Winning Space Race with Data Science

Sukun Cheng
Sept. 7, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis
  - Interactive Visual Analysis
  - Predictive Analysis
- Summary of all results including
  - Exploratory Data Analysis
  - Geospatial maps
  - Interactive dashboard
  - Predictions of classification models

Introduction

- SpaceX Falcon 9 rocket launch costs as low as $62 million dollars due to reuse the first stage of Rocket. In contrast, other providers cost upward of $165 million dollars.

- The launch cost depends on whether the first stage will land successful. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- This project will ultimately predict if the Space X Falcon 9 first stage will land successfully.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology
- Perform data wrangling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Methodology

Executive Summary

- Data collection methodology
  - Request the SpaceX launch data using the GET request
  - Web scraping using beautiful soup
- Perform data wrangling
  - Deal with missing data
  - Count numbers of launches (launch site, orbit, outcome, occurrences, etc)
  - Convert data types and one hot encoding
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Visualize variables by python pandas and matplotlib
  - Manipulate dataset by SQL with specific queries on db2
- Perform interactive visual analytics using Folium and Plotly Dash
  - Geospatial analysis by Folium
  - Interactive dashboard by Ploty Dash
- Perform predictive analysis using classification models
  - Using scikit-learn library:
  - Pre-process the data (Standardize)
  - Split the train and test data using train_test_split( )
  - Train four classification models (Logistic regression, decision tree, support vector machine, k nearest neighbor)
  - Get hyperparameter using GridSearchCV( )
  - Present confusion metrics
  - Select the best score of each classification models

Data Collection – SpaceX API

- Using the SpaceX API to retrieve data about launches, including the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

1
- request rocket launch data from SpaceX API as response
- Convert the response to a .json file and save as a Pandas DataFrame

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

3
- Create a Pandas DataFrame from the constructed dictionary dataset

```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
```

4
- Filter the DataFrame to only include Falcon 9 launches
- Reset the FlightNumber column
- Replace missing values of PayloadMass with the mean PayloadMass value

```
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
# Calculate the mean value of PayloadMass column
mean_PayloadMass = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(mean_PayloadMass)
```

2
- Use custom logic to clean the data (see Appendix)
- Define lists for data to be stored in
- Call custom functions (see Appendix) to retrieve data and fill the lists
- Use these lists as values in a dictionary and construct the dataset

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
# Call getBoosterVersion
getBoosterVersion(data)

# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(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}
```

Data Collection - Scraping

- Web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches

1
- Request the HTML page from the static URL
- Assign the response to an object

```
static_url = "https://en.wikipedia.org/w/index.php?
title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
data = response.content
```

2
- Create a BeautifulSoup object from the HTML response object
- Find all tables within the HTML page

```
soup = BeautifulSoup(data,'html.parser')
html_tables = soup.find_all('table')
```

3
- Collect all column header names from the tables found within the HTML page

```
column_names = []
# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to
get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into
a list called column_names
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if(name != None and len(name) > 0):
        column_names.append(name)
```

4
- Use the column names as keys in a dictionary
- Use custom functions and logic to parse all launch tables (see Appendix) to fill the dictionary values

```
launch_dict= dict.fromkeys(column_names)
# Remove an irrelvant column
del launch_dict['Date and time ( )']
# Let's initial the launch_dict with each value to be an empty list
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']=[]
```

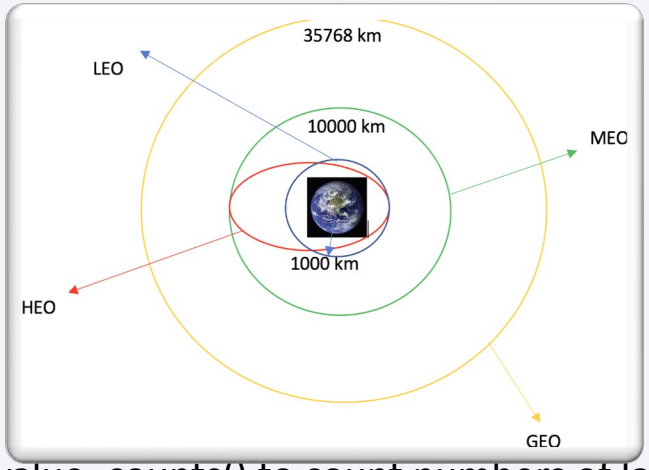
5
- Convert the dictionary to a Pandas DataFrame ready for export

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

Data Wrangling

Context:

- The dataset contains several Space X launch facilities. Each location is in the *LaunchSite* column.
- The Orbit type associated with each launch is in the *Orbit* column. Some common orbit types are illustrated below.



- Using the .value_counts() to count numbers of launches

  (launch site, orbit, outcome, occurrences, etc)

1

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

2

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

3

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

Data Wrangling - continued

Context:

The mission outcome in the *Outcome* column

- `True Ocean` – the mission outcome was successfully landed to a specific region of the ocean
- `False Ocean` – unsuccessfully landed to a specific region of the ocean.
- `True RTLS` – successfully landed to a ground pad
- `False RTLS` – unsuccessfully landed to a ground pad.
- `True ASDS` – successfully landed to a drone ship
- `False ASDS` – unsuccessfully landed to a drone ship.
- `None ASDS` and `None None` – these represent a failure to land.

Data Wrangling:

Using binary to indicate if a booster is successfully landed. 1 - success or 0 - unsuccess.

- Define unsuccessful (bad) outcomes as a list, `bad_outcome`
- Create a list, `landing_class` whose element is 0 if the corresponding row in `Outcome` is in the set `bad_outcome`. Otherwise, it's 1.
- Add a *Class* column to the DataFrame that contains the values from the list `landing_class`
- Export the resulting DataFrame as a .csv file.

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = []

for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

