IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Bartosz Borkowski
4/13/2022

Outline

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

Executive Summary

- Summary of methodologies
  - Database was created using SpaceX API connection using Python and its scientific libraries
  - Exploratory Data analysis was done using:
    - Web scrapping library was used to extract data from Wikipedia
    - SQL database connection
  - Geographical data was presented using Folium Python library
  - Non-geographical data was presented using dashboards
  - Machine learning techniques for predicting Falcon 9 booster landing probability
- Summary of all results
  - Using ML techniques, it was proven that, although the beginnings were tought, now the customers can have high confidence in the low-cost access to space offered by SpaceX Falcon 9.

Introduction

- Project background and context

  SpaceX was founded by Elon Musk after failed attempt to buy soviet Dnepr ICBM rocket from Russia in 2001. He wanted to put an experimental greenhouse on Mars. SpaceX goal, as stated by Musk is to create self-sustaining Mars colony. To achieve it, there is a need to develop rapidly and fully reusable rockets. Falcon 9 is a first step to achieve this goal. Its first stage can land after stage separation on landing pad or on autonomous drone ship stationed offshore.

  Landing and reusing a booster can decrease launch cost significantly , but its still not perfect. This project's goal is to calculate a probability of a booster recovery.

- Problems you want to find answers
  - What are decisive factors for a successful landing?
  - Is there any relation between this factors that determines a successful landing?
  - If so, what is the combination with higher chance for a successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Launch data was collected using various methods: SpaceX API, web scrapping and SQL.
- Perform data wrangling
  - Data was processed using Python's scientific libraries like pandas and numpy. Not essential columns we removed, and missing data (e.g. Payload mass) was replaced with means.
- 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 was normalized , transformed and split to train and test sets and then fitted to several classification models.

Data Collection

- Data was collected using various methods and techniques using python libraries and SQL. One dataset was retrieved using SpaceX API via requests python library. Response was parsed to json file and saved as pandas DataFrame for easy access. Data from table was filtered – e.g. Nonessential data was dropped or unwanted characted were removed using predefined functions. Web scrapping was used to retrieve launch manifest from Wikipedia page. In this project popular web scrapping library – Beautiful Soup was used. Data was later converted to pandas DataFrame. To improve numerical data, missing information about payload mass were replaced by mean values.
- Information of successful landing by launch site, orbit were collected coded as binary information (series of 0's for failed attempt and 1's for successful one).

Data Collection – SpaceX API

SpaceX API usage: request method and creating a normalized Panda DataFrame from json response.

- GitHub link:

[Data_collection_notebook]()

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [41]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

In [42]:
```
response.status_code
```

Out[42]: 200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

In [43]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

In [44]:
```
# Get the head of the dataframe
data.head()
```

Out[44]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules
0	2006-03-	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason':	Engine failure at 33 seconds	[]	[]	[]

Data Collection - Scrapping

- I requested Falcon 9 Launch Wiki page from its URL, then collected all column names.
- All this extracted launch records was put into pandas DataFrame.

- GitHub link:

[Web_scraping]

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

Create a `BeautifulSoup` object from the HTML `response`

In [11]:
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, "html.parser")
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [12]:
```python
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, pleas lab

In [15]:
```python
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

In [16]:
```python
# Let's print the third table and check its content
```

In [81]:
```python
df=pd.DataFrame.from_dict(launch_dict)
df
```

Out[81]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

121 rows × 11 columns

Data Wrangling

- For all the launches information dataframe was created.
- Launches were grouped by launch complex, destination orbit, landing site.
- Github link: data_wrangling.

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

In [6]:
```
# Apply value_counts on Orbit column
df.Orbit.value_counts()
```

Out[6]:
```
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
```

TASK 3: Calculate the number and occurence of mission outcome per orbit type

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes`.Then assign it to a variable landing_outcomes.

In [9]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

Out[9]:
```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

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

In [10]:
```
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
```

We create a set of outcomes where the second stage did not land successfully:

In [11]:
```
bad_outcomes = set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Out[11]:
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

TASK 4: Create a landing outcome label from Outcome column

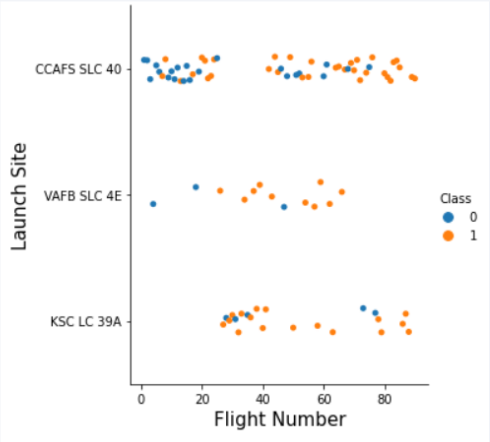
Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

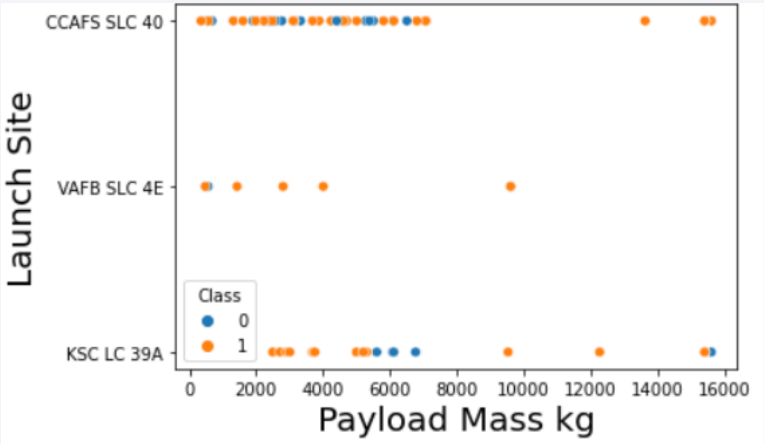
In [13]:
```
landing_class = df.apply(lambda row: 0 if row.Outcome in bad_outcomes else 1, axis=1)
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
```

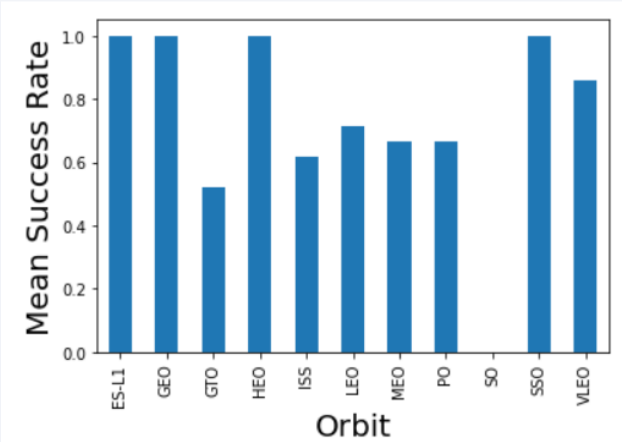
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the

