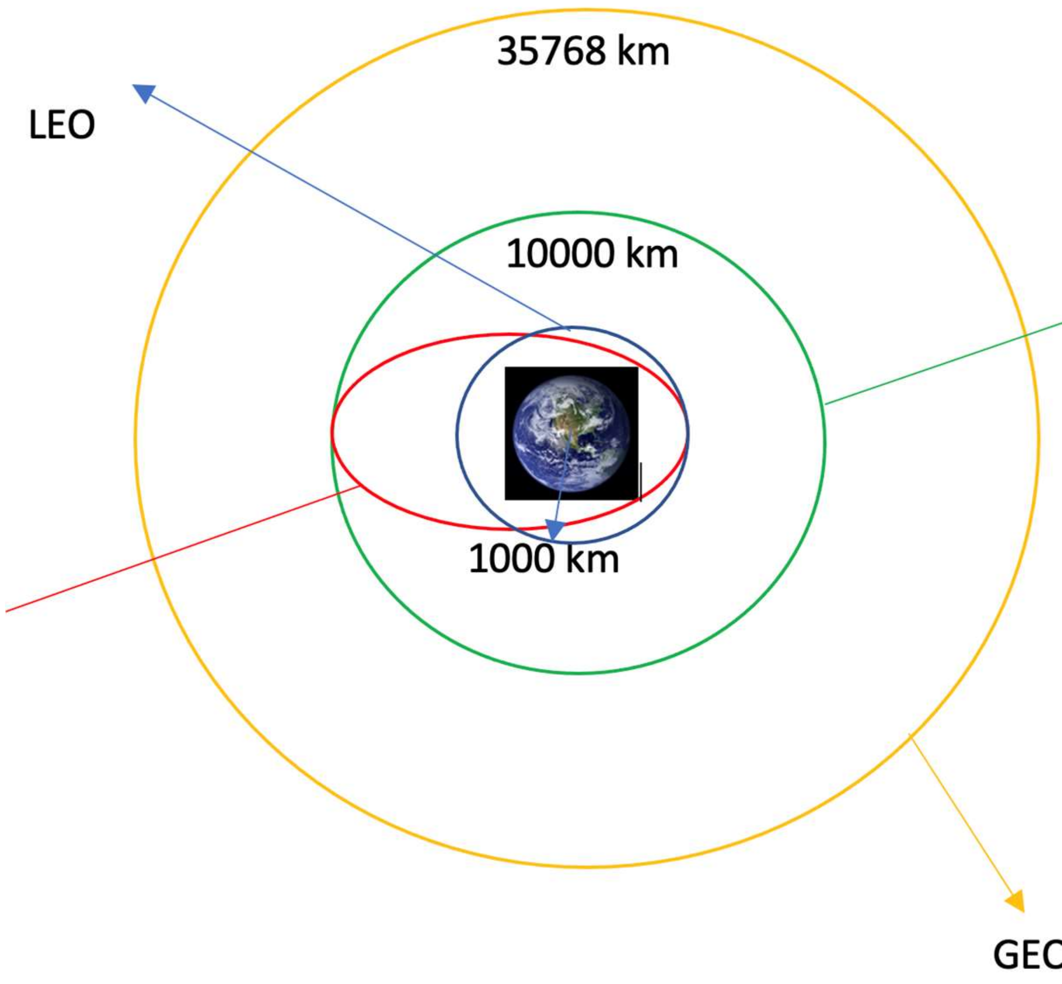
Data Collection – Webscrapping Wiki

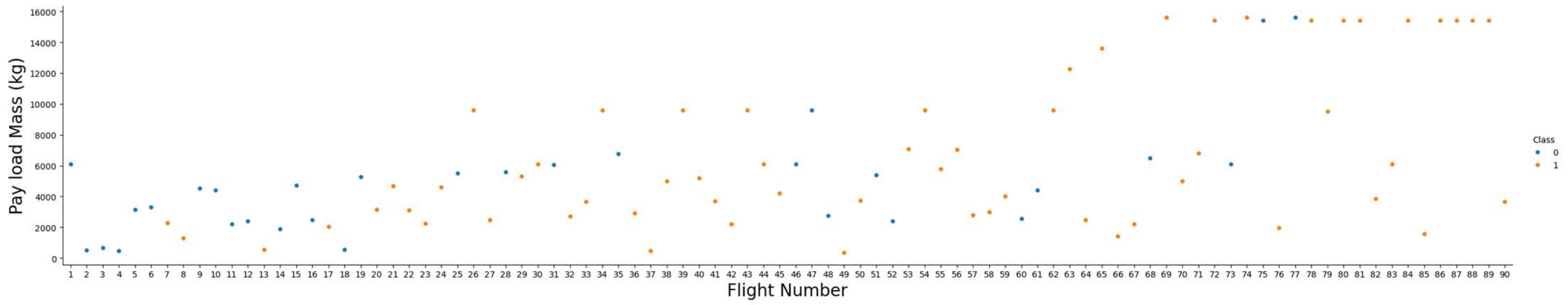
- By utilizing BeautifulSoup and making requests from the SpaceX wiki page on launches, we're able to utilize the neatly formatted tables that exist there.
- Once requested and souped, we're able to use the HTML tags to break the table apart and assign tags as needed to our data frame.



Data Wrangling

- Our data wrangling sets us on course for success. We start by identifying missing values, getting our data types, and identifying orbits and outcomes.
- By utilizing the outcomes column, which has eight different outcomes, we're able to create a 'Class' column with a Boolean value: we make a 'Success or Fail' column from a more complex column.
- By doing so, we're able to then identify average success rates.



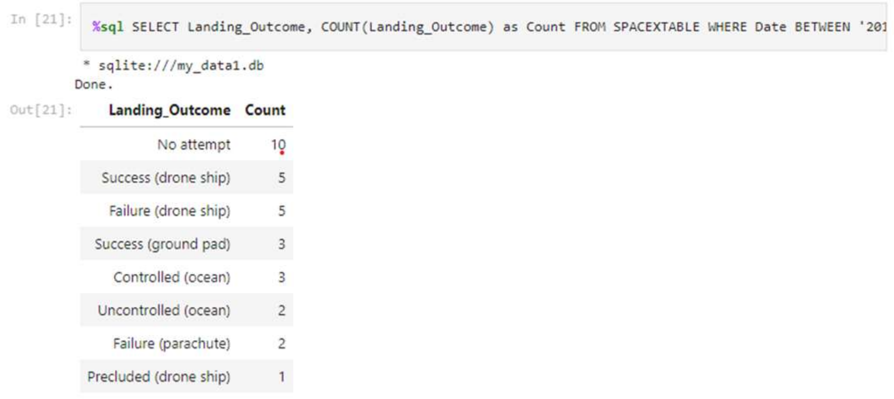


EDA with Visualizations

- Our dataset has a lot of interesting relationships worth visualizing. While all of them will be shown later, the viz shown displays flights by payload mass, and whether they succeeded or not.
- We were able to produce visualizations utilizing outcome/class, payload mass, launch sites, orbit, and average success over measurements such as year.

EDA with SQL

- In Jupyter Notebook we use SQLAlchemy, connect to a database, and load our data into a database to play around with.
- Some more interesting explorations from this exercise were a list of successful and failed missions, landing outcome counts, and booster versions that carried the maximum possible payloads.