```
df['Class']=landing_class
```

```
df.to_csv("dataset_part\_2.csv", index=False)
```

EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts
- Scatter charts - visualize the relationships or correlation between two variables
  - Flight Number and Launch Site
  - Payload and Launch Site
  - Orbit Type and Flight Number
  - Payload and Orbit Type
- bar charts - compare a numerical value to a categorical variable. Horizontal or vertical bar charts can be used, depending on the size of the data.
  - Success Rate and Orbit Type
- Line charts - the change of a variable over time
  - Success Rate and Year (i.e. the launch success yearly trend)

EDA with SQL

The SQL queries performed on the data set were used to:

- 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 the average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome on a ground pad was achieved
- List the names of the boosters which had success on a drone ship and a payload mass between 4000 ~ 6000 kg
- List the total number of successful and failed mission outcomes
- List the names of the booster versions which have carried the maximum payload mass
- List the failed landing outcomes on drone ships, their booster versions, and launch site names for 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

The following steps were taken to visualize the launch data on an interactive map:

1. Mark all launch sites on a map
   - Initialise the map using a Folium Map object
   - Add a `folium.Circle` and `folium.Marker` for each launch site on the launch map

2. Mark the success/failed launches for each site on a map
   - As many launches have the same coordinates, it makes sense to cluster them together.
   - Before clustering them, assign a marker colour of successful (class = 1) as green, and failed (class = 0) as red.
   - To put the launches into clusters, for each launch, add a `folium.Marker` to the `MarkerCluster()` object.
   - Create an icon as a text label, assigning the `icon_color` as the `marker_colour` determined previously.

3. Calculate the distances between a launch site to its proximities
   - To explore the proximities of launch sites, calculations of distances between points can be made using the `Lat` and `Long` values.
   - After marking a point using the `Lat` and `Long` values, create a `folium.Marker` object to show the distance.
