IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Prarthana kolhe

19-03-2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection with SpaceX REST API
  - Data collection with web scraping from Wikipedia
  - Data wrangling
  - Data analysis with SQL
  - Data visualization with matplotlib
  - Visual analytics with Folium and Plotly Dash
  - Predictive analysis with machine learning
- Summary of all results
  - Exploratory data analysis results
  - Predictive analysis results

Introduction

- Project background and context

  Most rocket launchers advertise the cost of rocket launches as, at least, 165 million dollars. Interesting, SpaceX advertises same task with a cost 62 million dollars which is about 62% lower than its competitors. We noticed that SpaceX's launch cost is due to the reusability of the rocket's first stage. Thus, if we could predict the landing outcome of a rocket's first stage, we could determine a rocket launch cost, and our results can be used by companies that would like to bid against SpaceX for a rocket launch. A recent increase in commercial space travel makes our work relevant.

- Problems you want to find answers

  We want to understand the correlation between the first stage landing success and other factors such as the payload mass, launch sites, type of rockets, launching pad used, launching dates, e.t.c.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected with API (SpaceX) and webscraping (Wikipedia)
- Perform data wrangling
  - We used one-hot encoding method to convert relevant features into classes
- 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, evaluate classification models

Data Collection

- Describe how data sets were collected
  - Data was collected with the SpaceX REST API, and from Wikipedia using webscraping. Data collected gave us information about rockets, payload, launching and landing specifications, and finally the landing outcomes
  - Rocket launch data JSON was obtained from one of SpaceX REST API endpoints using get request
  - Obtained json was turned into a dataframe with .json_normalize() method
  - Obtained dataframe was filtered to only include Falcon 9 booster version
  - Replaced missing values with the average of their repective columns
  - With webscraping on Wikipedia, BeautifulSoup was used to extract Falcon 9 launch records
  - Data collected and cleaned were used for the subsequent analysis

Data Collection - SpaceX API

- Using requests.get(), we obtained launch data from SpaceX REST API
- We converted the response using .json() and turned the response to a Pandas data frame
- Cleaning the data by replacing missing values in their respective columns
- Click here for the GitHub URL to the full notebook

1. Get launch data from SpaceX API with get resquest

```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

2. Decode the response using `.json()` and turn into a Pandas dataframe using `.json_normalize`

```
# Use json_normalize meethod to convert the json result into a dataframe
response = response.json()
data = pd.json_normalize(response)
```

3. Filter the dataframe for only `Falcon 9` launches

```
# Get dataframe for Falcon 9
data_falcon9 = launch_dict_data[launch_dict_data['BoosterVersion']!='Falcon 1']
```

4. Locate missing values and replace with `average` of values in the column

```
# Calculate the mean value of PayloadMass column
data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(
    to_replace = np.nan, value = data_falcon9['PayloadMass'].mean())
```

Data Collection - Scraping

- We web scrap with BeautifulSoup to get the launch records for Falcon 9
- Extract needed column names and create a data frame by parsing the obtained table
- Click here for the GitHub URL to the full notebook

1. Get Falcon9 Lauch HTML page on Wkipedia in `static_url` with `requests.get()`

```
static_url = "https://en.wikipedia.org/w/index.php? \n
title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"


response = requests.get(static_url)
# assign the response to a object
data = response.text
```

2. From the response object in `data`, use `BeautifulSoup()` to create a `soup` object

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

3. Extract relevant column names

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

column_names = []
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name != None and len(name) > 0:
        column_names.append(name)
```

4. Create data frame by parsing html tables

```
launch_dict= dict.fromkeys(column_names)
df=pd.DataFrame(launch_dict)
```

Data Wrangling

- Calculate launches from each launch site
- Calculate number of occurrences from each orbit type
- Calculate landing outcomes and create labels for each outcome (0 for failures, 1 for successes)
- Click here for the GitHub URL to the full notebook

1. Get information for the launch sites and orbits

```
# For each launch site, calculate the number of launches
# using .value_counts()

df['LaunchSite'].value_counts()


# For each orbit type, calculate the number of occurrences
# using .value_counts()

df['Orbit'].value_counts()
```

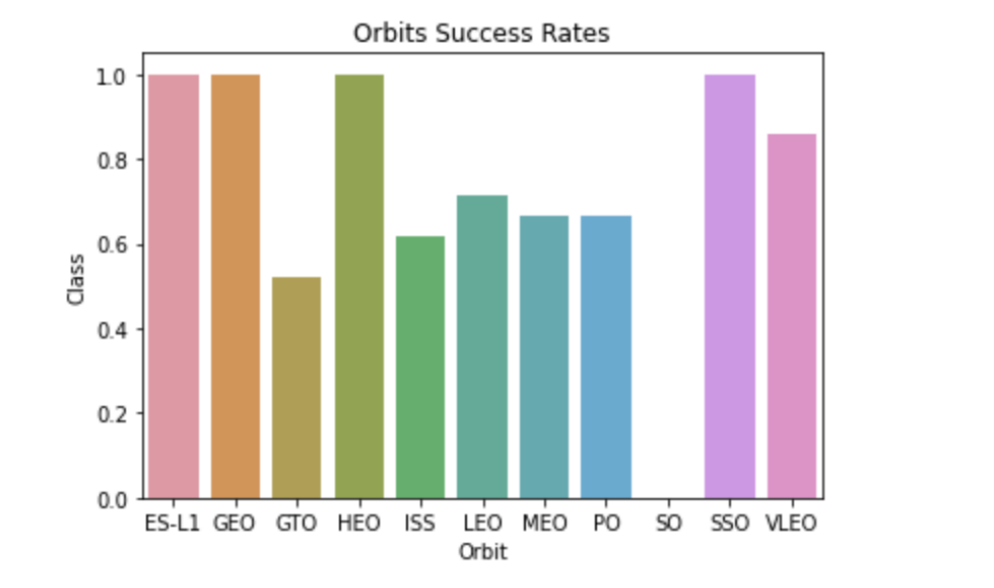
2. Get landing outcomes for each orbit type and creating outcome labels for landing classes

