IBM Data Science Capstone

SpaceX - Iain Campbell 10-10-2021

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data Collection
  - Data wrangling
  - EDA with Visualization / EDA with SQL
  - Folium interactive maps
  - Dashboards with Plotly, Dash
  - Predictive analysis – Classification
- **Summary of all results**
  - EDA results
  - Analytics outcomes
  - Predictive analytics conclusions

Introduction

Project background and context:

We are building a case for competing with SpaceX on the basis of determining the risks associated with launching and reusing first stage rockets.

Publicly available data suggests that SpaceX's Falcon 9 rocket could cost 62 million USD, while competitors' rockets can cost upward of 165 million.

SpaceX has the competitive advantage of being able to reuse the first stage of its propulsion system by landing the components with minimal damage after use.

If we examine and determine the varied factors involved with successful landings, we can use the data as a basis for competitively bidding with SpaceX on launch projects.

Common problems

- What are the factors that contribute to a successful rocket landing?
- What factors are most important in determining the success rate of rocket landings?
- In order for rocket landings to have the highest success rate, what conditions must be met?

Methodology

Section 1

Methodology

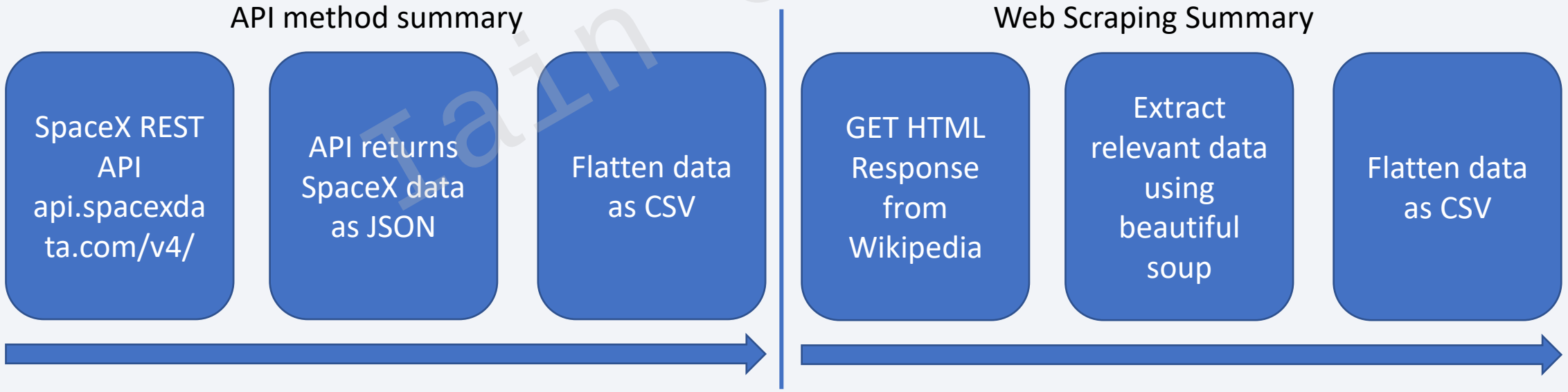
Executive Summary

- Data collection methodology:
  - SpaceX public data via REST API.
  - Wikipedia web scraping (Link)
- Perform data wrangling
  - Conversion via one hot encoding
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly, Dash
- Perform predictive analysis using classification models
  - How to build, tune and, evaluate classification models

Data Collection

SpaceX launch data is publicly available via the SPACEX REST API

- Rocket Versions, Payloads,, Outcomes, Customers, Launch sites, specifications and more.
- Additional information related to the falcon 9 rocket launch data is obtainable using Beautiful Soup Web scraping.



Data Collection – SpaceX API

1. Get API response
2. Convert response to JSON
3. Clean data
4. Create dictionary and assign to data frame*
5. Filter data frame and export as .CSV

[Github Link]

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

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

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

```
getBoosterVersion(data)
```

```
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)
```

*

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

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

Data Collection - Scraping

1. Get HTML response

```
page = requests.get(static_url)
```

2. Create Soup object

```
soup = BeautifulSoup(page.text, 'html.parser')
```

3. Locate tables

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

4. Get column names

```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5. Create Dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant 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'] = []
```

