IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Wael Nabil Abdelfattah

19th of Sep 2024

Outline

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

Executive Summary

- Summary of methodologies
  - SpaceX Data Collection using:
    - SpaceX API
    - Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA)
  - EDA DataViz using:
    - Python pandas
    - Matplotlib
  - SpaceX Interactive Visual Analytics with Folium
  - SpaceX Interactive Dashboard with Ploty Dash
- Summary of all results
  - Data Visualization and Dashboard.
  - Predictive Analysis results

Introduction

- Project background and context
  - we will predict if the Falcon 9 first stage will land successfully. 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.
- Problems you want to find answers
  - 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.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Using SpaceX API
    - Using Web Scraping to collect data from a Wikipedia page titled `List of Falcon 9 and Falcon Heavy launches.
- Perform data wrangling
    - perform EDA to find some patterns in the data and determine what would be the label for training supervised models.
    - In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed, for example, 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 on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.
    - In this lab we will mainly convert those outcomes into Training Labels with `1` means the booster successfully landed `0` means it was unsuccessful. How to build, tune, evaluate classification models

Methodology

Executive Summary

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

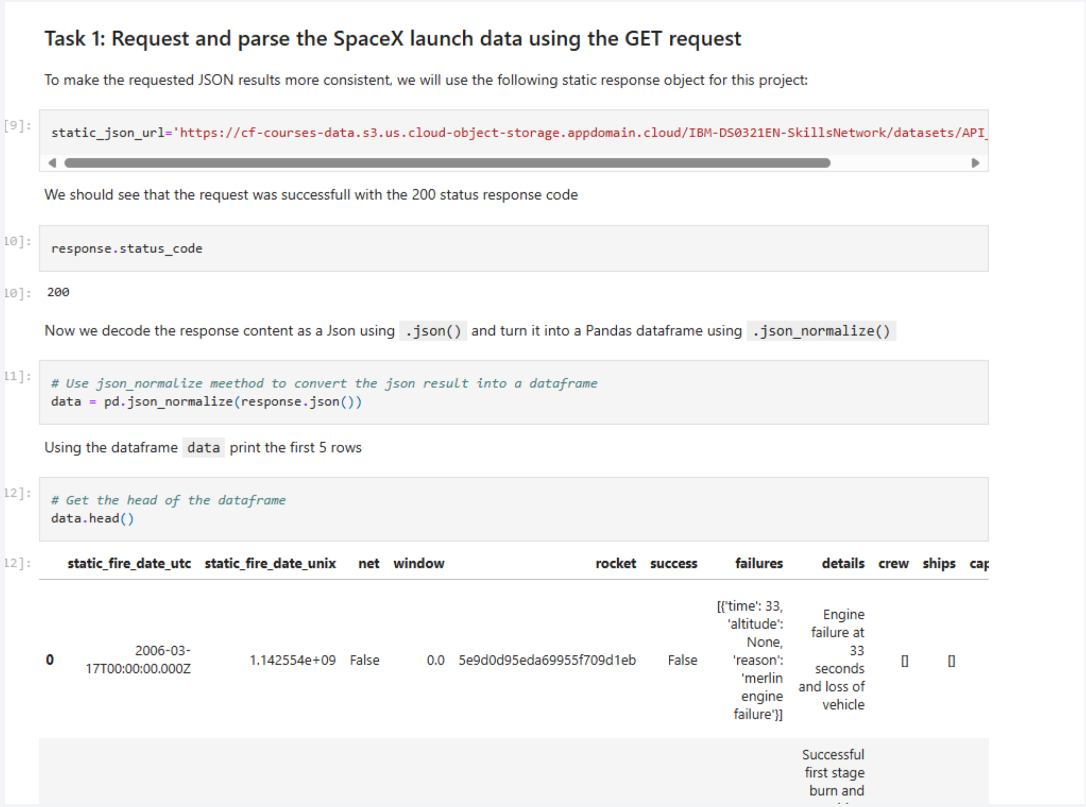
Data Collection

- Describe how data sets were collected.
  - Using SpaceX API
  - Using Web Scraping to collect data from a Wikipedia page titled `List of Falcon 9 and Falcon Heavy launches
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- GitHub URL of the completed SpaceX API calls notebook https://github.com/Wael-Nabil/spacex/blob/main/01-jupyter-labs-spacex-data-collection-api.ipynb



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:

[9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
```

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

10]:
```
response.status_code
```

10]: 200

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

11]:
```
# 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

12]:
```
# Get the head of the dataframe
data.head()
```

12]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	cap
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	
								Successful first stage burn and			

Data Collection - Scraping

- Web scrap Falcon 9 launch records with BeautifulSoup:
  - Extract a Falcon 9 launch records HTML table from Wikipedia
  - Parse the table and convert it into a Pandas data frame
- GitHub URL [spacex/02-jupyter-labs-webscraping.ipynb at main · Wael-Nabil/spacex · GitHub]()

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 [4]:
```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 [5]:
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, 'html.parser')
```

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

In [6]:
```python
# Use soup.title attribute
soup.title
```

Out[6]:
```
<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`, please check the external reference link towards the end of this lab

In [7]:
```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 [8]:
```python
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
```

Data Wrangling

- we will perform some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, 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 on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.
- In this lab we will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.
- Falcon 9 first stage will land successfully.
- Add the GitHub URL [spacex/03-labs-jupyter-spacex-Data wrangling.ipynb at main · Wael-Nabil/spacex · GitHub]

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

```
# Apply value_counts() on column LaunchSite
LaunchSite_count=df.value_counts('LaunchSite')
LaunchSite_count
```

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

TASK 2: Calculate the number and occurrence of each orbit

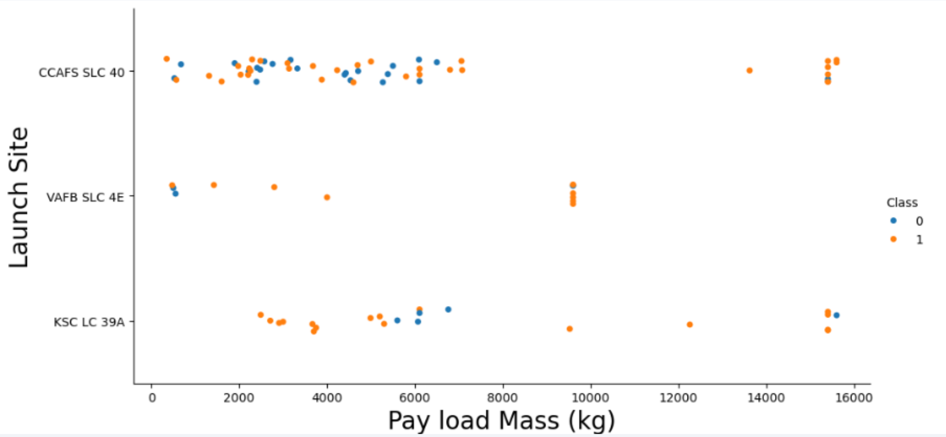
Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

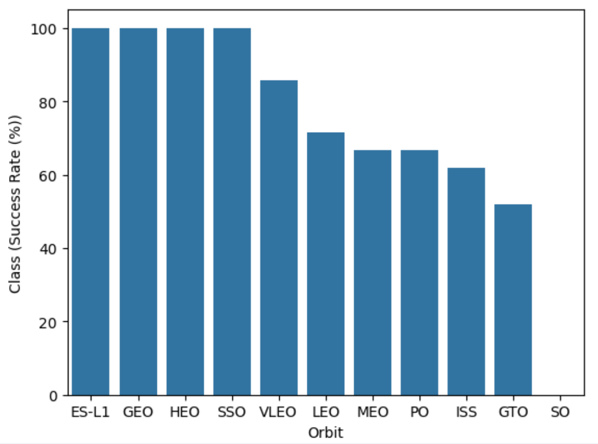
```
# Apply value_counts on Orbit column
orbit_count=df.value_counts("Orbit")
orbit_count
```

