IBM Developer
SKILLS NETWORK

Applied Data Science Capstone Project - SpaceX

Olga Ignatenko
21.09.2021

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

Summary of methodologies

- Data collection
- Data wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)

Summary of all results

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

Introduction

Project background and context

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. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Problems you want to find answers

- Identify how the different variables would affect the launch outcome
- Predict if the Falcon 9 first stage will land successfully

Section 1

Methodology

Methodology

- Data collection methodology:
  - SpaceX Rest API
  - Web scraping to collect Falcon 9 historical launch records from Wikipedia
- Perform data wrangling:
  - Dealing with Missing Values
  - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

SpaceX API

1. Request data from SpaceX API
2. Decode the response content and turn it into a Pandas dataframe
3. Clean the requested data
   Data Wrangling, filtering, ...
4. Export it to a CSV

Web Scrapping

1. Request the Falcon9 Launch Wiki page from its URL
2. Create a BeautifulSoup object fro m the HTML respons
3. Create a data frame by parsing the launch HTML tables
4. Export it to a CSV

Data Collection – SpaceX API

GitHub URL:
[jupyter-labs-spacex-data-collection-api.ipynb]()

1. Request data from SpaceX API

```
Ввод [7]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
Ввод [8]: response = requests.get(spacex_url)
```

2. Decode the response content as a Json **using .json()** and turn it into a Pandas dataframe using **.json_normalize()**

```
Ввод [48]: # Use json_normalize meethod to convert the
           response.json()
           data = pd.json_normalize(response.json())
```

3. Apply helper functions

```
[52]: # Call getBoosterVersion
      getBoosterVersion(data)
```

```
вод [54]: # Call getLaunchSite
          getLaunchSite(data)
```

```
вод [55]: # Call getPayloadData
          getPayloadData(data)
```

```
вод [56]: # Call getCoreData
          getCoreData(data)
```

4. Create a Pandas data frame

```
[58]: # Create a data from launch_dict
      df = pd.DataFrame.from_dict(launch_dict)
```

```
# Show the head of the dataframe
df.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None

5. Filter the dataframe

```
[60]: # Hint data['BoosterVersion']!='Falcon 1'
      data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
      data_falcon9.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	Launch
4	6	2010-06-04	Falcon 9	NaN	LEO	CC SL

6. Data Wrangling (Dealing with Missing Values)

Data Collection - Scraping

GitHub URL: [jupyter-labs-webscraping.ipynb]

1. Perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response

```
[7]: # use requests.get() method with the provi
# assign the response to a object
response = requests.get(static_url)
```

2. Create a BeautifulSoup object from the HTML response

```
[8]: # Use BeautifulSoup() to create a BeautifulSoup object from a
BeautifulSoup =  BeautifulSoup(response.text,"html.parser")
```

3. Find tables on the wiki page

```
0]: # Use the find_all function in the BeautifulSoup 
# Assign the result to a list called `html_tables
html_tables = BeautifulSoup.find_all('table')
```

4. Extract column name one by one

```
column_names = []

# Apply find_all() function with `th` element on first_l
th_data = first_launch_table.find_all('th')
# Iterate each th element and apply the provided extract_
# Append the Non-empty column name (`if name is not None

for row in th_data:
    column_name = extract_column_from_header(row)
    if column_name != None and len(column_name) > 0:
        column_names.append(column_name)
```

5. Create an empty dictionary with keys from the extracted column names fill up the launch_dict with launch records extracted from table rows

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each v
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']=[]
```

```
: extracted_row = 0
#Extract each table
for table_number,table in enumerate(Bea
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table he
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.s
                flag=flight_number.isd
```

6. Create a dataframe from dictionary

```
df = pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df.head()
```

40]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n

Data Wrangling

GitHub URL:
[/jupyter-labs-spacex-Data_wrangling.ipynb]

In the data set, there are several different cases :

- True Ocean - the mission outcome was successfully landed to a specific region of the ocean
- False Ocean - the mission outcome was unsuccessfully landed to a specific region of the ocean
- True RTLS - the mission outcome was successfully landed to a ground pad
- False RTLS - the mission outcome was unsucce ssfully landed to a ground pad
- True ASDS - the mission outcome was successfully landed on a drone ship
- False ASDS means the mission outcome was unsuccessfully landed on a drone ship

We will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Data wrangling process

Identify the missing values in each attribute

Calculate the number of launches on each site

Calculate the number and occurrence of each orbit

Calculate the number and occurrence of mission outcome per orbit type

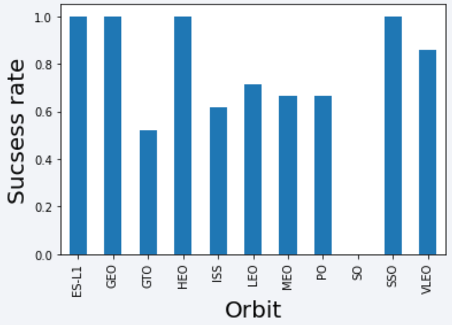
Create a landing outcome label from Outcome column

EDA with Data Visualization

GitHub URL:
[jupyter-labs-eda-dataviz.ipynb]

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

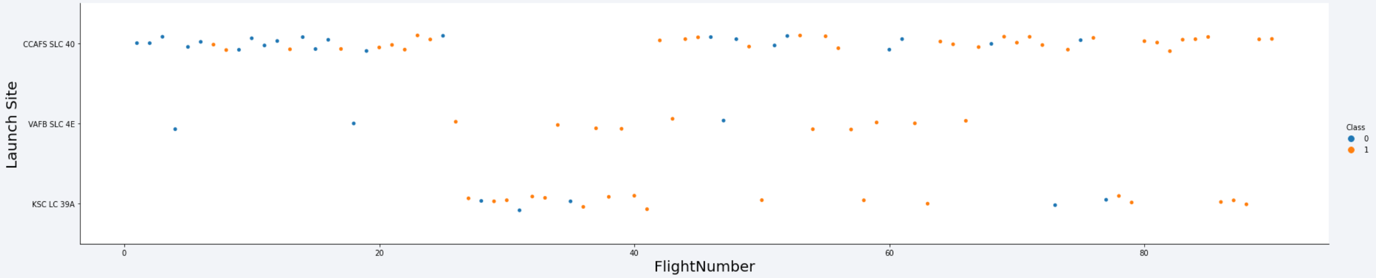


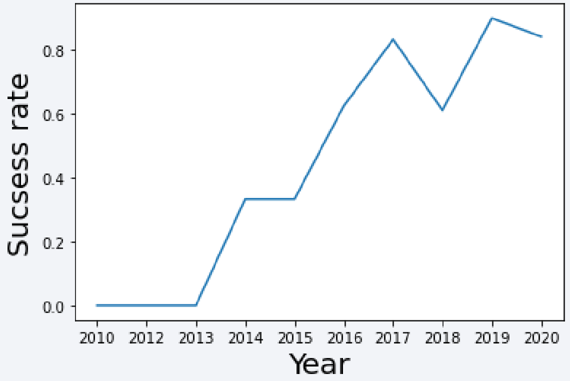
Bar Graph:

- Mean VS. Orbit

Line Graph:

Success Rate VS. Year





Scatter plots show how much one variable is affected by another.
Line graphs show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded
A bar diagram makes it easy to compare sets of data between different groups at a glance.

EDA with SQL

GitHub URL:
[jupyter-labs-eda-sql.ipynb]