6. Append to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable pl
ainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding
to launch a number
```

7. Convert Dictionary to data frame

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

8. Export data frame as. CSV

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

[Github Link]

Data Wrangling

Data Wrangling – Cleaning and unifying complex data sets.
It was necessary to convert categorical data to numerical data.

Sample:
The notation for landings offered significant variety within the dataset:

- True ocean – Successfully landed in a specific ocean region
- False ocean – Failed landing in a specific ocean region
- True RTLS – Successfully landed on a landmass pad
- False RTLS failed landing on a landmass pad
- True ASDS – Successful landing on drone ship
- False ASDS – Failed Landing on a drone ship

In this case assigning a 1 for Successful landing and, 0 for Failed landing

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

Perform exploratory data analysis(EDA) on Dataset

Calculate the number of launches at each site.

Calculate the number and occurrence of each orbit

Calculate number and occurrence of mission outcomes per orbit type

Calculate landing outcome labels from the outcome column.

Export as .CSV

Calculate success

Github Link

EDA with Data Visualization

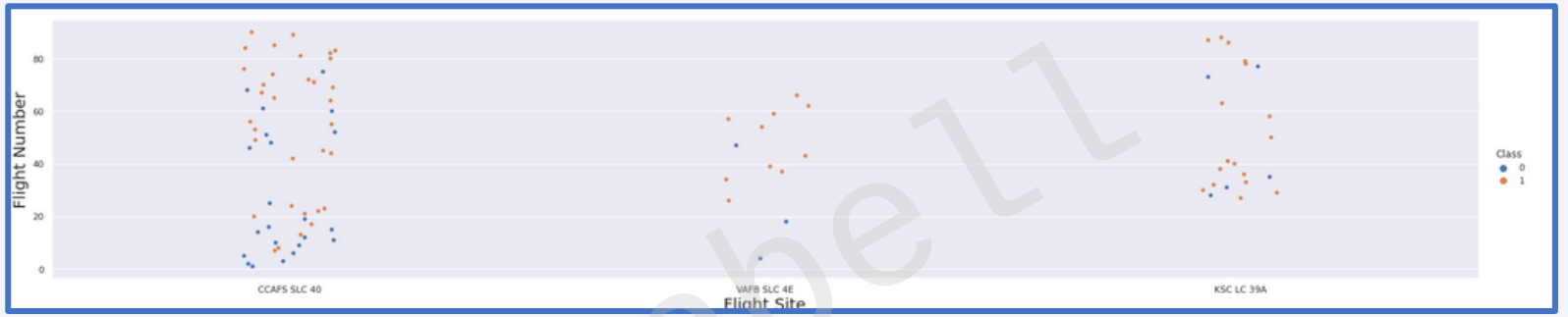
Exploratory data analysis – Analyzing data sets to summarize key observations using statistical plots and other visualization graphics.

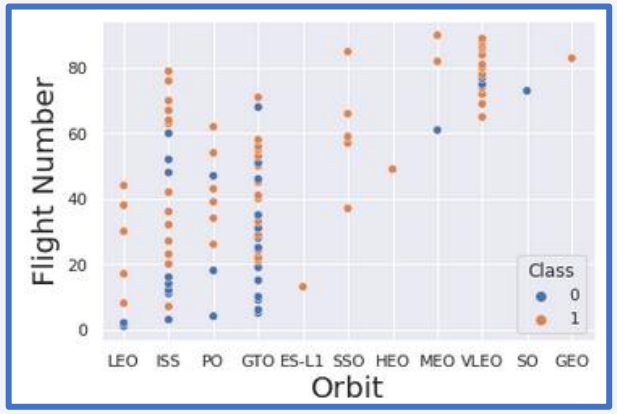
Scatter Graphs:

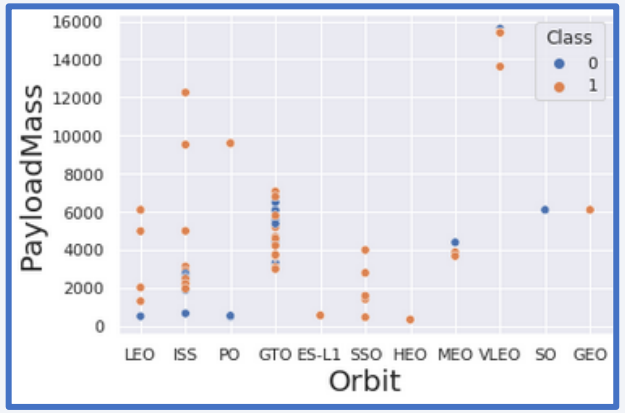
- Flight number vs Payload Mass
- Flight number vs Launch Site
- Payload vs Launch Site
- Orbit vs Flight number
- Payload vs Orbit Type
- Orbit vs Payload Mass

Samples can be seen to the right. This type of plot is effective at showing how much one variable is impacted by another and other relationships.

[Github Link]





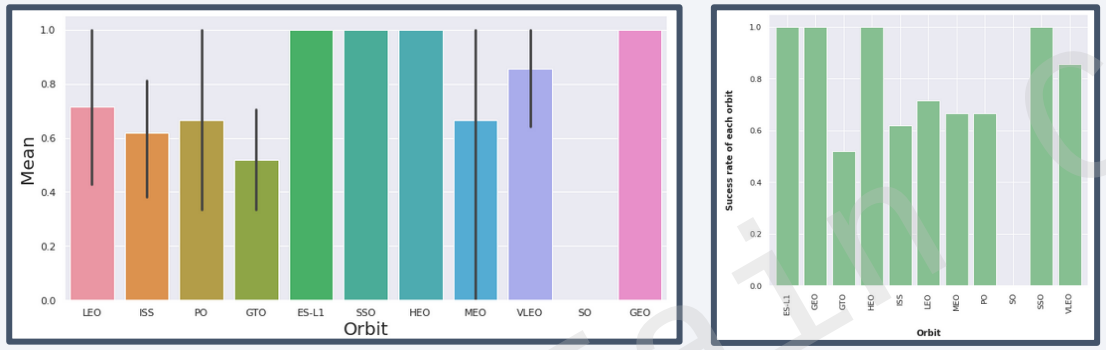


EDA with Data Visualization

Bar Graph:

- Mean Vs Orbit

Excellent for comparing data from different groups and are effective at showing changes in data over time.
Sample:



Key takeaway:

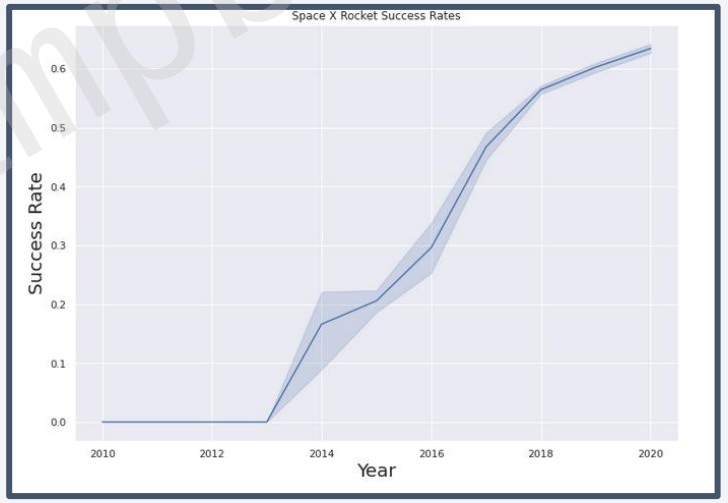
- ES-L1, GEO, HEO, SSO have the highest Success rates. SO has not been successful.

[Github Link]

Line Graph:

- Success Rate Vs Year

Useful for observing trends. The line graph can enable predications to be made.
Sample:



Key takeaway:

- You can observe that the success rate since 2013 kept increasing till 2020.

EDA with SQL

SQL was chosen to handle the required relational database operations. For this project I used IBMs DB2 cloud service.

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

The following Queries were performed:

- 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. (Using a subquery)
- 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

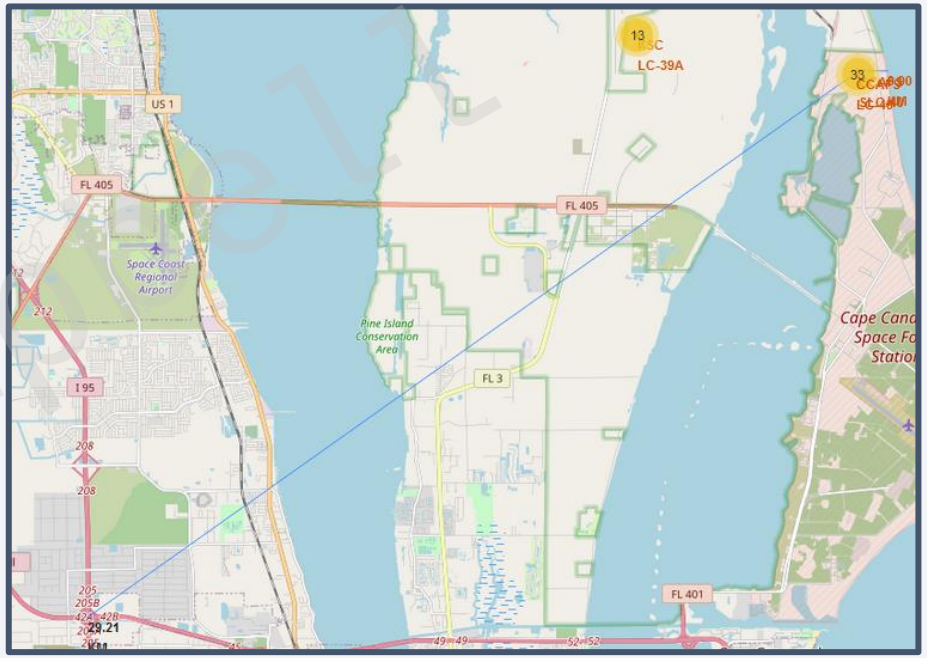
[Github Link]

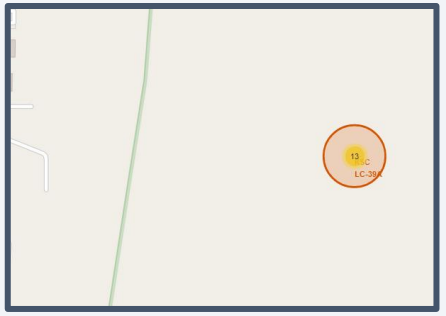
Build an Interactive Map with Folium

Using folium, I visualized manipulated data on an interactive leaflet map.
The latitude and longitude coordinates for each launch site were marked appropriately using circle markers and labels.
I used the polyline feature to visualize the calculations to the following the following questions and answers*

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

*Example: Using Haversine's formula I calculated the distance from a Launch Site to various landmarks.





[Github Link]

Build a Dashboard with Plotly Dash

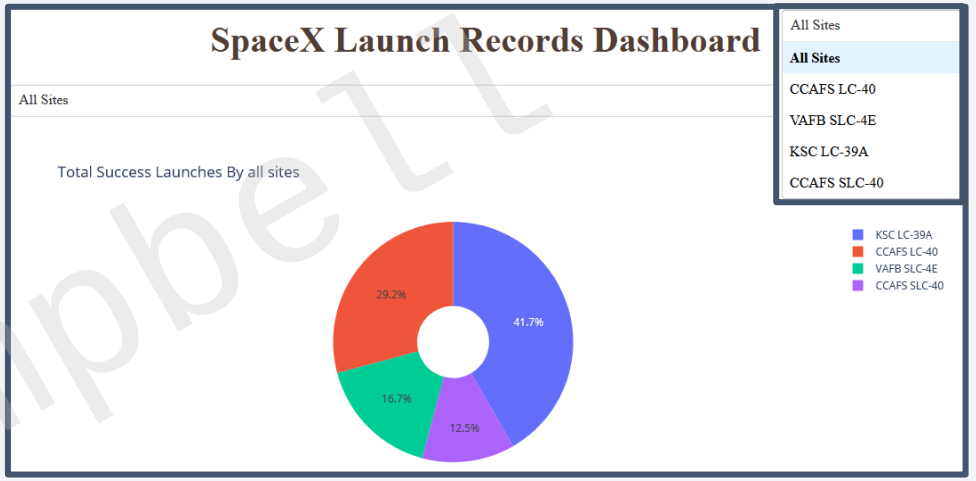
Used PythonAnywhere to build a dashboard with the Flask and Dash web framework.

Interactive Pie Chart:

- Total successes by all sites
  - Percentage of successes relative to the launch site
  - Displays relative proportions of data
  - Slice size is proportional to the quantity

Scatter Graph:

- Shows the relationship of Outcome and Payload Mass (Kg) for different Booster Versions
  - Shows the relationship between variables
  - Appropriate for non linear patterns
  - Max and min values are easily determinable



[Github Link]

Predictive Analysis (Classification)

Model Building summary:

- Load feature engineered data into data frame
- Transform into NumPy arrays
- Standardise and transform the data
- Split data into training and test data sets
- Check the number of test samples created
- Collate machine learning algorithms
- Set our parameters and algorithms to GridSearchCV
- Fit datasets into the GridSearchCV objects and train our model

Model Evaluation:

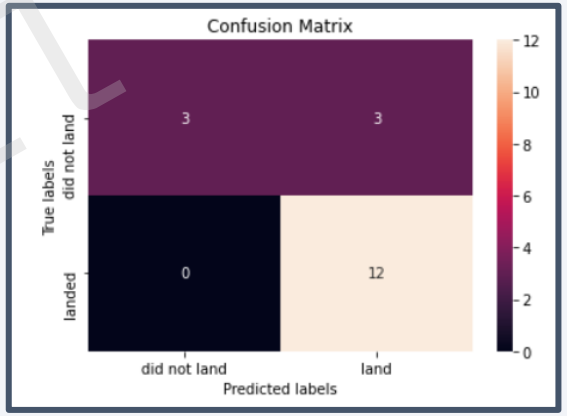
- Check the accuracy of each model
- Get tuned hyperparameters for each algorithm
- Plot confusion Matrix

Model Improvement:

- Iterate on engineered features
- Tune Algorithms

Identify best classification model

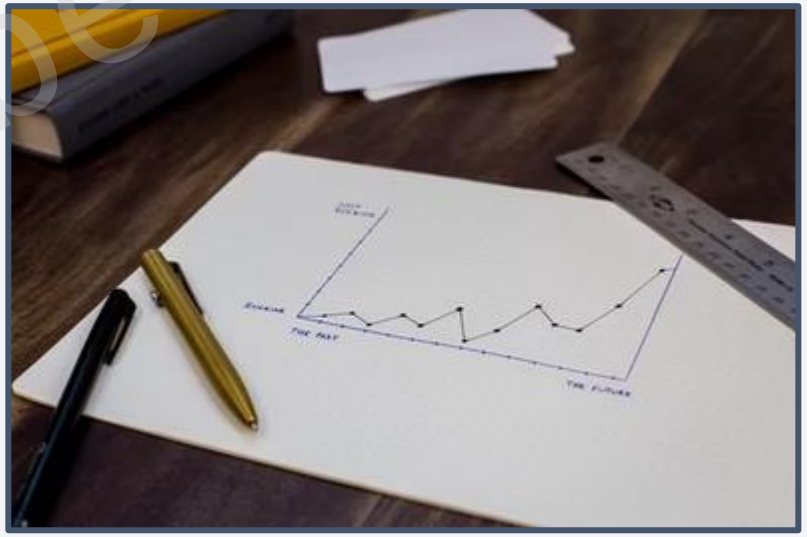
- Identify the model with the best accuracy score
- Scores are found at the bottom of the Jupyter notebook.



	Algorithm	Accuracy
0	Logistic Regression	0.846429
1	SVM	0.848214
2	KNN	0.848214
3	Decision Tree	0.873214

Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



End of Methodology Section

Insights Drawn from EDA

Section 2

Flight Number vs. Launch Site

- There is a positive correlation between the total number of launches and the number of successful launches.
- There is a significant increase in success for flight numbers greater than 30

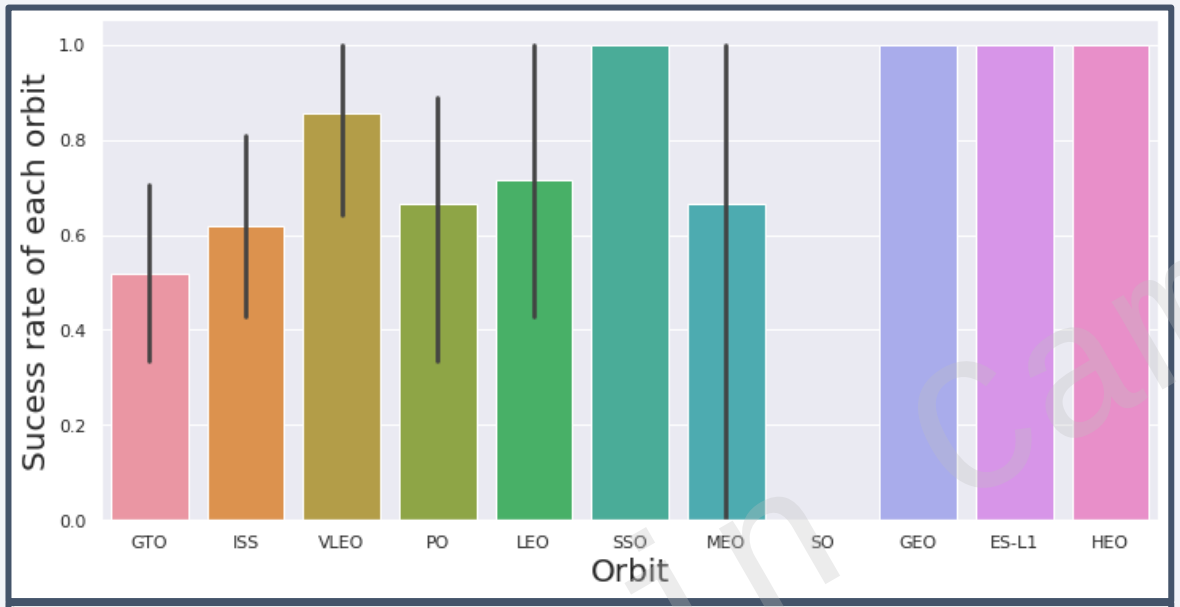
The relationship between fight number and success at a launch site may not be meaningful for decision making purposes.

Payload vs. Launch Site

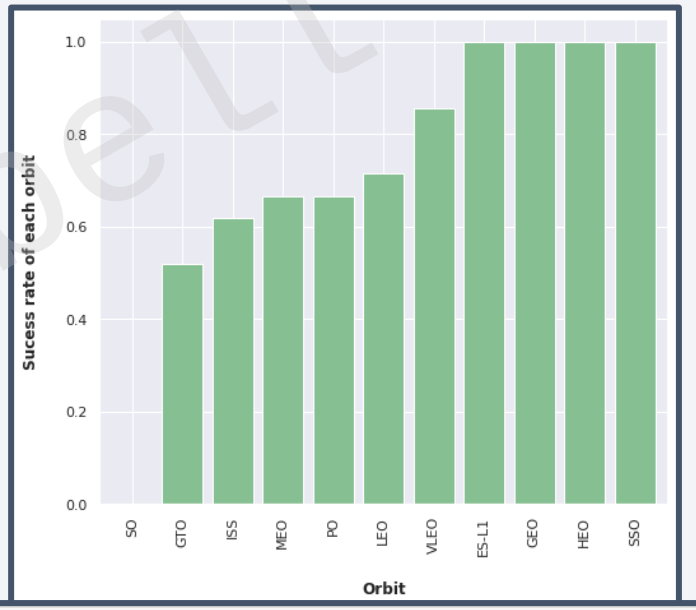
- The greater payload mass at launch site CCAFS SLC 40 achieves a higher success rate.
- The data suggests that a payload mass greater than 8000kg could contribute to increased success rate.

The pattern is not conclusive. We cannot identify if increased payload mass at each launch site is meaningful with respect to successful launches.

Success Rate vs. Orbit Type