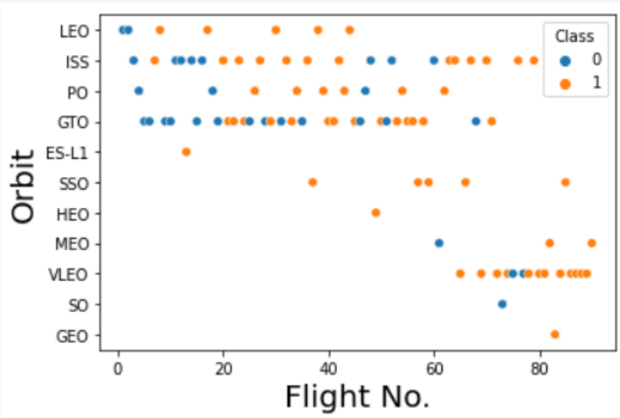
EDA with Data Visualization

- The plots used in this assessment:
  - Scatter plots for: payload mass vs. flight number, flight no. vs. launch site, payload mass vs. launch site, orbit vs. Success rate, flight no. vs orbit.
- Findings:
  - The success ratio of SpaceX launches increased in time.
  - Recent launches (flight numbers > 40) were more prone to success on CCAFS SLC 40 Launch Site
  - Fewer launches were done on VAFB SLC 4E than in any other launch site due to complex launch corridor.
  - GitHub URL:









EDA with SQL

- Using bullet point format, summarize the SQL queries you performed:

```
### Task 1
##### Display the names of the unique launch sites  in the space mission
%sql select Unique(LAUNCH_SITE) from SPACEXTBL;

### Task 2
##### Display 5 records where launch sites begin with the string 'CCA'
%sql SELECT Date, time, Booster_Version, LAUNCH_SITE from SPACEXTBL where LAUNCH_SITE LIKE ('CCA%')
LIMIT 5;

### Task 3
##### Display the total payload mass carried by boosters launched by NASA (CRS)
%sql select sum(PAYLOAD_MASS__KG_) as payloadmasskg from SPACEXTBL where Customer = ‘NASA (CRS)’;

### Task 4
##### Display average payload mass carried by booster version F9 v1.1
%sql select avg(PAYLOAD_MASS__KG_) as payloadmasskg from SPACEXTBL where booster_version = 'F9 v1.1';

### Task 5
##### List the date when the first successful landing outcome in ground pad was acheived.

*Hint:Use min function*
sql select min(DATE) as mindate from SPACEXTBL where landing_outcome like '%ground pad%';
```

```
### 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 from SPACEXTBL where LANDING__OUTCOME='Success (drone ship)' and
PAYLOAD_MASS__KG BETWEEN 4001 and 5999;

### Task 7
##### List the total number of successful and failure mission outcomes
%sql select MISSION_OUTCOME , count(*) as missionoutcomes from SPACEXTBL GROUP BY
MISSION_OUTCOME

### Task 8
##### List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
%sql select BOOSTER_VERSION  from SPACEXTBL where PAYLOAD_MASS__KG_=(select
max(PAYLOAD_MASS__KG_) from SPACEXTBL);

### Task 9
##### List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year
2015
%sql SELECT MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where EXTRACT(YEAR
FROM DATE)='2015' and landing_outcome like '%Failure%';

### Task 10
##### Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the
date 2010-06-04 and 2017-03-20, in descending order
%sql SELECT LANDING_OUTCOME from SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
ORDER BY DATE DESC;
```


- GitHub URL: [EDA with SQL]

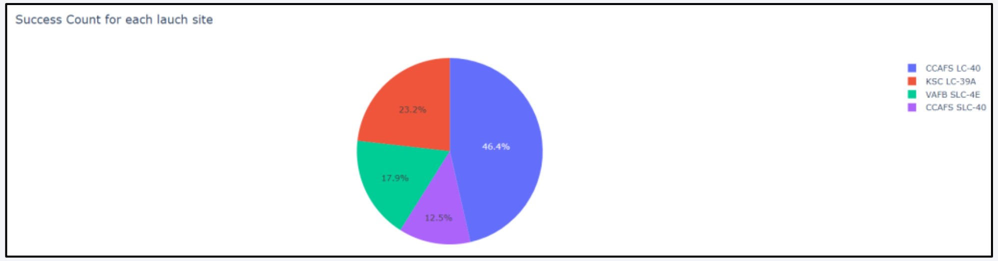
Build an Interactive Map with Folium

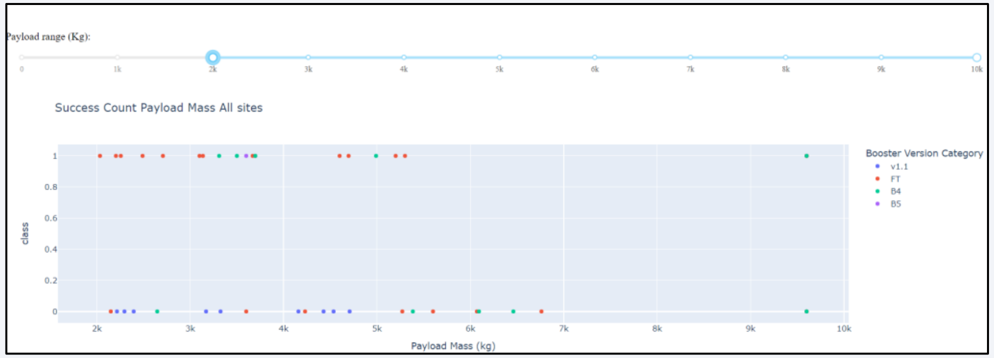
- Using Folium Map all the launch sites used by SpaceX were added to the map using translucent circles with text labels. All individual launches were added color-coded by success outcome. Lines denoting calculated distances from launch sites to highways, railroads and closest city. And by using clusters, it was shown which launch site had higher success rate.

- **GitHub URL:** SpaceX_Folium

Build a Dashboard with Plotly Dash

- Plotly/Dash dashboard was used as an overview tool for all previously gathered information about Falcon 9 launches:
  - There is a dropdown element with Launch sites for dynamic filtering
  - Pie chart shows launch count for all launch sites . When specific site is selected it presents Success vs. Failure of booster landing
  - Slider element is present to filter by payload mass.
  - Scatter plot is present to inform about payload mass vs. launch success





- GitHub URL: [dashboard.py]

Predictive Analysis (Classification)

- Launch data was loaded to pandas dataframe with normalized and encoded information. Dataset was split to train and test data using sklearn method.
- Each classification model used: Logistics Regression, SVM, Decision Trees, KNN were fitted using various parameters using GridSearchCV. Finally best parameters based on score and accuracy were used for test set.