```
%sql SELECT DISTINCT launch_site from SPACEXDATASET
```

```
 * ibm_db_sa://qbp91844:***@fbd88901-ebdb-4a4f-a32e-9822b
Done.
```

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

SQL queries were executed to answer next questions:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was acheived.
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use 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

Build an Interactive Map with Folium

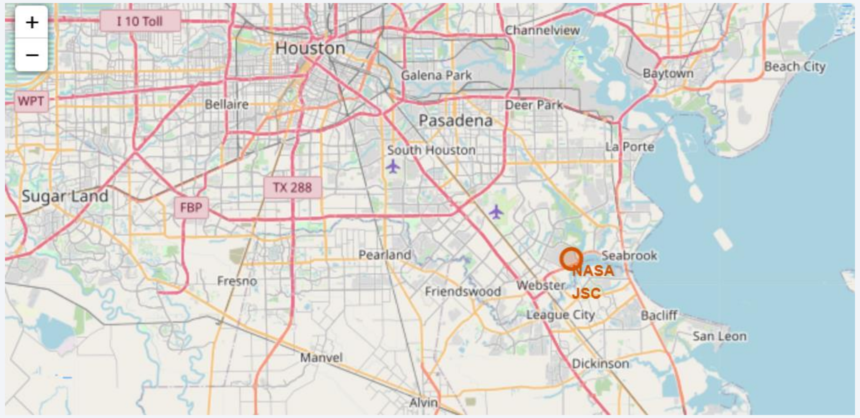
GitHub URL: jupyter_lab_launch_site_location.ipynb

We use Folium for performing more interactive visual analytics.

Map objects created and added to a folium map:

- `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate
- `folium.Circle` and `folium.Marker` for each launch site on the site map
- `folium.MarkerCluster` object (to simplify a map containing many markers having the same coordinate)
- assigned the dataframe launch_outcomes(failures, successes) to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster()
- `folium.MousePosition` to get coordinate for a mouse over a point on the map
- calculated the distance from the Launch Site to various landmarks
- `folium.PolyLine` to show distance from the Launch Site to various landmarks

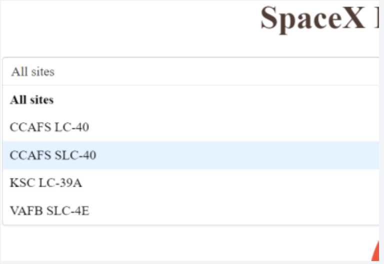
```python
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its na
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```



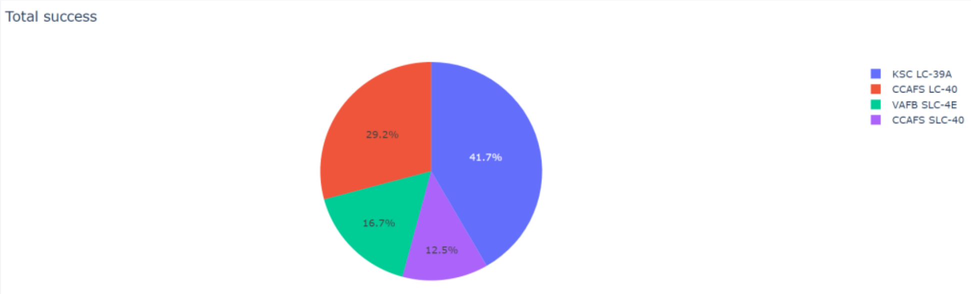
Build a Dashboard with Plotly Dash

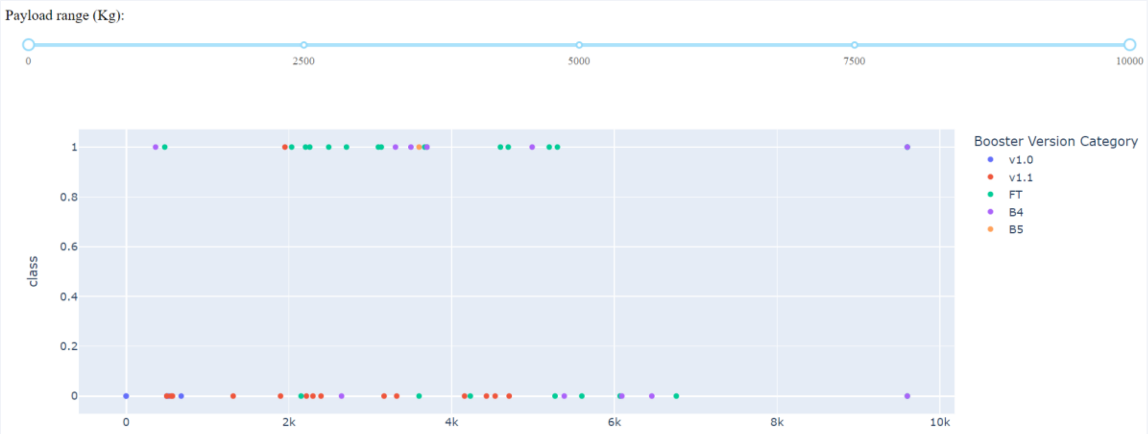
GitHub URL:
[spacex_dash_app.py]()

Plots/graphs and interactions added to a dashboard:



- Site Drop-down Input Component to let us select different launch sites
- A callback function to render success-pie-chart based on selected site dropdown
- Pie Chart:
  - show the total launches by a certain site/all sites
  - display relative proportions of multiple classes of data.





- Range Slider to Select Payload
- A callback function to render the success-payload-scatter-chart scatter plot
- Scatter Graph show the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions

Predictive Analysis (Classification)

GitHub URL: [SpaceX_Machine_Learning_Prediction.py]

1. Import Libraries and Define Auxiliary Functions
2. Load the dataframe
3. Create a NumPy array from the column Class in data, then assign it to the variable Y
4. Standardize the data in X then reassign it to the variable X
5. Split the data into training and testing data
6. Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
7. Calculate the accuracy on the test
8. Plot the confusion matrix
9. Find the method performs best

```
[7]: Y = data['Class'].to_numpy()
     Y

[7]: array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
            1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
            1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1])
```

```
X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
```

```
X.head(100)
```

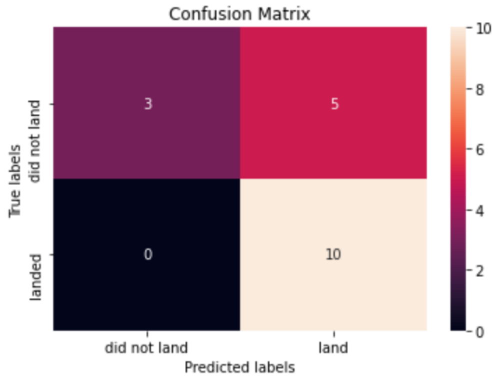
[4]:

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Orbit_GTO
0	1.0	6104.959412	1.0	1.0	0.0	0.0	0.0	0.0
1	2.0	525.000000	1.0	1.0	0.0	0.0	0.0	0.0
2	3.0	677.000000	1.0	1.0	0.0	0.0	0.0	0.0

```
]: logreg_cv = GridSearchCV(lr,parameters, cv = 10)
   logreg_cv.fit(X_train, Y_train)

]: GridSearchCV(cv=10, estimator=LogisticRegression(),
                param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                            'solver': ['lbfgs']})
```