```
df.groupby(['Orbit']).mean()
sns.catplot(x="Orbit",y="Class", kind="bar",data=df, aspect=2, order=df['Orbit'].value_counts().index)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Sucess rate of each orbit",fontsize=20)
plt.show()
```

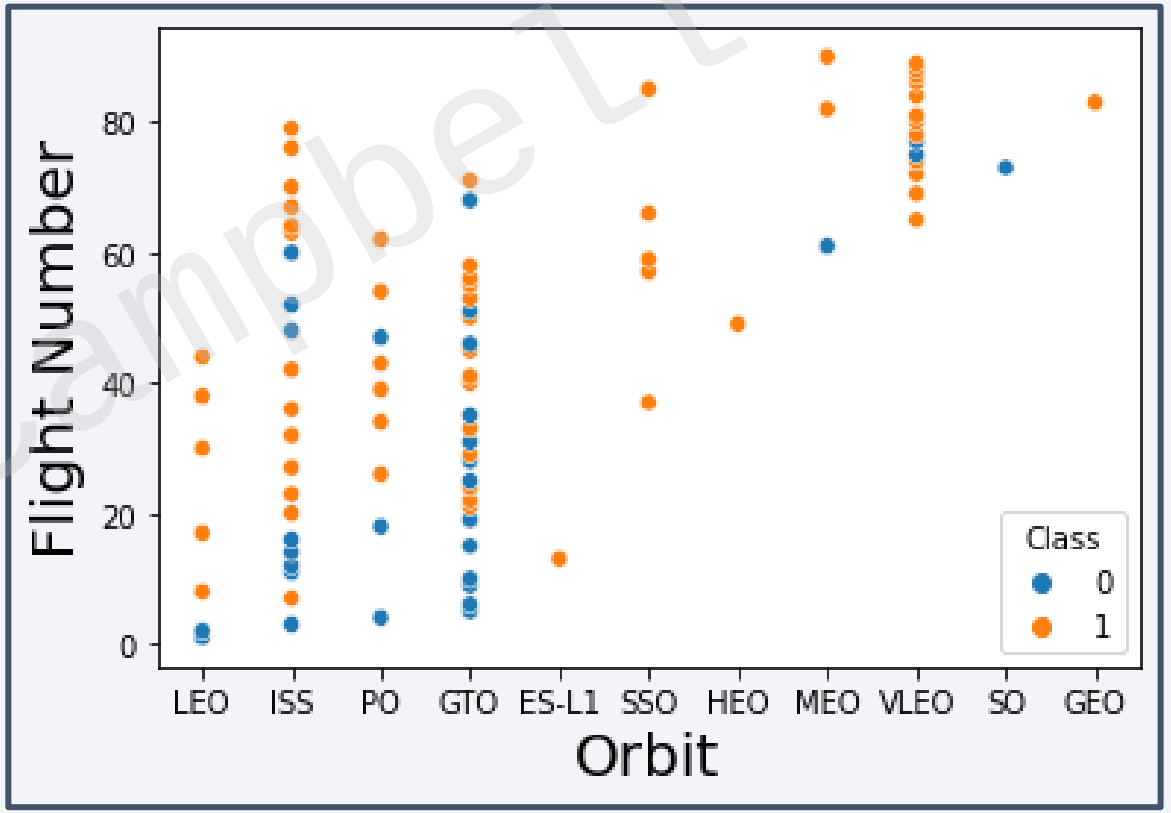