- GitHub URL: SpaceX_predictions

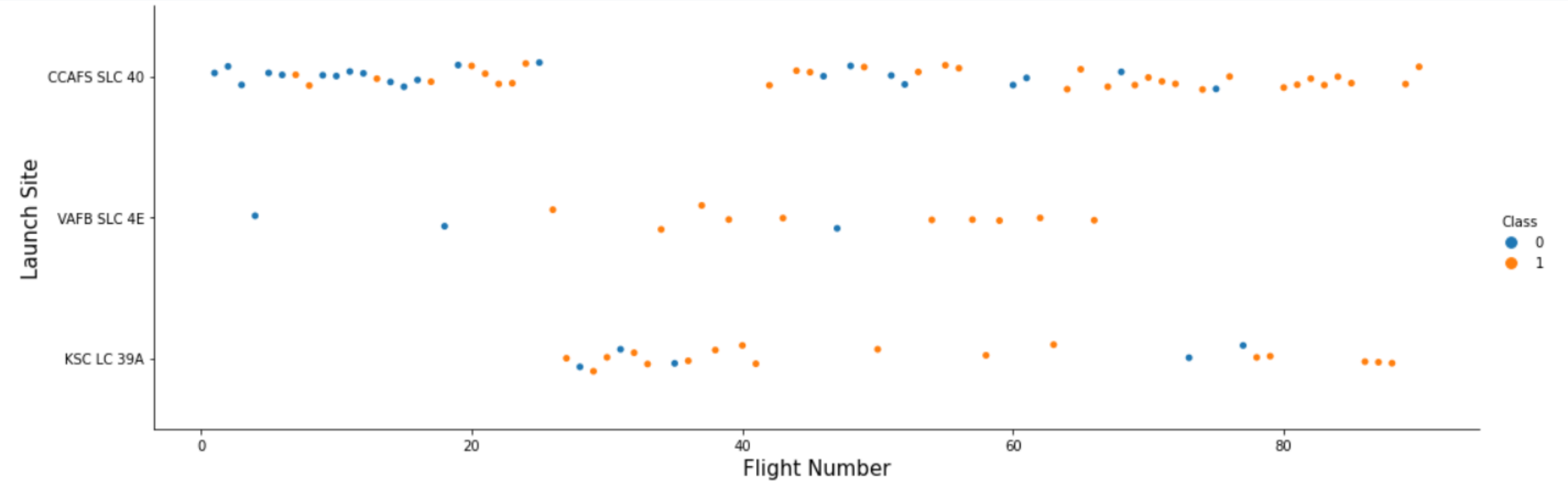
Results

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

Section 2

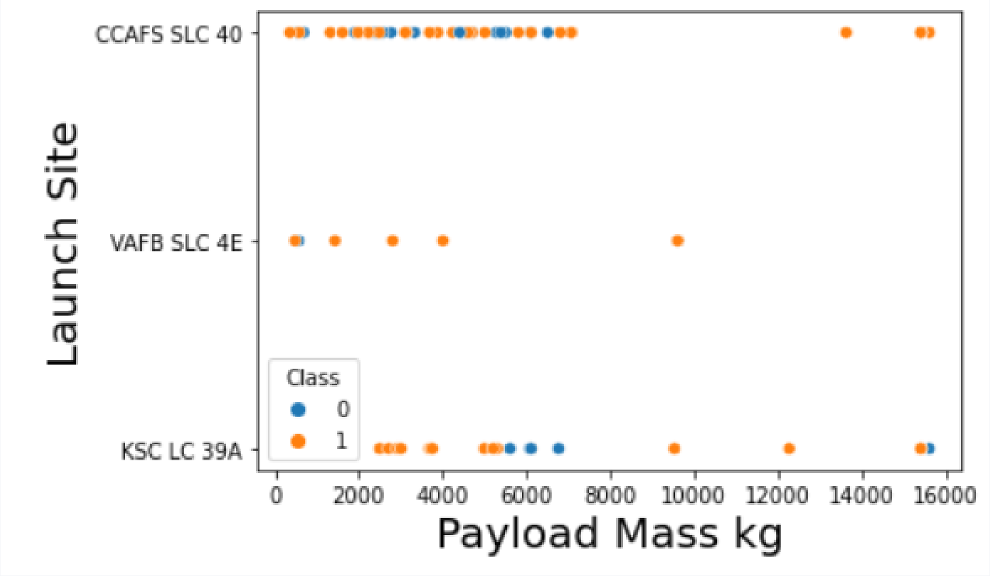
Insights drawn from EDA

Flight Number vs. Launch Site



- SpaceX is improving successful landing probability over time.
- Launchpad SLC-40 was used almost for all launches till planned flight 29 when it was partially destroyed due to COPV malfunction before static fire of the engines.
- Launch site in Vandenberg Airforce Base on west coast is used rarely due to retrograde orbit – mostly for military payloads.
- After launchpad SLC-40 was rebuilt, it was again hevily used by SpaceX.

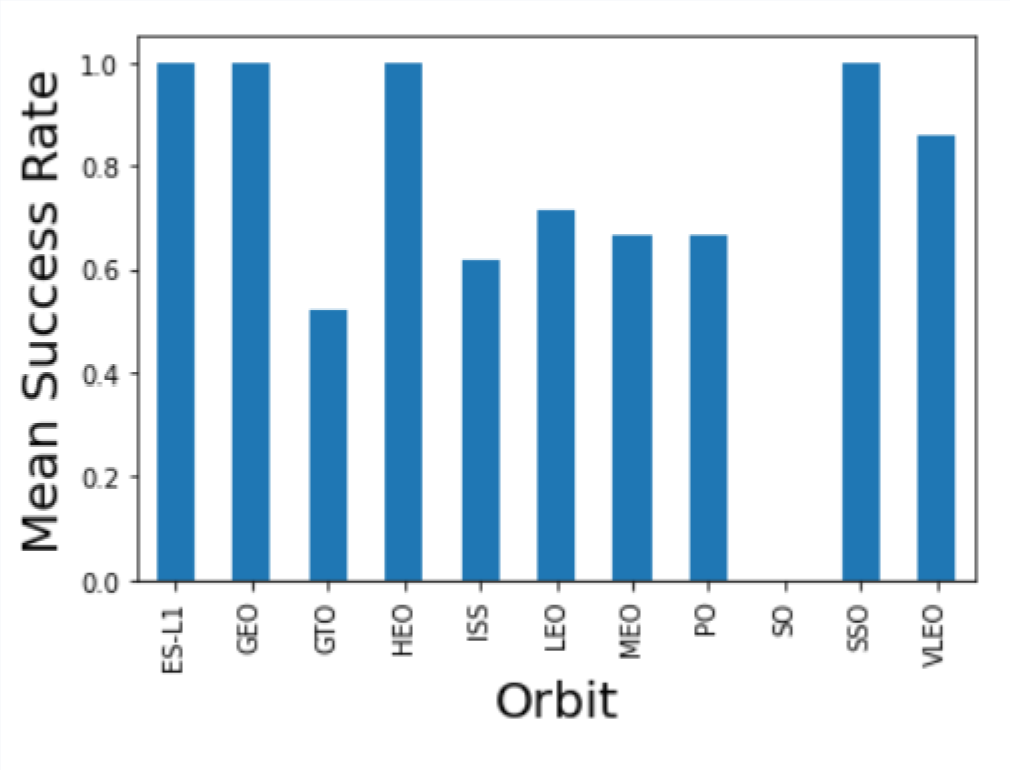
Payload vs. Launch Site



- For the VAFB-SLC launch site there are no heavy payloads launched due to retrograde orbit mass restriction.
- The greater the payload for CCAFS SLC 40 higher the success rate.