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



Results

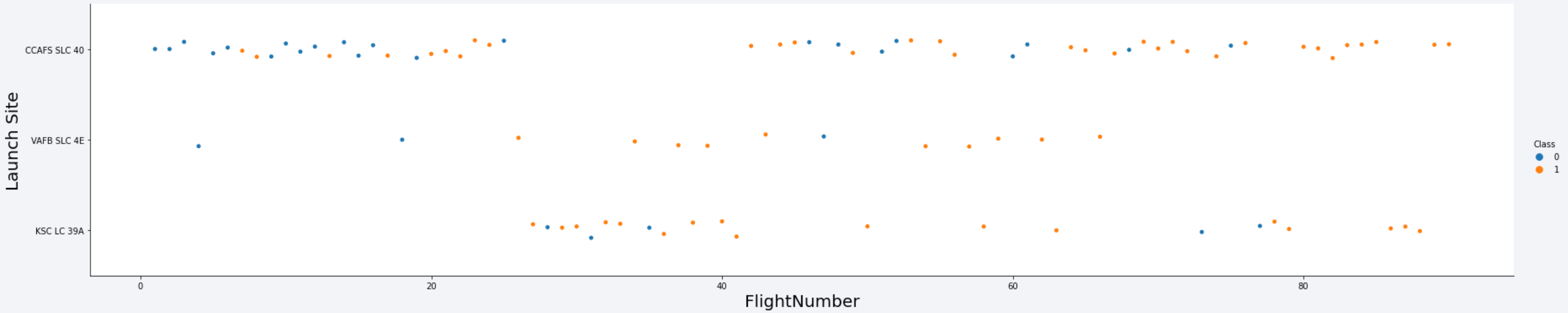
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

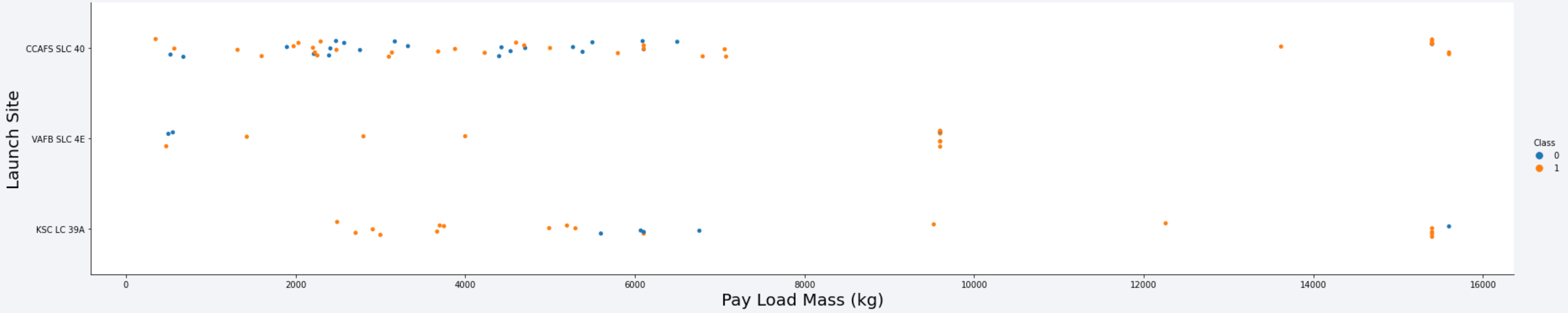
Scatter plot of Flight Number vs. Launch Site



We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.

Payload vs. Launch Site

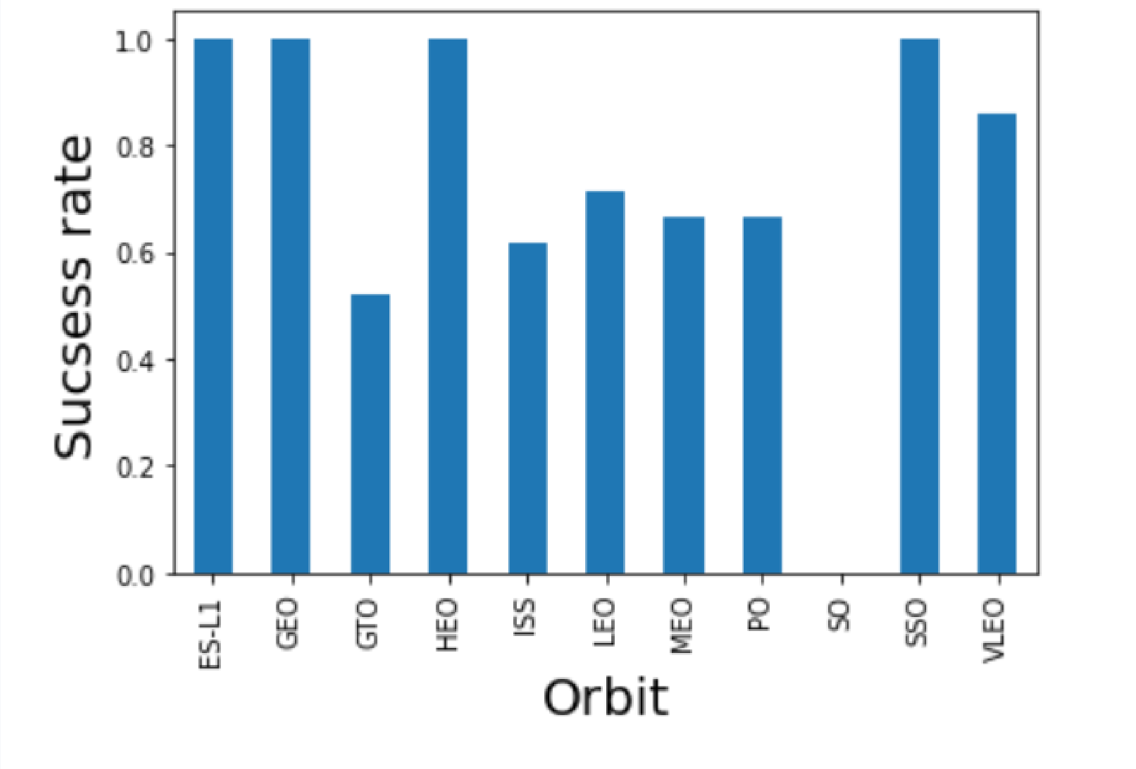
Scatter plot of Payload vs. Launch Site



For Launch Site CCAFS SLC 40 it seems the more massive the payload, the more likely the first stage will return. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch.

Success Rate vs. Orbit Type

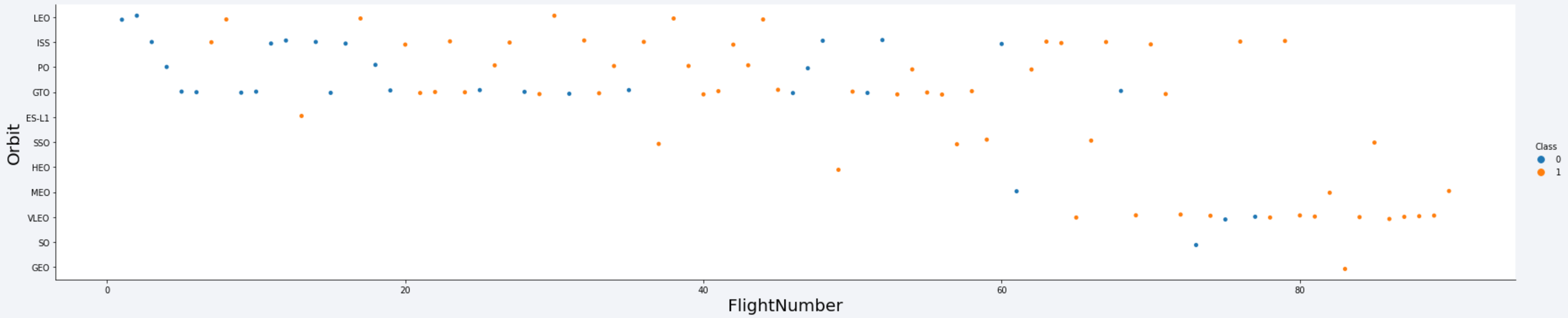
Chart for the success rate of each orbit type



Orbit GEO,HEO,SSO,ES-L1 have the highest success rate = 1.