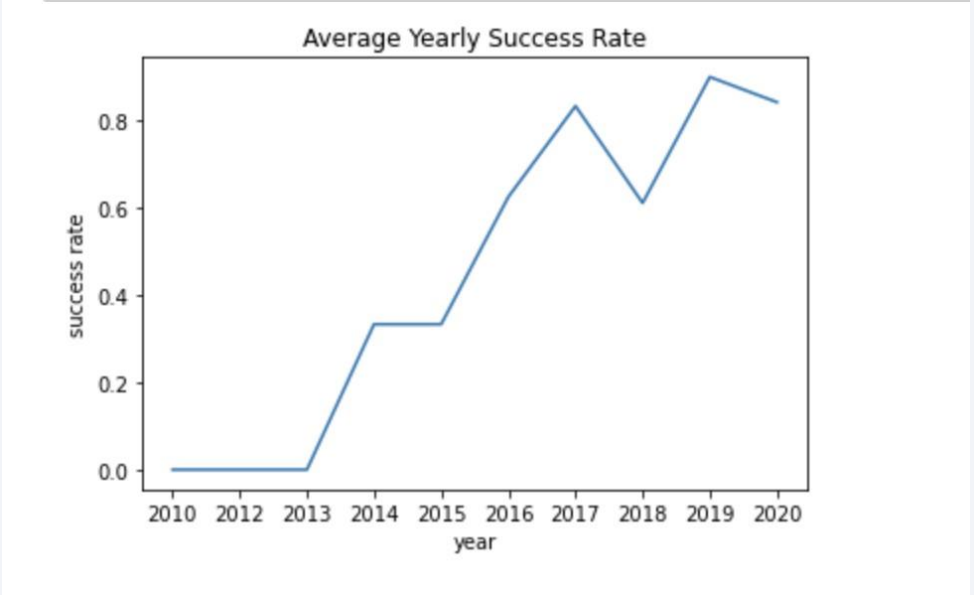
```
landing_outcomes = df['Outcome'].value_counts()

# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
n = len(df['Outcome'])
landing_class = np.zeros(n)
for i in range(n):
    if df['Outcome'].loc[i] in bad_outcomes:
        landing_class[i] = 0
    else:
        landing_class[i] = 1
landing_class = list(landing_class)
```

EDA with Data Visualization

- Visualizing two important charts
  - Success rate for orbit types: to know the safe region for commercial space trave
  - Success rate and yearly trend: to see how safe such exploration is, and how much progress has been made





- Click here for the GitHub URL to the full notebook

EDA with SQL

- SQL queries were used to:
  - Get unique launch sites in the space mission
  - Get the total payload mass carried by boosters launched by NASA(CRS)
  - Get average payload mass carried by booster version F9 v1.1
  - Get names of boosters with success in drone ships and payloaf >4000 and <6000
  - Get total successful and failed missions
  - Get boosters carrying maximum payload mass
- Click here for the GitHub URL to the full notebook

Build an Interactive Map with Folium

- Created map objects like circles and markers for each launch sites to mark launch sites on map
- Created MarkerCluster object to mark successful or failed lauches for each site. These were also colored-coded for ease of differentiation
- Created MousePosition objects to get coordinates of any points of interest
- We created a function to calculate distance between launch sites and places like cities, railways, highways, and coastlines. We needed to know these to evaulate the ease of transportation to and from launch sites, and possible effect of city-polution due to rocket launches.
- Click here for the GitHub URL to the full notebook

Build a Dashboard with Plotly Dash

- With plotly dash, we built an interactive dashboard that displays:
  - Pie charts of total launches from all sites
  - Pie charts of mission outcomes from each site
  - Scatter graph of the relationship of payload mass and mission outcomes for booster versions
- These results will help in focusing on how to maximize successful mission outcomes. We now know sites, boosters, and payload masses that give best outcomes.
- Click here for the GitHub URL to the full notebook

Predictive Analysis (Classification)

- Using numpy and pandas, we loaded our data and splitted into training and testing data steps
- We used GridSearchCV to tune and get the best hyperparameters for our machine learning models
- We used the .score() method to get the accuracy of each model
- We compared the accuracy of all four models
- Click here for the GitHub URL to the full notebook

Results

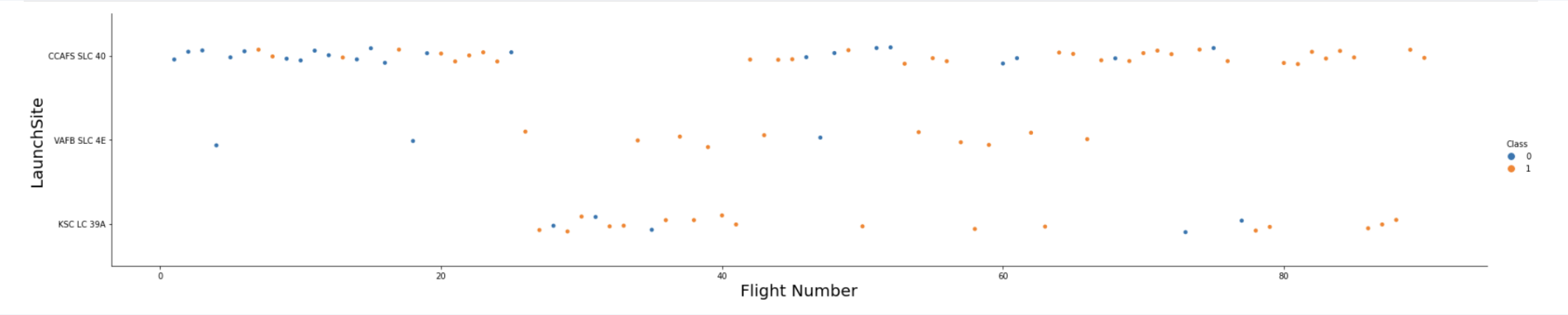
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