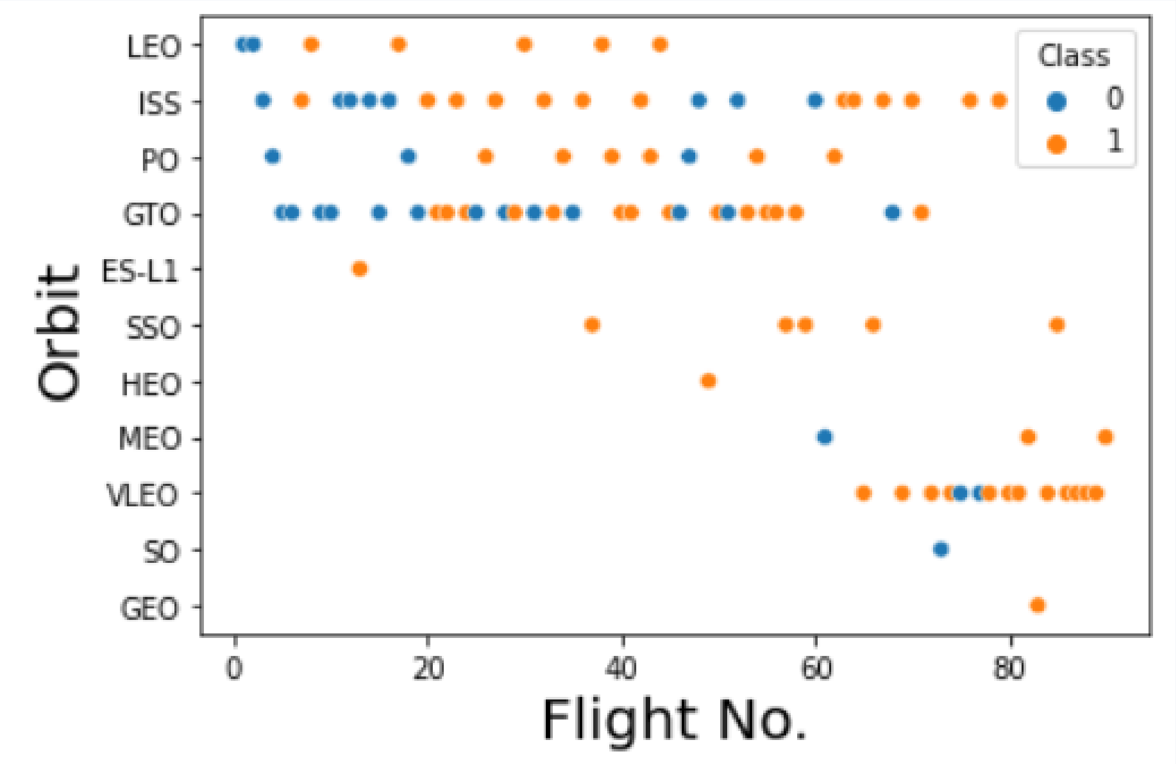
Success Rate vs. Orbit Type

- Most of the payloads are placed in LEO and GTO, therefore low sample orbitslike SO of Lagrange Point transfer orbit shows very high or very low success/failure rate.



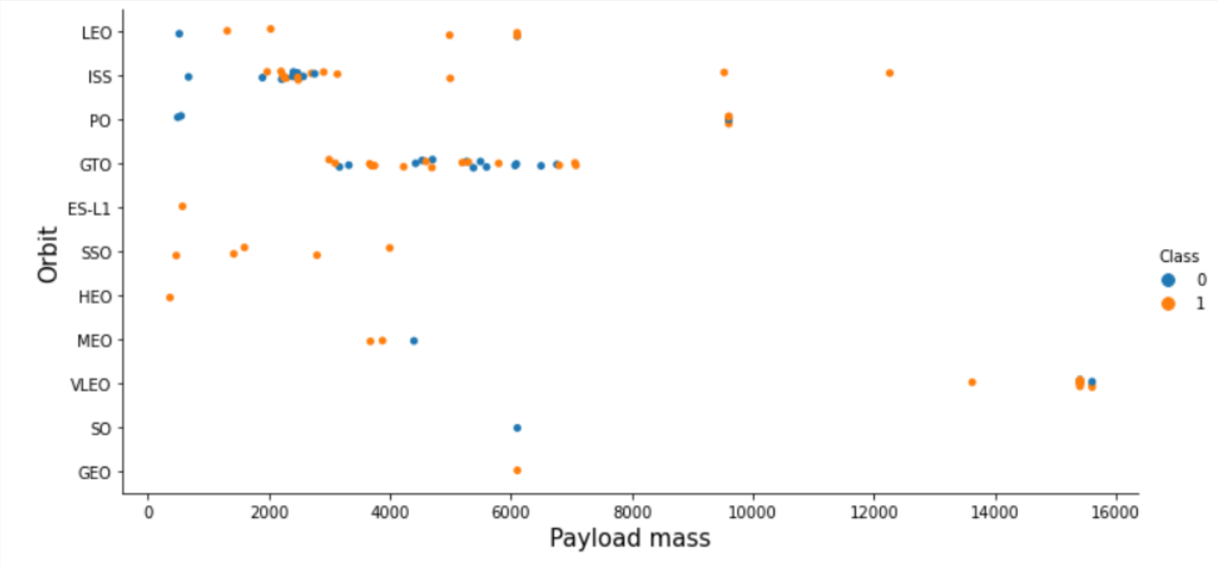
Flight Number vs. Orbit Type

- Based of flight number vs orbit plot it can be observed that success rate improved with later flights, but it is inconclusive if orbit type influences the success rate.