Flight Number vs. Orbit Type

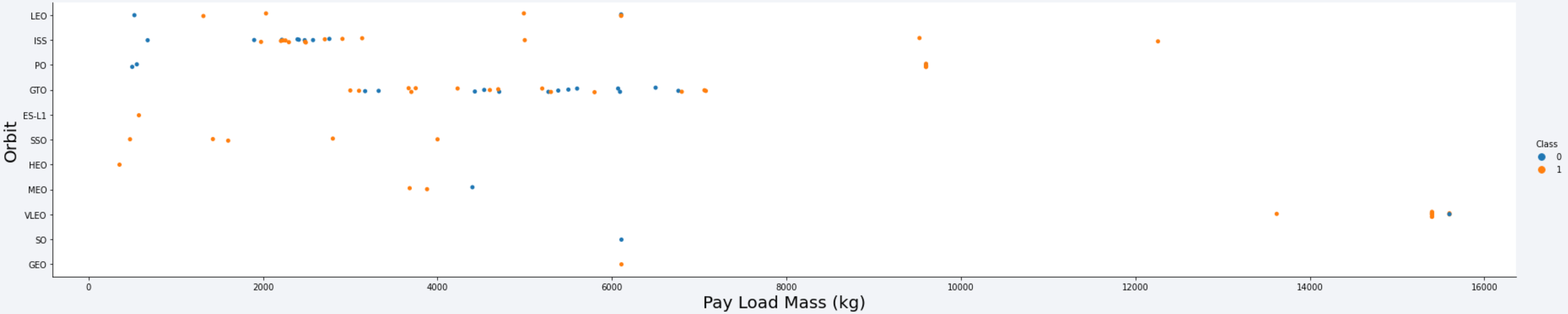
Scatter point of Flight number vs. Orbit type



The LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

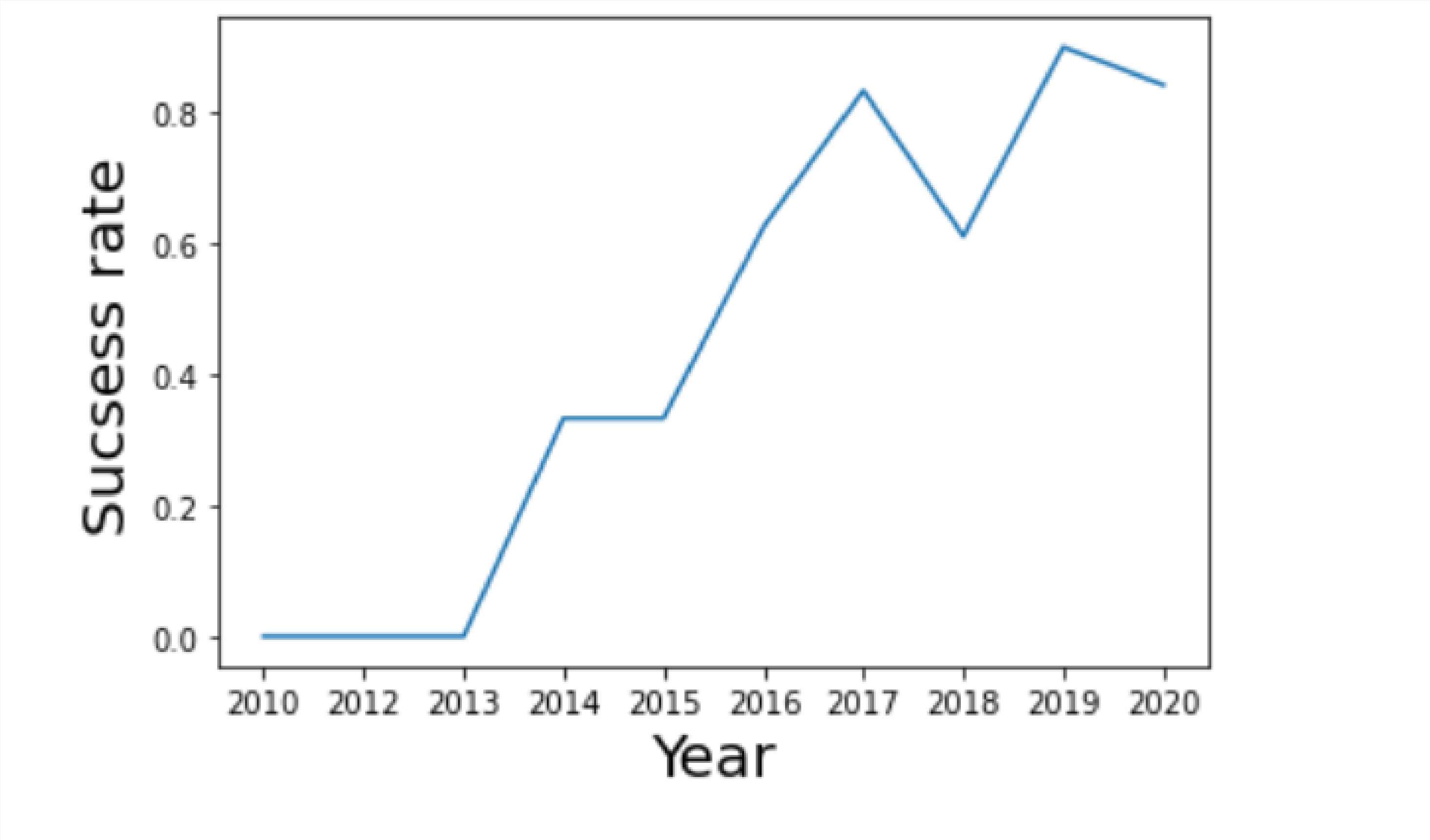
Payload vs. Orbit Type

Scatter point of 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 sucess rate since 2013 kept increasing till 2020

All Launch Site Names

Find the names of the unique launch sites

The SELECT DISTINCT statement is used to return only distinct (unique) values.

The following SQL statement selects only the DISTINCT values from the "launch sites" column in the "SPACEXDATASET" table:

query

```
%sql SELECT DISTINCT launch_site from SPACEXDATASET
```

query result

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

Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with `CCA`

The following SQL statement select records where launch sites begin with `CCA` in the "launch sites" column in the "SPACEXDATASET" table and show only 5 first record:

query

```
%sql SELECT * from SPACEXDATASET WHERE launch_site LIKE 'CCA%' LIMIT 5
```

query result

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

Total Payload Mass

Calculate the total payload carried by boosters from NASA

The SUM() function returns the total sum in the column PAYLOAD_MASS_KG_.
WHERE clause filters the dataset to only perform calculations on Customer NASA (CRS)

query

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) from SPACEXDATASET where customer = 'NASA (CRS)'
```

query result

1
45596

Average Payload Mass by F9 v1.1

Calculate the average payload mass carried by booster version F9 v1.1

The AVG() function returns the average value in the column PAYLOAD_MASS_KG_.
WHERE clause filters the dataset to only perform calculations on booster version F9 v1.1

query

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) from SPACEXDATASET where booster_version = 'F9 v1.1'
```

query result

1
2928

First Successful Ground Landing Date

Find the dates of the first successful landing outcome on ground pad

The `MIN()` function returns the smallest (first) date in the column Date
The WHERE clause filters the dataset to only perform calculations on successful landing outcome on ground pad

query

```
%sql SELECT min(DATE) from SPACEXDATASET where landing__outcome = 'Success (ground pad)'
```

query result

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Select only booster_version column
The WHERE clause filters the dataset to Landing_Outcome = Success (drone ship)
The AND clause specifies additional filter conditions Payload_MASS_KG 4000 AND Payload_MASS_KG 6000

query

```
%sql select booster_version from SPACEXDATASET where landing__outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000
```

query result

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

Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcomes

Combine two query into one.
Select records in the "mission outcomes " column in the "SPACEXDATASET" table where mission outcom contain `Success` for successful mission outcomes and 'Failure' for failure mission outcomes. The `LIKE` operator is used in a `WHERE` clause to search for a specified pattern in a column.
The `COUNT()` function returns the number of rows that matches a specified criterion.