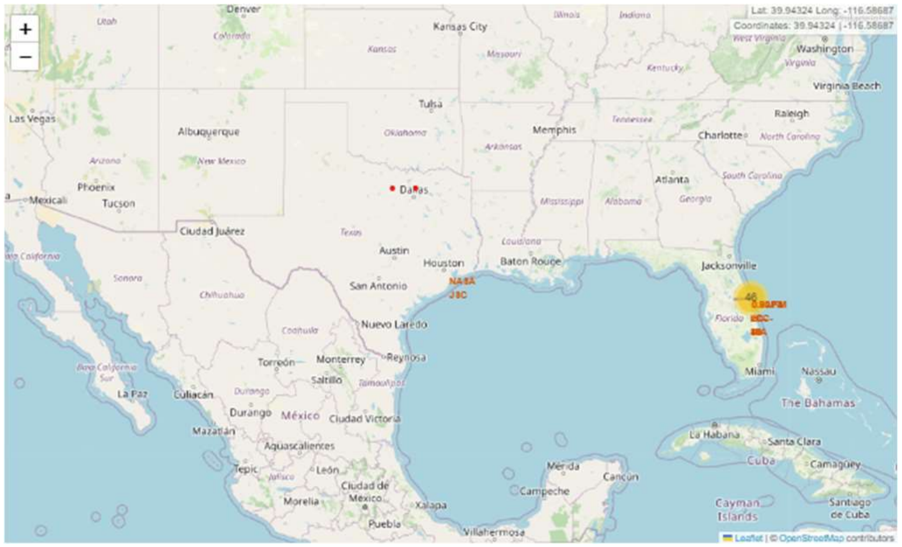


```
In [21]: %sql SELECT Landing_Outcome, COUNT(Landing_Outcome) as Count FROM SPACEXTABLE WHERE Date BETWEEN '201
 * sqlite:///my_data1.db
Done.
```

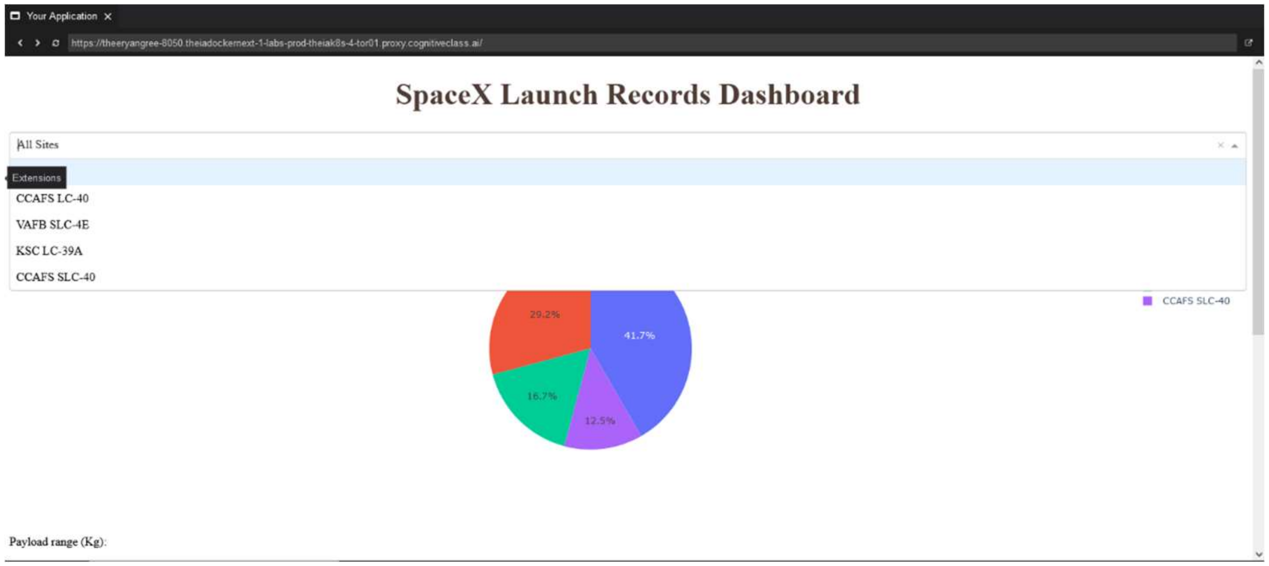
Out[21]:

Landing_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

Folium (for geographic mapping and viz)

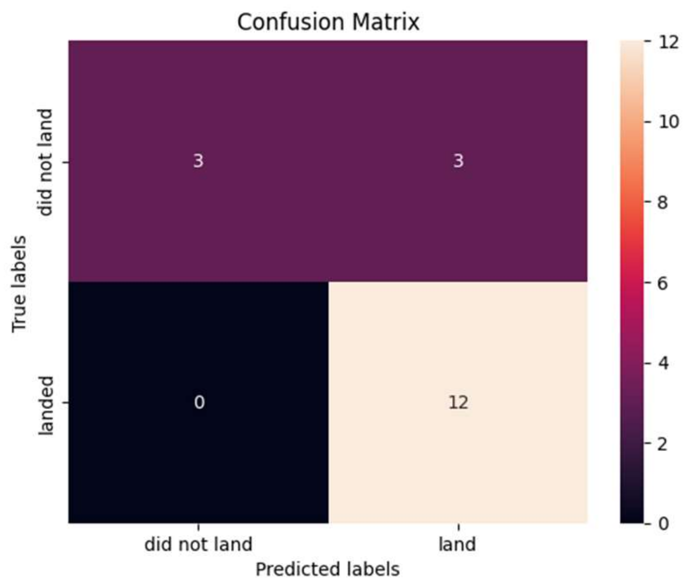


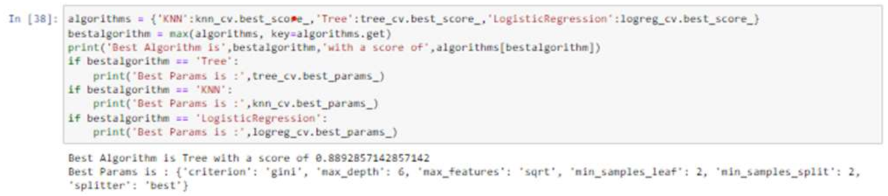
- Folium is an excellent tool for plotting geographic data.
- Utilizing the longitude and latitude of launch locations, we were able to visualize on a map the number of launches at each side. Additionally, we were able to (for our Florida SpaceX Launch sites) map and calculate the distance between our launch site and the nearest 1) coastline 2) highway 3) railway 4) city.



- Utilizing Plotly, we created a simple, interactive dashboard.
- The dashboardcontained pie charts for launches by sites, and a scatter graph for a relationship of outcome with payload mass by booster versions.

Plotly (for interactive dashboarding)





```
In [38]: algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
         bestalgorithm = max(algorithms, key=algorithms.get)
         print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
         if bestalgorithm == 'Tree':
             print('Best Params is :',tree_cv.best_params_)
         if bestalgorithm == 'KNN':
             print('Best Params is :',knn_cv.best_params_)
         if bestalgorithm == 'LogisticRegression':
             print('Best Params is :',logreg_cv.best_params_)

         Best Algorithm is Tree with a score of 0.8892857142857142
         Best Params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2,
         'splitter': 'best'}
```

Machine Learning

- At the end of our journey, we took in our data and split it into training and testing.