   - To display the distance line between two points, draw a `folium.PolyLine` and add this to the map.

Build a Dashboard with Plotly Dash

The Plotly Dash dashboard is used to interactively visualize the data by charts below:

1. Pie chart (`px.pie()`) showing the total successful launches per site
   - This makes it clear to see which sites are most successful
   - The chart could also be filtered (using a `dcc.Dropdown()` object) to see the success/failure ratio for an individual site

2. Scatter graph (`px.scatter()`) to show the correlation between outcome (success or not) and payload mass (kg)
   - This could be filtered (using a `RangeSlider()` object) by ranges of payload masses
   - It could also be filtered by booster version

Predictive Analysis (Classification)

Built, evaluated, improved, and found the best performing classification model

1. Model Development

To prepare the dataset for model development:

a. Load dataset
b. Perform necessary data transformations (standardize and pre-process)
c. Split data into training and test data sets, using `train_test_split()`
d. Decide which type of machine learning algorithms are most appropriate

For each chosen algorithm:

a. Create a `GridSearchCV` object and a dictionary of parameters
b. Fit the object to the parameters
c. Use the training data set to train the model

2. Model Evaluation

For each chosen algorithm:

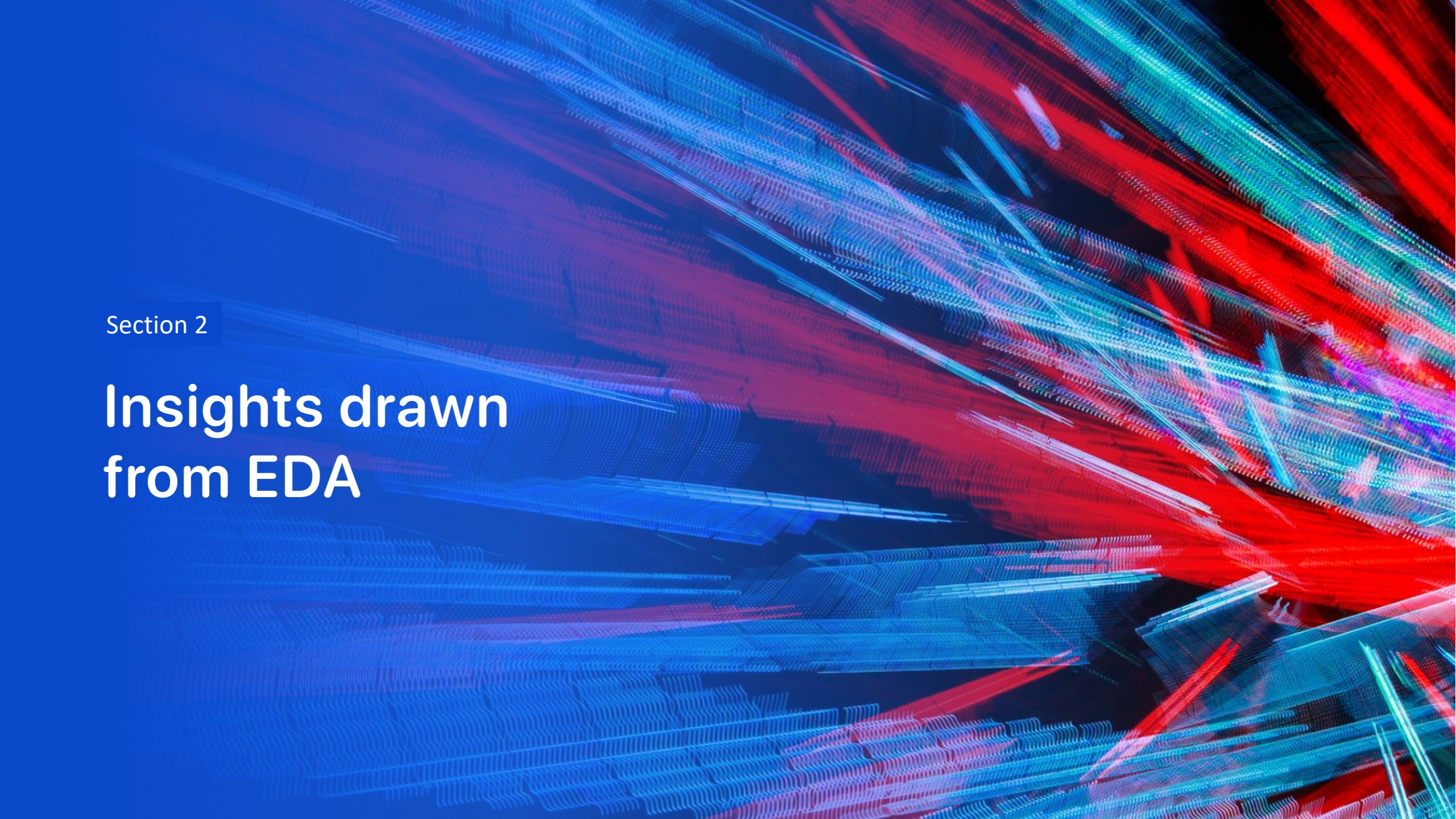
a. Using the output GridSearchCV object:
   Check the tuned hyperparameters (`best_params_`)
   Check the accuracy (`score` and `best_score_`)
b. Plot and examine the Confusion Matrix

3. Finding the Best Classification Model

a. Review the accuracy scores for all chosen algorithms
b. The model with the highest accuracy score is determined as the best performing model

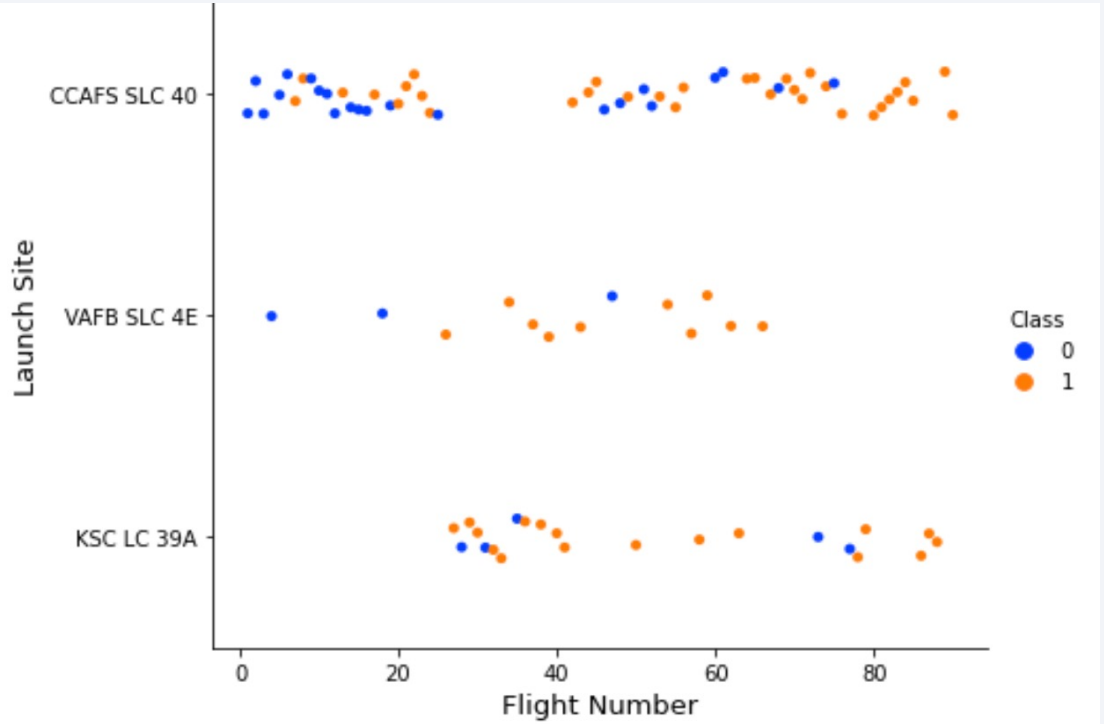
Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

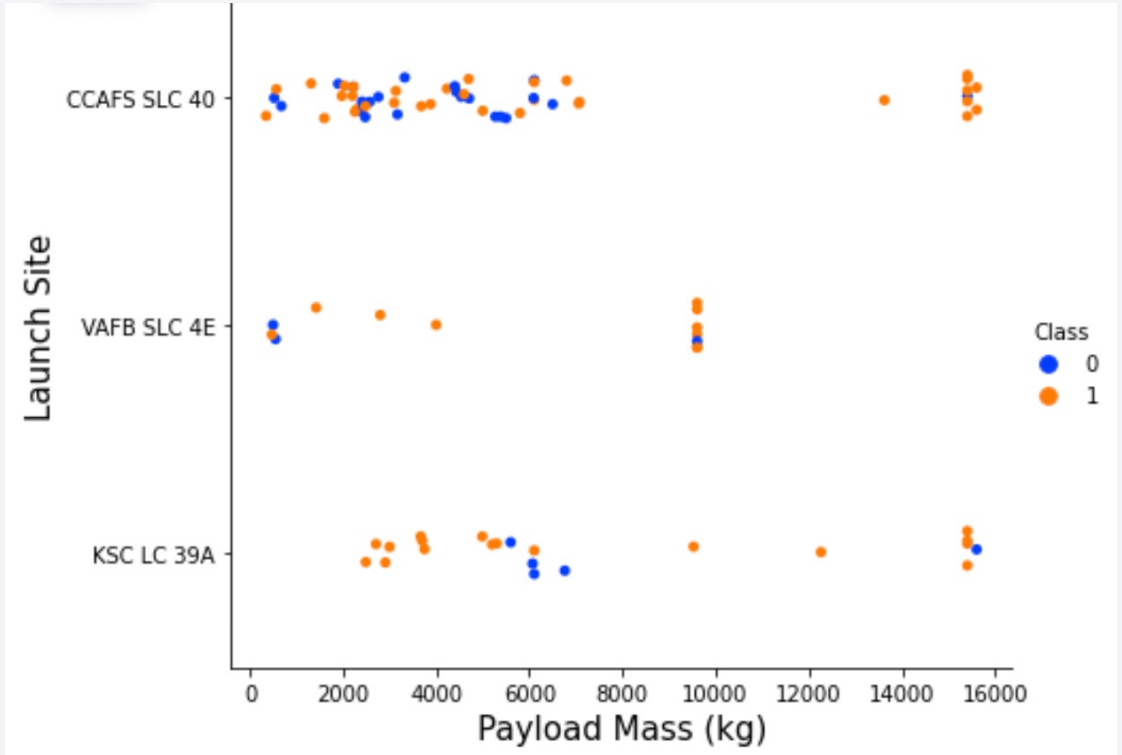
- Success rate increases as the number of flights increases .
- Early flights (flight numbers < 30) launched were mostly unsuccessful
  - Mostly launched from CCAFS SLC 40.
  - The launches from VAFB SLC 4E has two flights, which are all failed
  - No early flights were launched from KSC LC 39A.