query

```
%sql select * from(select count(mission_outcome) as Success_mission_outcomes\
                from SPACEXDATASET where mission_outcome like '%Success%'),\
                (select count(mission_outcome) as Failure_mission_outcomes\
                from SPACEXDATASET where mission_outcome like '%Failure%')
```

query result

success_mission_outcomes	failure_mission_outcomes
100	1

Boosters Carried Maximum Payload

List the names of the booster which have carried the maximum payload mass

Use a subquery for finding maximum payload mass. The MAX( ) function returns the largest value of the selected column.

Use the SELECT DISTINCT statement to return only unique names of the booster which have carried the maximum payload mass

query

```
%sql select distinct booster_version from SPACEXDATASET \
            where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) from SPACEXDATASET)
```

query result

booster_version
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

2015 Launch Records

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

Select landing_outcomes, booster_versions, and launch_site columns from table SPACEXDATASET.

Use AND operator for two condition.

query

```
%sql select landing__outcome, launch_site, booster_version from SPACEXDATASET \
    where landing__outcome = 'Failure (drone ship)' and DATE like '%2015%'
```

query result

landing__outcome	launch_site	booster_version
Failure (drone ship)	CCAFS LC-40	F9 v1.1 B1012
Failure (drone ship)	CCAFS LC-40	F9 v1.1 B1015

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

query

```
%sql select landing__outcome,count (*) as total from SPACEXDATASET \
     where date between date('2010-06-04') and date('2017-03-20') \
     GROUP BY landing__outcome ORDER BY count(*) DESC
```

query result

landing__outcome	total
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

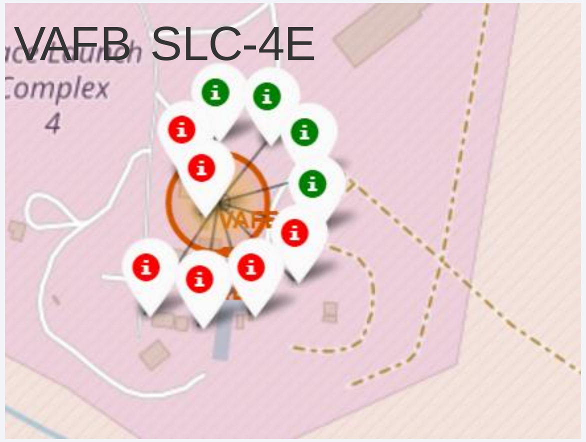
Section 4

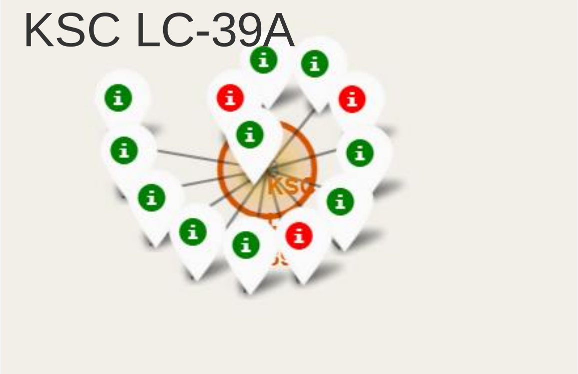
Launch Sites Proximities Analysis

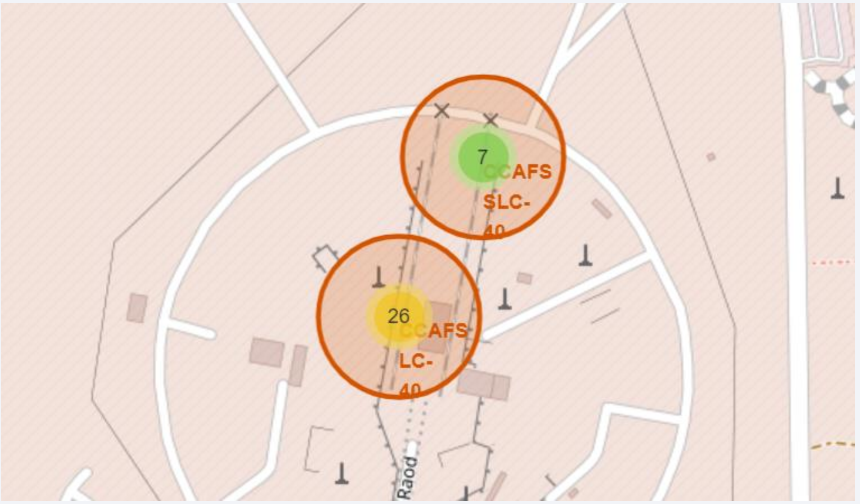
Layout of Launch sites

Launch sites located on west and east coast.
West coast launch sites are in proximity to the Equator line.
There are more launch sites on the western coast.