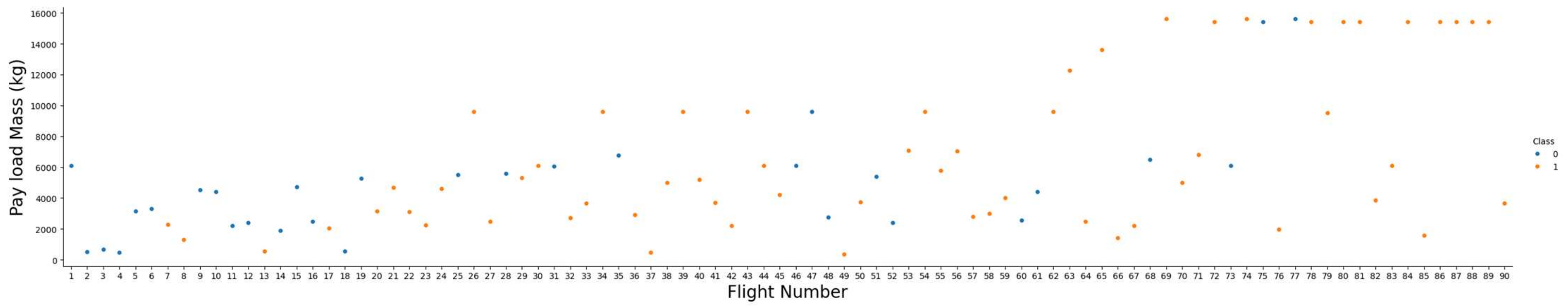
- We utilized a variety of ML-ing algorithms and tuned them with GridSearchCV (parameters like poly, linear, rfb, and sigmoid kernels).

- Storing the results by accuracy together, we were able to determine the best classification method (shown above).

Conclusions

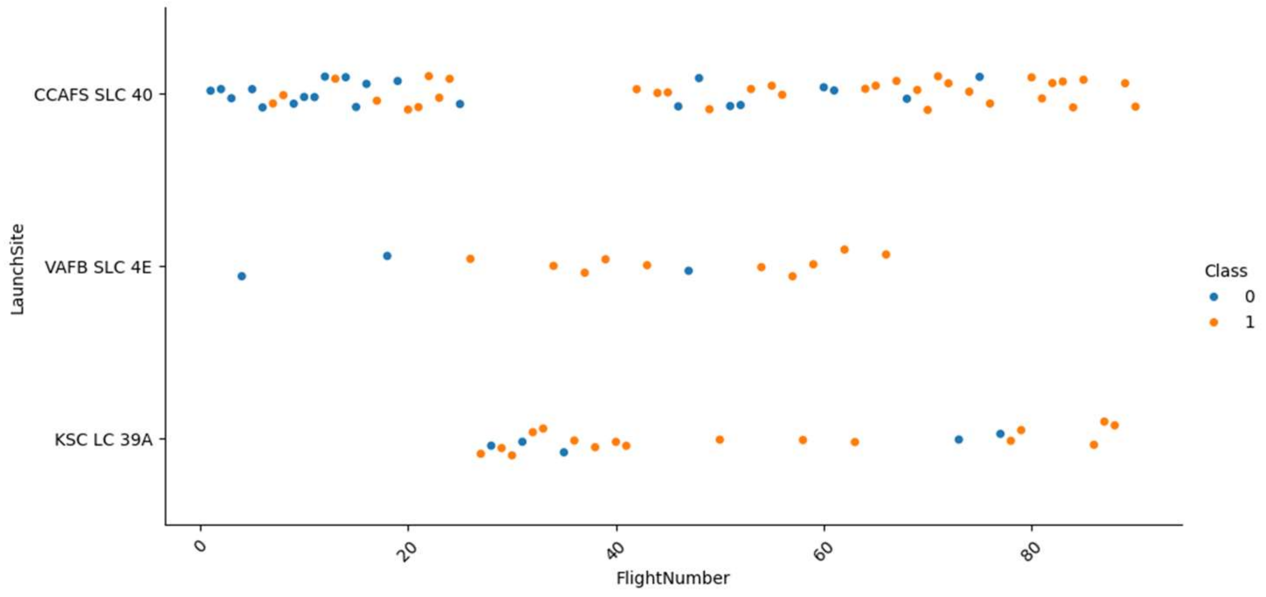
Payload Mass and Flight Number

- Given that 1s (orange) are successes, we can see that as time goes on, starting with flight 7, our successes become more successful, and with that, our payload mass also increases.



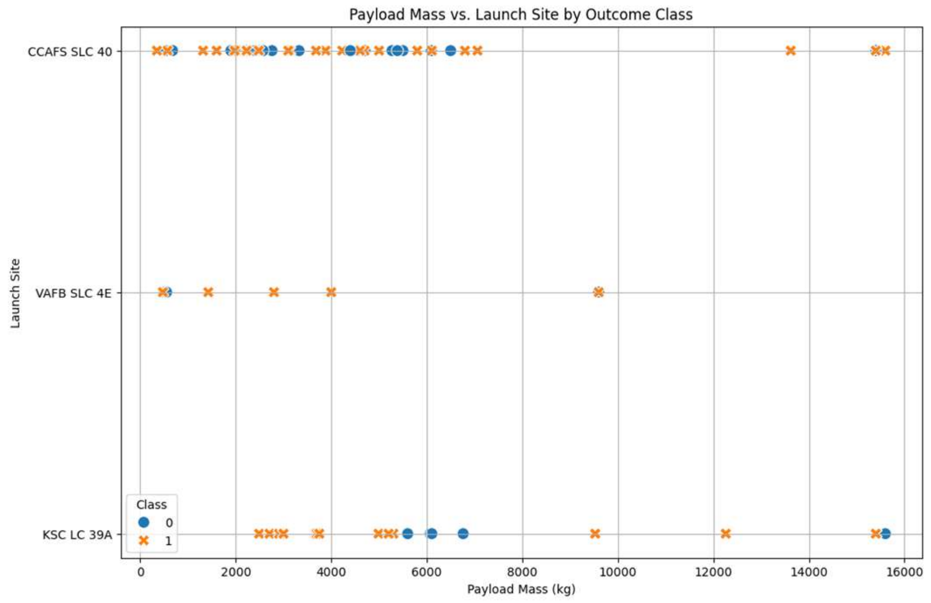
Launch Sites and Outcomes

- We can see how launch sites were utilized and make assumptions about which facility is newer vs older. Their oldest launch sites, CCAFS SLC 40 did not have initial success understandably, but returned to much more frequent use and success as SpaceX launched more flights.