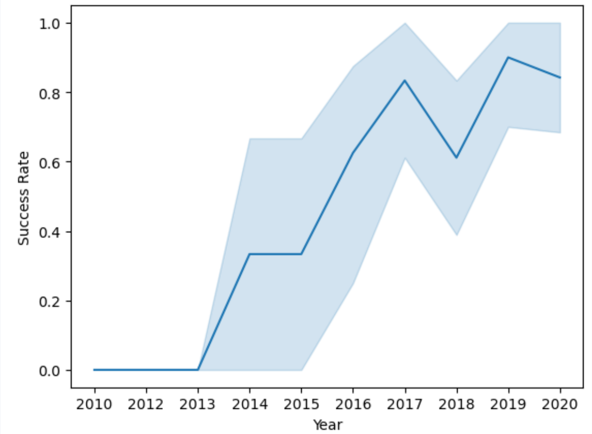
```
Orbit
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
GEO       1
HEO       1
SO        1
dtype: int64
```

EDA with Data Visualization

- Summarize what charts were plotted
  - scatter point chart FlightNumber vs. PayloadMassand overlay the outcome of the launch.
  - scatter point chart FlightNumber vs LaunchSite
  - Visualize the relationship between Payload Mass and Launch Site
  - bar chart for the success rate of each orbit
  - Line Chart to visualize the launch success yearly trend
- GitHub URL [spacex/05-edadataviz.ipynb at main · Wael-Nabil/spacex · GitHub]()







EDA with SQL

- SQL queries performed:
  - Display the names of the unique launch sites in the space mission
    - %sql select distinct Launch_site from SPACEXTABLE
  - Display 5 records where launch sites begin with the string 'CCA’
    - %sql select * from SPACEXTABLE where Launch_site like 'CCA%' limit 5
  - Display the total payload mass carried by boosters launched by NASA (CRS)
    - %sql select Sum(PAYLOAD_MASS__KG_), Customer from SPACEXTABLE where Customer = 'NASA (CRS)’
  - Display average payload mass carried by booster version F9 v1.1
    - %sql select avg(PAYLOAD_MASS__KG_), Booster_Version from SPACEXTABLE where Booster_Version like 'F9 v1.1%’
  - **List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.**
    - %sql SELECT substr(Date,1,4) as 'Year', substr(Date, 6, 2) as 'Month' ,Booster_Version, Launch_Site, Payload, PAYLOAD_MASS__KG_, Mission_Outcome, Landing_Outcome FROM SPACEXTABLE WHERE substr(Date,1,4)='2015' AND Landing_Outcome = 'Failure (drone ship)'

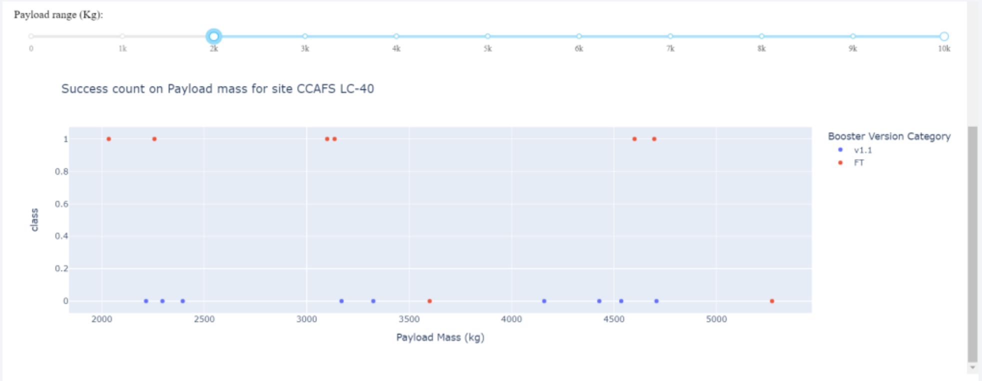
- GitHub URL [spacex/04-jupyter-labs-eda-sql-coursera_sqllite.ipynb at main · Wael-Nabil/spacex · GitHub]

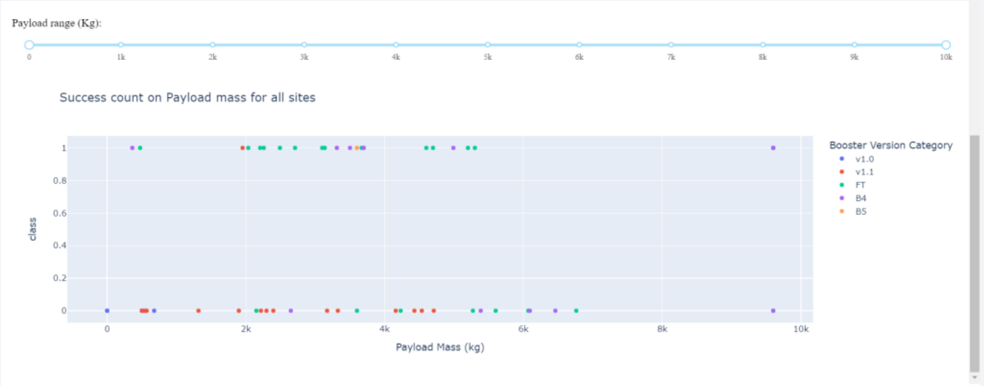
Build an Interactive Map with Folium

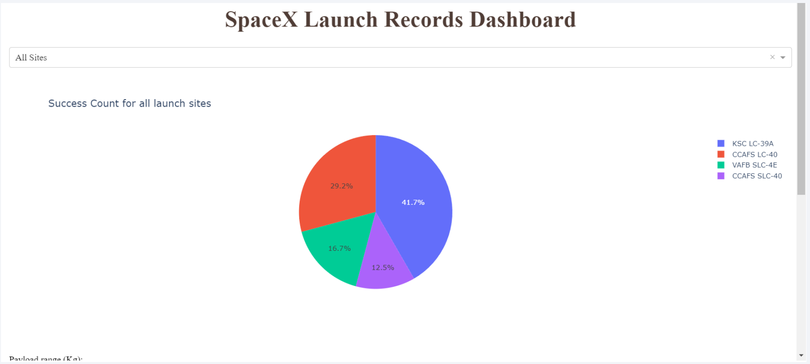
- folium map contains markers, circles and lines
- Those Objects are to mark success and failure for each launch site.
- GitHub URL spacex/06-lab_jupyter_launch_site_location.ipynb at main · Wael-Nabil/spacex · GitHub

Build a Dashboard with Plotly Dash

- what plots/graphs and interactions added to a dashboard:
  - dropdown list to enable Launch Site selection (default select value is for ALL sites)
  - slider to select payload range.
  - pie chart to show the total successful launches count for all sites.
  - scatter chart to show the correlation between payload and launch success
- GitHub URL [spacex/07-spacex_dash_app.ipynb at main · Wael-Nabil/spacex · GitHub]()







Predictive Analysis (Classification)

- Perform exploratory Data Analysis and determine Training Labels
    - create a column for the class
    - Standardize the data
    - Split into training data and test data
- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
    - Find the method performs best using test data
- Create a NumPy array from the column Class in data,.
- Standardize the data in X then reassign it to the variable X using the transform
- split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2
- Create a logistic regression object then create a GridSearchCV object logreg_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.
- Calculate the accuracy on the test data using the method score
- GitHub URL [spacex/SpaceX_Machine Learning Prediction_Part_5.ipynb at main · Wael-Nabil/spacex · GitHub]()

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

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

```
dtype('int64')
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

```
# students get this
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
X
```

```
array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.67441914e+00, -1.19523159e+00, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.63592675e+00, -1.16267307e+00, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       ...,
       [ 1.63592675e+00,  1.99100483e+00,  3.49060516e+00, ...,
         1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
       [ 1.67441914e+00,  1.99100483e+00,  1.00389436e+00, ...,
         1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
       [ 1.71291154e+00, -5.19213966e-01, -6.53912840e-01, ...,
        -8.35531692e-01, -5.17306132e-01,  5.17306132e-01]])
```

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

```
print ('Train set:')
print('X-train= ', X_train.shape,'Y-train= ',Y_train.shape)
print ('Test set:')
print('X-test= ', X_test.shape,  'Y-test= ',Y_test.shape)
```