Colour Labelled Markers

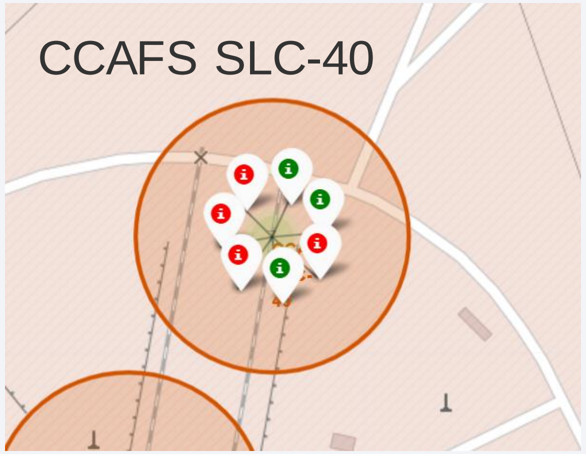


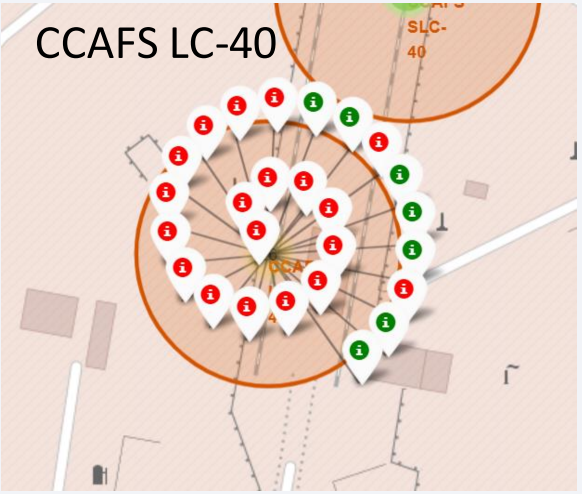




Green Marker shows successful Launches and Red Marker shows Failures

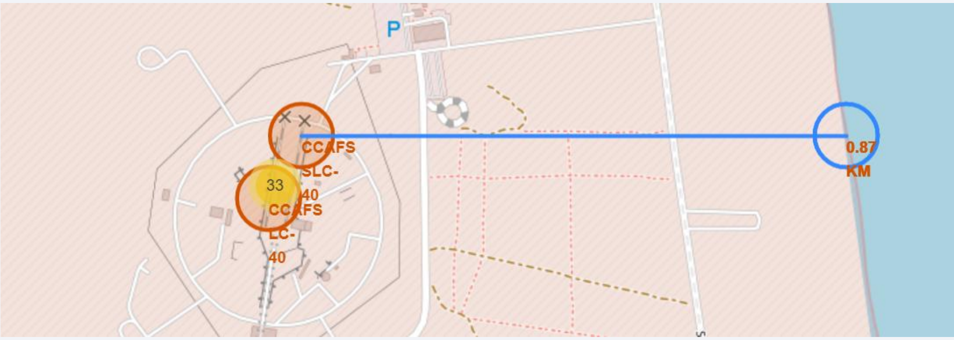
KSC LC-39A launch site has relatively high success rates.



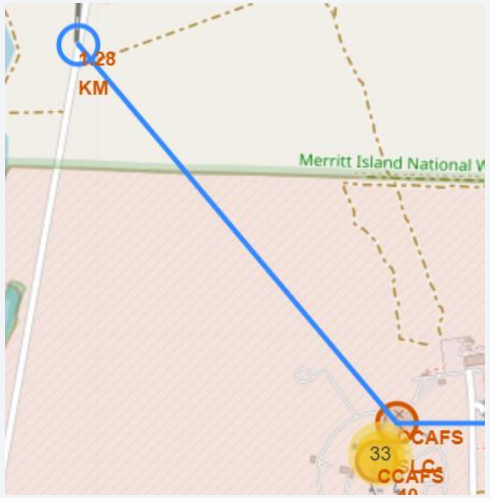


Distances between a launch site CCAFS SLC-40 to its proximities

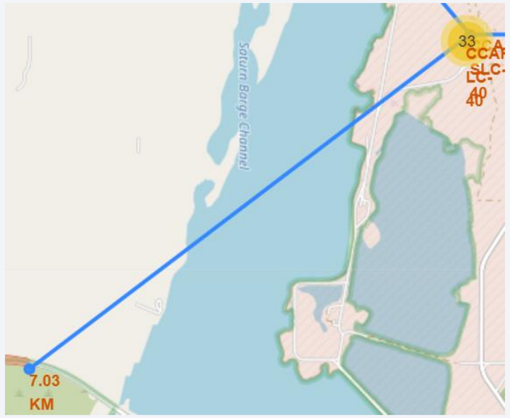
coastline



railway

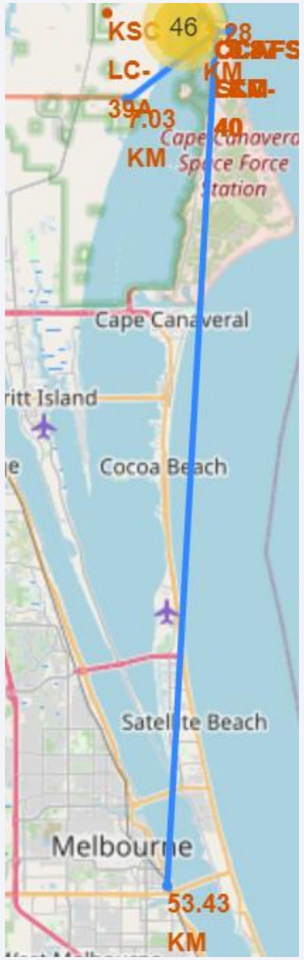


highway



- Launch site CCAFS SLC-40 is in close proximity to railway. Distance between Launch site and the nearest railway is 1.2 km
- Launch site CCAFS SLC-40 is non in close proximity to highways. Distance between Launch site and the nearest highway is more then 7 km
- Launch site CCAFS SLC-40 is in close proximity to coastline. Distance between Launch site and the nearest coastline is 0,87 km
- Launch site CCAFS SLC-40 keep certain distance away from cities. The nearest city, Melbourne, is 53 km away.