Payload Mass, Launch Sites, and Outcomes

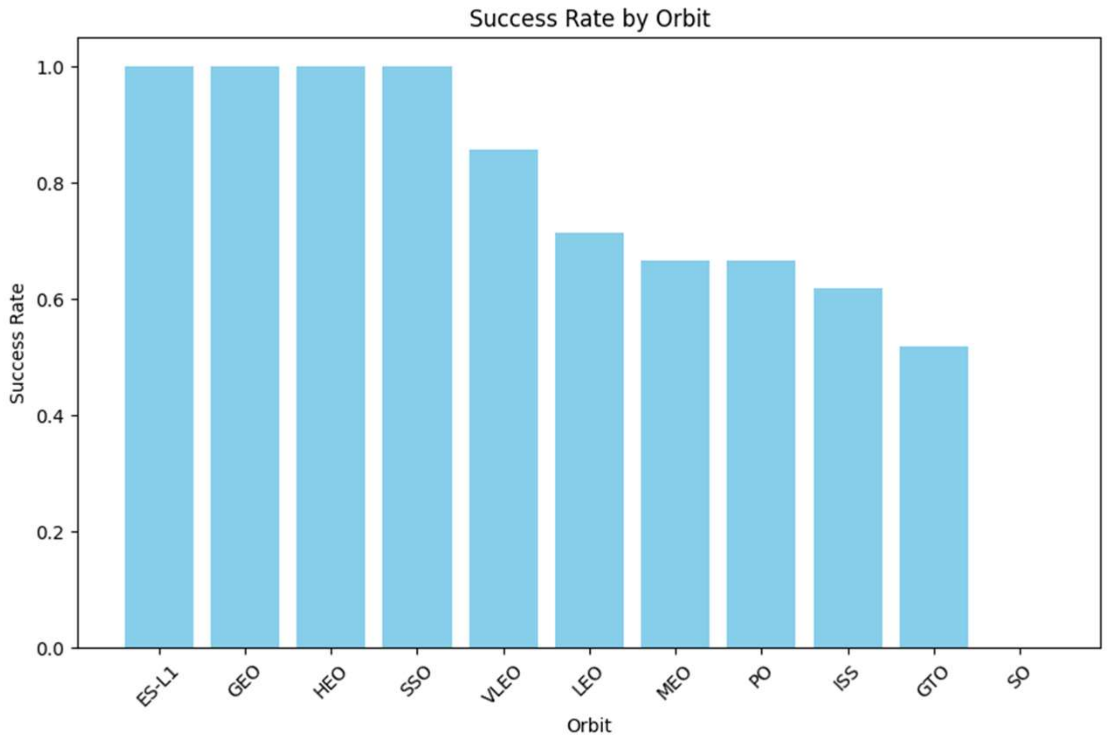
- Wutilizing the graph we can see (excluding one or two flights) that the heavier payloads demonstrate higher success. Keeping in mind that they added more weight as they conducted more flights, I think this demonstrates growing mastery in SpaceX's endeavors.



Success Rates by Orbit

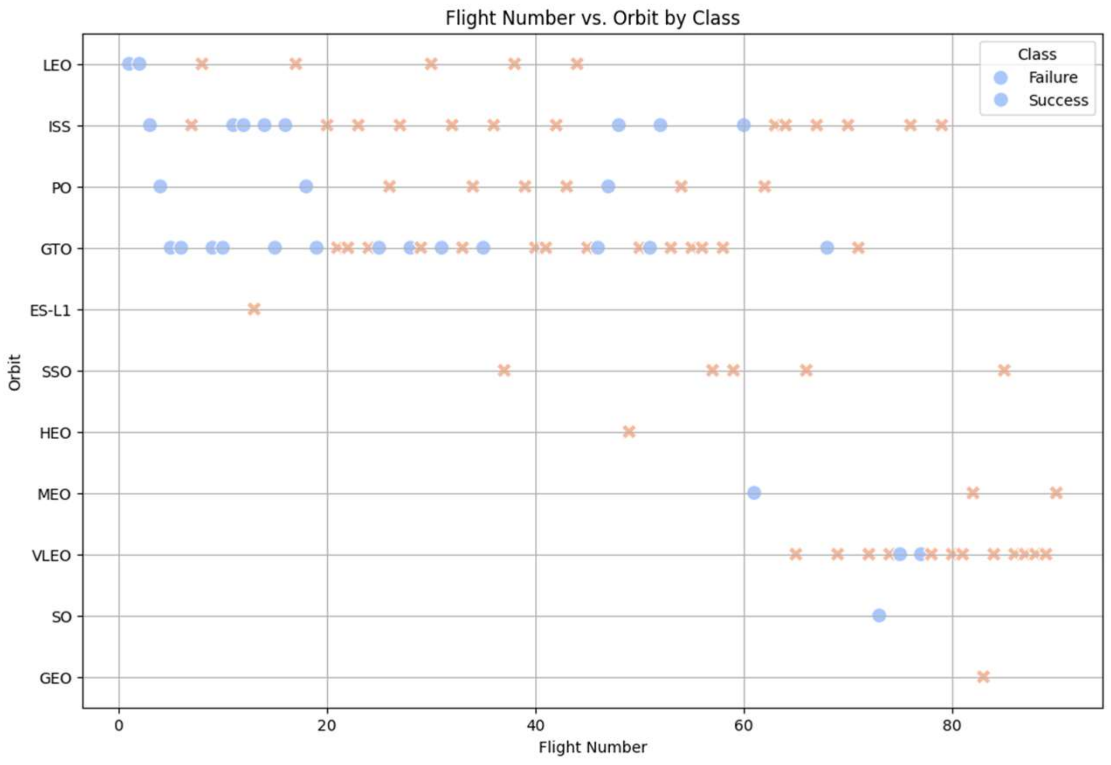
1. ESL1: Earth-Sun Lagrange Point 1
2. GEO: Geostationary Orbit
3. HEO: Highly Elliptical Orbit
4. SSO: Sun-Synchronous Orbit
5. LEO: Low Earth Orbit
6. VLEO: Very Low Earth Orbit
7. MEO: Medium Earth Orbit
8. PO: Polar Orbit
9. ISS: International Space Station
10. GTO: Geostationary Transfer Orbit
11. SO: Suborbital

ES-L1, GEO, HEO, and SSO all have 100(ish) success rates.



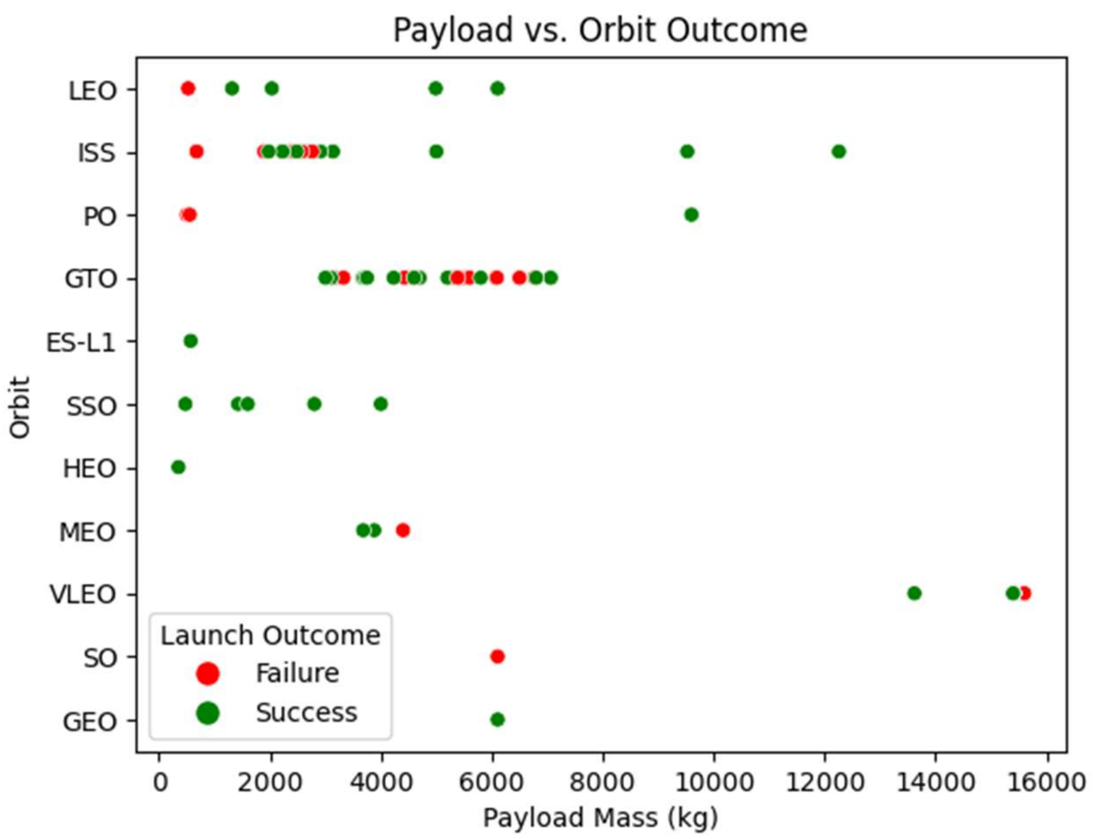
Flight Number vs Orbit by Class

- Based on this graph we can see their progression for orbiting targets, as well as their outcomes. It seems typically they the more flights done with a particular orbital target, the more likely they are to succeed (understandably).