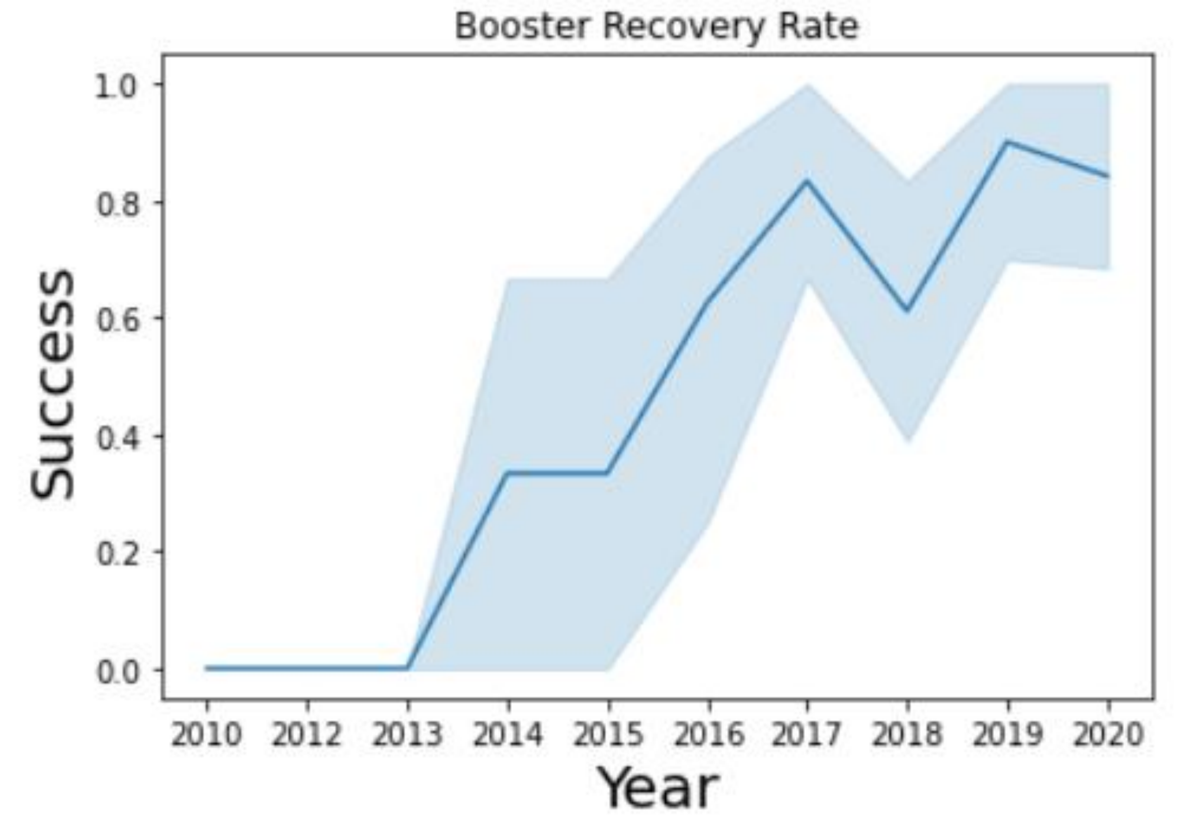
Payload vs. Orbit Type

- ISS orbit shows a cluster of points around 2000 kg, with 3 outliers highly above this value. Maximum payload capability of Dragon capsule is 6000 kg , so most likely higher values are errorous.
- There is an increase of the successful landing accompanied by an increase on the payload mass for the orbits PO, LEO and ISS



Launch Success Yearly Trend

- From the start of booster recovery program in 2013 there was a constantly increase of the success rate
- There is a decline in this trend in 2018 but it is recovered in 2019.



All Launch Site Names

- Data selection using SQL using the function DISTINCT

```
%sql select Unique(LAUNCH_SITE) from SPACEXTBL;
```

Out[6]:

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

Launch Site Names Begin with 'CCA'

%sql SELECT Date, time, Booster_Version, LAUNCH_SITE from SPACEXTBL where LAUNCH_SITE LIKE ('CCA%') LIMIT 5;

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

%sql select sum(PAYLOAD_MASS__KG_) as total_payload_mass_kg from SPACEXTBL where Customer = ‘NASA (CRS)’;

Out[15]:

total_payload_mass_kg
45596

Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as avg_payload_mass_kg from SPACEXTBL where booster_version = 'F9 v1.1';
```

Out[16]:

avg_payload_mass_kg
2928

First Successful Ground Landing Date

```
%sql select min(DATE) as mindate from SPACEXTBL where landing_outcome like '%ground pad%';
```

Out[20]:

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

%sql select BOOSTER_VERSION from SPACEXTBL where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG BETWEEN 4001 and 5999;

Out[21]:

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

Total Number of Successful and Failure Mission Outcomes

%sql select MISSION_OUTCOME , count(*) as mission_outcome from SPACEXTBL GROUP BY MISSION_OUTCOME

Out[23]:

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

Boosters Carried Maximum Payload

```
%sql select BOOSTER_VERSION  from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

Out[26]:

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

%sql SELECT MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where EXTRACT(YEAR FROM DATE)='2015' and landing_outcome like '%Failure%';

Out[32]:

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

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

- %sql SELECT LANDING__OUTCOME, COUNT(*), counter from SPACEXTBL WHERE landing__outcome in ('Failure (drone ship)', 'Success (ground pad)') and date BETWEEN '2010-06-04' and '2017-03-20' group by landing__outcome order by counter desc;

Out[32]:

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

Section 3

Launch Sites Proximities Analysis

Launch sites locations

All SpaceX launch sites are situated on the northern hemisphere at USA on its west coast in California and on its east coast in Florida. There is a higher success rate for the launches in Florida Cape Canaveral sites. Most of the rockets launches from Florida due to prograde orbit.

The closer launch pad is to equator the higher boost effect from Earth’s rotation.

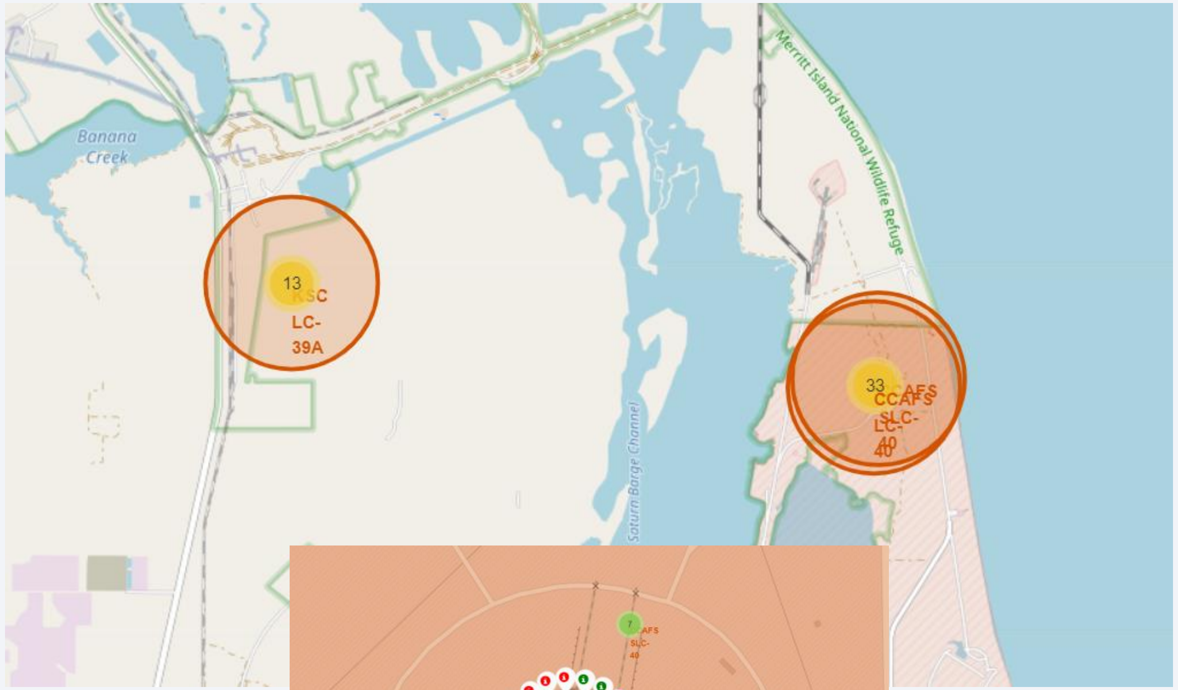
All launches in USA are performed in direction of the ocean for populations safety.