```
xh = df.groupby('Orbit')['Class'].mean().sort_values()
ax = xh.plot(kind='bar', figsize=(8, 7), color='#86bf91', zorder=2, width=0.8)
ax.set_xlabel("Orbit", labelpad=20, weight='bold', size=12 )
ax.set_ylabel("Sucess rate of each orbit", labelpad=20, weight='bold', size=12);
```


ES-L1, GEO, HEO, SSO have the highest Success rates. SO has seen limited success.

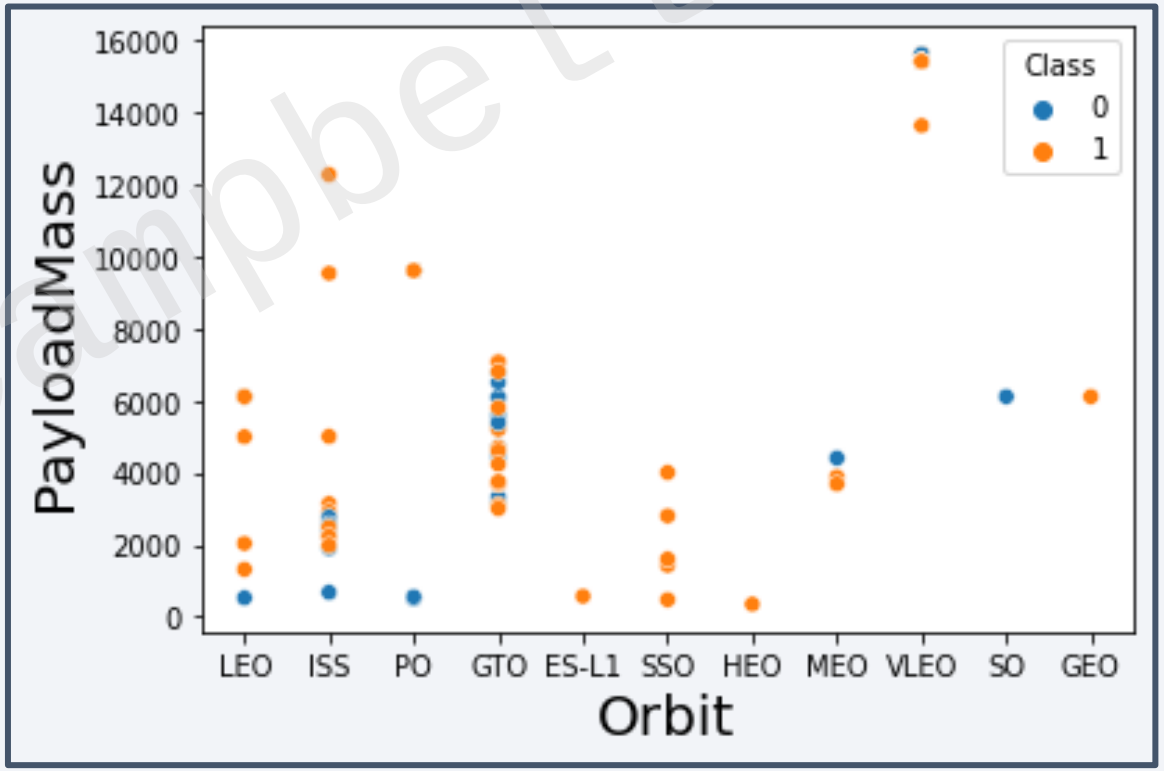
Flight Number vs. Orbit Type

- LEO orbit the Success appears related to the number of flights.
- There seems to be no relationship between flight number when in GTO orbit



Payload vs. Orbit Type

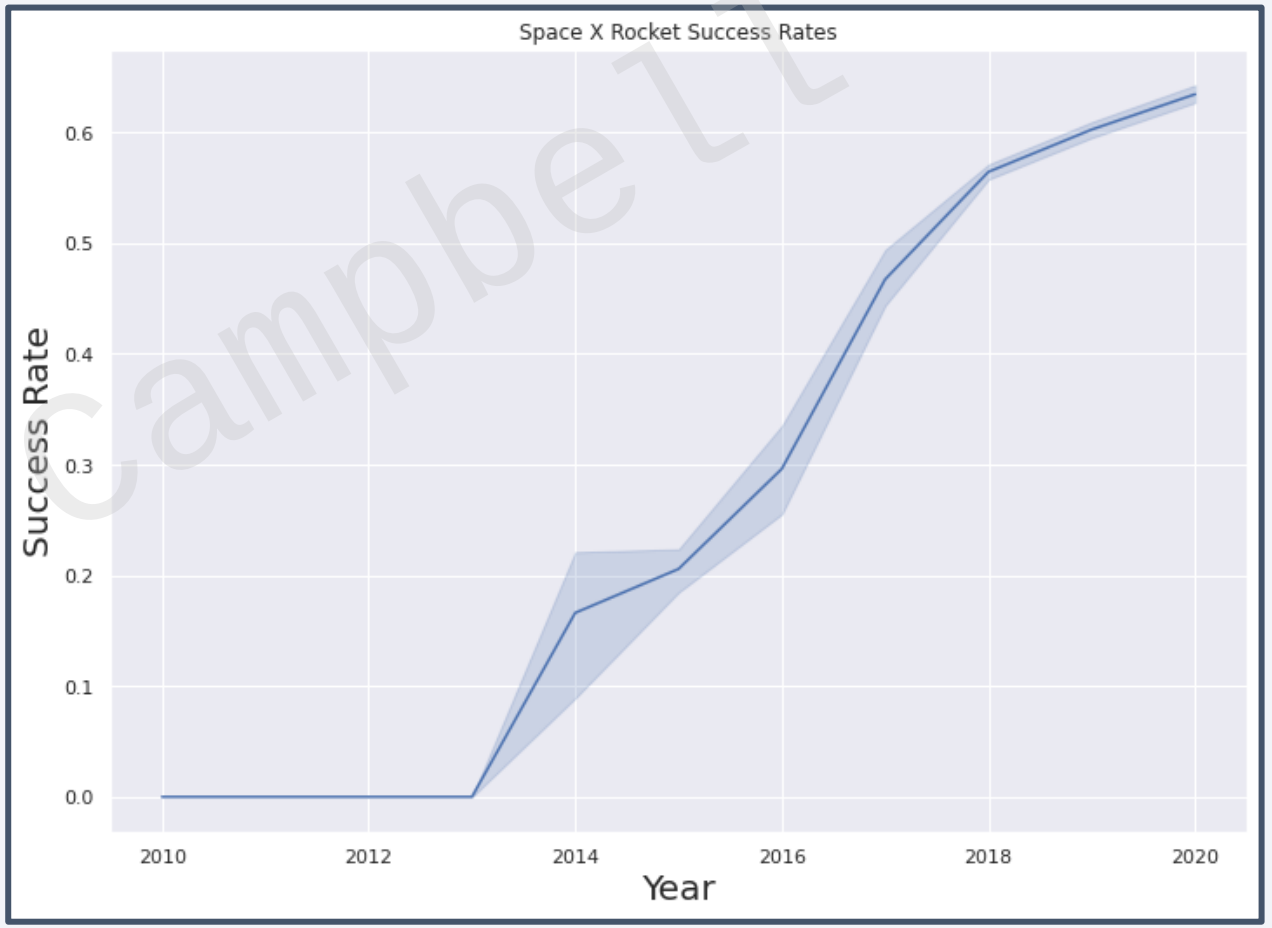
Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.



Launch Success Yearly Trend

You can observe that the success rate has continued to increase since 2013.

- The trend continued until the end of our time series 2020




EDA With SQL
Section 3
Iain Campbell

All Launch Site Names

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