```
Train set:
X-train=  (72, 83) Y-train=  (72,)
Test set:
X-test=  (18, 83) Y-test=  (18,)
```

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
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']})
```

TASK 5

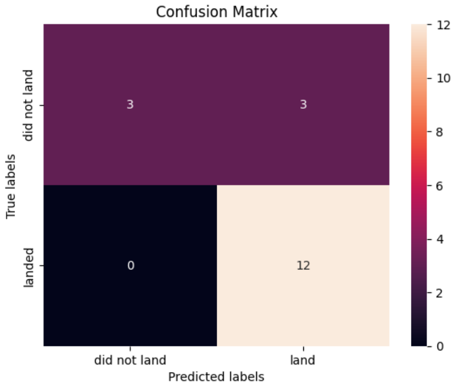
Calculate the accuracy on the test data using the method `score` :

```
logreg_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```

Lets look at the confusion matrix:

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



Results

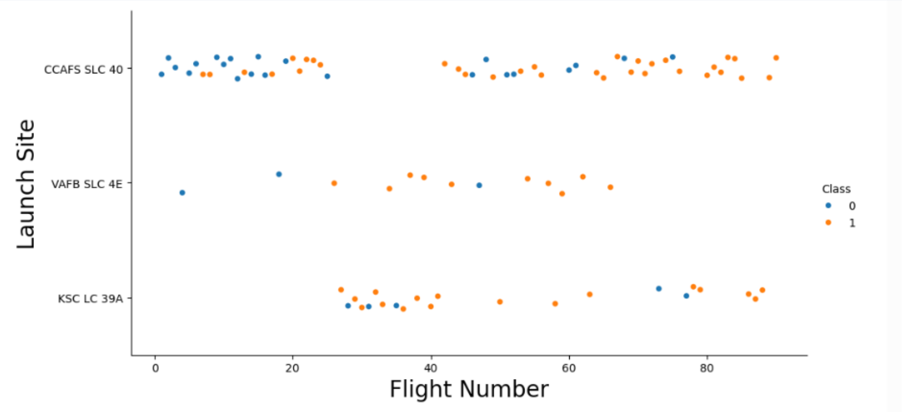
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

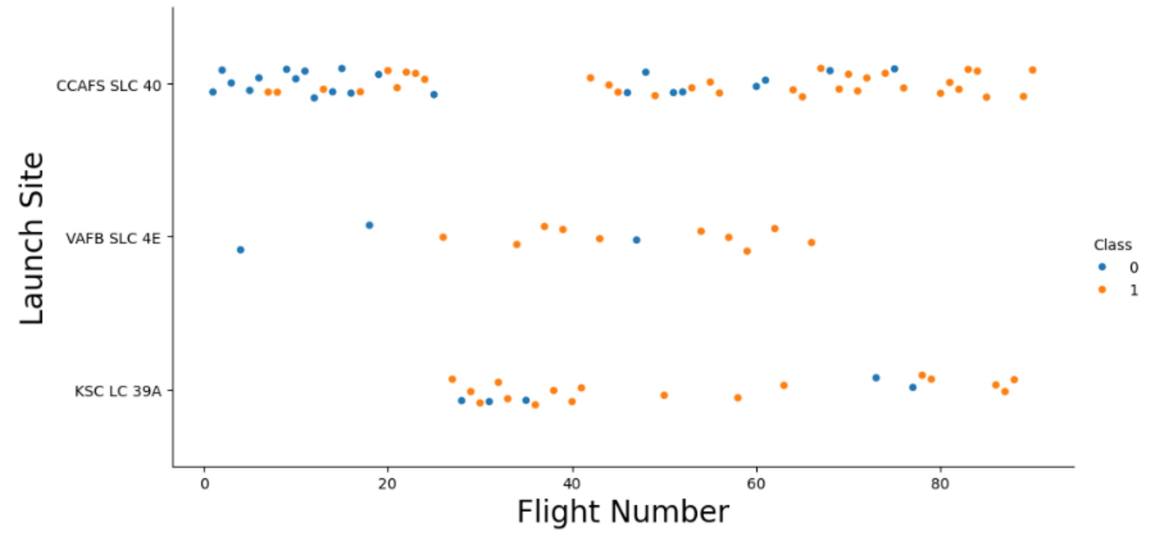
Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site



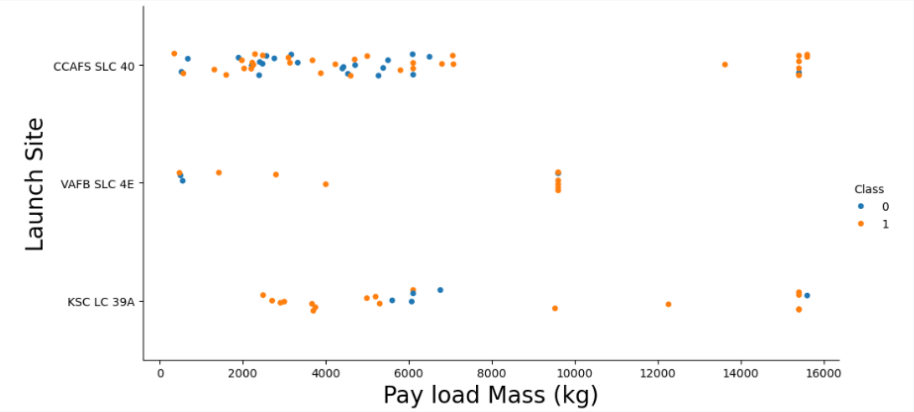
- Show the screenshot of the scatter plot with explanations

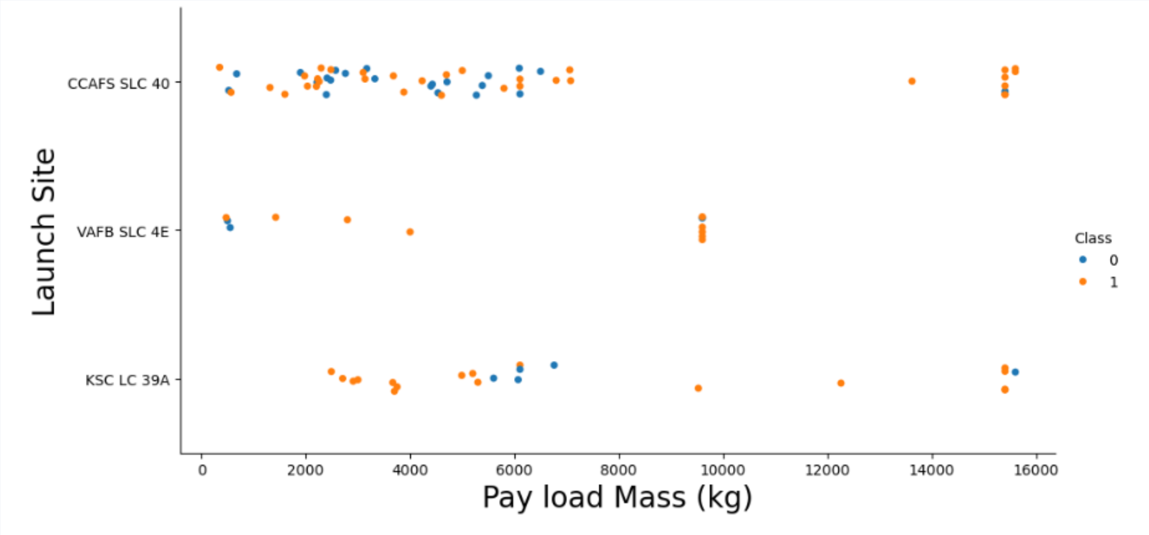


Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots. as the flight number increases, the success rate increases too. the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80th flight.

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site
- Show the screenshot of the scatter plot with explanations



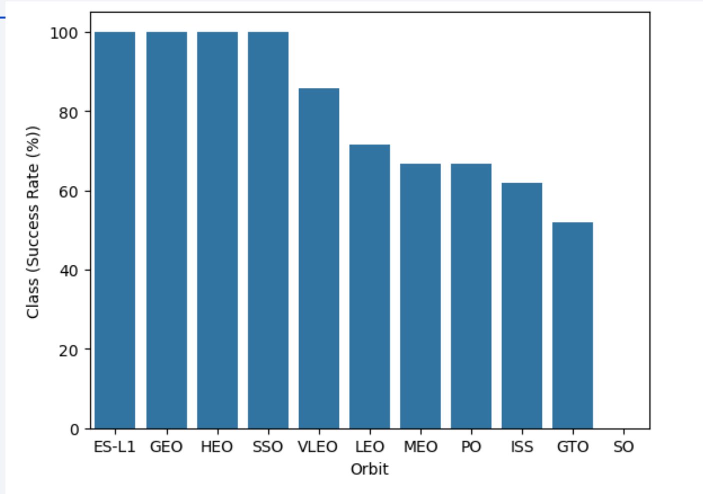


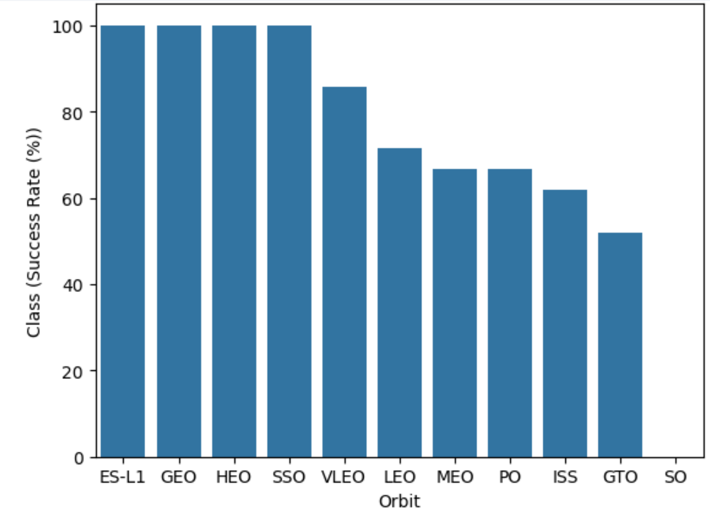
Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

- Show the screenshot of the scatter plot with explanations





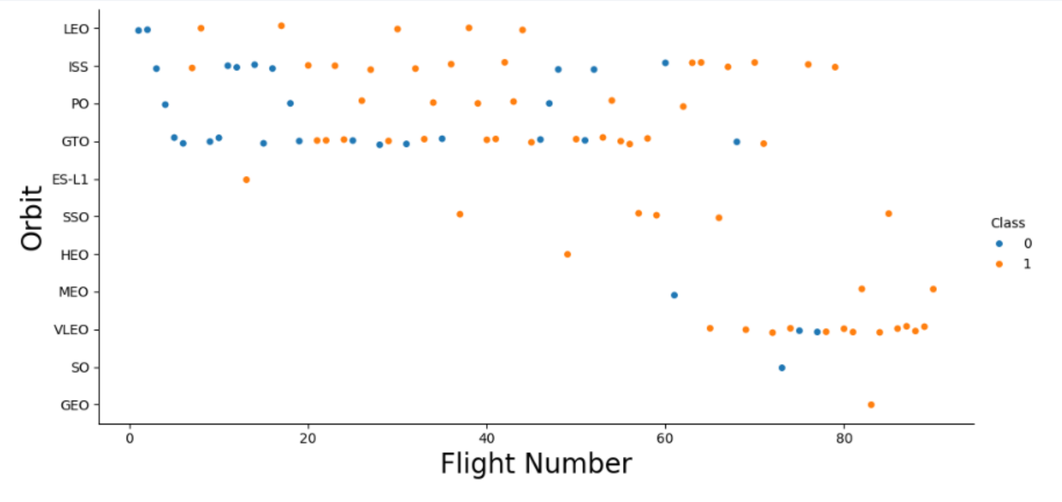
Analyze the plotted bar chart to identify which orbits have the highest success rates.

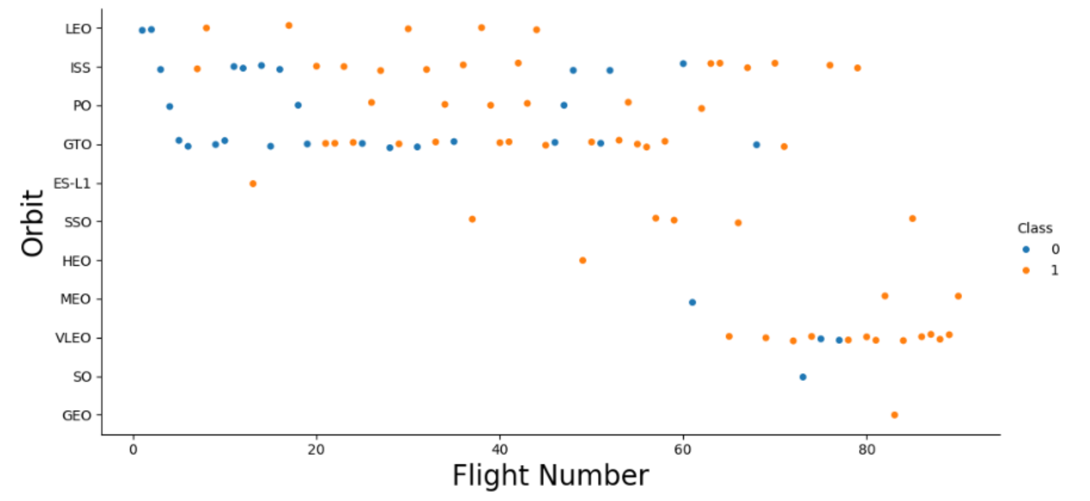
Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.

Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type

- Show the screenshot of the scatter plot with explanations



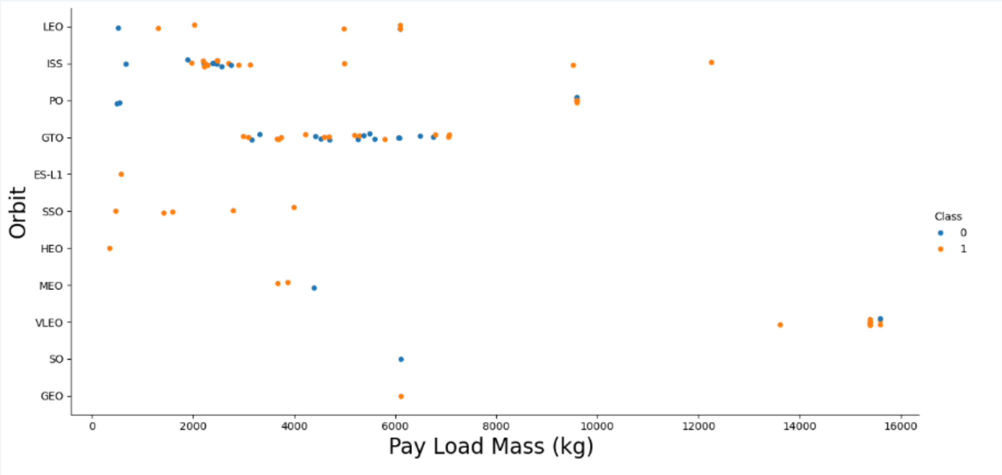


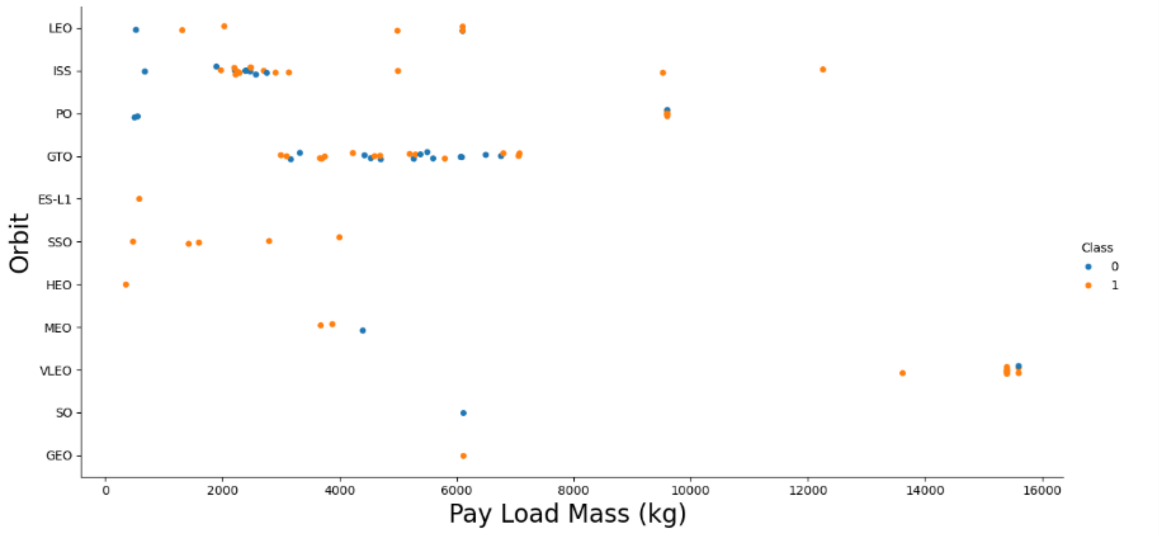
You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

- Show the screenshot of the scatter plot with explanations





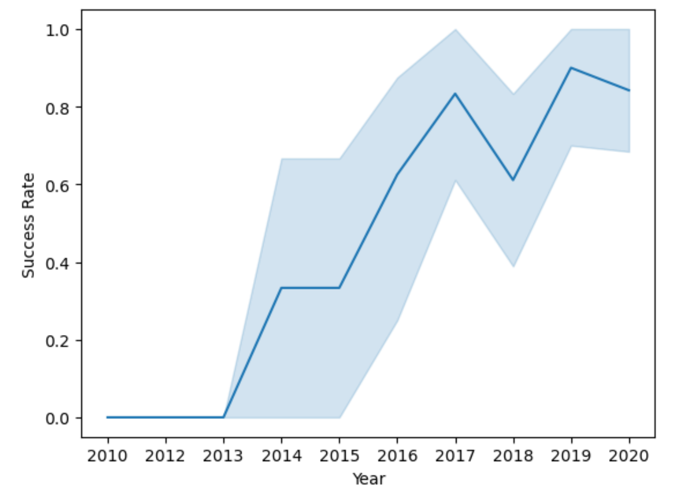
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

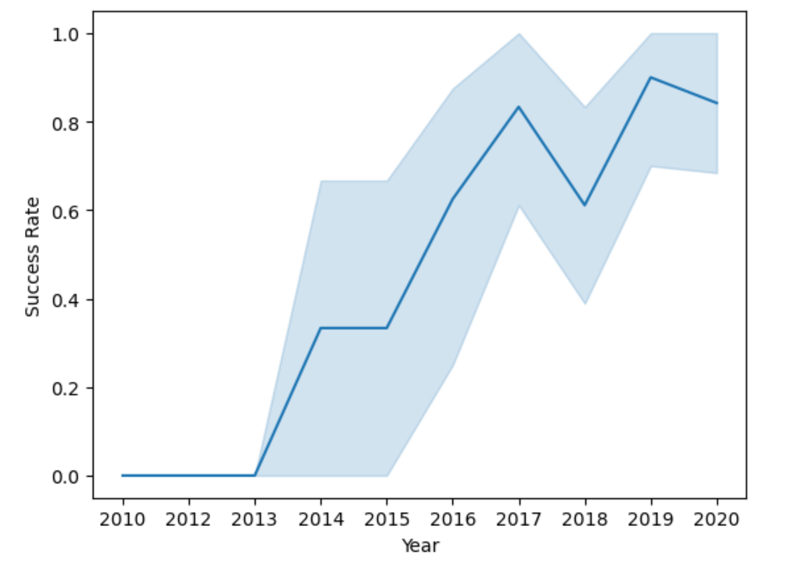
However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- Show the screenshot of the scatter plot with explanations





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
  - CCAFS SLC 40
  - KSC LC 39A
  - VAFB SLC 4E
- Present your query result with a short explanation here
  - Select launch site from table using distinct function to retrieve unique names

Task 1

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