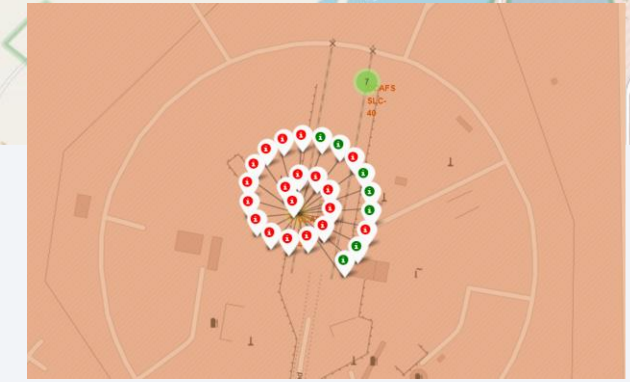
Florida's launch sites

```
In [45]: # Function to assign color to launch outcome
         def assign_marker_color(launch_outcome):
             if launch_outcome == 1:
                 return 'green'
             else:
                 return 'red'

         spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)
         spacex_df.tail(10)
```

Out[45]:

	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red
53	CCAFS SLC-40	28.563197	-80.576820	0	red
54	CCAFS SLC-40	28.563197	-80.576820	1	green
55	CCAFS SLC-40	28.563197	-80.576820	0	red

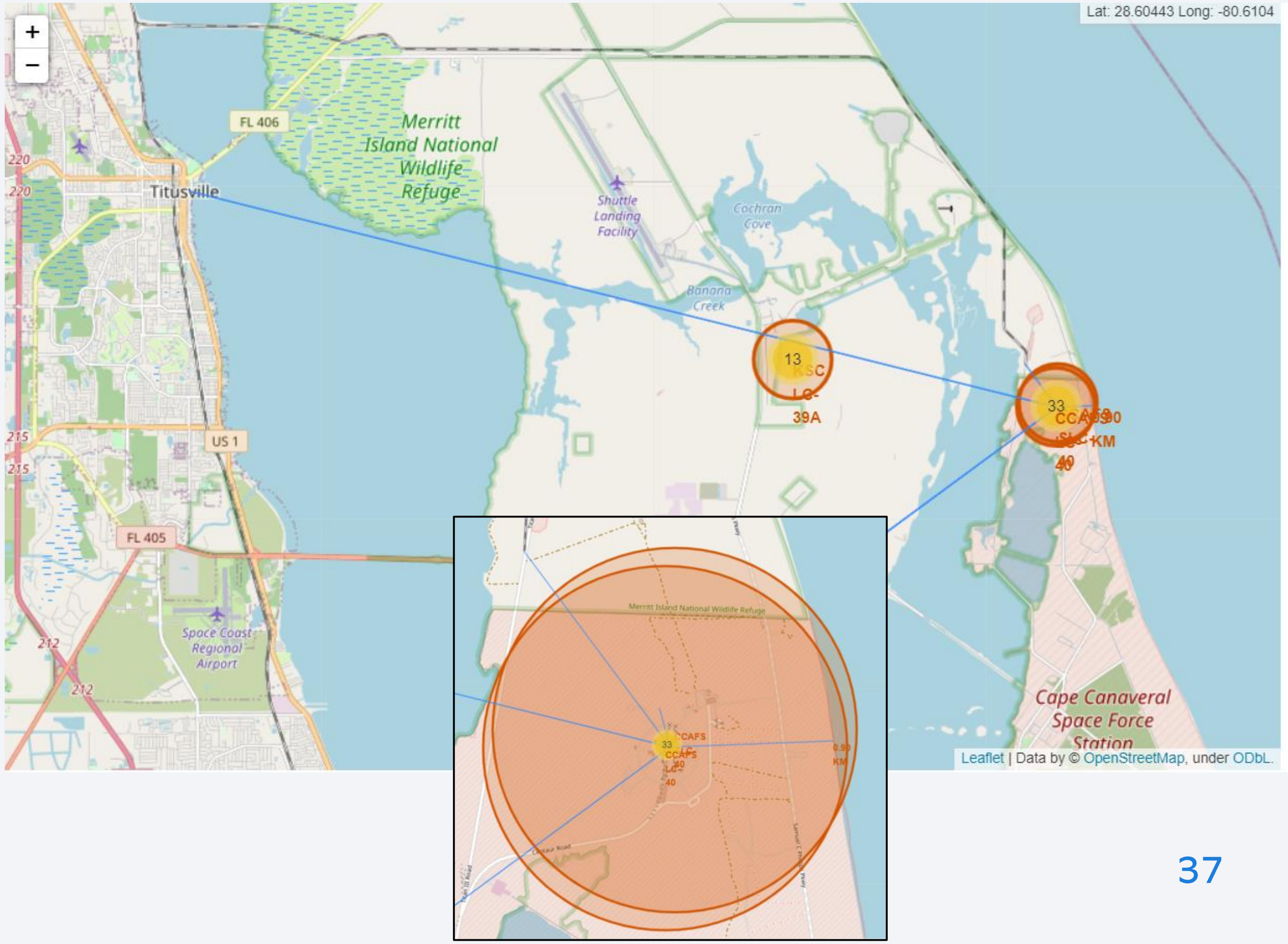




Florida's launch sites

In [55]:
```python
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the marker to the launch site

#city
lat3 = 28.61229
long3 = -80.80611

#railway
lat4 = 28.57206
long4 =-80.58533

#highway
lat5 = 28.52538
long5 = -80.63431
```