```
%sql SELECT DISTINCT LAUNCH_SITE AS "Launch_Sites" FROM SPACEXTBL;
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

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

Description:

Keyword **DISTINCT** will pull unique data values from the **Launch_Sites** attribute in the **SPACEXTBL** data table.

Launch Site Names Begin with 'CCA'

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

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

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

Description:

Keyword **LIMIT** followed by integer 5 fetches the integer number of records from **SPACEXTBL**.

Keyword **LIKE** combined with a wild card **'%'** appended to the end of a string suggests that the **Launch_Site** name must start with '**KSC**'.

Total Payload Mass

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEXTBL WHERE
CUSTOMER = 'NASA (CRS)';
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

Total Payload Mass by NASA (CRS)
45596

Description:

Function **SUM** calculates the total records within the **PAYLOAD_MASS_KG** attribute in **SPACEXTBL**. The clause **WHERE** applies a filter to the data, to fetch only the records containing a specific request. In this case 'NASA (CRS)'.

Average Payload Mass by F9 v1.1

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM
SPACEXTBL \
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

Average Payload Mass by Booster Version F9 v1.1
2928

Description:

Function **AVG** calculates the Average from the records within the **PAYLOAD_MASS_KG** attribute in **SPACEXTBL**.

The clause **WHERE** applies a filter to the data, in order to apply the calculation to a specific subset of the data. – In this case Booster_Version records with a “F9 v1.1” value.

First Successful Ground Landing Date

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

Hint:Use min function

```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEXTBL \
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

First Succesful Landing Outcome in Ground Pad
2015-12-22

Description:

Function **MIN** calculates the minimum date from the records within the **DATE** attribute in **SPACEXTBL**.

The clause **WHERE** applies a filter to the data, in order to apply the calculation to a specific subset of the data. – In this case the Landing_Outcomes with a “Success (ground pad)” value.

Successful Drone Ship Landing with Payload between 4000 and 6000

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 SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

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

Description:

The clause **WHERE** applies a filters our data to Landing _outcomes = success (drone ship)

Logical operator **AND** specifies additional mandatory conditions for PAYLOAD_MASS records.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful
Mission", \
    sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" \
FROM SPACEXTBL;
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

Successful Mission	Failure Mission
100	1

Description:

Code in parenthesis is read first and bottom up. The code can be broken down into two main components to fetch both the failure and success counts in the same query.

I have used CASE in order to nest sub queries inside another query. **LIKE** with **%** prefixed and appended to a string searches for records with the string anywhere within the record.

Boosters Carried Maximum Payload

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

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload M
ass" FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

Description:

DISTINCT will return only show Unique values in the Booster_Version attribute of the SPACEXTBL. WHERE Filters the results by a condition.

The Same result could have been achieved by:

```SELECT DISTINCT Booster_Version, MAX(PAYLOAD_MASS_KG_) AS 'Maximum Payload mass'
FROM SPACEXTBL GROUP BY Booster_Version ORDER BY 'Maximum Payload Mass DESC```

- **GROUP BY orders the list based on a condition**
- **DESC ensures the list is in descending order**

2015 Launch Records

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql SELECT {fn MONTHNAME(DATE)} as "Month", BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHER
E year(DATE) = '2015' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

Month	booster_version	launch_site
January	F9 v1.1 B1012	CCAFS LC-40
April	F9 v1.1 B1015	CCAFS LC-40

Description:

{fn MONTHNAME(DATE)} is used to get the month as a string from a date. AS aliasing is used to name the attribute appropriately.

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 as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" F
ROM SPACEXTBL \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

Landing Outcome	Total Count
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 COUNT(LANDING__OUTCOME) AS "Rank success count between 2010-06-04 and 2017-03-20"
FROM SPACEXTBL \
WHERE LANDING__OUTCOME LIKE '%Success%' AND DATE > '2010-06-04' AND DATE < '2017-03-20' ;
```