city

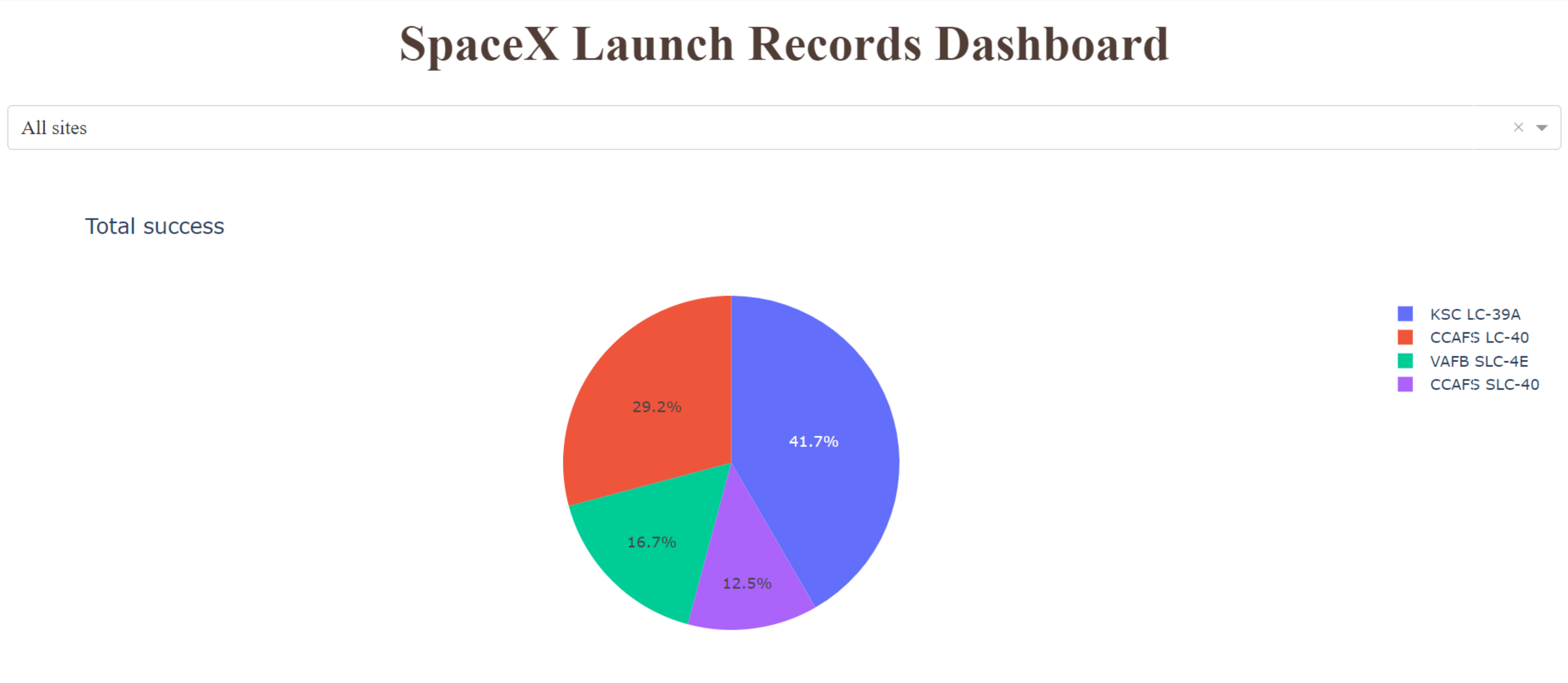


Section 5

Build a Dashboard with Plotly Dash

Pie chart 1 - successful launches for all sites

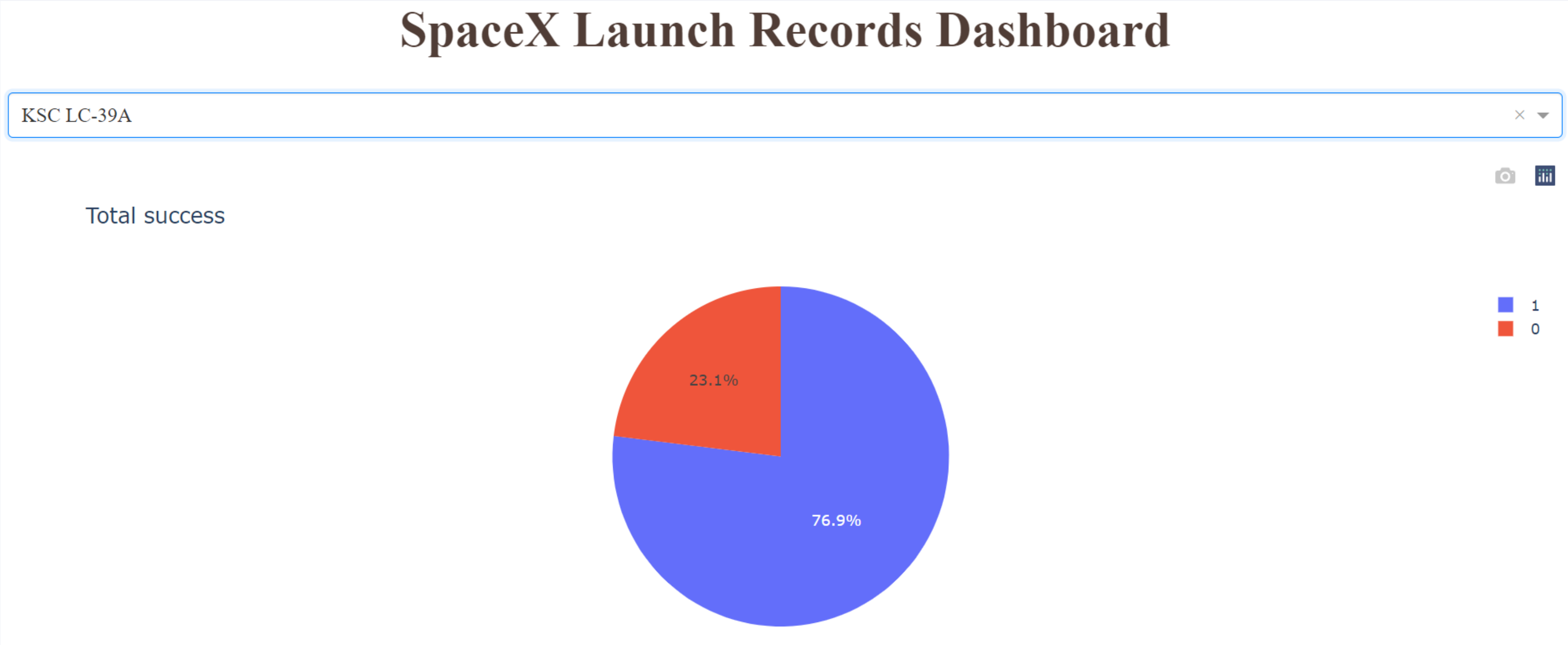
Which site has the largest successful launches?



We can see that KSC LC-39A had the most successful launches from all the sites

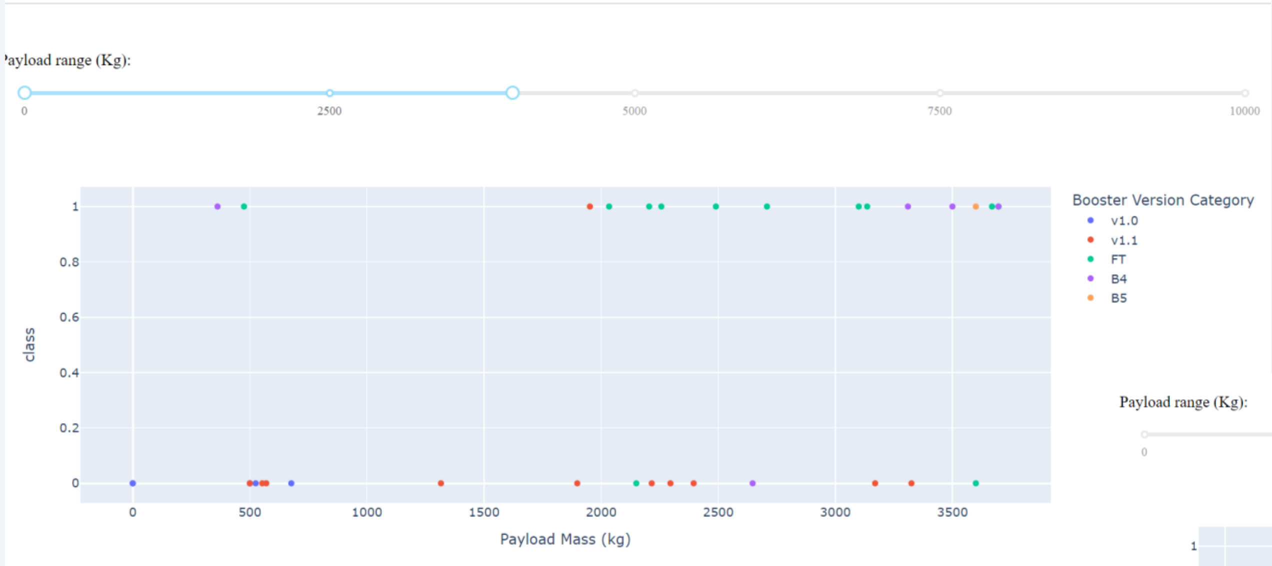
Pie chart 2 - launch site with highest launch success ratio