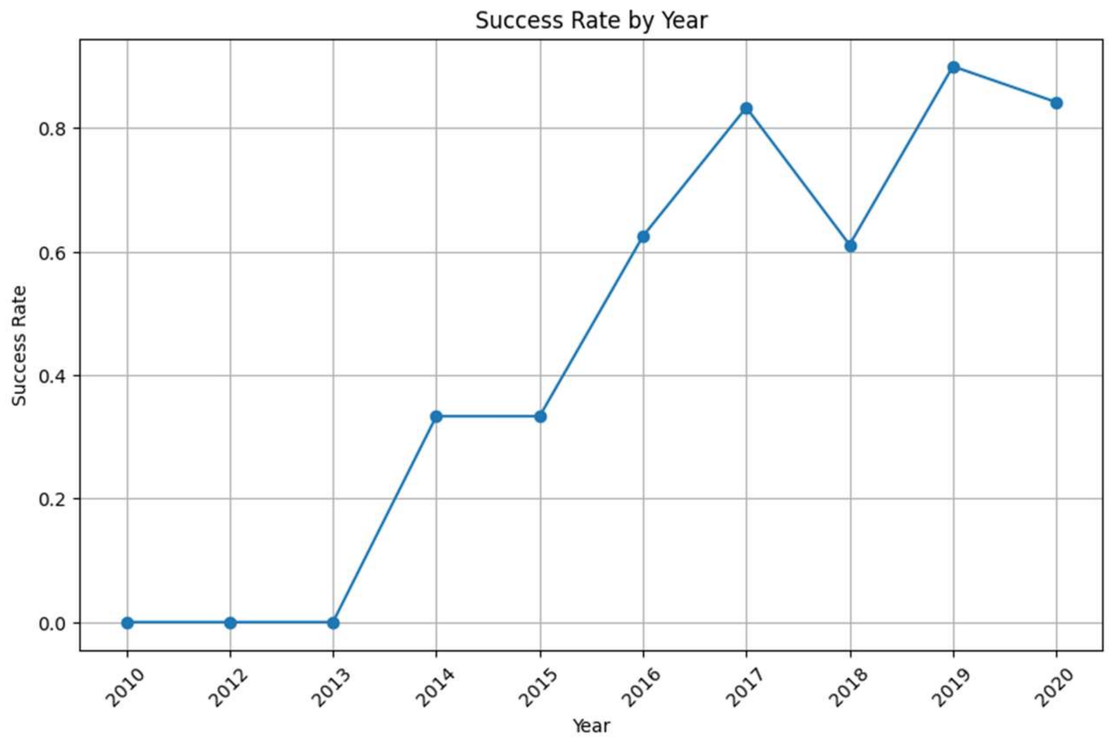
Payload vs Orbit Outcome

- Across LEO, ISS, PO, and GTO, it seems typically that they operate using lower payloads, and these lower paylods typically have more chance to fail.



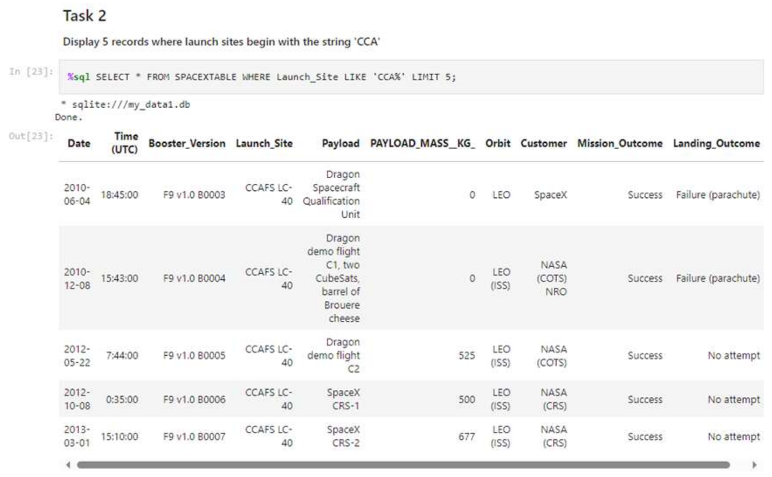
Success Rate by Year

- We can see that as time went on, SpaceX’s mastery or rockets improved. That said, 2018 was a rougher year than most. One might also wonder if in 2020 that their success rate is higher because they were less rocket flights (due to Covid-19).



SQL EDA 1-2

Going by task, these first two tasks are to show us a little bit about the data utilizing SQL.



Task 2

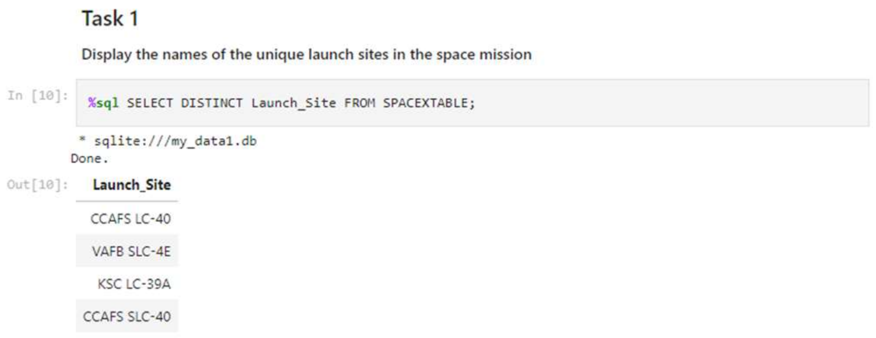
Display 5 records where launch sites begin with the string 'CCA'

```
In [23]: %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[23]:

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



Task 1

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

```
In [10]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

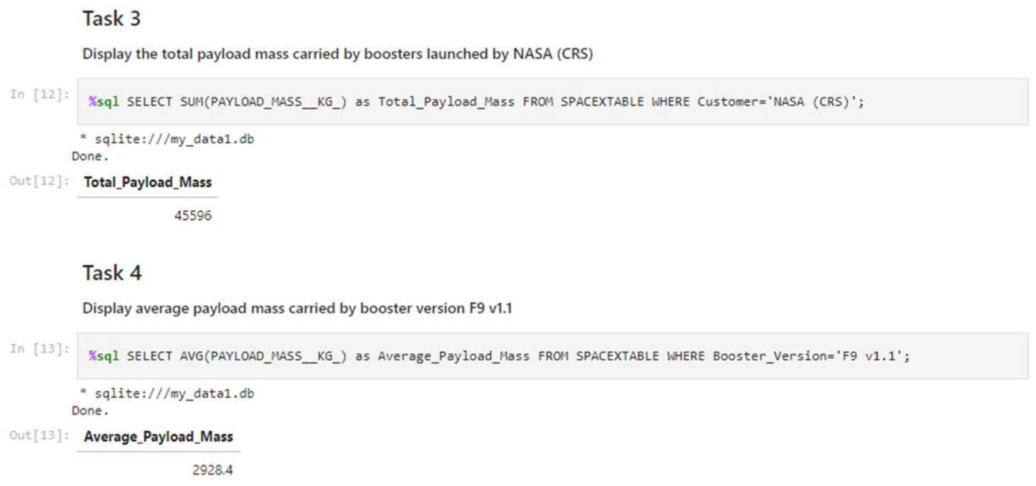
```
 * sqlite:///my_data1.db
Done.
```

Out[10]:

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

SQL EDA 3-4

We're able to see total payload mass over all flights, as well as the average payload.