```
 * ibm_db_sa://hpr96438:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databas
es.appdomain.cloud:31321/bludb
Done.
```

Rank success count between 2010-06-04 and 2017-03-20
8

Description:

Selecting only the LANDING_OUTCOME

- WHERE DATE BETWEEN filters the data by two dates.
- COUNT calculates the number of records
- GROUP BY orders the list based on a condition
- DESC ensures the list is in descending order

Interactive Maps with Folium

Section 4

All launch sites

When markers for launch site locations are applied to a global map we can observe that all of the SpaceX sites are in coastally located in the United States of America.

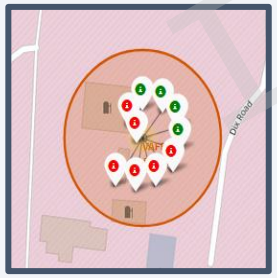
East and West costs are represented. – California and Florida

Launch records for each site

Red – Failure Green - Success

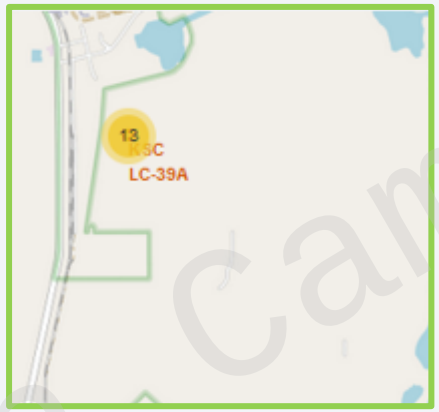
KSC LC39A has the highest number of successful launches.

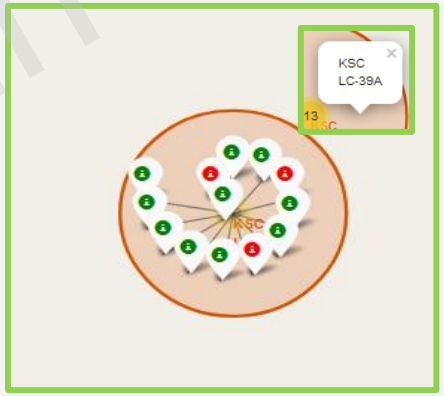
VAFB SLC-4E



Florida

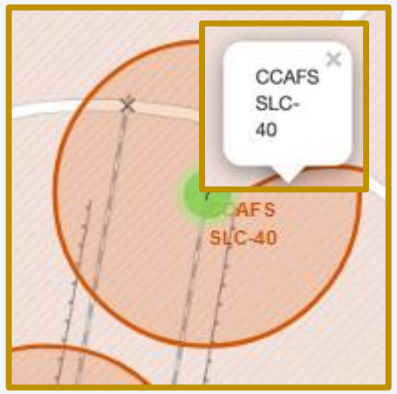
KSC LC-39A

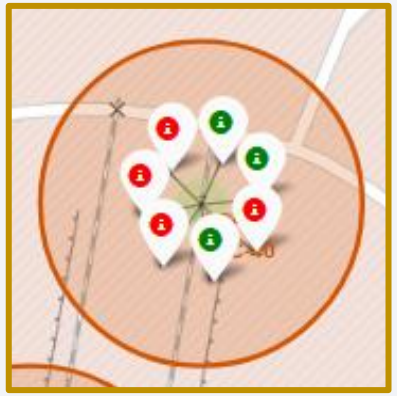




CCAFS LC-40

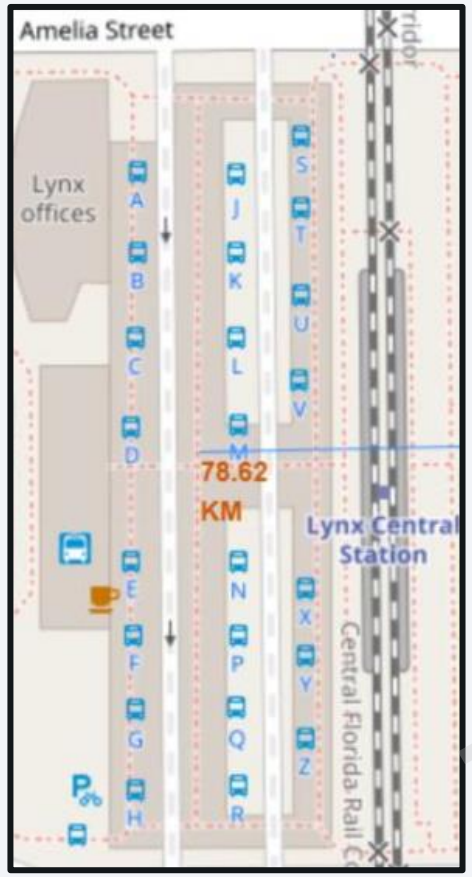
CCAFS SLC-40



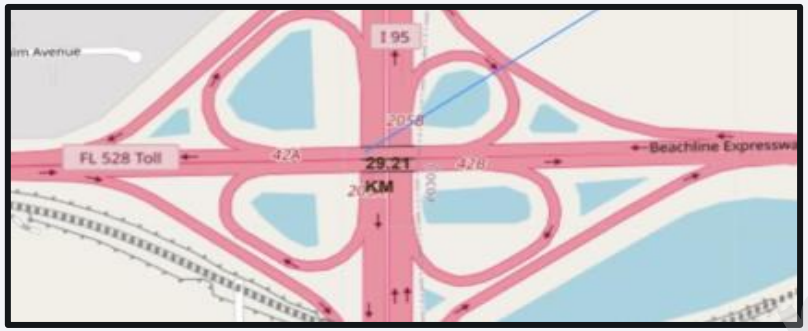


California

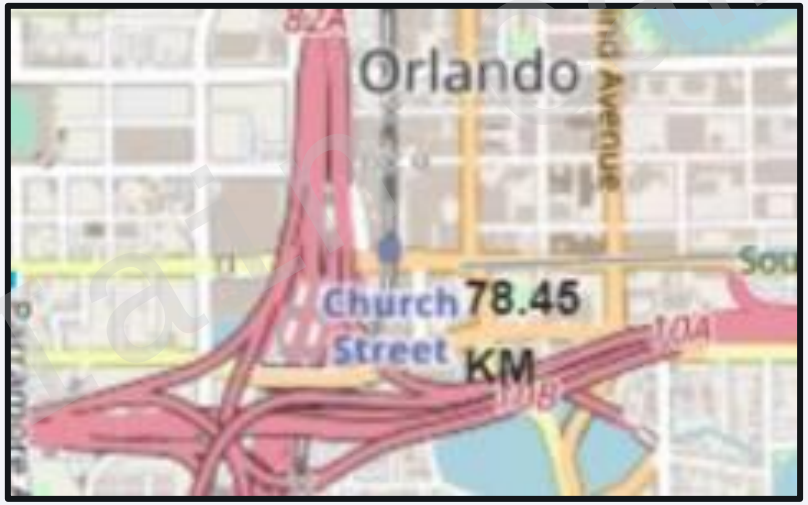
Distances between a launch site to its proximities



Proximity to Railway station 78.62km



Proximity to Highway 29.21km



Proximity to City 74.45km

Proximity to Coast 0.90km

CCAFS-SLC-40 was used as reference for the haversine formula.

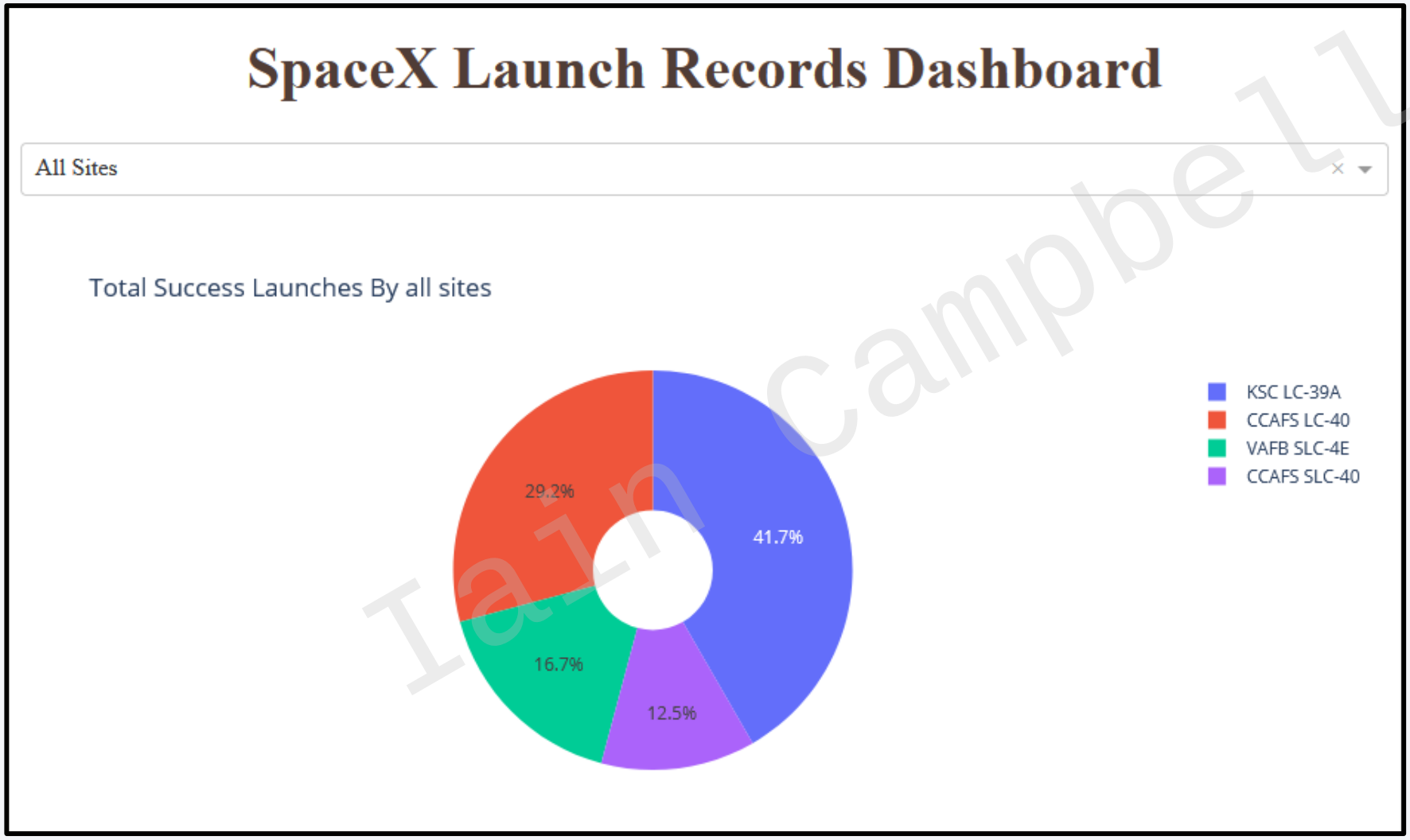
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Dashboard with **Plotly**

Dash

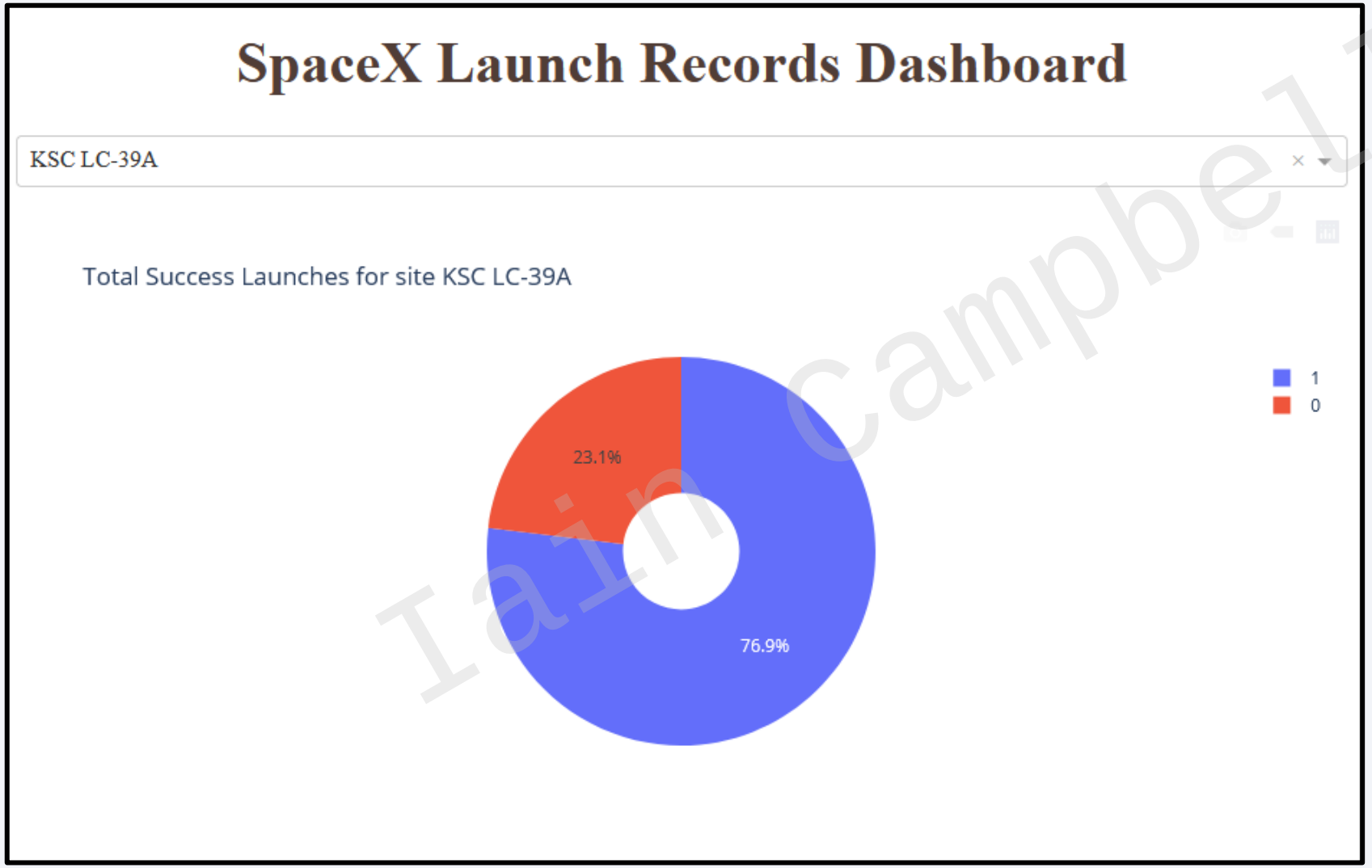
Section 5