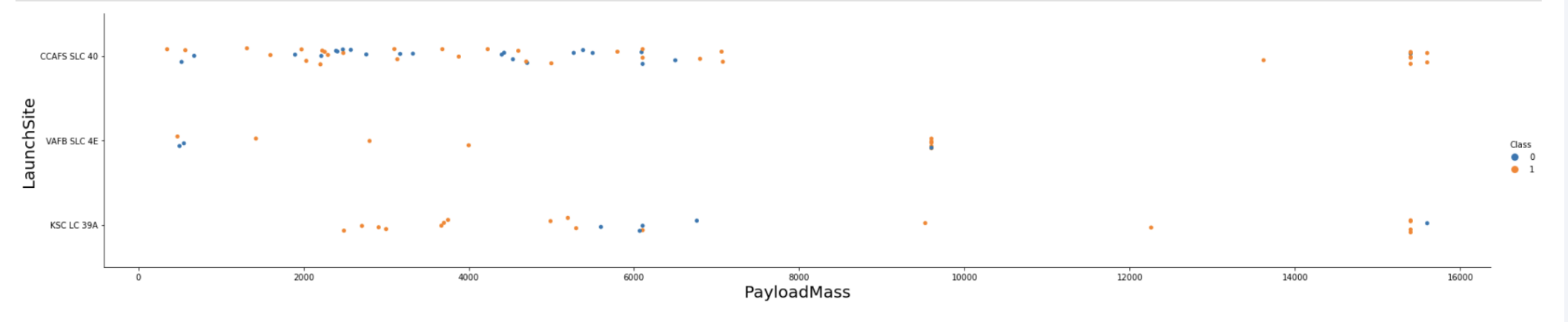
Flight Number vs. Launch Site

- For each launch site, success rate increases with increasing flight number



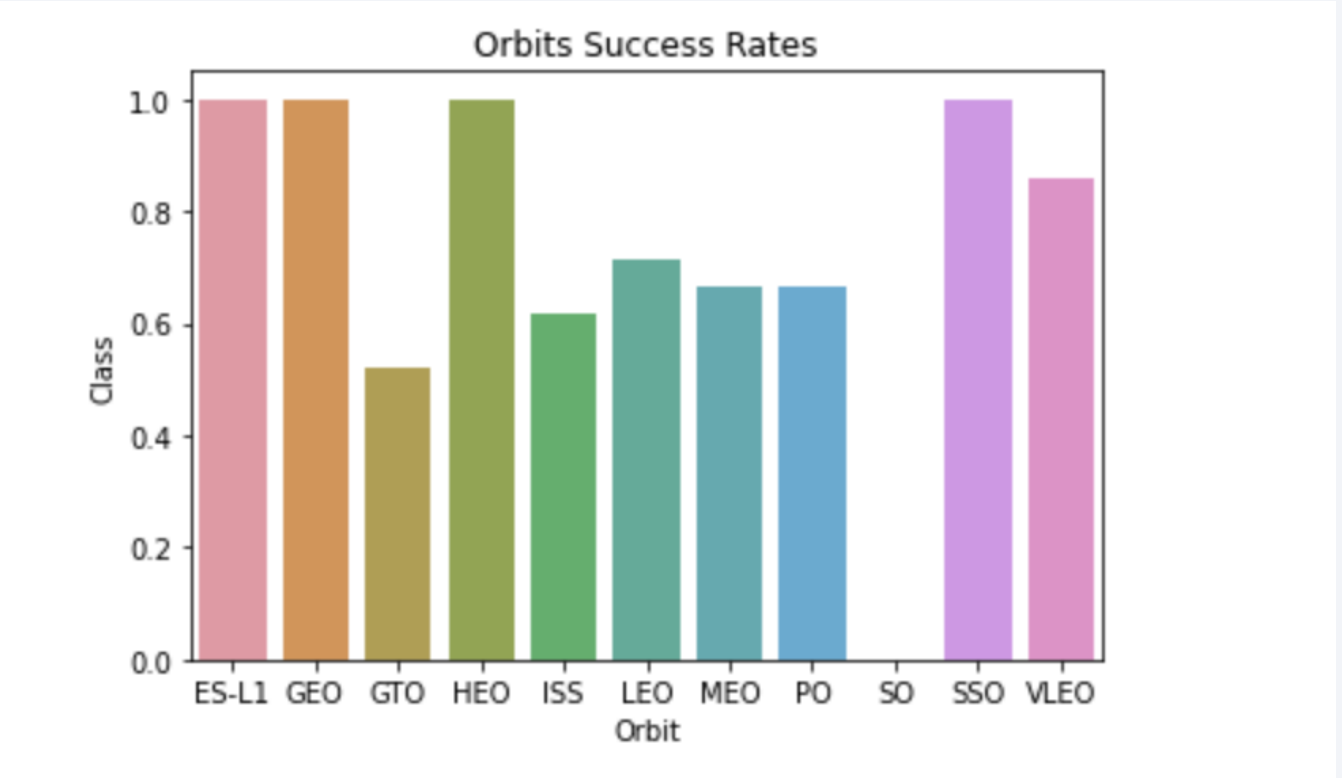
Payload vs. Launch Site

- Success rate for CCAFS SLC 40 is 100% when the payload mass is > 8000 kg
- No launches in VAFB SLC 4E for payload mass is > 10000 kg
- Success probability is very low in KSC LC 39A for payload around 6000 kg



Success Rate vs. Orbit Type

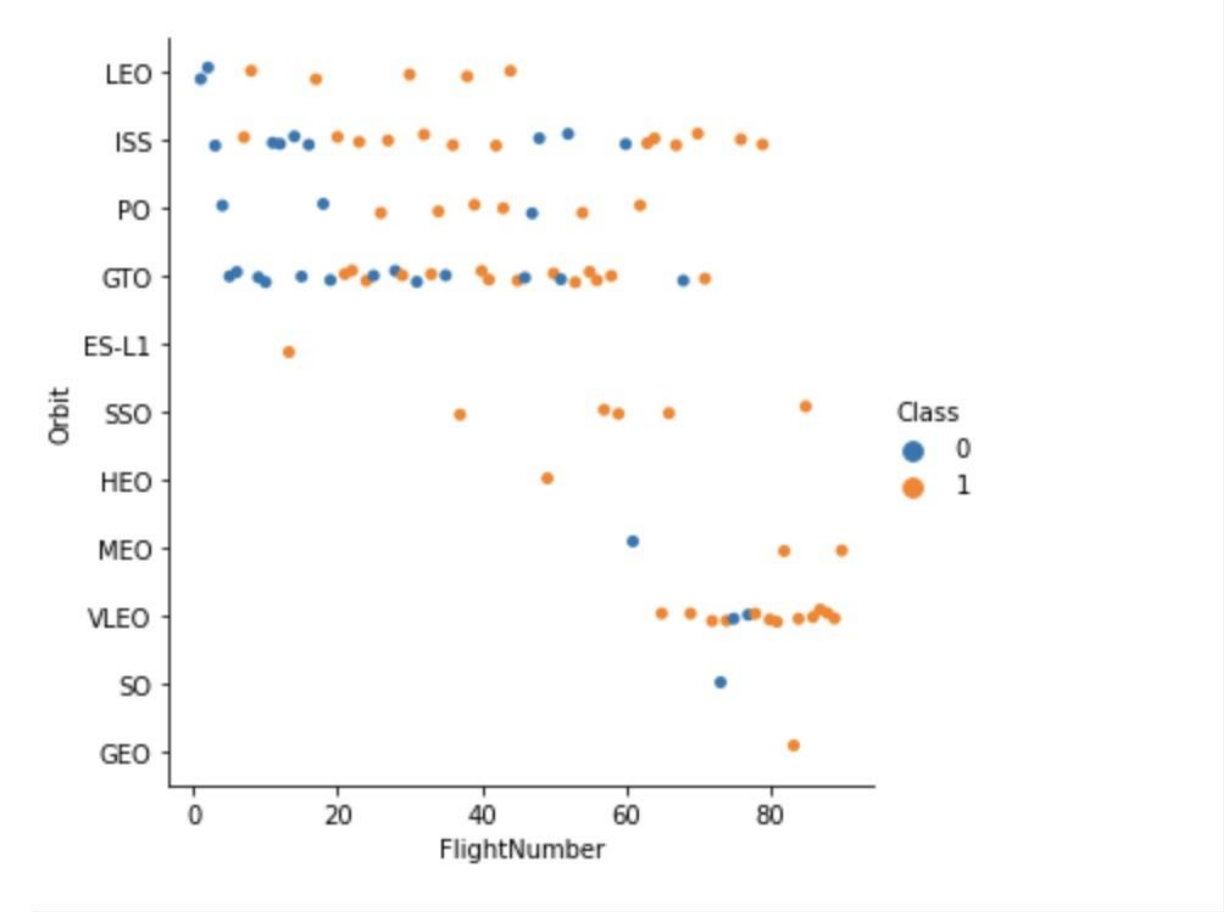
- Best success rates are observed in ES-L1, GEO, HEO, and SSO orbit types. No success rates recorded for SO orbit type