Task 3

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

```
In [12]: %sql SELECT SUM(PAYLOAD_MASS__KG_) as Total_Payload_Mass FROM SPACEXTABLE WHERE Customer='NASA (CRS)';
```

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

Out[12]:

Total_Payload_Mass
45596

Task 4

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

```
In [13]: %sql SELECT AVG(PAYLOAD_MASS__KG_) as Average_Payload_Mass FROM SPACEXTABLE WHERE Booster_Version='F9 v1.1';
```

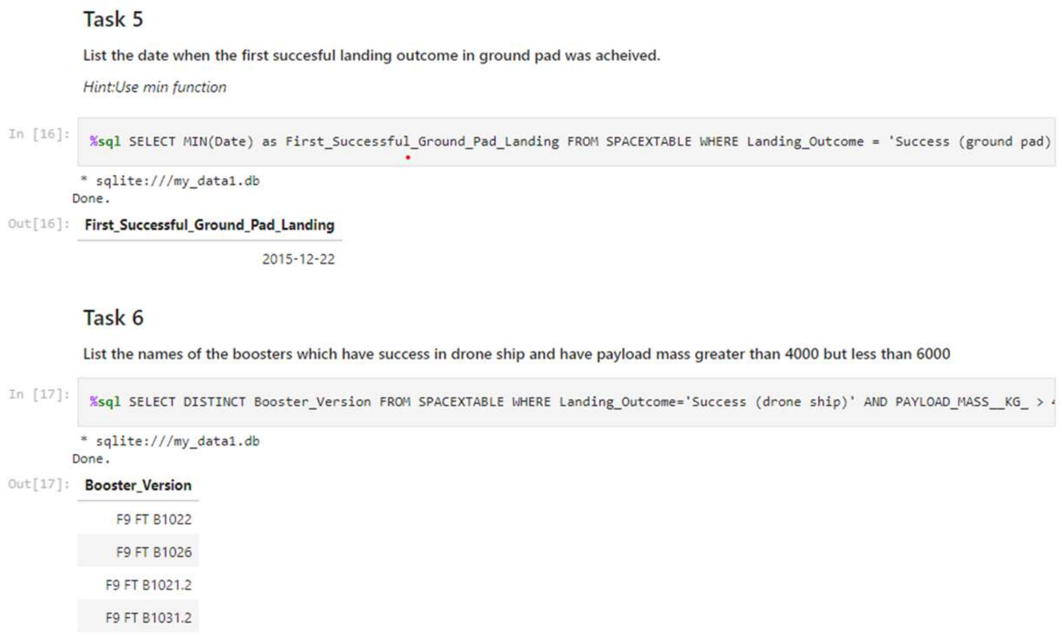
```
 * sqlite:///my_data1.db
Done.
```

Out[13]:

Average_Payload_Mass
2928.4

SQL EDA 5-6

- These tasks demonstrate the first successful ground pad landing, and additionally, drone landing successes for payload masses more than 4,000kg, and less than 6,000kg.



Task 5

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

Hint:Use min function

```
In [16]: %sql SELECT MIN(Date) as First_Successful_Ground_Pad_Landing FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)
```

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

Out[16]:

First_Successful_Ground_Pad_Landing
2015-12-22

Task 6

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

```
In [17]: %sql SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome='Success (drone ship)' AND PAYLOAD_MASS__KG_ >
```

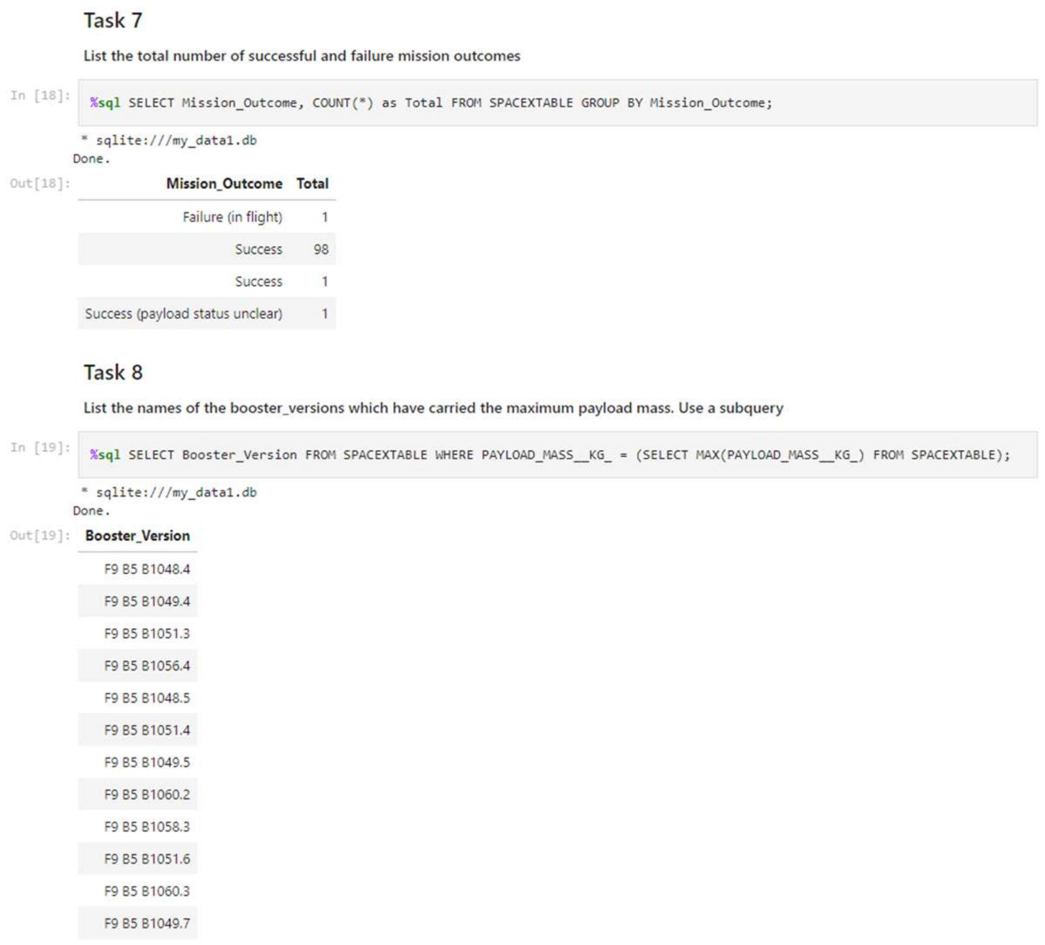
```
 * sqlite:///my_data1.db
Done.
```

Out[17]:

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

SQL EDA 7-8

- We're able to see the total mission outcomes, as well as which booster versions were able to carry the heaviest payloads (arguably making them the most powerful).