Launch Success Count for All Sites



KSC LC-39A has achieved the most successful launches relative to the other sites.

Launch Site with highest success ratio

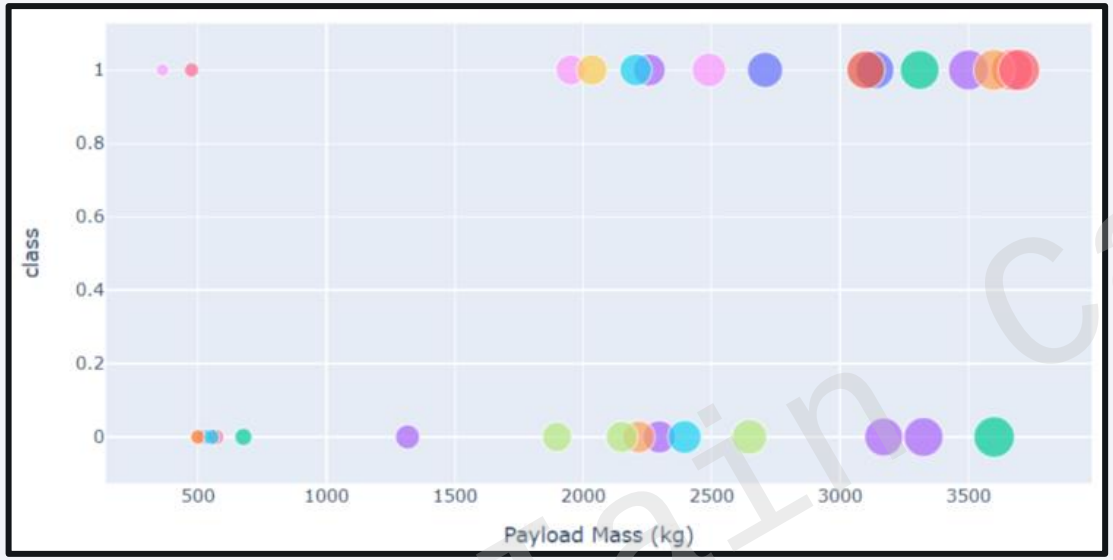


KSC LC-39A has achieved the most successful launches relative to the other sites.

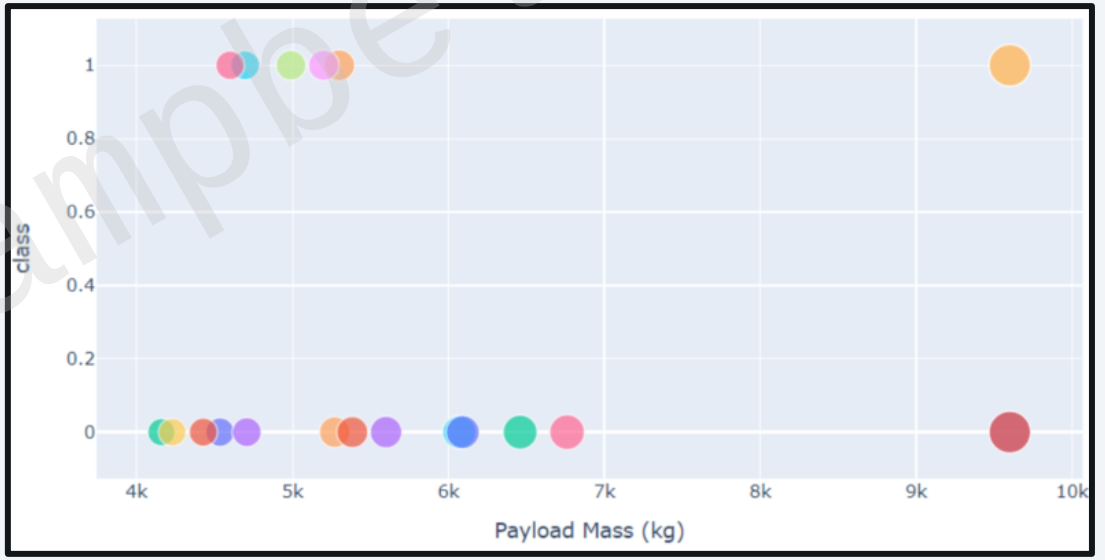
76.9% success
23.1% failure

Payload vs Launch outcomes

Success for rates for lower payloads (0kg-4000kg) is meaningfully higher than heavier payloads (4000kg-10000kg)

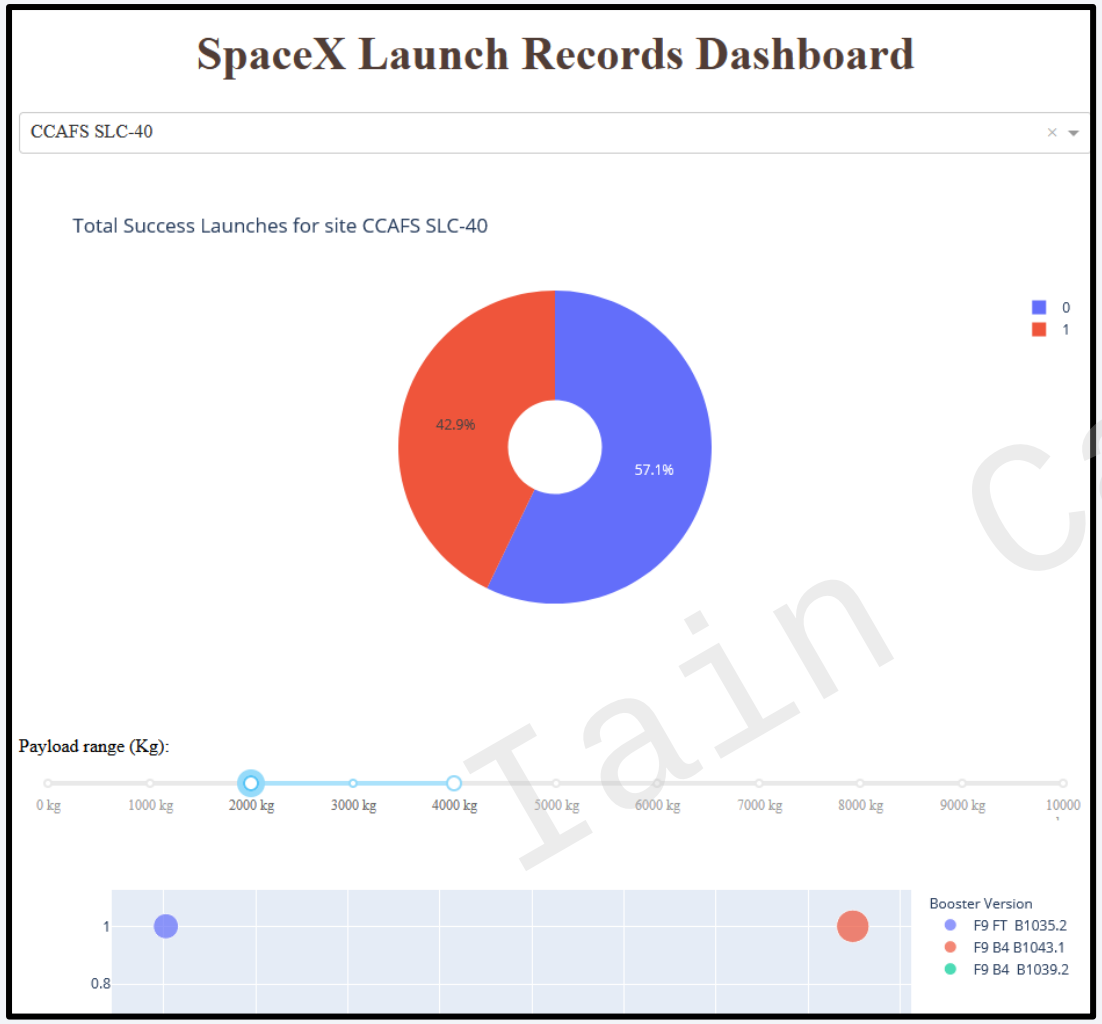


Light Weight Payloads 0kg-4000kg



Heavy weight Payloads 0kg-4000kg

Dashboard Insights



We are able to obtain a number of insights from the dashboard:

- Which site has the highest launch success rate?
  - KSC LC-39A
- Which Payload range has the highest launch success rate?
  - 2000kg – 10000kg
- Which payload range has the lowest success rate?
  - 0kg - 1000kg
- Which F9 booster version has the highest success rate?
  - FT

Predictive analysis (Classification)

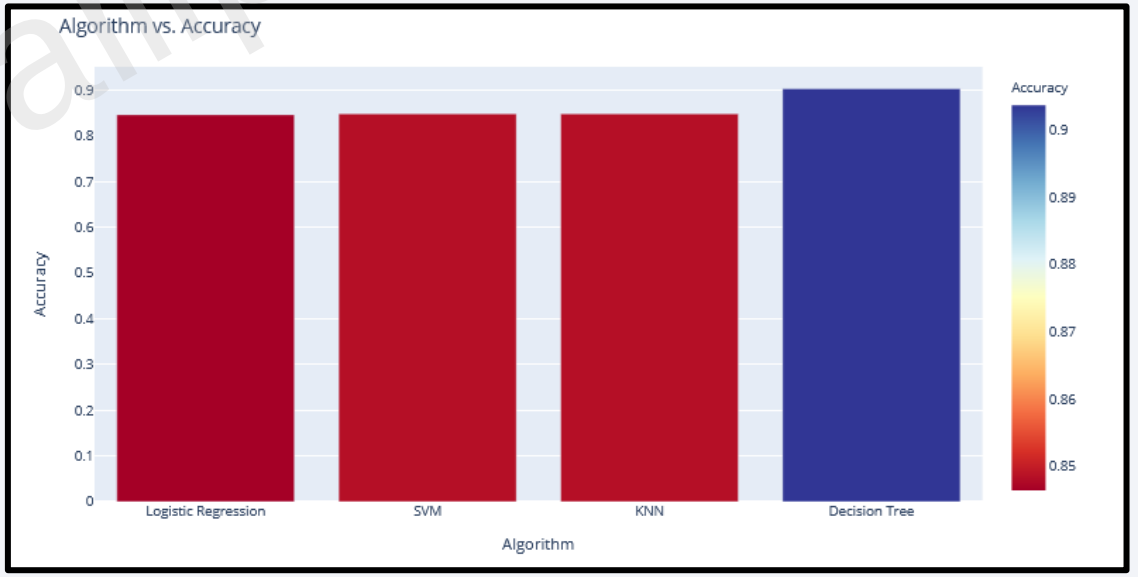
Section 6

Classification Accuracy

The accuracy between each of the models is exceptionally close. However, Decision tree is arrived at 0.903571 which is closest to 1.0 and is objectively the most accurate.

It is also worth noting that each algorithm performed equally well on test data by each achieving a score of 83.3333%

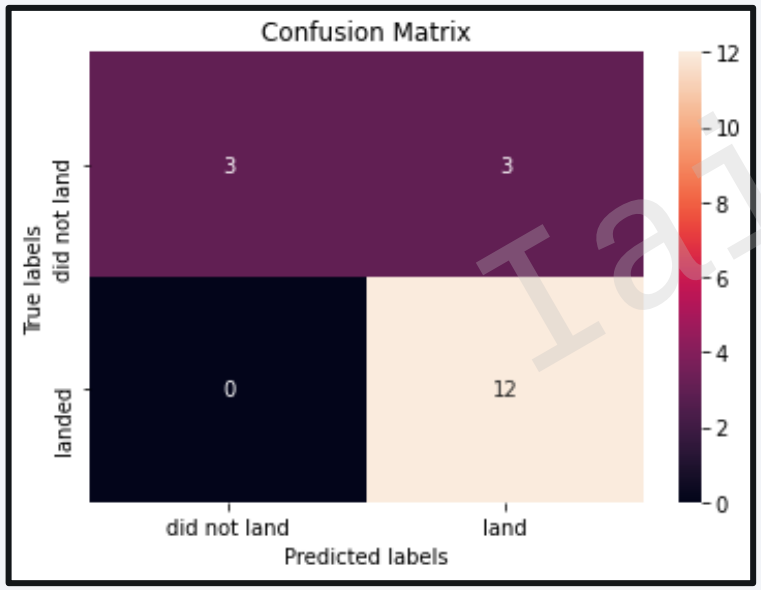
	Algorithm	Accuracy
0	Logistic Regression	0.846429
1	SVM	0.848214
2	KNN	0.848214
3	Decision Tree	0.903571



Confusion Matrix

Our decision tree can distinguish between different classes.

The most notable issues with our model is that it has some difficulty handling false positives and true negatives.