Flight Number vs. Orbit Type

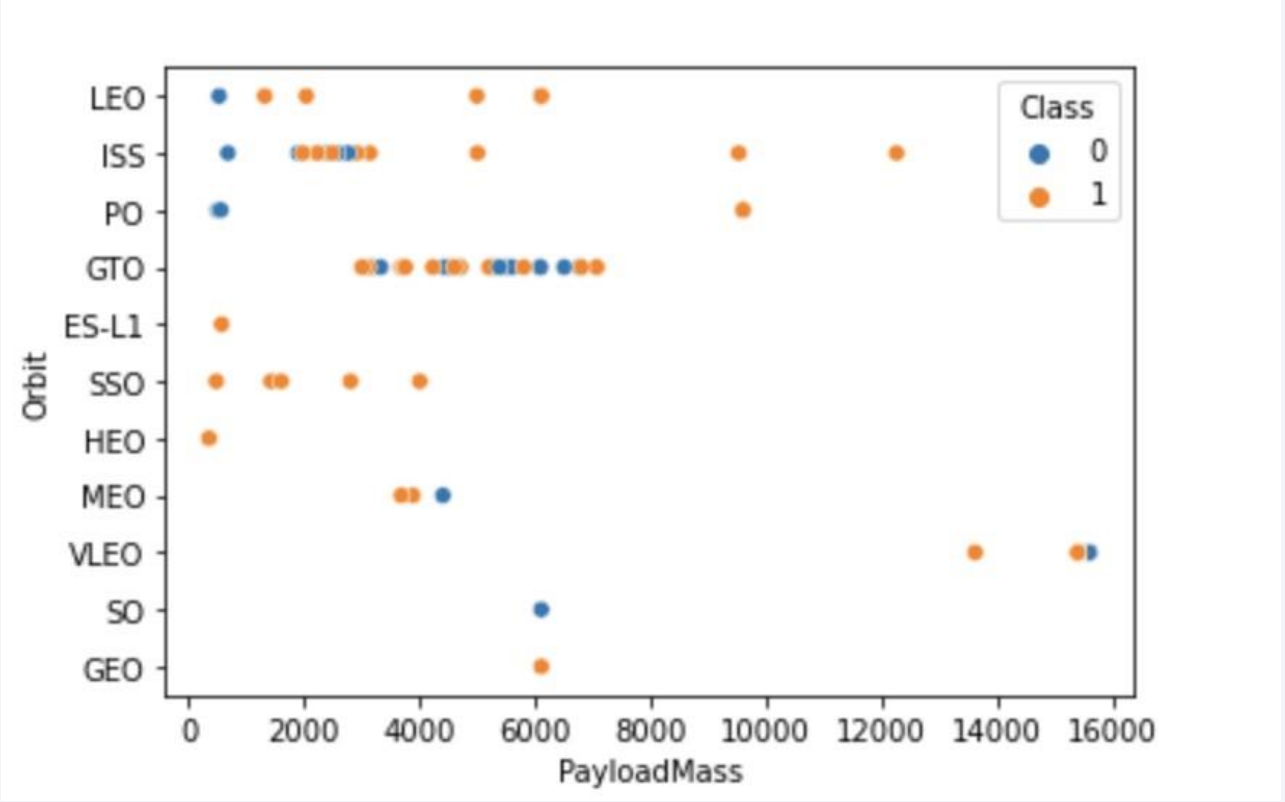
- LEO orbit has a strong relationship of success rate with the number of flights. Success rate is 100% for flight number > 10

- On the other hand, GTO orbit shows no clear relationship of success rate with flight number



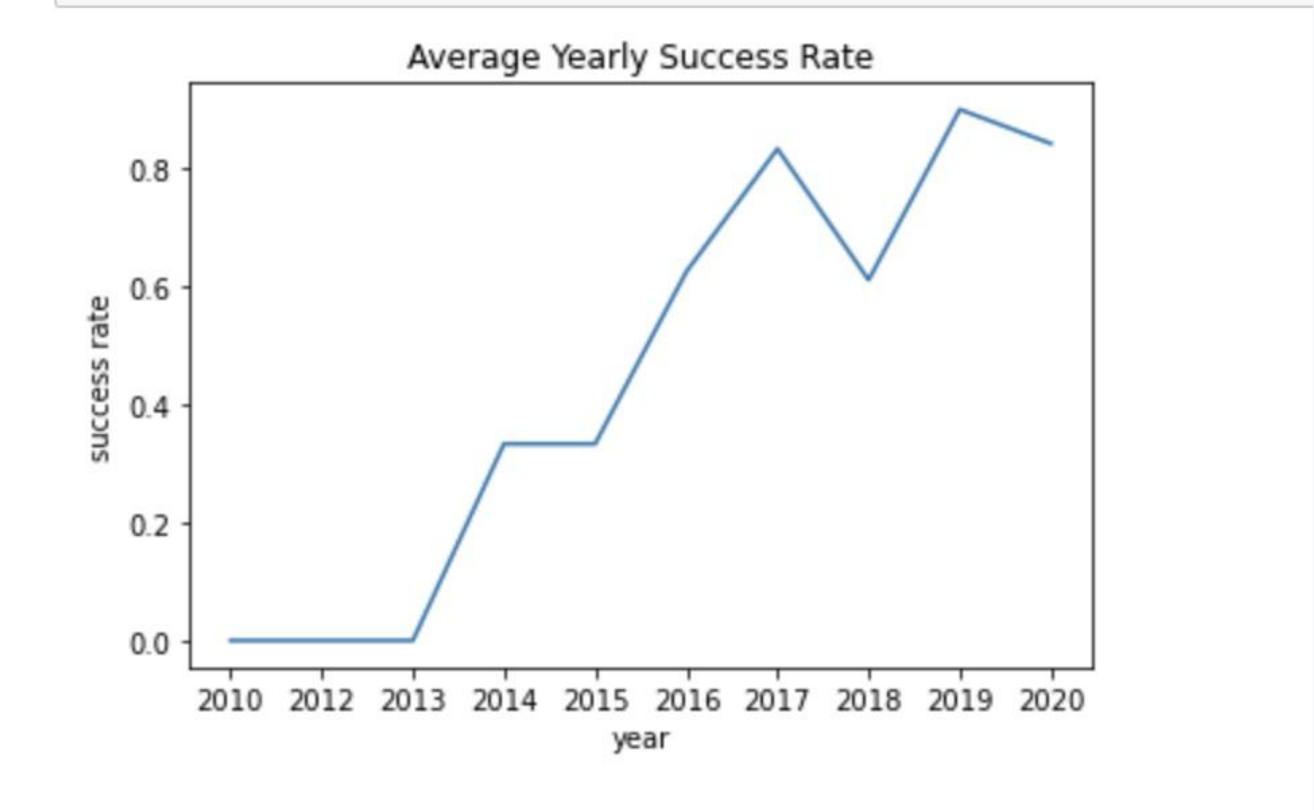
Payload vs. Orbit Type

- LEO, ISS, and PO orbits have high success rates with heavy payloads
- SS0 orbit has maximum success rates for low payload mass.
- GTO orbit does not show any clear relationship with payload mass