```
%sql select distinct Launch_site from SPACEXTABLE
```

```
 * sqlite:///my_data1.db
Done.
```

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CC

Task 2

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

```
%sql select * from SPACEXTABLE where Launch_site like 'CCA%' limit 5
```

- Present your query result with a short explanation here
  - Selecting data that begins with CCA and limit the results to 5 records

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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
- Present your query result with a short explanation here
  - Select statement using Sum() function and selecting only boosters from NASA

Task 3

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

```
%sql select Sum(PAYLOAD_MASS__KG_), Customer from SPACEXTABLE where Customer = 'NASA (CRS)'
```

```
 * sqlite:///my_data1.db
Done.
```

Sum(PAYLOAD_MASS__KG_)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here
  - Select statement using avg() function and selecting only booster version F9 v1.1

Task 4

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

```
%sql select avg(PAYLOAD_MASS__KG_), Booster_Version from SPACEXTABLE where Booster_Version like 'F9 v1.1%'
```

```
 * sqlite:///my_data1.db
Done.
```

avg(PAYLOAD_MASS__KG_)	Booster_Version
2534.6666666666665	F9 v1.1 B1003

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here
  - Selecting the first date using min() function and only landing outcome that has the value “Success (ground pad)”

Task 5

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

Hint:Use min function

```
%sql SELECT min(DATE) FROM 'SPACEXTABLE' WHERE Landing_Outcome = 'Success (ground pad)'
```

```
 * sqlite:///my_data1.db
Done.
```

min(DATE)
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
  - %sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ between 4000 AND 6000;
- Present your query result with a short explanation here
  - Selecting unique boosters names with payload between 4000 and 6000 with “'Success (drone ship)” landing outcome.

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

```
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
  - %sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
- Present your query result with a short explanation here
  - Using nested select statements boosters names that carried maximum payload mass

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - %sql SELECT substr(Date,7,4), "Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome", "Landing _Outcome" FROM SPACEXTBL WHERE substr(Date,7,4)='2015' AND "Landing _Outcome" = 'Failure (drone ship)';
- Present your query result with a short explanation here
  - select only year 2015 from date field using substr() function

substr(Date,7,4)	substr(Date, 4, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing _Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone ship)

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order
  - %sql SELECT * FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORDER BY Date DESC;
- Present your query result with a short explanation here