	Predicted No	Predicted Yes	
Actual No	True Negative TN=3	False Positive FP=3	6
Actual Yes	False Negative TN=0	True Positive TN=12	12
	3	15	Total Cases 18

Accuracy: 0.8333
Misclassification: 0.1657
Precision: 0.8000
Prevalence: 0.6667

Conclusions

1. The Tree Classifier Algorithm performs best with the SpaceX dataset
2. Lighter Payloads perform better than heavy payloads
3. A proportional relationship between successful launches and time can be seen with the current data set indicating steady improvement.
4. Site KSC LC-39A has achieved the most successful launches
5. Site KSC LC-39A has seen the most success with lighter payload weights.
6. Orbits ES-L1, GEO, HEO and SSO have the highest success rates.

Appendix

- A combination of matplotlib plotly and seaborn were used to accomplish the plots in the project.
- I chose pythonanywhere.com to host my live dashboard. The site is implemented usin Flask in /www/ within a docker container.
- Folium maps were used in conjunction with the ADGGoogleMaps Module – The MeasureControl plugin for folium maps were experimented with as part of the project but ultimately omitted for simplicity.
- The Haversine formula mentioned in this project is used to measure spherical distance.

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{(1-a)}\right)$$

$$d = R \cdot c$$

- I performed secondary validation of my data finding via IBM Cognos. Cognos is a tool built for the purpose of providing analysis and insights from a dataset. My intention was to derive meaningful differences in the findings.
- Royalty free images have been sourced via Pexels.com.


Thank you
End of presentation