Launch Success Yearly Trend

- Increasing success trend rate from 2013 till 2020



All Launch Site Names

- To display names of unique launch site, we use the DISTINCT function in SQL

Task 1

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

```
%sql select distinct(launch_site) from spacex
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e61
Done.
```

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

Launch Site Names Begin with 'CCA'

- We use the 'like' keyword to filter any string or character we are looking for

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

```
%sql select * from spacex where launch_site like 'CCA%' limit 5
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/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

Total Payload Mass

- Total payload = 45596

Task 3

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

```
%sql select sum(payload_mass__kg_), customer from spacex group by customer
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu
Done.
```

Average Payload Mass by F9 v1.1

- Total average payload mass carried by booster version F9 v1.1 = 2928

Task 4

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

```
%sql select avg(payload_mass__kg_), booster_version from spacex group by booster_version
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

First Successful Ground Landing Date

- First successful ground landing date is 2015-12-22

```
%sql select date,landing__outcome from spacex where landing__outcome like 'Success%' order by date limit 5
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

DATE	landing__outcome
2015-12-22	Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

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

Task 6

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

```
%sql select booster_version, payload_mass__kg_,landing__outcome from spacex where (landing__outcome = 'Success (drone ship)' and payload_m
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

booster_version	payload_mass__kg_	landing__outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failure mission outcomes after filtering with WHERE function

```
%sql select count(*) from spacex where (mission_outcome != 'Success (payload status unclear)')
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

1
100

```
%sql select count(*) from spacex where (mission_outcome = 'Success (payload status unclear)')
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

1
1

Boosters Carried Maximum Payload

- The names of the booster which have carried the maximum payload mass. We filtered the results with the WHERE and MAX functions

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

```
%sql select payload_mass__kg_,booster_version from spacex where payload_mass__kg_ = (select max(payload_mass__kg_) from spacex)
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

payload_mass__kg_	booster_version
15600	F9 B5 B1048.4
15600	F9 B5 B1049.4
15600	F9 B5 B1051.3
15600	F9 B5 B1056.4
15600	F9 B5 B1048.5
15600	F9 B5 B1051.4
15600	F9 B5 B1049.5
15600	F9 B5 B1060.2
15600	F9 B5 B1058.3
15600	F9 B5 B1051.6
15600	F9 B5 B1060.3
15600	F9 B5 B1049.7

2015 Launch Records

- Failed landing_outcomes in drone ship, their booster versions, and launch site names for year 2015.

```
%sql select landing__outcome, date, booster_version, launch_site from spacex where (landing__outcome = 'Failure (drone ship)' and date lik
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

landing__outcome	DATE	booster_version	launch_site
Failure (drone ship)	2015-01-10	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	2015-04-14	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

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, count(landing__outcome) as count from spacex where (date between '2010-06-04' and '2017-03-20') group by lar
```

```
 * ibm_db_sa://sjm82238:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

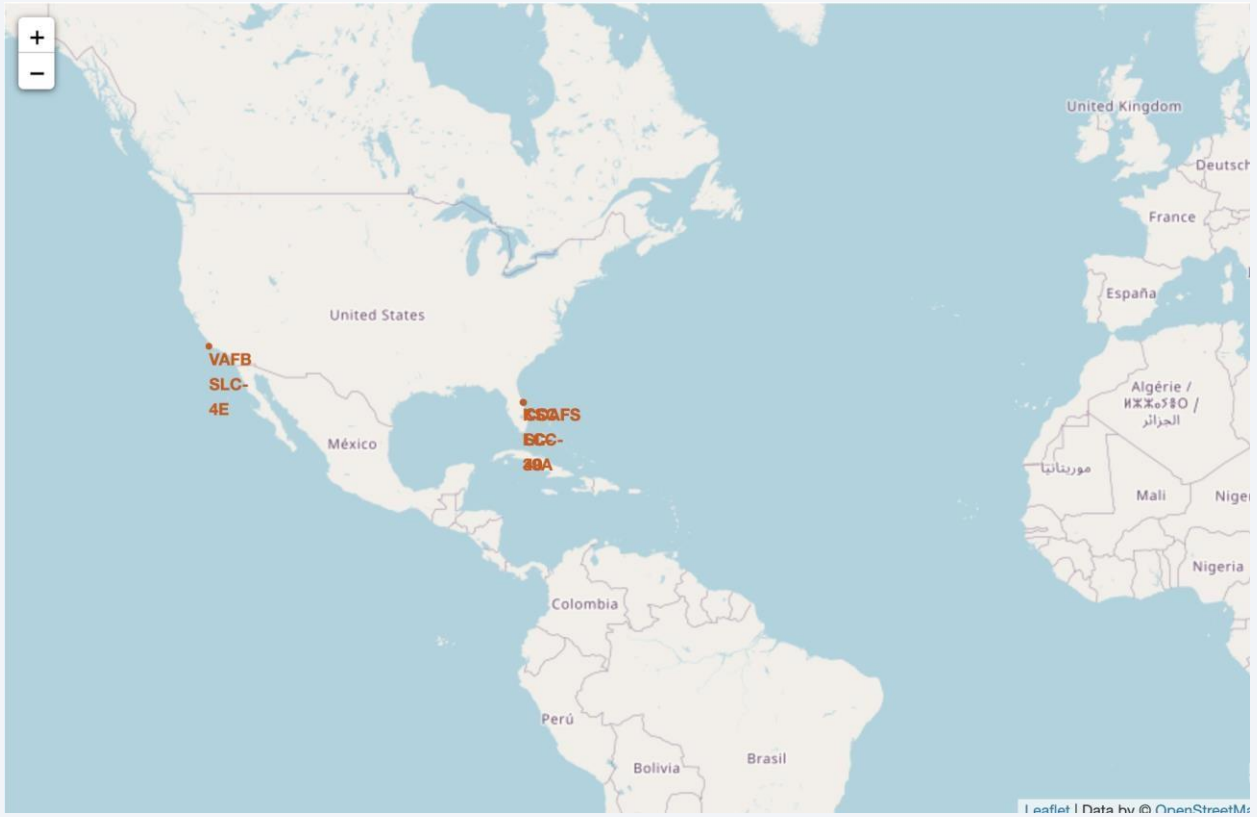
landing__outcome	COUNT
Controlled (ocean)	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	10
Precluded (drone ship)	1
Success (drone ship)	5
Success (ground pad)	3
Uncontrolled (ocean)	2