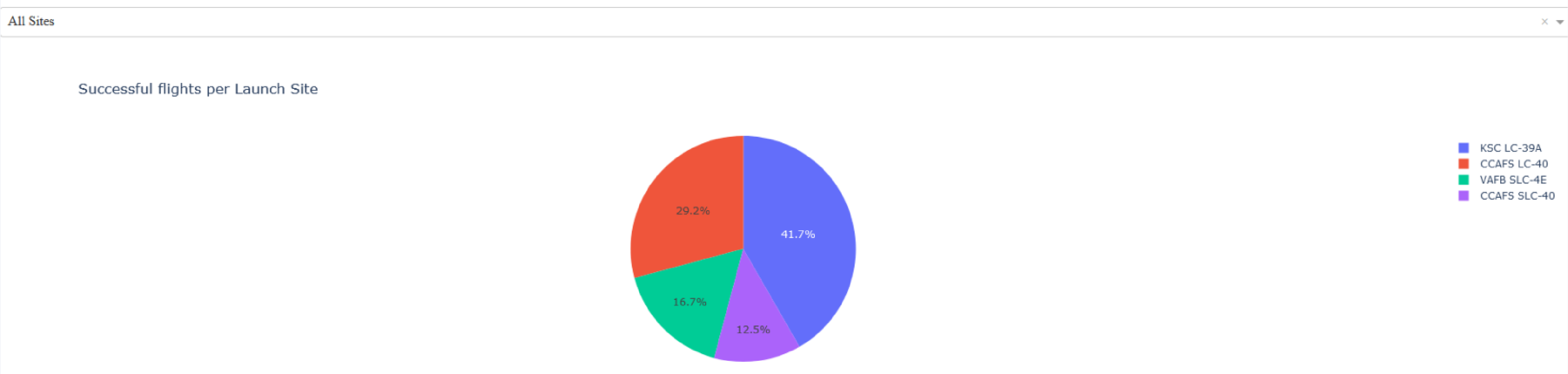
In [56]:
```python
lines=folium.PolyLine(locations=([lat1, long1], [lat3, long3]), weight=1)
site_map.add_child(lines)
lines=folium.PolyLine(locations=([lat1, long1], [lat4, long4]), weight=1)
site_map.add_child(lines)
lines=folium.PolyLine(locations=([lat1, long1], [lat5, long5]), weight=1)
site_map.add_child(lines)
```



Section 4

Build a Dashboard with Plotly Dash

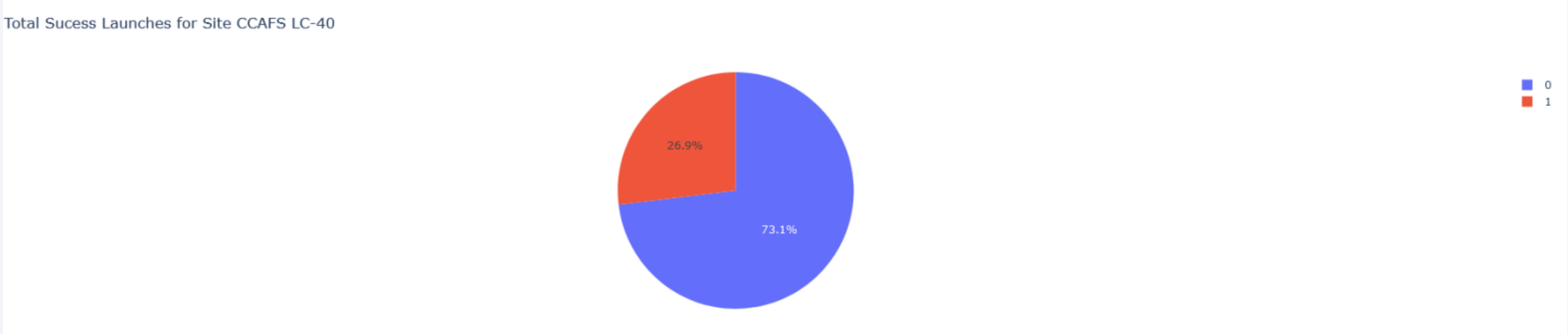
All launch sites success rate



Dropdown element is set to ALL (label: “All sites”), used as input on callback to pie chart showing the successful flights per Launch. The pie chart allows us to visualize the success rate between launch sites.

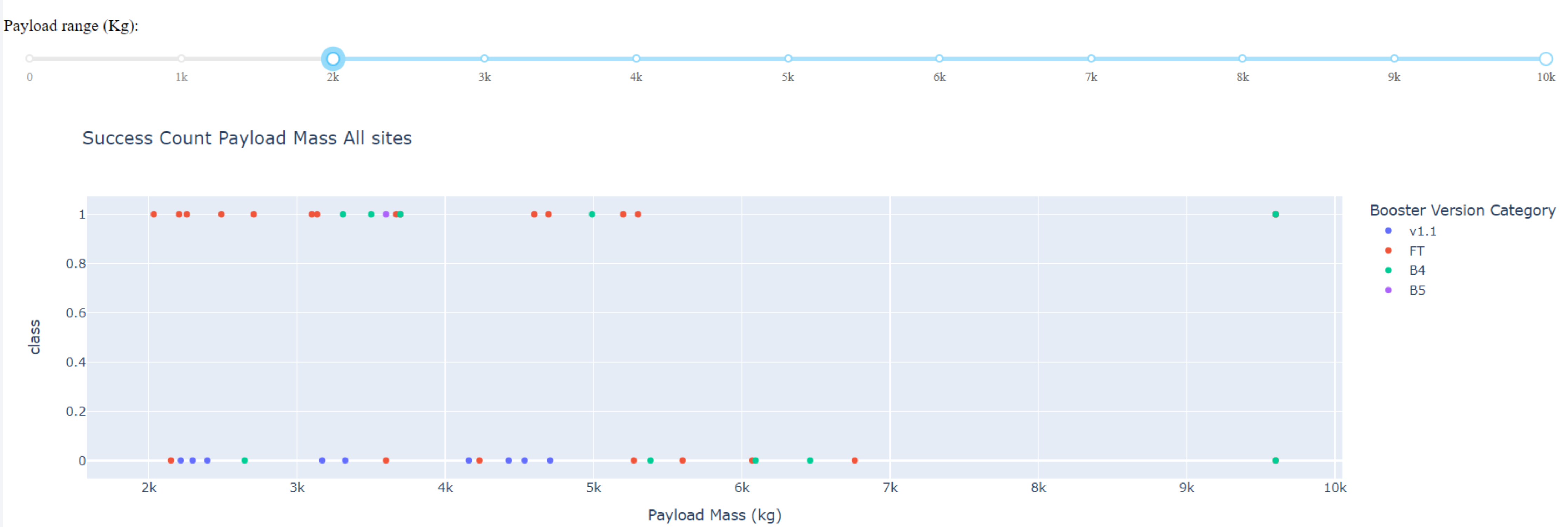
Most successful landings were for rockets launched from historic SLC-39A

Successful landing from CCAFS SLC-40



- Dropdown element is set to LC-40, used as input on callback to pie chart showing the successful flights per Launch. The pie chart allows us to visualize the success rate for particular site.