Task 7

List the total number of successful and failure mission outcomes

```
In [18]: %sql SELECT Mission_Outcome, COUNT(*) as Total FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

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

Out[18]:

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

Task 8

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

```
In [19]: %sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

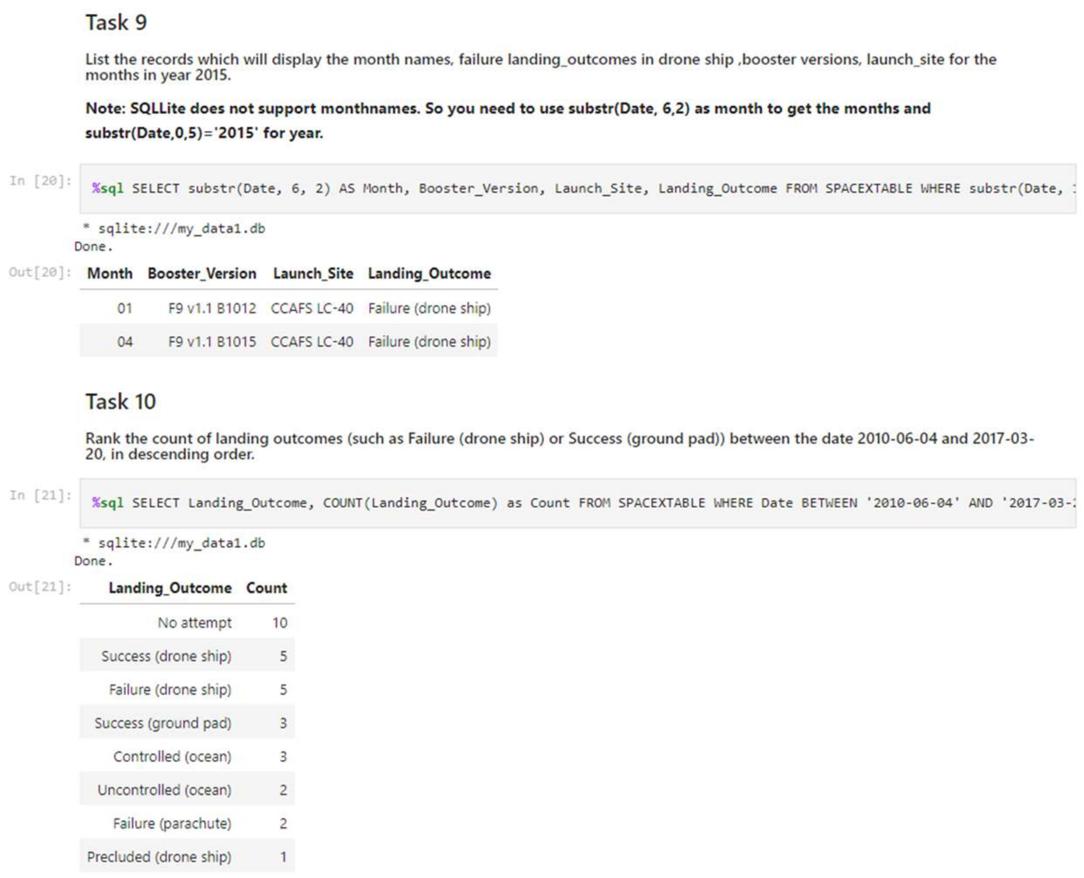
```
 * sqlite:///my_data1.db
Done.
```

Out[19]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

SQL EDA 9-10

- Our last set of tasks show us successes and failures between some specific date ranges.



Task 9

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.

```
In [20]: %sql SELECT substr(Date, 6, 2) AS Month, Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTABLE WHERE substr(Date, :
```

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

Out[20]:

Month	Booster_Version	Launch_Site	Landing_Outcome
01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Task 10

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.

```
In [21]: %sql SELECT Landing_Outcome, COUNT(Landing_Outcome) as Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-:
```

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

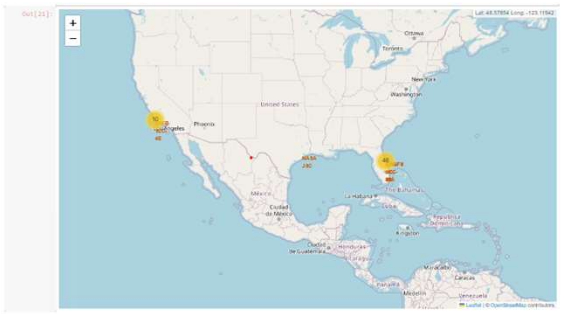
Out[21]:

Landing_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

Folium & Proximity

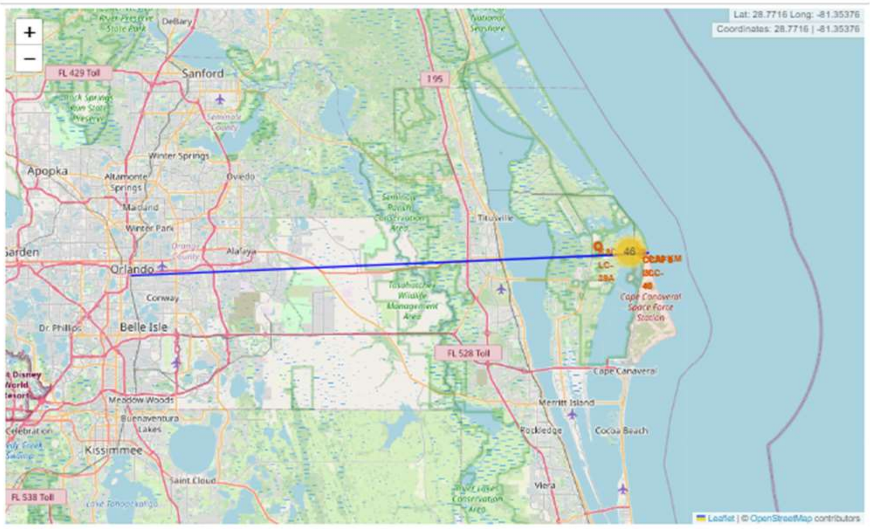
- Our folium setup is supposed to answer four main questions:
- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities?

- Based on our findings illustrated in our maps, visualizations, and calculations: our launch site is infact close to infrastructure and the coastline, but not quite so close to the city.