Section 3

Launch Sites Proximities Analysis

Lauch sites global map markers

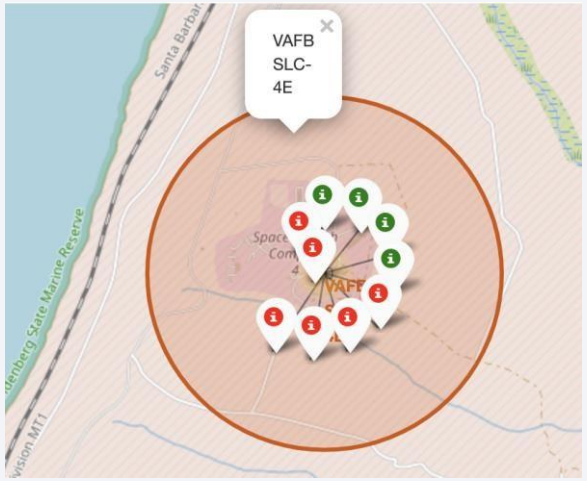
- All launch sites are close to the coastline in their locations

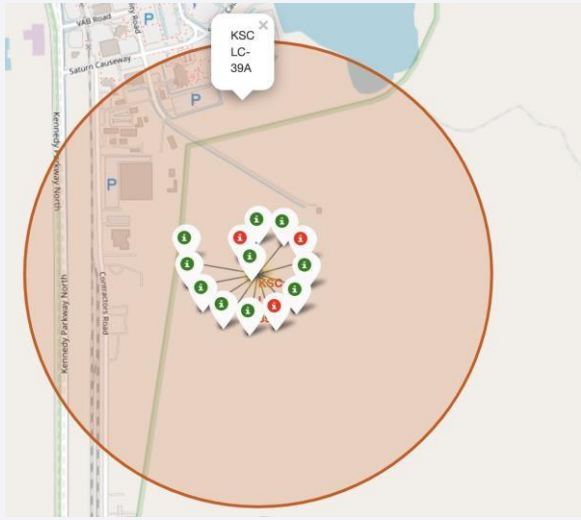


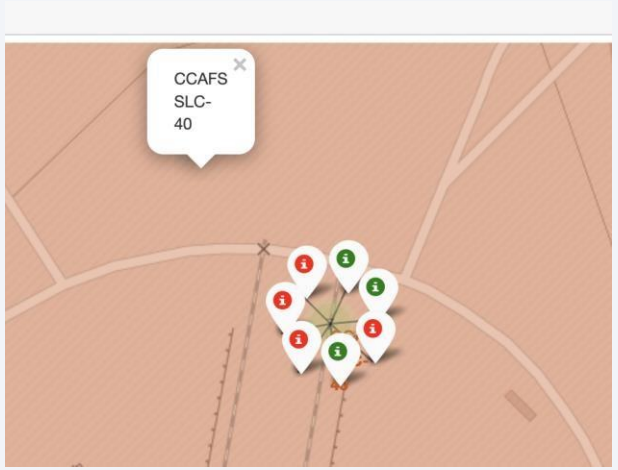
Launch sites color-coded launch outcomes

Key: Green (successful launches), Red (failed launches)