- Recent flights are mostly successful.
  - Launches from VAFB SLC 4E was mostly success with only one failure.
  - Launched from CCAFS SLC 40 has most failures
- CCAFS SLC 40 launches most missions.



Class=1, success
Class=0, failure

Payload vs. Launch Site

- No clear correlation between payload mass and success rate for a given launch site.
- Crossing launch sites, launches with heavy payload mass are likely successful
  - Very few failure landings for payload mass > 7000 kg. The heaviest payload is close to 16000 kg.
- All sites launched a variety of payload masses
  - Major launches are from CCAFS SLC 40 being.
  - No launch with payload mess > 10000 kg is found in VAFB SLC 4 site.

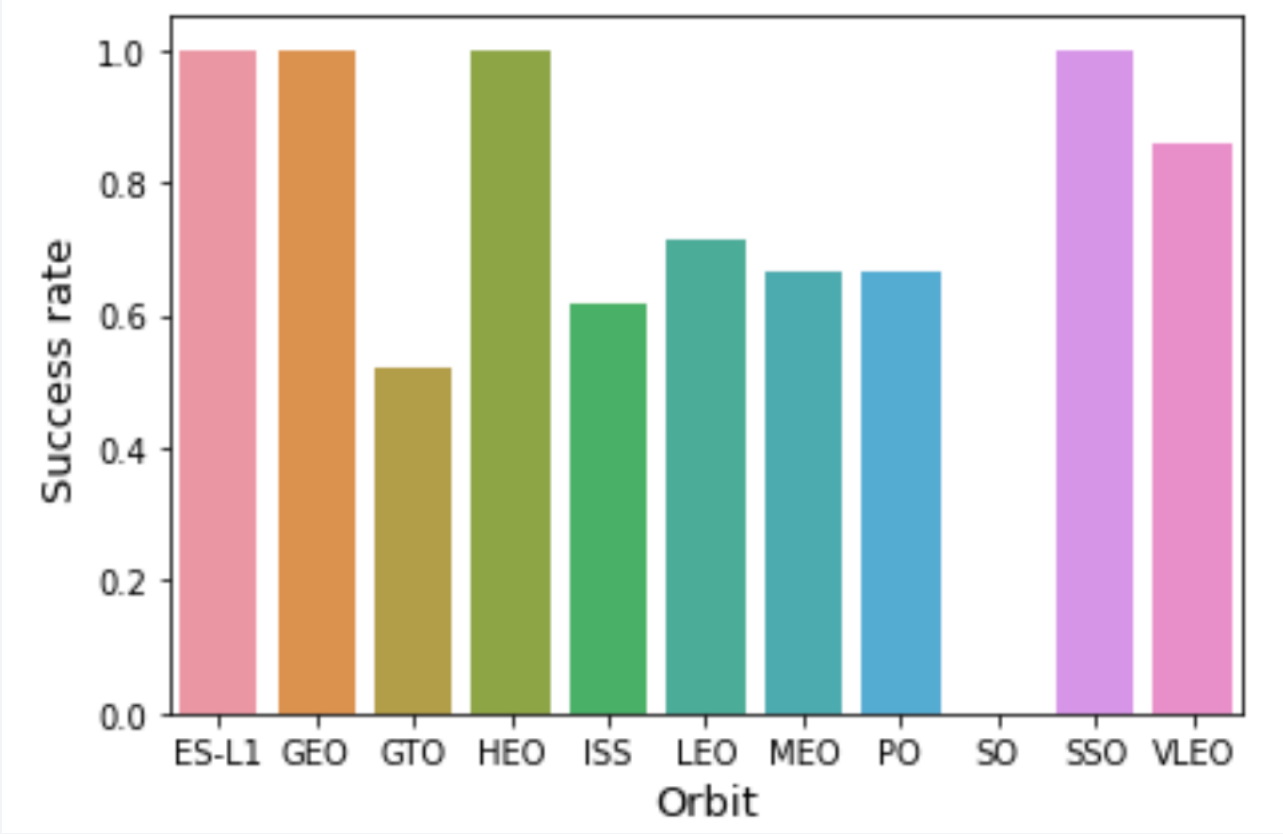


Class=1, success
Class=0, failure

Success Rate vs. Orbit Type

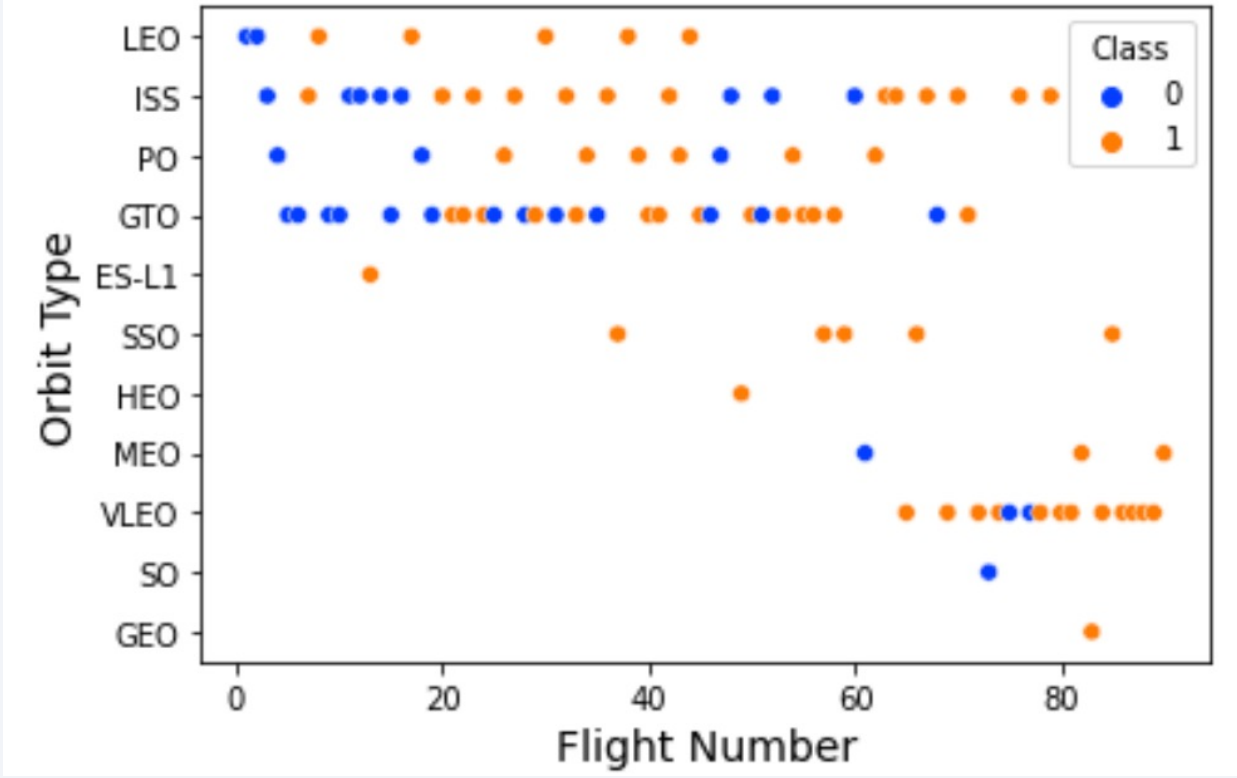
- Four orbits are successful with 100% success rate:
  - ES-L1 (Earth-Sun First Lagrangian Point)
  - GEO (Geostationary Orbit)
  - HEO (High Earth Orbit)
  - SSO (Sun-synchronous Orbit)

- No successful launch is associated with the SO (Heliocentric Orbit)
- The success rates of launches on other orbits are above 50 %



Flight Number vs. Orbit Type

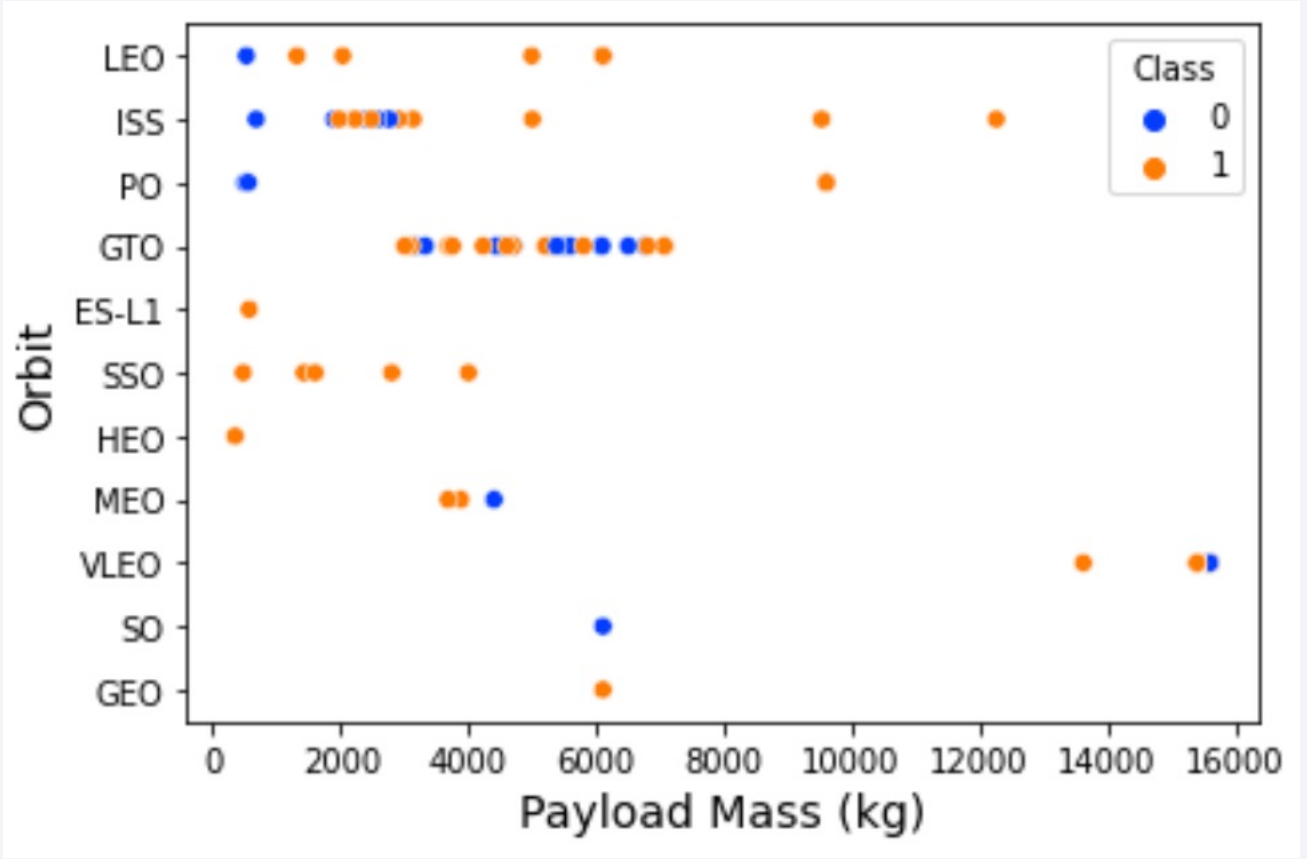
- The success rate increases as flight number increases
- 100% success rate of GEO, HEO, and ES-L1 orbits is ONLY related to one flight
- 100% success rate in SSO has 5 successful flights.
- No relationship between Flight Number and Success Rate for GTO.



Class=1, success
Class=0, failure

Payload vs. Orbit Type

- Orbit types (ES-L1, SSO, HEO) are always related to successful launch with low payload mass.
- Orbit types (PO, ISS, LEO) have more success with heavy payloads
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO launches are associated with heavier payloads
- Orbit SO in one launch was failed

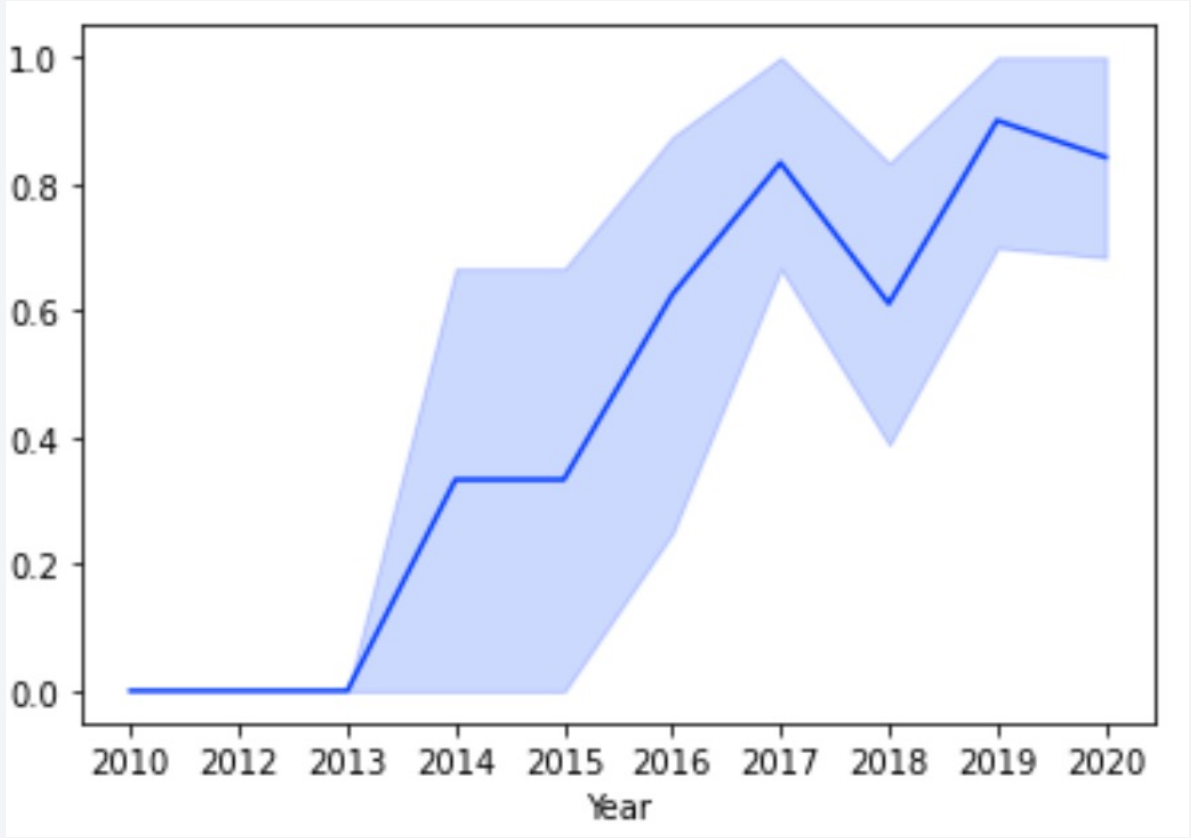


Class=1, success
Class=0, failure

Launch Success Yearly Trend

- All landings were failed before 2013.
- After 2013, the success rate generally increased
- After 2016, success rates are above 60%

- The success rate dropped slightly in 2018 and 2020.



All Launch Site Names

- Display the names of the unique launch sites
- The key word UNIQUE returns unique values from the LAUNCH_SITE column of the SPACEX_DATA table.