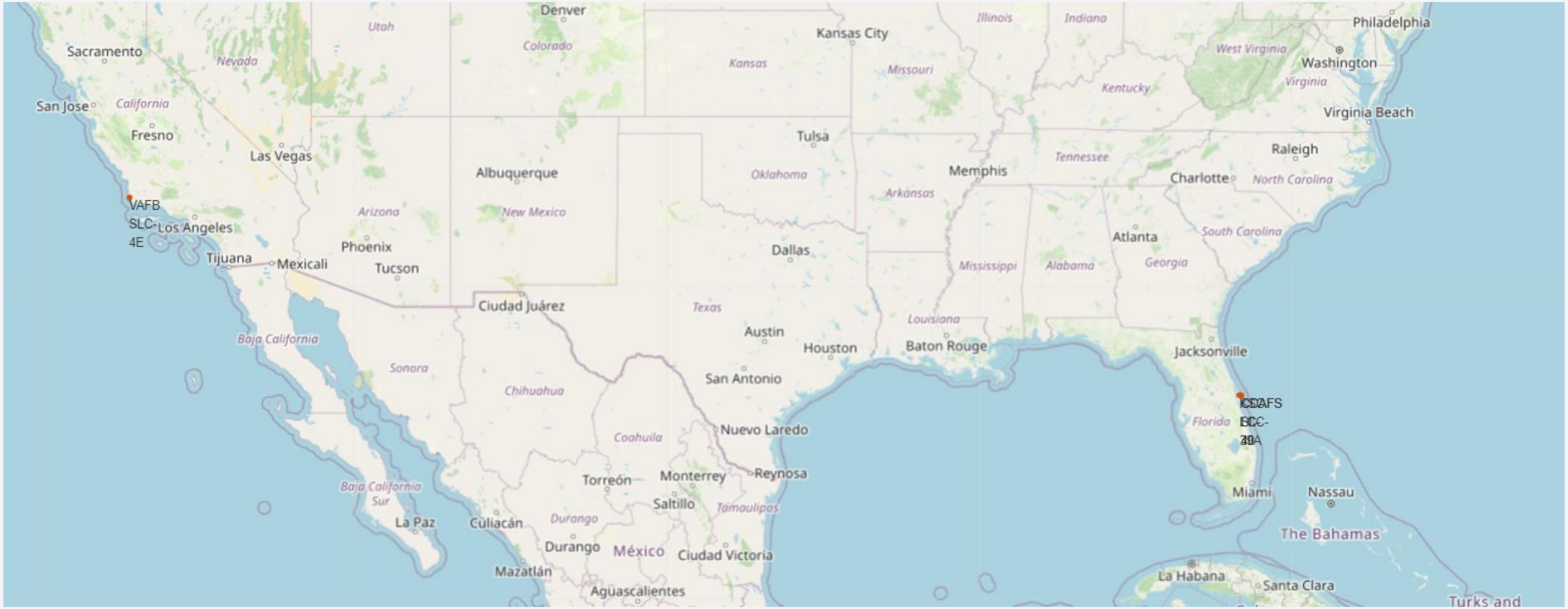
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-10-2020	12:25:57	F9 B5 B1051.6	KSC LC-39A	Starlink 13 v1.0, Starlink 14 v1.0	15600	LEO	SpaceX	Success	Success
18-08-2020	14:31:00	F9 B5 B1049.6	CCAFS SLC-40	Starlink 10 v1.0, SkySat-19, -20, -21, SAOCOM 1B	15440	LEO	SpaceX, Planet Labs, PlanetIQ	Success	Success
18-07-2016	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-04-2018	22:51:00	F9 B4 B1045.1	CCAFS SLC-40	Transiting Exoplanet Survey Satellite (TESS)	362	HEO	NASA (LSP)	Success	Success (drone ship)
17-12-2019	00:10:00	F9 B5 B1056.3	CCAFS SLC-40	JCSat-18 / Kacific 1, Starlink 2 v1.0	6956	GTO	Sky Perfect JSAT, Kacific 1	Success	Success
16-11-2020	00:27:00	F9 B5B1061.1	KSC LC-39A	Crew-1, Sentinel-6 Michael Freilich	12500	LEO (ISS)	NASA (CCP)	Success	Success
15-12-2017	15:36:00	F9 FT B1035.2	CCAFS SLC-40	SpaceX CRS-13	2205	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)

Section 3

Launch Sites Proximities Analysis

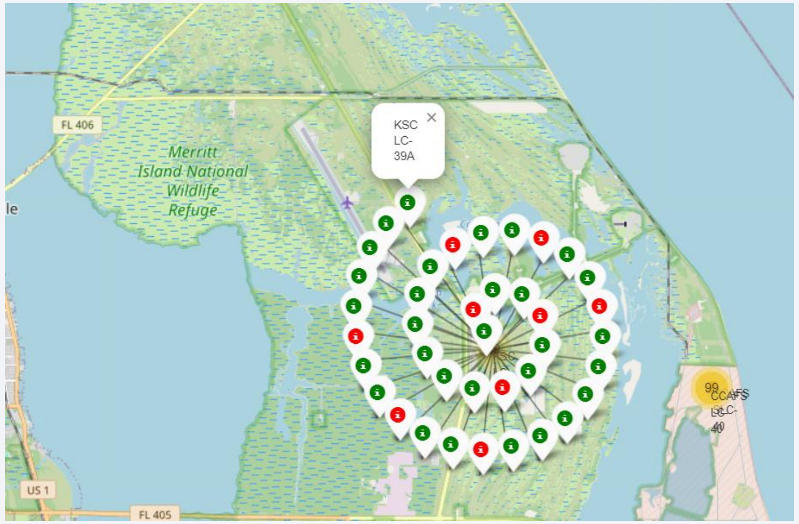
Launch Sites on map

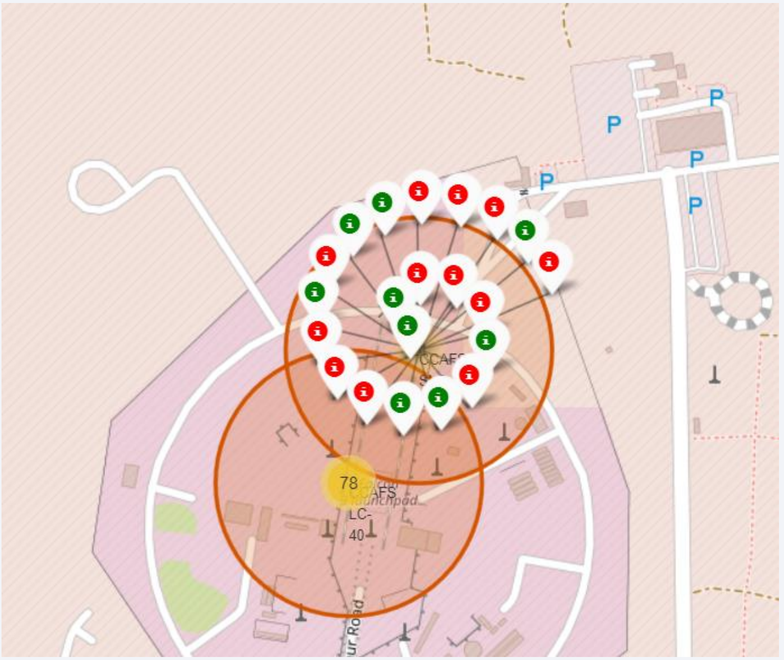
- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map



Launch outcomes for each site

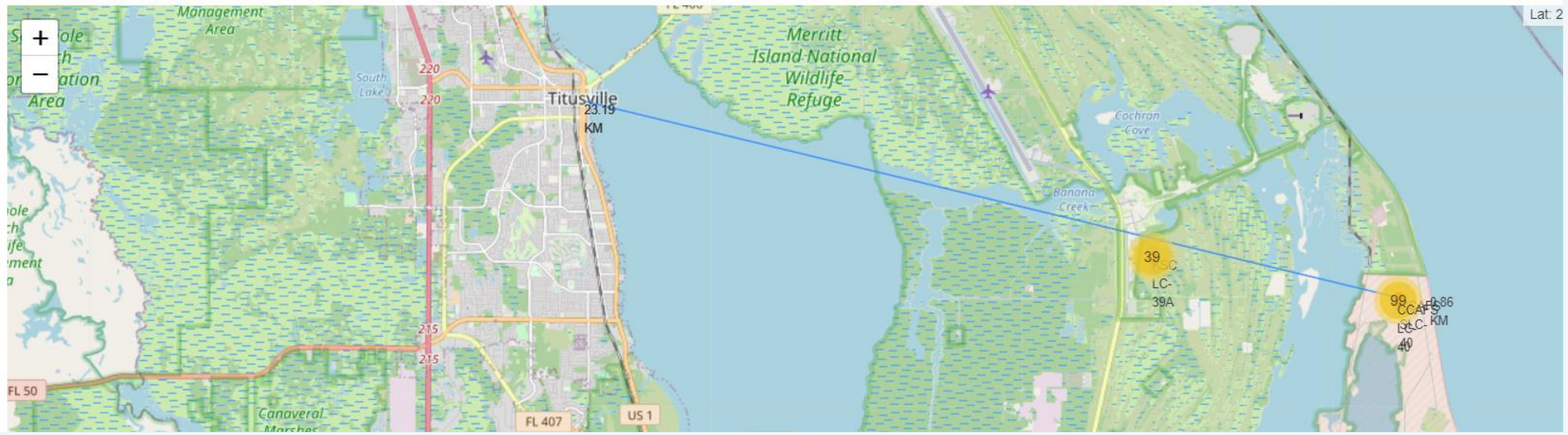
- Explain the important elements and findings on the screenshot