Lauch site KSC LC-39A has highest number of successful launches

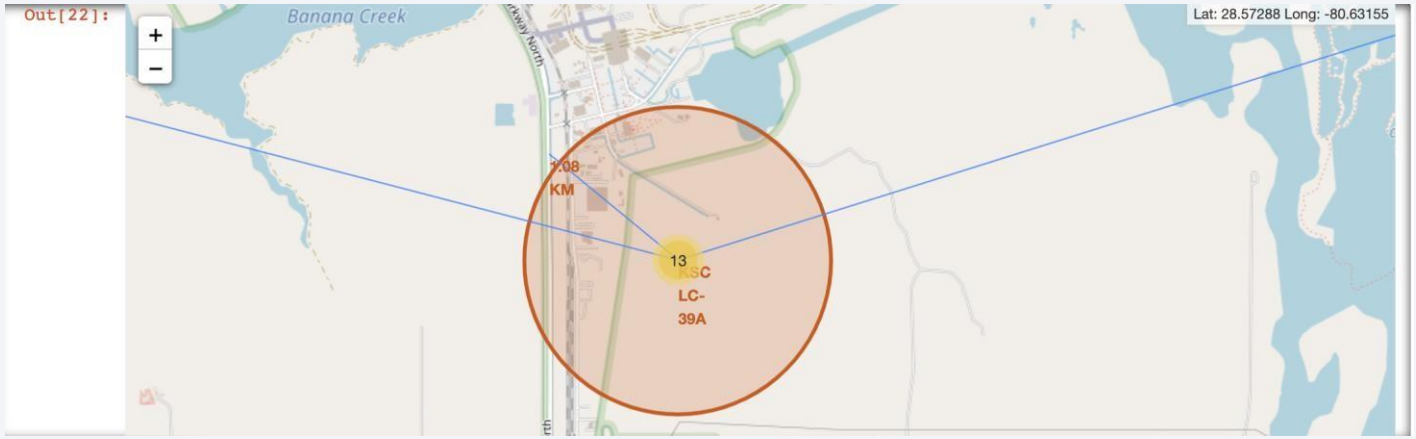


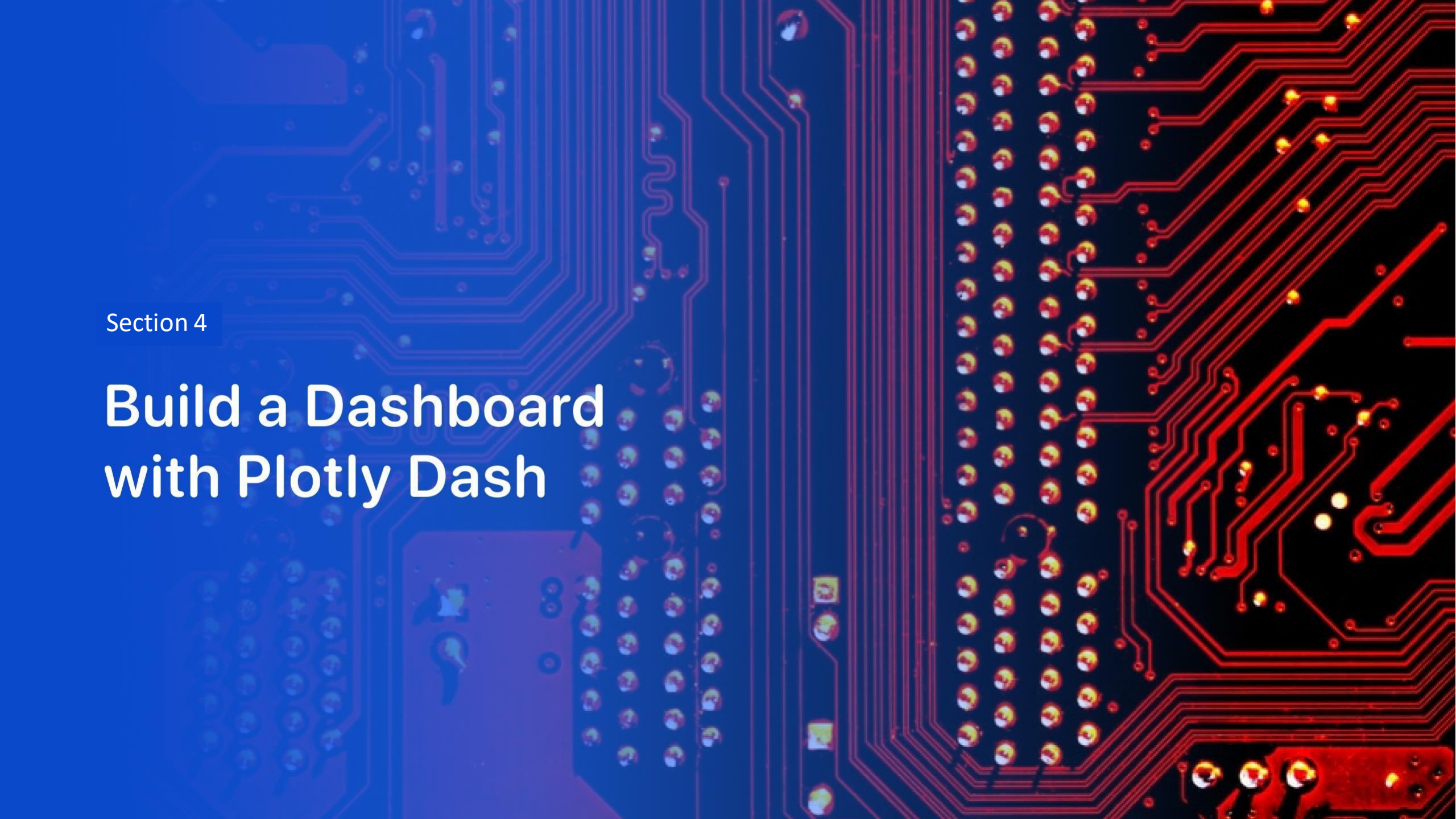




Launch sites and distances to landmarks

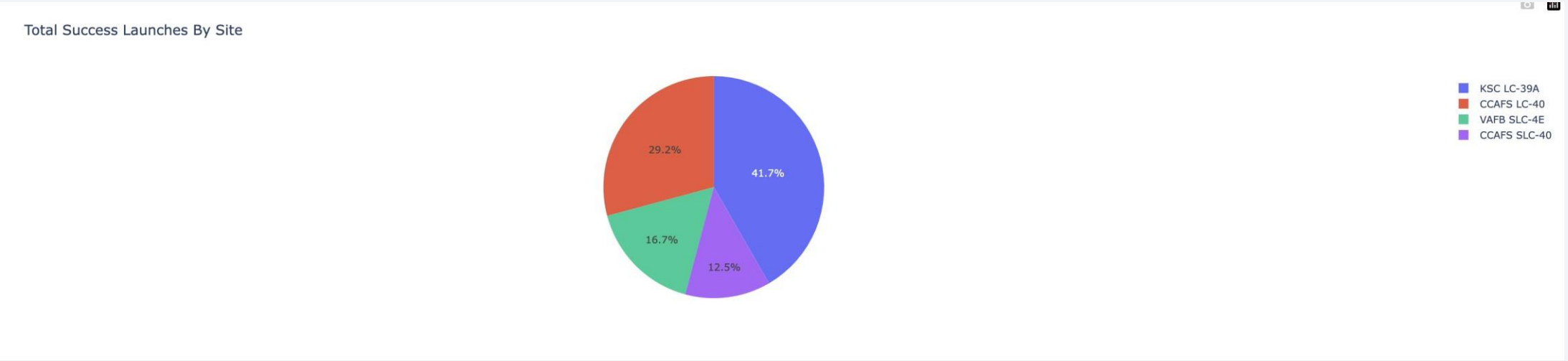
- Launch site KSC LC-39A, farthest from the city.
- In general, launch sites are closer to the coastline and farthest from the city.




Section 4
Build a Dashboard with Plotly Dash

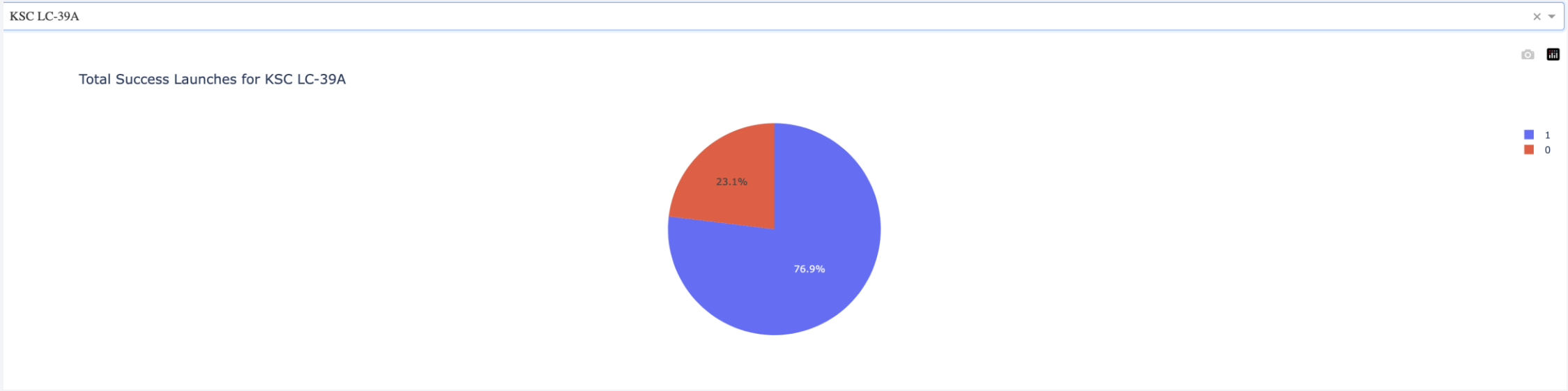
Rate of successful launches from all sites

- KSC LC-39A has highest percentage of successful launches



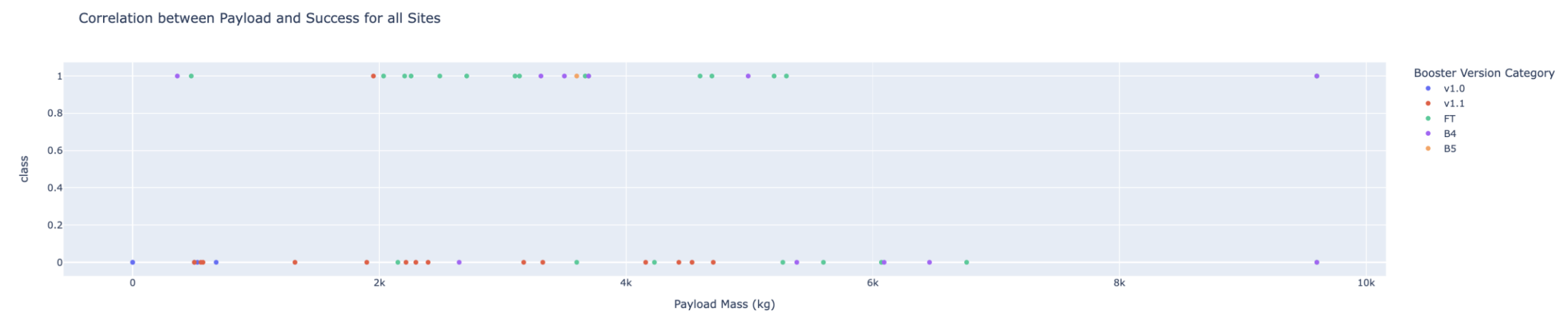
Launching rate at KSC LC-39A

- 76.9% successful launches from KSC LC-39A. This is the highest from all sites



Payload vs Launch Outcome for all sites

- High success rate observed with payload between 2k(kg) and 4k(kg). Booster version FT has highest success rate