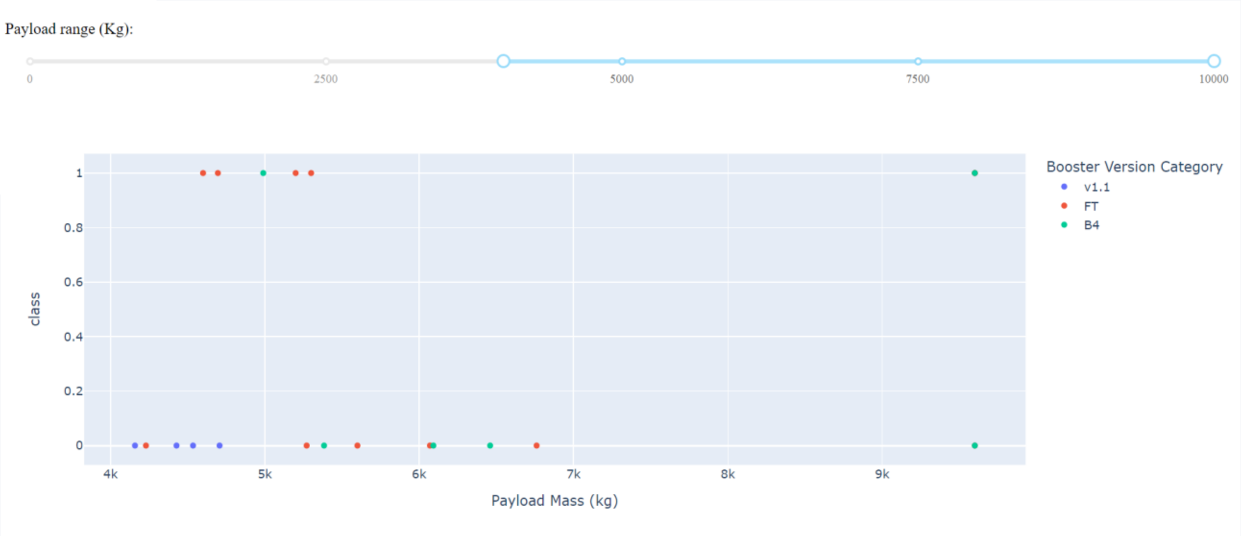
Which site has the highest launch success rate?



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs. Launch Outcome scatter plot for all sites





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

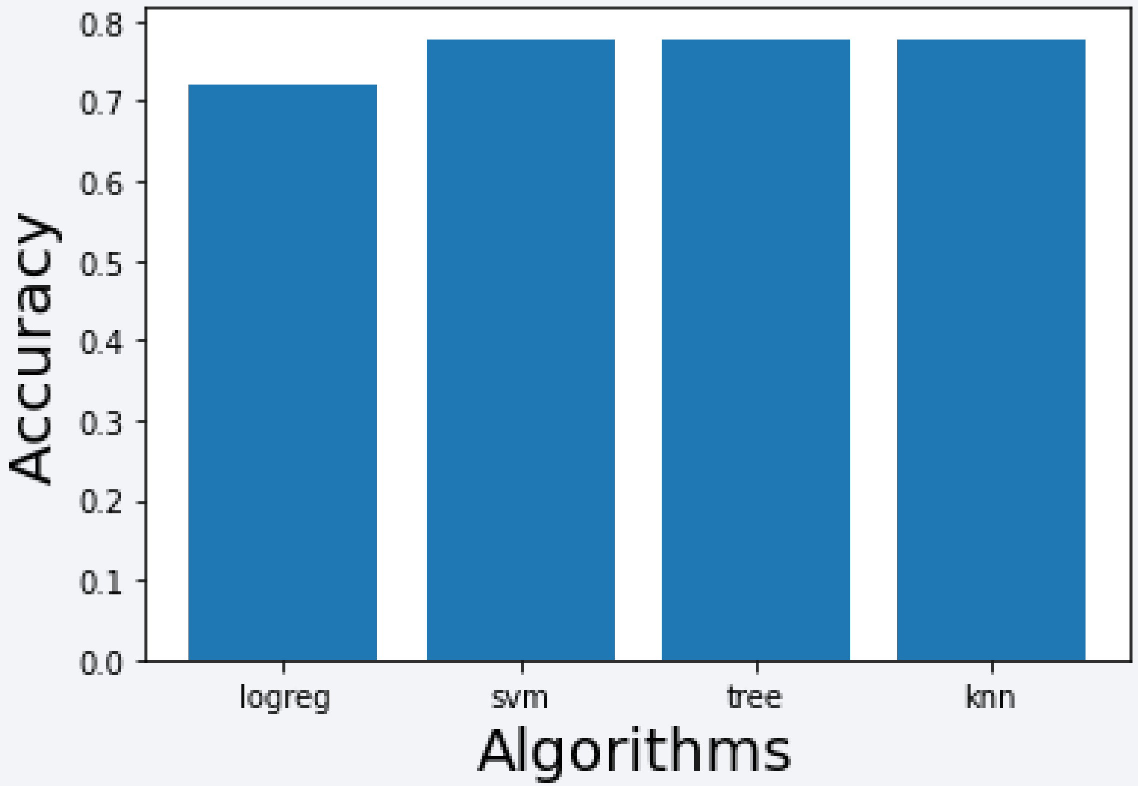
Predictive Analysis (Classification)

Classification Accuracy

Practically all these algorithms give the same result.

SVM, Classification Trees and KNN have accuracy **0.77**

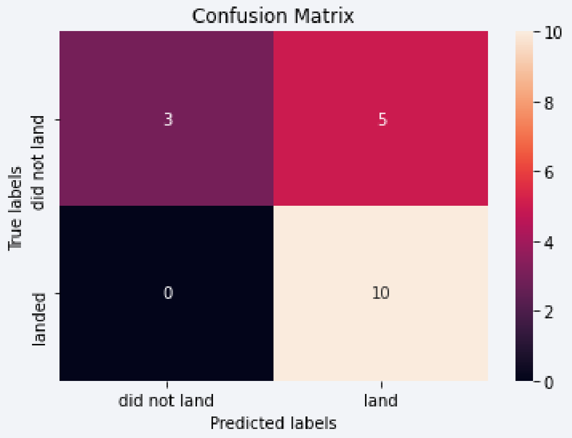
Logistic Regression has accuracy **0.72**



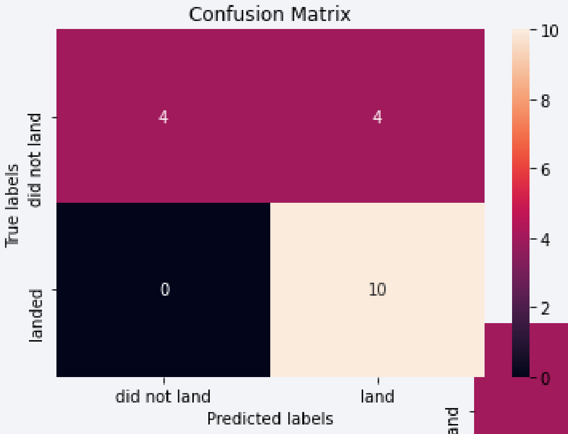
Confusion Matrix

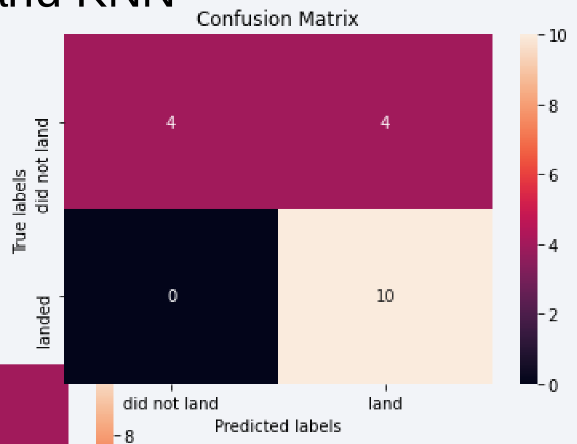
Examining the confusion matrix, we can see that all algorithms can distinguish between the different classes. We see that the major problem is false positives.

Logistic Regression



SVM, Classification Trees and KNN





Conclusions

- The sucess rate since 2013 kept increasing till 2020
- KSC LC-39A had the most successful launches from all the sites. KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate
- Low weighted payloads perform better than the heavier payloads
- Orbit GEO,HEO,SSO,ES L1 has the best Success Rate
- Practically all algorithms give the same result. Examining the confusion matrix, we can see that all algorithms can distinguish between the different classes. We see that the major problem is false positives.

Thank you!