```
%sql SELECT UNIQUE(LAUNCH_SITE) FROM SPACEXTBL;
```

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

Launch Site Names Begin with 'CCA'

- Show 5 records where launch sites begin with `CCA`
- LIKE keyword is used to retrieve string values beginning with ‘CCA’.
- LIMIT 5 fetches only 5 records

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

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Use SUM keyword to sum `PAYLOAD_MASS__KG_` column as the total payload
- The WHERE condition filters the records boosters from NASA (CRS).

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

total_payload_mass
45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1.
- Use AVG keyword to calculate the average of the PAYLOAD_MASS__KG_ column.
- WHERE keyword (and the associated condition) filters the records of the F9 v1.1 booster version.

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

average_payload_mass
2928

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Use the MIN keyword to calculate the earliest date in the DATE column
- The WHERE keyword filters the records of the successful ground pad landings.

```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_GROUND_LANDING FROM SPACEXTBL \
    WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

first_successful_ground_landing
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 kg.
- Use the WHERE keyword to filter the records satisfying both conditions in the brackets (as the AND keyword is also used). Select payload range by the BETWEEN keyword.

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL \
    WHERE (LANDING__OUTCOME = 'Success (drone ship)') AND \
          (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000);
```

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
- The COUNT keyword is used to calculate the total number of mission outcomes, and the GROUPBY keyword is also used to group these results by the type of mission outcome.

```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

mission_outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Use the DISTINCT keyword to obtain the unique booster versions. The WHERE conditions find the maximum payload by subquery of the maximum payload mass by MAX(PAYLOAD_MASS__KG_)

```
%sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL \
    WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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
- Use the WHERE keyword to select the records of failed landing outcomes, AND only for the year of 2015.

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL \
    WHERE (LANDING__OUTCOME = 'Failure (drone ship)') AND \
        (EXTRACT(YEAR FROM DATE) = '2015');
```

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

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order
- Use the WHERE keyword filters records in the data range by the BETWEEN keyword. The results are also grouped by the keyword GROUP BY and then and ordered using the keyword ORDER BY, where DESC is applied for the descending order.

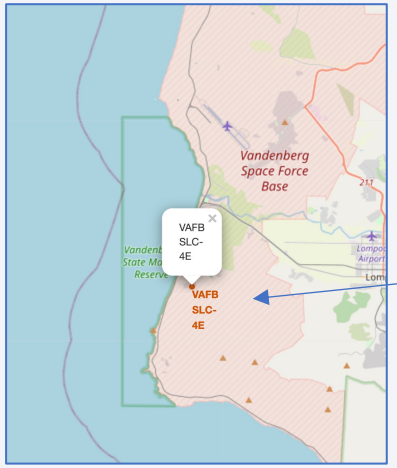
```
%sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL \
    WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
    GROUP BY LANDING__OUTCOME \
    ORDER BY TOTAL_NUMBER DESC;
```

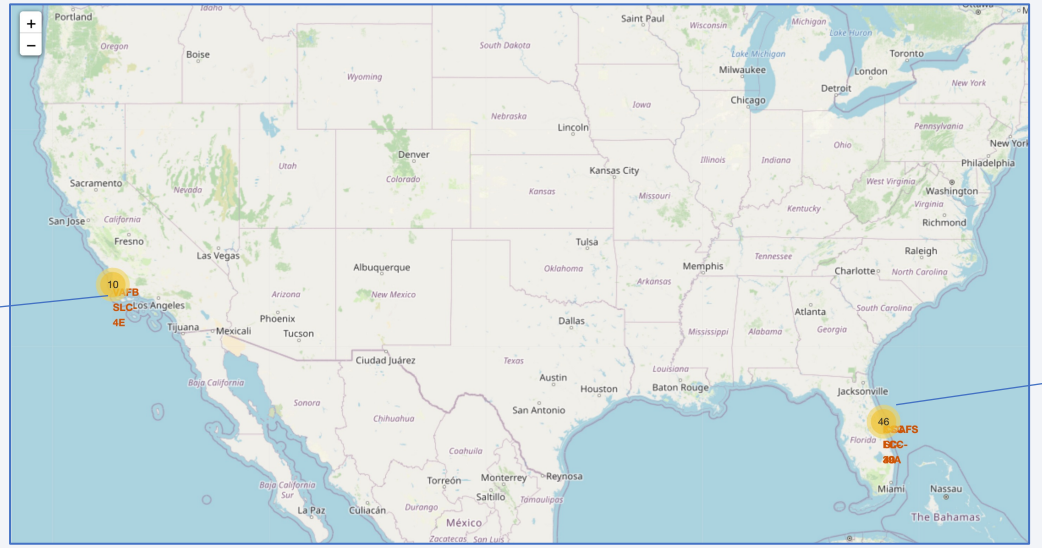
landing__outcome	total_number
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

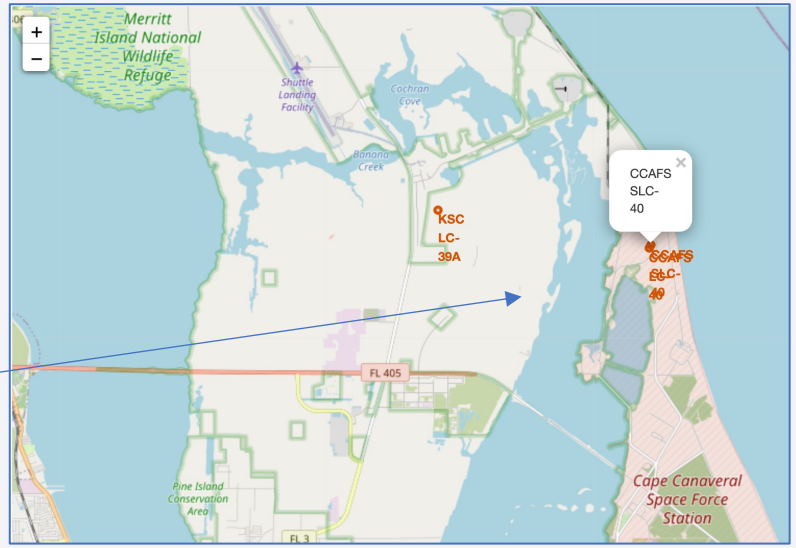

Section 3
Launch Sites
Proximities Analysis

Overview of Launch Sites

- All SpaceX launch sites are on the coasts of Florida and California, the United States.
- Zoom in the coasts of Florida, showing three launch sites located closely.
- One launch site is located on the coast of California.