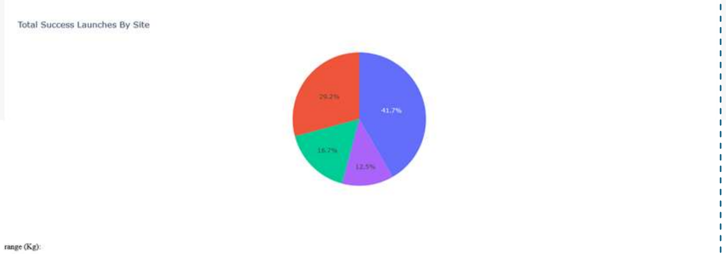
```
In [30]: # Calculate distances
         distance_to_railroad = calculate_distance(launch_site_lat, launch_site_lon, *railroad_coord)
         distance_to_highway = calculate_distance(launch_site_lat, launch_site_lon, *highway_coord)
         distance_to_city = calculate_distance(launch_site_lat, launch_site_lon, *city_coord)

         distance_to_railroad, distance_to_highway, distance_to_city

Out[30]: (1.2168834892954372, 7.688436563401537, 78.58591633997781)
```

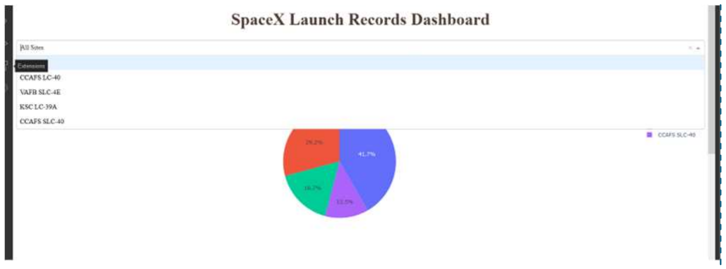
Out[29]:

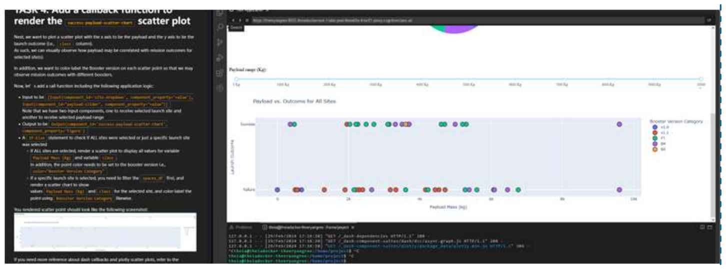

Lat: 28.7716 Long: -81.35376
Coordinates: 28.7716 | -81.35376
Sanford
Apopka
Orlando
Belle Isle
Kissimmee
Titusville
Cape Canaveral
Cocoa Beach
Leaflet | © OpenStreetMap contributors

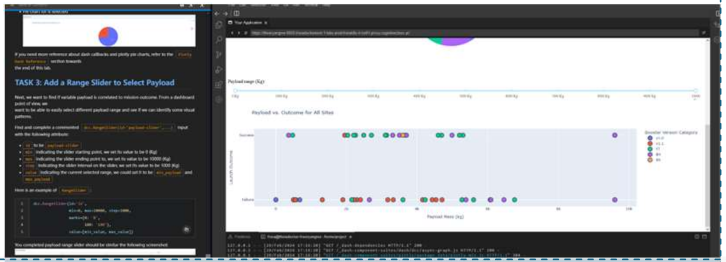
Plotly Dashboarding

- Utilizing our Plotly Dashboard (which, my copy and paste would not work on), we're able to illustrate our success rate by launch site, and additionally, see that lower weighted payloads are often more successful than heavier ones.





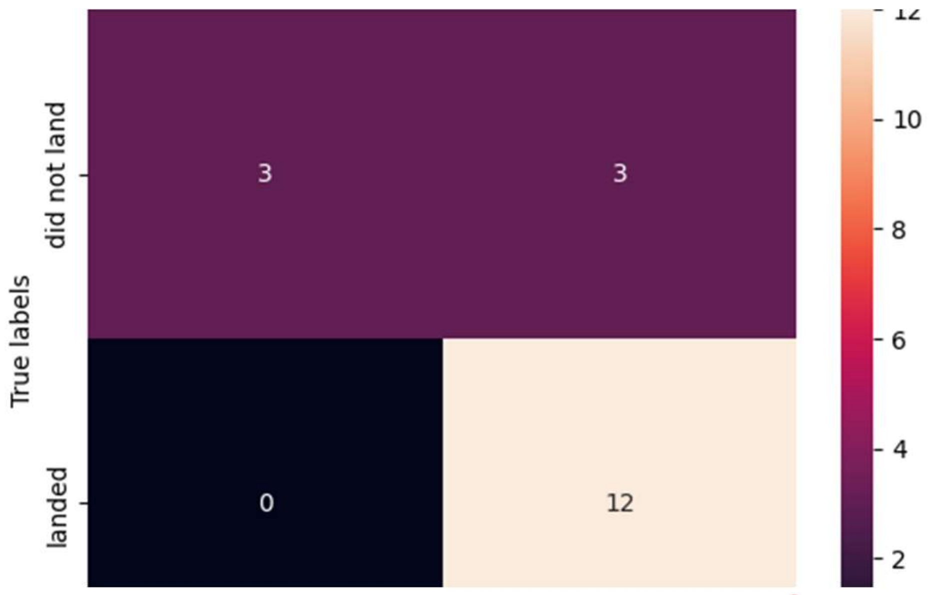


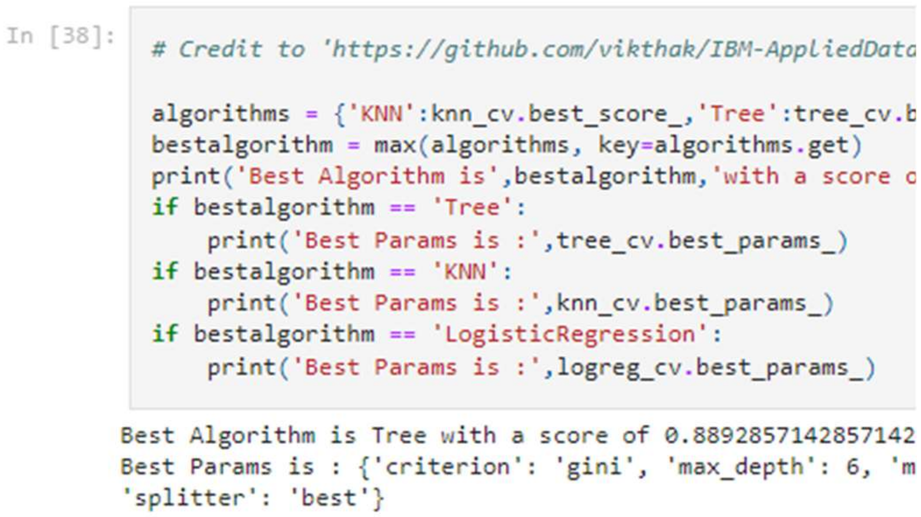


TASK 3: Add a Range Slider to Select Payload
Payload range (Kg):
Payload vs. Outcome for All Sites
Booster Version Category
Payload Mass (kg)

Machine Learning

- As stated before in the earlier machine learning slide, our best model is the Decision Tree.
- As far as our predictability goes, the confusion demonstrates that while our model works well, it does occasionally throw a false positive (crashed marked as landed).





```
In [38]: # Credit to 'https://github.com/vikthak/IBM-AppliedData

algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.b
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score o
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8892857142857142
Best Params is : {'criterion': 'gini', 'max_depth': 6, 'm
'splitter': 'best'}
```

Final Remarks

As time goes on, SpaceX has improved their flight capabilities, and hit their goals more consistently/often.

SpaceX is especially good at hitting it’s ES-L1, GEO, HEO, and SSO targets (100% or near 100% rates).

The data demonstrates that higher payload targets are more successful, but additionally, these flights are later. Keeping context in mind, does weight mean better flights? Or is more weight a result of time? I am inclined to believe the later.

When making predictions to hit targets: our Decision Tree algorithm will be best.

Thank you!