Payload vs. Launch Outcome



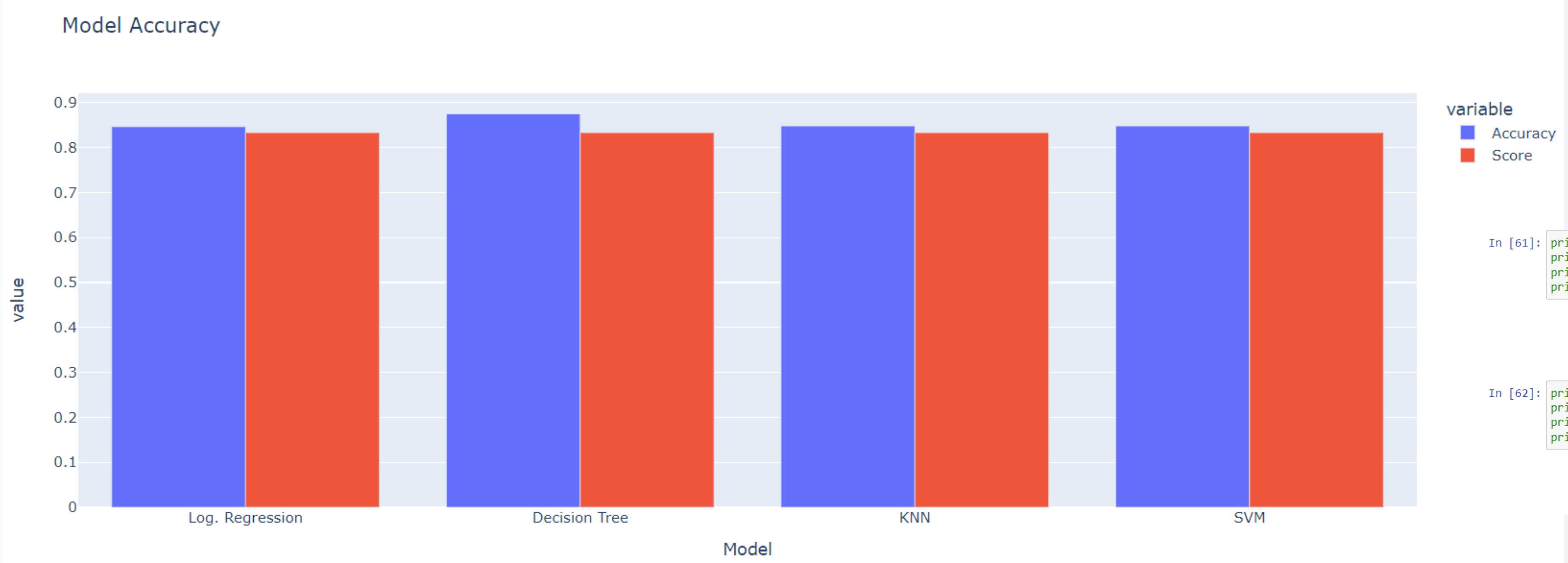
Range and selection were used as input on callback app. This plot showing Payload Mass (kg) vs Success rate for all launch sites. Where we can identify:

- Which payload ranges have the highest launch success rate? 2500 - 5500
- Which payload ranges have the lowest launch success rate? 2000 - 3000
- Full thrust is most successful booster version of Falcon 9

Section 5

Predictive Analysis (Classification)

Classification Accuracy

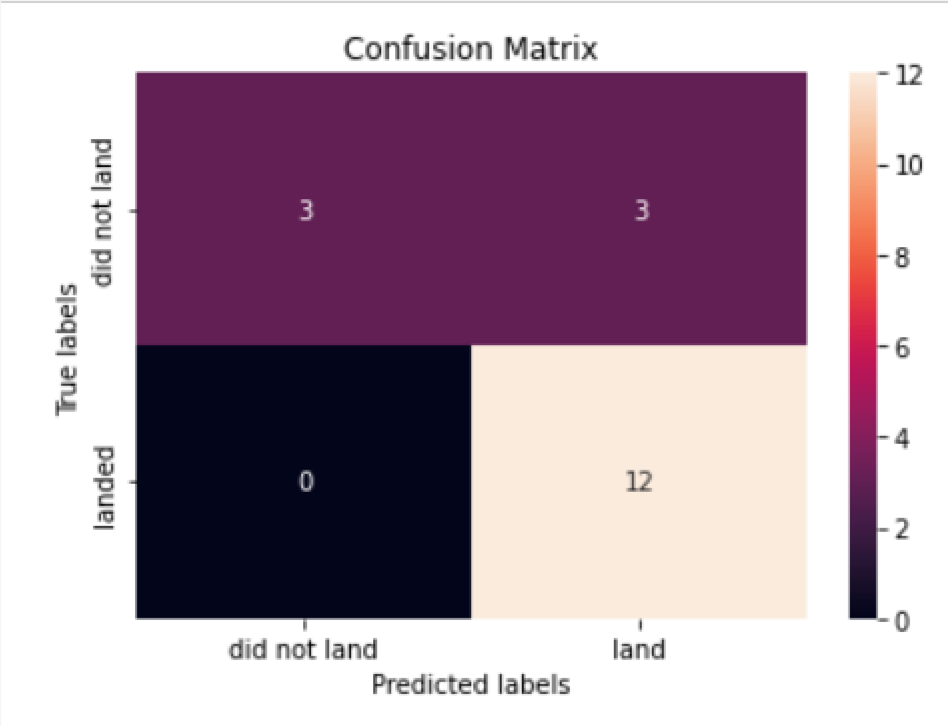



```
In [61]: print(knn_score)
         print(tree_score)
         print(svm_score)
         print(lr_score)

0.8333333333333334
0.8333333333333334
0.8333333333333334
0.8333333333333334

In [62]: print(f'Training sample accuracy for Log Regression {logreg_cv.best_score_}')
         print(f'Training sample accuracy for Decision Tree {tree_cv.best_score_}')
         print(f'Training sample accuracy for KNN {knn_cv.best_score_}')
         print(f'Training sample accuracy for SVM {svm_cv.best_score_}')

Training sample accuracy for Log Regression 0.8464285714285713
Training sample accuracy for Decision Tree 0.875
Training sample accuracy for KNN 0.8482142857142858
Training sample accuracy for SVM 0.8482142857142856
```

- Best score in this assessment acquired decision tree model with 0.875
- The accuracy for all models was: 0.8333

Confusion Matrix

- As Decision Tree was the best performing model its Confusion Matrix shows two classes very distinguished.
- Type 1 errors (False positives) are still quite an issue with this model.



Conclusions

- Over time starting from 2013 SpaceX is gradually more successful with booster landing for all scenarios with slight decrease in 2018
- Largest number of successful launches were found at KSC LC-39A site with over 41%
- Latest iteration of booster design – Full Thrust is most reliable in terms of landing success rate.
- Prediction model with highest score is Decision Tree, with 0.8875.
- The accuracy for
- Accuracy for all models was: 0.8333
- The average payload mass carried by booster version F9 v1.1 was 2928 kg

Appendix

- GIT repository: link

Thank you!