Nearest city to launch site

- you can draw a line between a launch site to its closest city, railway,



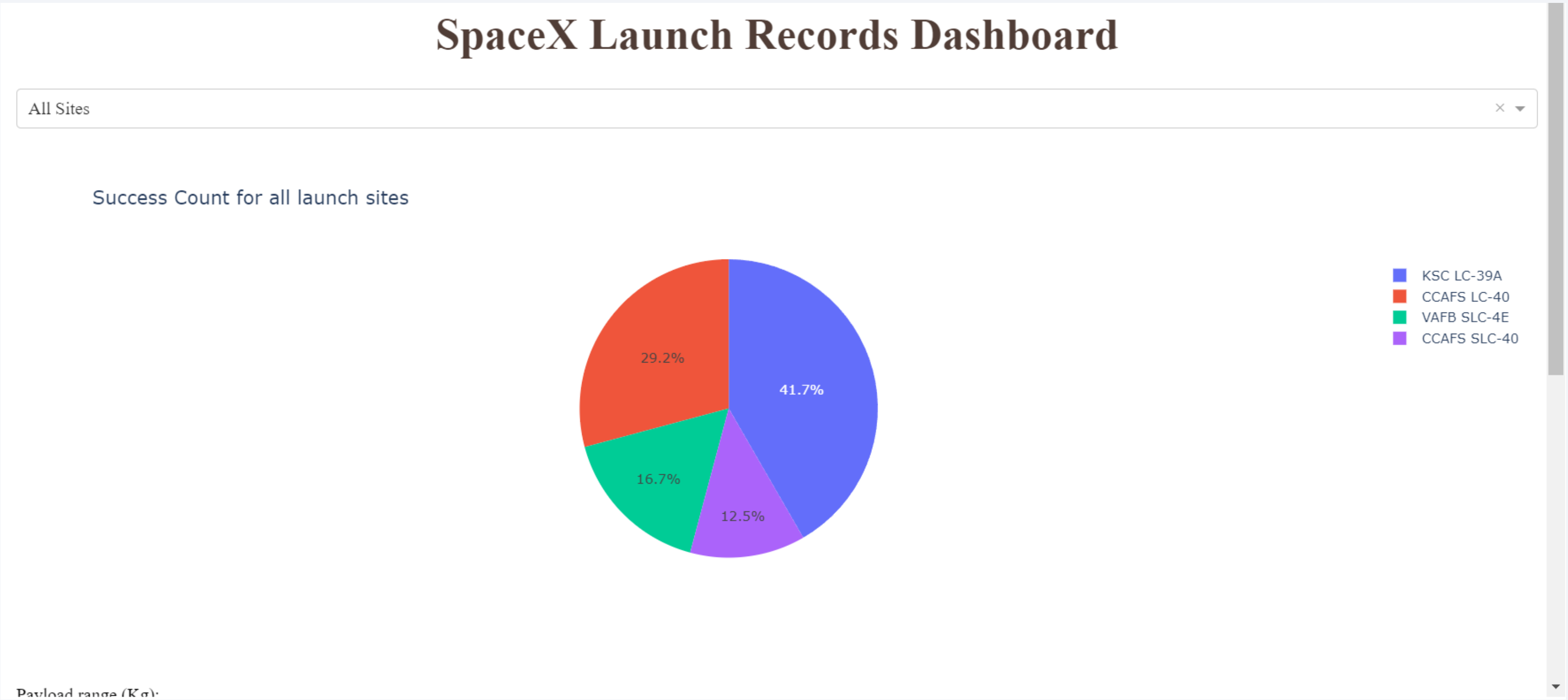
- Explain the important elements and findings on the screenshot
  - The line between the launch site and the near city
  - A city map symbol may look like

Section 4

Build a Dashboard with Plotly Dash

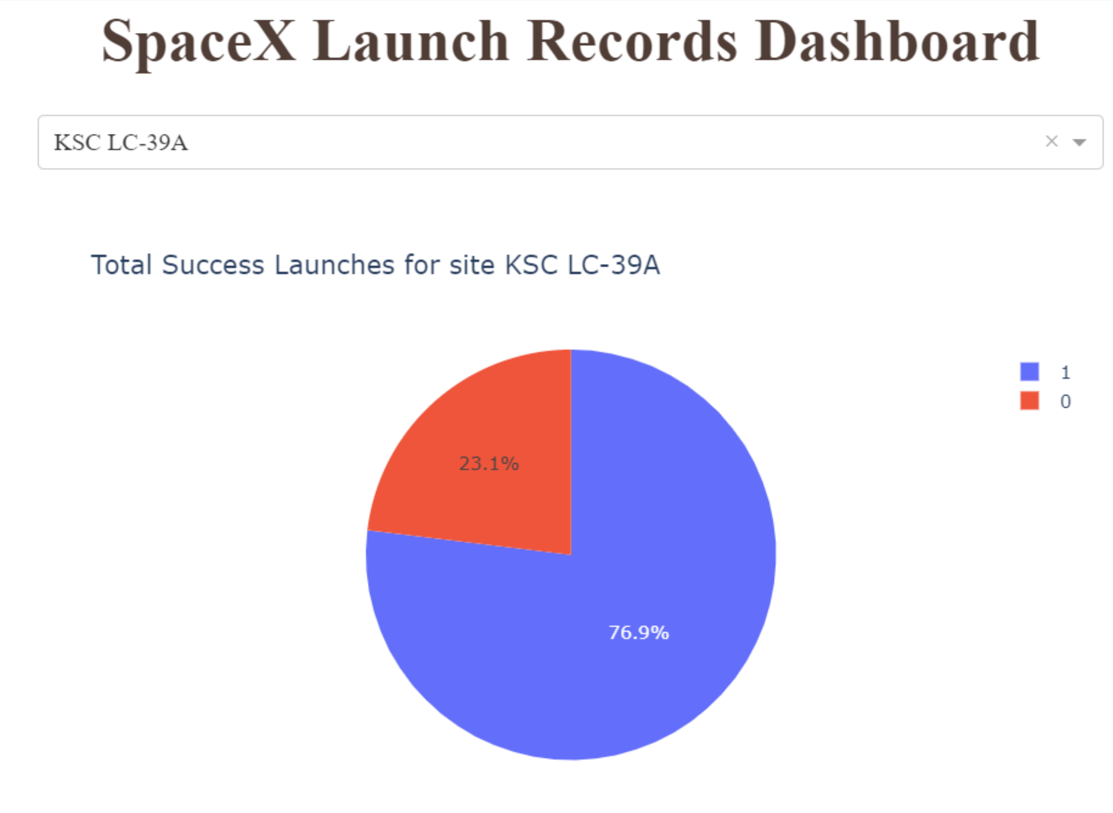
Success count for all sites

- Explain the important elements and findings on the screenshot
  - Using filter dropdown to select All Sites
  - Chart will show all sites count colored accordingly to the legend on the right side of the visual