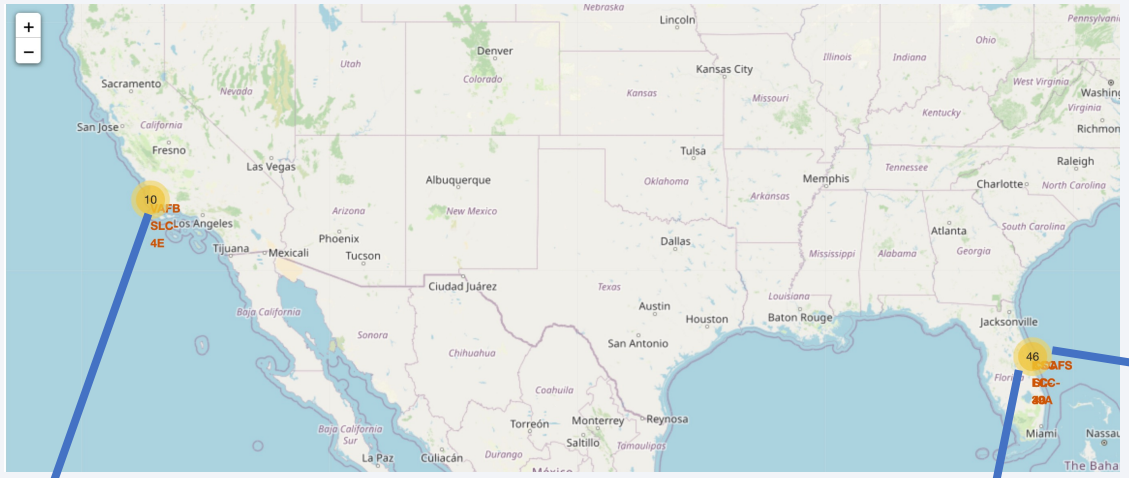


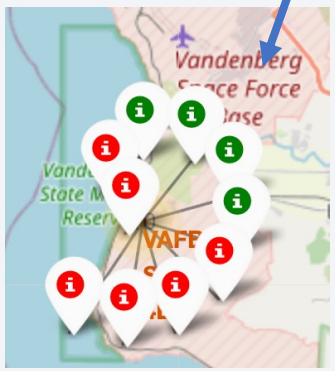




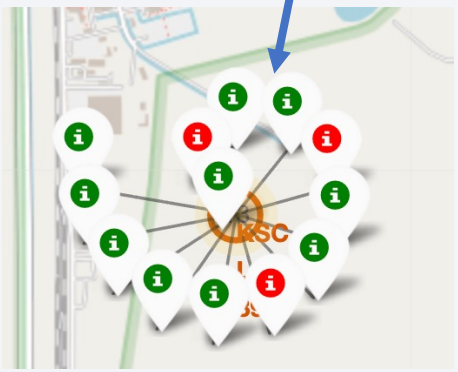
Launch Outcomes

- Successful/unsuccessful launches are annotated with green/red icons, respectively. Launches are grouped at four launch sites.