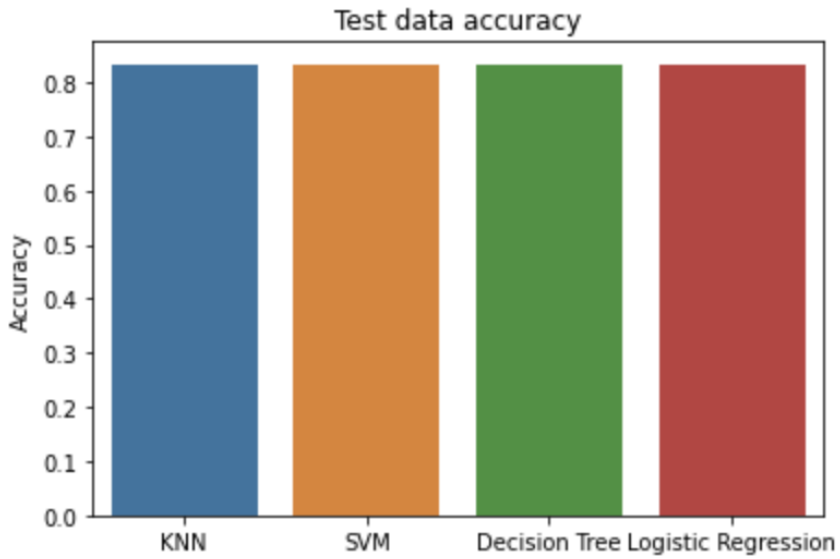
Section 5

Predictive Analysis (Classification)

Classification Accuracy

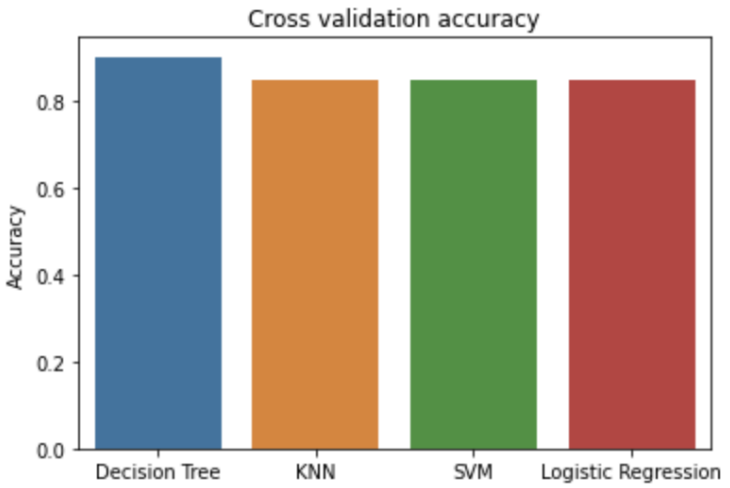
- With test data, all models have same accuracy.
- With cross validation, decision tree classifier has highest accuracy

```
Text(0.5, 1.0, 'Test data accuracy')
```



	Accuracy
KNN	0.833333
SVM	0.833333
Decision Tree	0.833333
Logistic Regression	0.833333

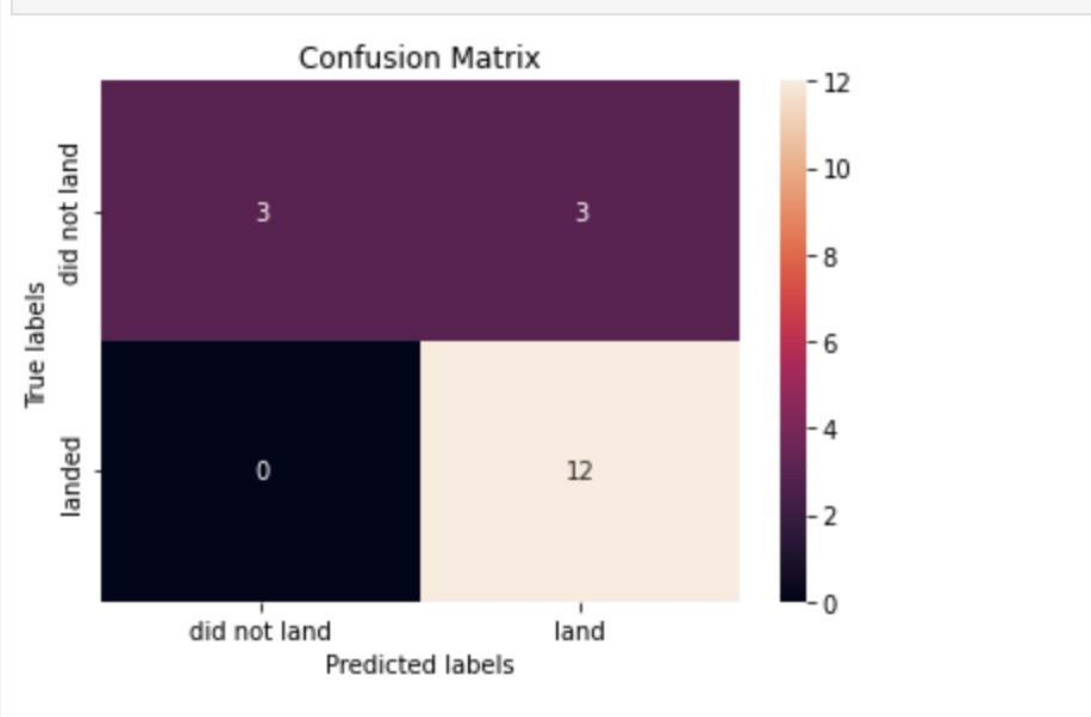
```
Text(0.5, 1.0, 'Cross validation accuracy')
```




```
KNN best score: 84.8%
tree best score: 90.0%
svm best score: 84.8%
logreg best score: 84.6%
```

Confusion Matrix

- All models have similar confusion matrix. Best model gives 3 false positives



Conclusions

- Success rate for rocket launches has been increasing from 2013 till 2020, a good indicator of progress in the 'field'
- Launch site KSC LC-39A has most successful launches
- Orbit types ES-L1, GEO, HEO, and SSO have best success rates
- For each launch site, success rate increases with increasing flight number
- Test data accuracy is the same for all four models

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

Thank you!