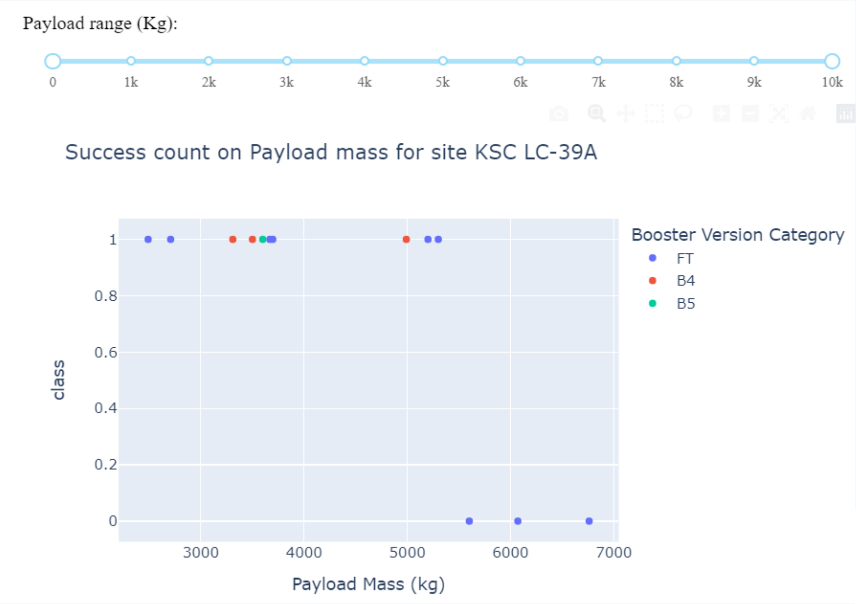


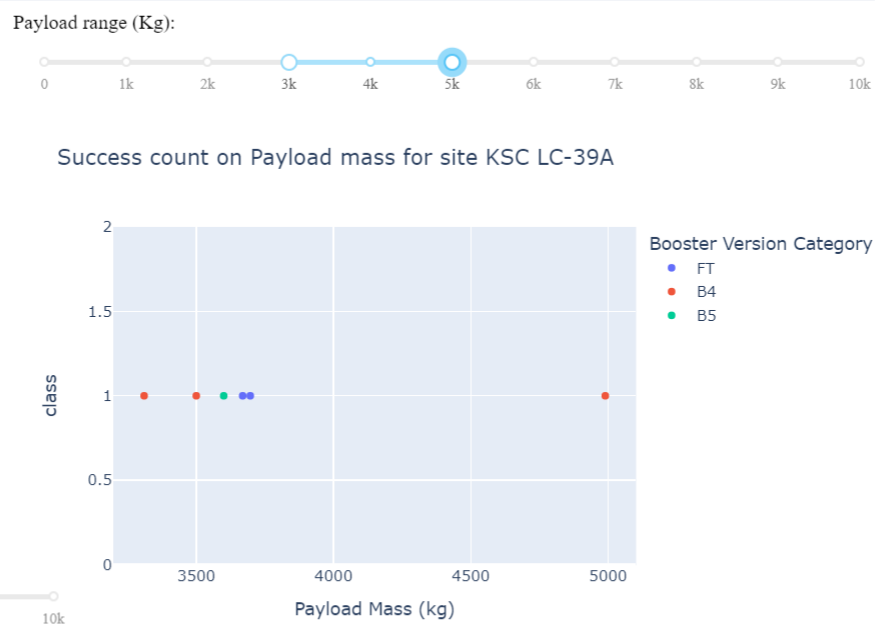
Launch site with the highest launch success ratio

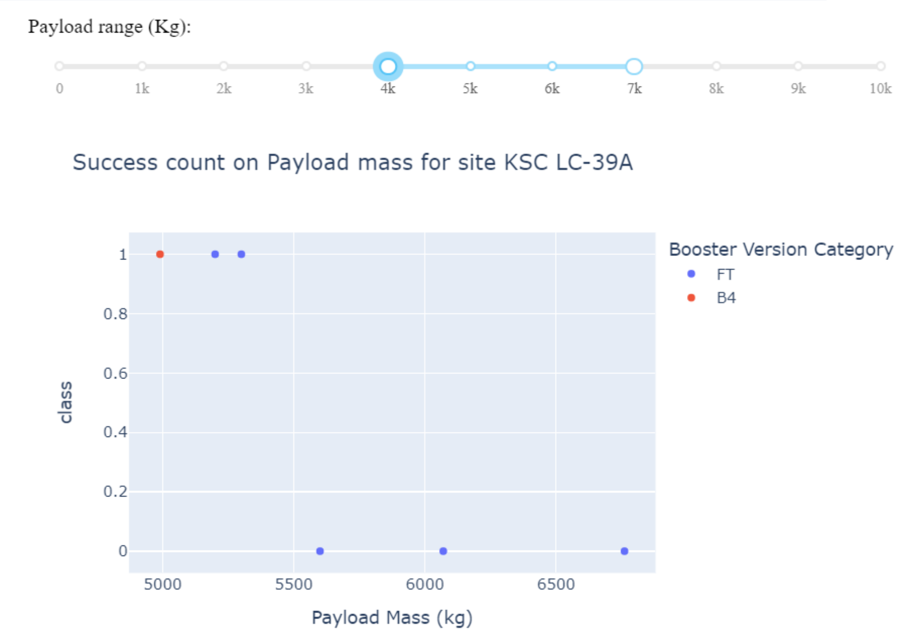
- KSC LC-39A has the highest success launch ratio



Payload vs. Launch Outcome







Section 5

Predictive Analysis (Classification)

Classification Accuracy

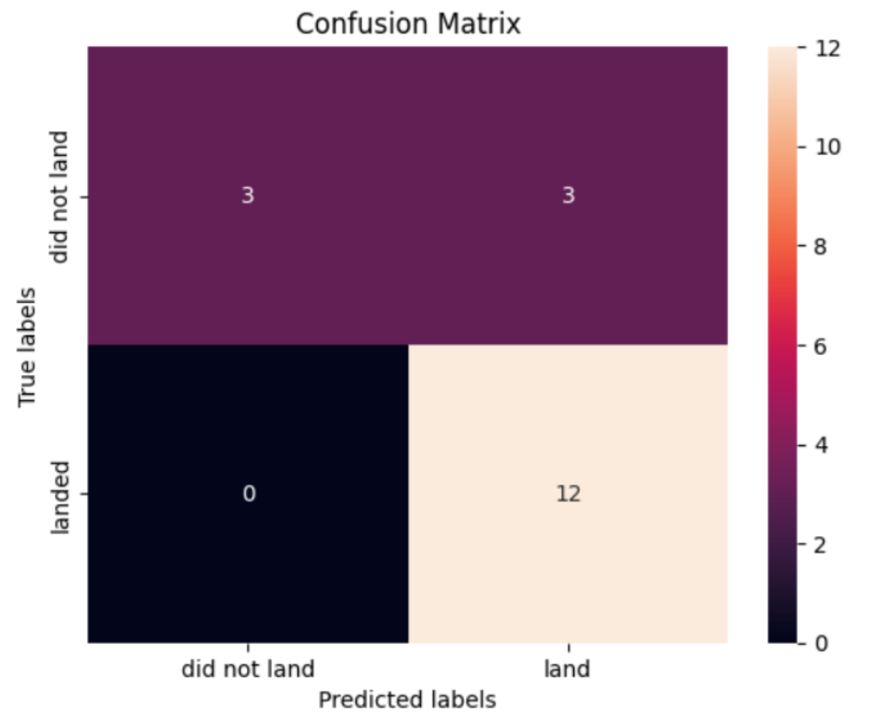
- Find which model has the highest classification accuracy

Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation:
  - classifier can distinguish between the different classes.
  - problem is false positives for all models.

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
plt.show()
```



Conclusions

- Launch sites has different landing success rates:
- When flight number increases the landing success rate is increases too.
- Whan pay load increases the landing success rate is increases too.
- Orbits ES-L1, GEO, HEO & SSO have the highest landing success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.
- Landing success rate since 2013 kept increasing till 2020.

Appendix

- Data collection api spacex/01-jupyter-labs-spacex-data-collection-api.ipynb at main · Wael-Nabil/spacex · GitHub
- Data collection webscraping spacex/02-jupyter-labs-webscraping.ipynb at main · Wael-Nabil/spacex · GitHub
- Data wrangling spacex/03-labs-jupyter-spacex-Data wrangling.ipynb at main · Wael-Nabil/spacex · GitHub
- Eda SQL spacex/04-jupyter-labs-eda-sql-coursera_sqllite.ipynb at main · Wael-Nabil/spacex · GitHub
- Exploring and Preparing Data spacex/05-edadataviz.ipynb at main · Wael-Nabil/spacex · GitHub
- Launch site location spacex/06-lab_jupyter_launch_site_location.ipynb at main · Wael-Nabil/spacex · GitHub
- Dashboard spacex/07-spacex_dash_app.ipynb at main · Wael-Nabil/spacex · GitHub
- Machine Learning Prediction spacex/SpaceX_Machine Learning Prediction_Part_5.ipynb at main · Wael-Nabil/spacex · GitHub

Thank you!