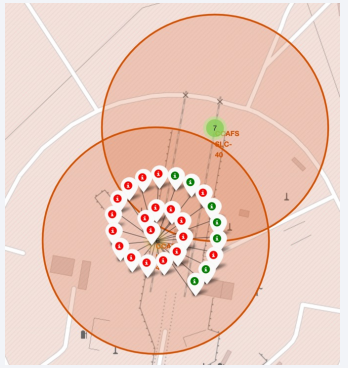




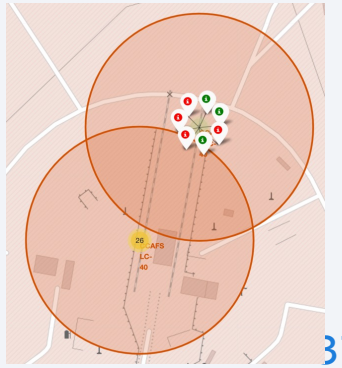
VAFB SLC-4E



KSC LC-39A



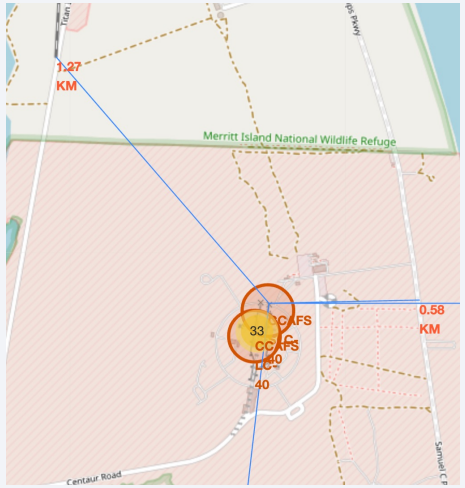
CCAFS SLC-40

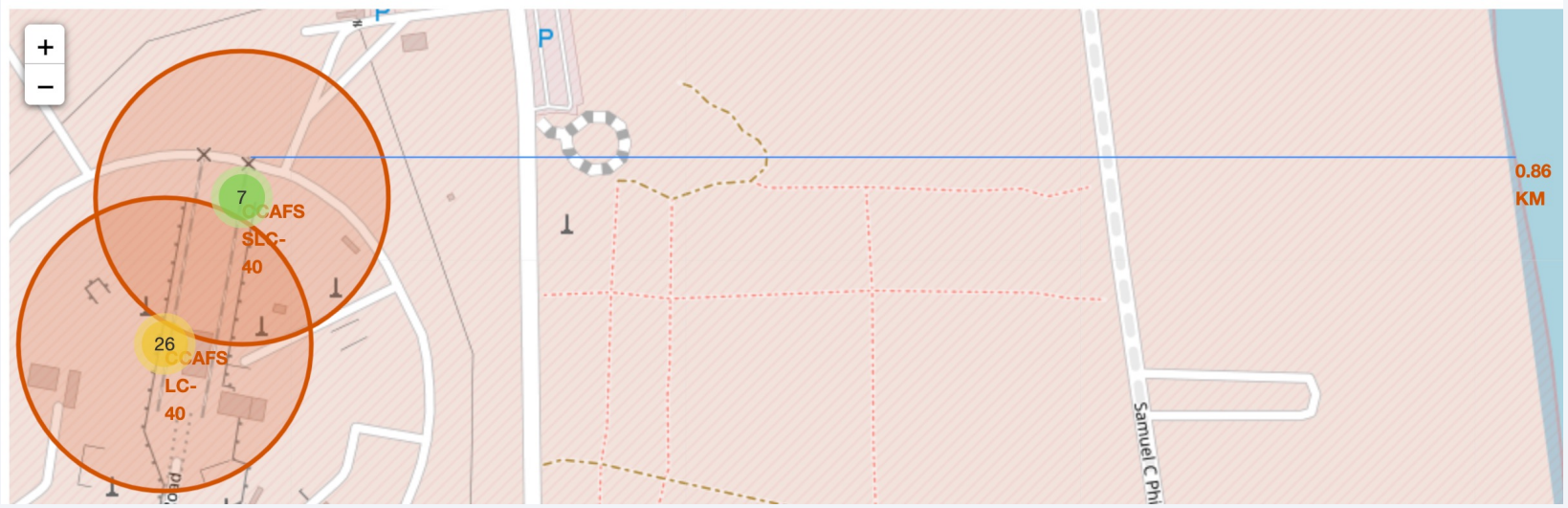


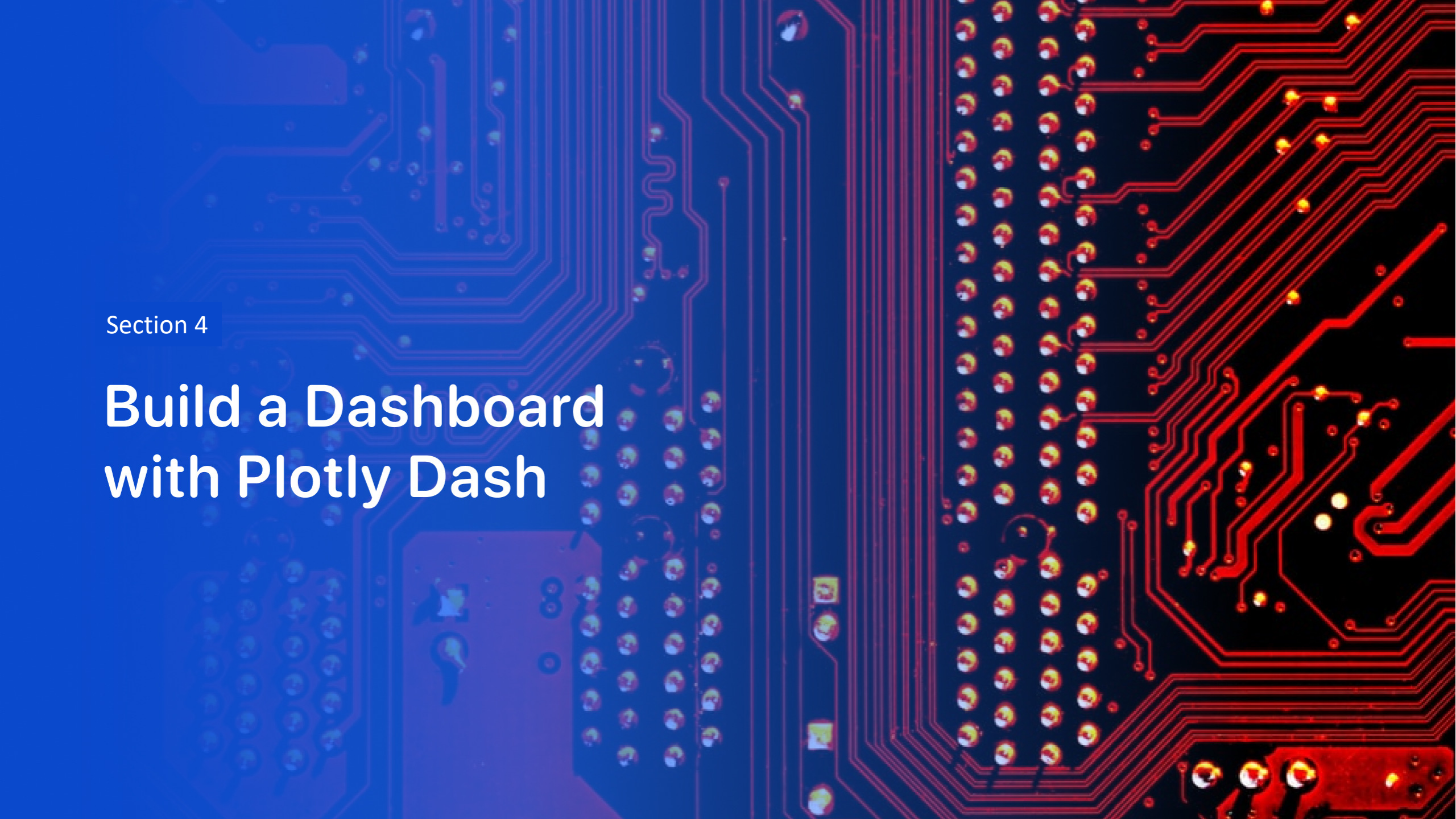
CCAFS LC-40

Proximity of launch sites

- Investigation of the CCAFS SLC-40 launch site, it shows more details of geo-information around the launch site.
- Are launch sites in close proximity to railways? YES. The coastline is only 0.86 km due East.
- Are launch sites in close proximity to highways? YES. The nearest highway is only 0.58km away.
- Are launch sites in close proximity to railways? YES. The nearest railway is only 1.27 km away.
- Do launch sites keep certain distance away from cities? YES. The nearest city Orlando is 54.67 km away.





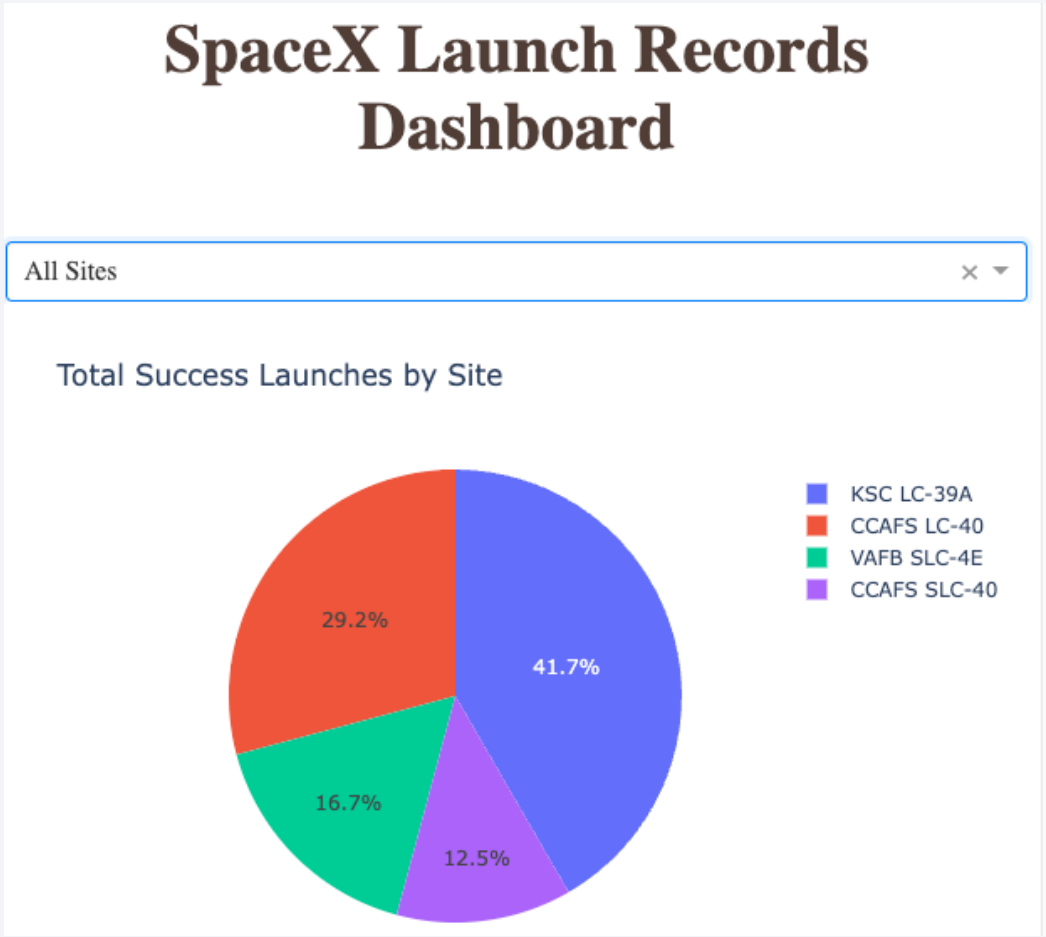

Section 4
Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard

- The launch site KSC LC-39 A (blue partition) has launched most of the successful launches, i.e., 41.7% of the total successful launches.

Class = 1, success

Class = 0, failure

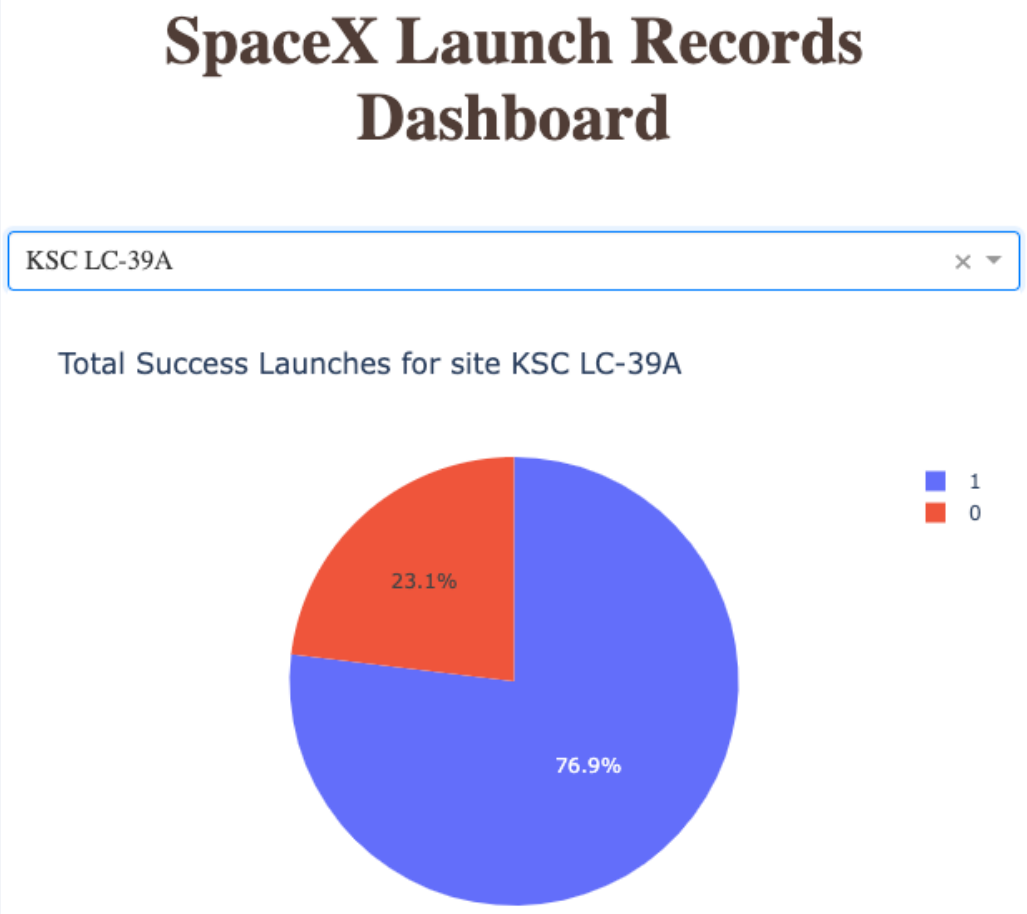


Pie chart for the launch site with highest launch success ratio, KSC LC-39 A

- KSC LC-39 A shows the highest rate of successful launches, with a 76.9% success rate.

Class = 1, success

Class = 0, failure



Payload vs. Launch Outcome scatter plot for all sites

- Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

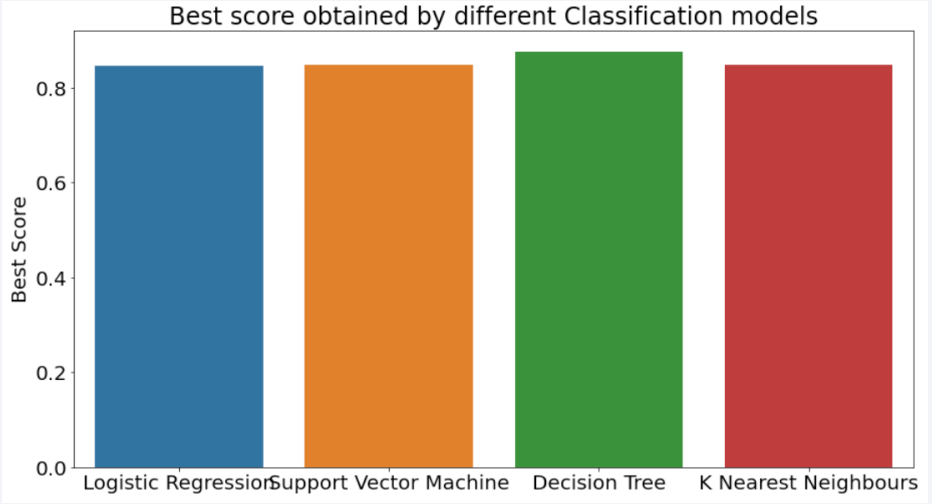
- Only Booster types (B4 and FT) have been launched with massive payloads (> 6000 kg).
- Only one successful launch with massive payloads (> 6000 kg)

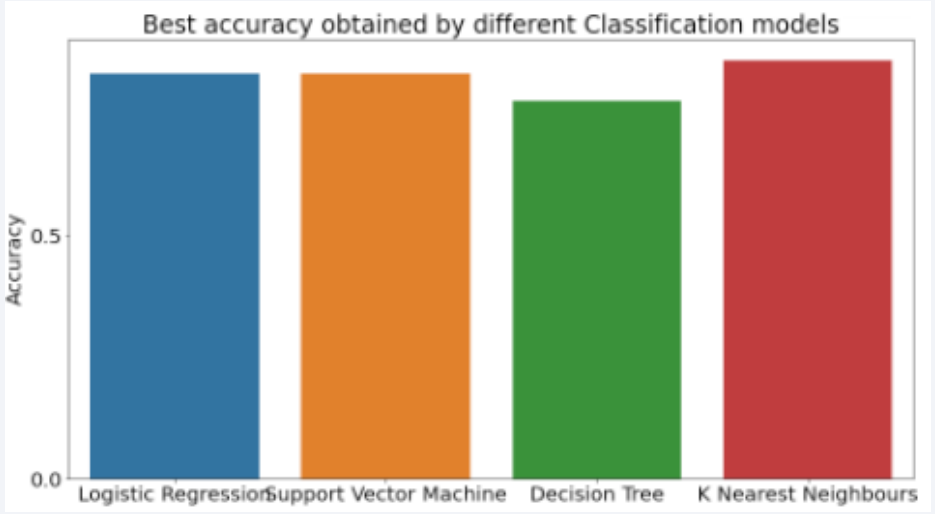
Section 5

Predictive Analysis (Classification)

Classification Accuracy

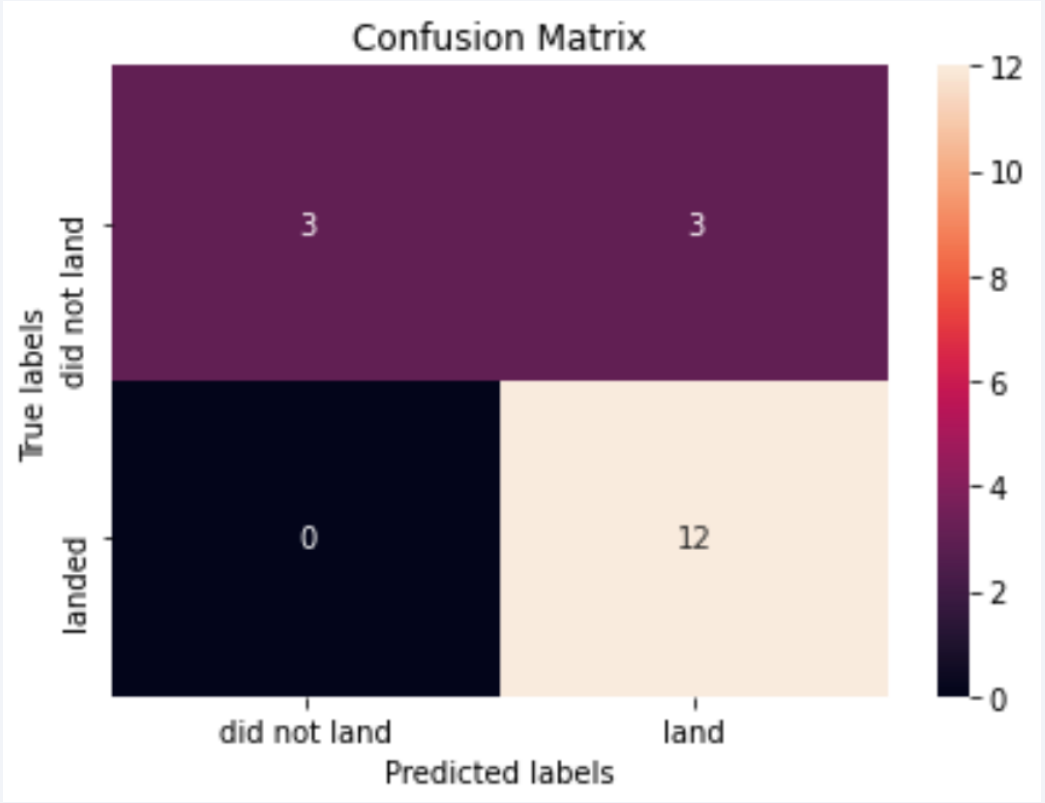
- Visualize the built model accuracy for all built classification models, in a bar chart.
- The K Nearest Neighbours model has the highest classification accuracy.
  - The accuracy score by test dataset is 86.11%
  - The best score by train dataset is 84.82%
- Note the decision tree model is not robust, which accuracy varies due to random initialization.





Confusion Matrix

- The confusion matrix of the best performing model (KNN) shows
- confusion matrix, which shows only 3 out of 18 total results classified incorrectly (a false positive, shown in the top-right corner).
- The other 15 results are correctly classified, where 12 results did land successfully, while 3 results failed landing.



Conclusions

Analysis of Space X Falcon 9 Launches success rate

- Success rate regarding years from 2010 to 2020. As the number of flights increases, the rate of success at a launch site increases over year
  - No landings were unsuccessful before 2013
  - The success rate generally increased from 2013, and above 80% since 2017
- Success rate regarding Orbit types
  - Orbit types ES-L1, GEO, HEO, and SSO are always success.
  - The orbit types PO, ISS, and LEO, have more success with heavy payloads:
  - Very Low Earth Orbit launches are associated with heavier payloads, which makes intuitive sense.
- Launch site KSC LC-39 A is most successful over 4 sites located in coasts of Florida and California
  - It has 41.7% of the total successful launches over 4 sites
  - The success rate is 76.9% at KSC LC-39 A
- The success for massive payloads (> 6000kg) is lower.
- The best performing classification model is the K nearest neighbor model, with an accuracy of 86.11%. The accuracy of other models are a little bit lower around 83%

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

```
algorithms = ['Logistic Regression', 'Support Vector Machine', 'Decision Tree', 'K Nearest Neighbours']

scores = [lr_score, svm_score, tree_score, knn_score]

best_scores = [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_]

column_names = ['Algorithm', 'Accuracy Score (test)', 'Best Score (train)']

df = pd.DataFrame(list(zip(algorithms, scores, best_scores)),columns = column_names)
```

	Algorithm	Accuracy Score (test)	Best Score (train)
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.722222	0.876786
3	K Nearest Neighbours	0.861111	0.848214

Thank you!