IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ryan Hernandez
10/23/2021

Outline

Executive Summary

- Using exploratory data analysis with sql I was able to manipulate some data and find information about how common certain launch outcomes are, what boosters are the best, and information about payloads
- Using Exploratory data analysis with Visuals I could visualize the corrleation between factors like flight number, launch site, orbit, and year with success giving us something to work with in order to increase the odds of space Y's rocket success
- Using Folium, I figure out spaceX's criteria for launch sites that being that they need to be in proximity to the equator, the coast, and transportation. Along with this I visualized the success of certain launch sites
- Using dash, I was able to isolate variables to find correlations between boosters and success, and how KSC LC 39-A is the best launch site
- Finally with the predictive analysis I found that almost all my models had the same benchmarks on the testing data shown by their identical confusion matrices and accuracy scores. Unfortunately, they didn't provide much new insight

Introduction

- This project was made for the capstone project at the end of the IBM data science certificate program. It's made to show my application of the data science skills the program has taught me, and my audience is peers who will review me.
- I'm trying to find out how to save costs for a fictional company SpaceY by studying how SpaceX saves cost by reusing the first stage of their rockets and how we can predict whether a launch will fail or succeed based on a variety of factors.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology: Data was collected from IBM provided spacex api and webscraped from wikipedia
- Performed data wrangling where I normalized data, put it into pandas dataframes, fixed column labels, filled in missing values, and filtered the dataframe for certain processes.
- Performed exploratory data analysis (EDA) using visualization and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models by using the logistic regresion, support vector machine, decision tree classifer, and k nearest neighbors methods from scikit learn

Data Collection

Data was provided by the SpaceX API provided and web scraped from Wikipedia with the beautifulsoup4 library. The extracting of the data from the IBM cloud is shown to the right and the web scraping is shown below. These are the datasets that I would later reformat and analyze in order to get our results.

To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021`

In [5]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

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 [6]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
import requests
getresponse = requests.get(static_url)
html = getresponse.text
```

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

In [7]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html, 'html.parser')
```

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

In [8]:
```
# 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`, please check the external reference link towards the end of this lab

In [9]:
```
# 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')
```

In [18]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [19]:
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
        Longitude.append(response['longitude'])
        Latitude.append(response['latitude'])
        LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [20]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

In [21]:
```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [37]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [38]:
```
response = requests.get(spacex_url)
```

Data Collection – SpaceX API

- First, I requested and parsed the spacex data from the ibm cloud and made the results more consistent by making a static response object then decoded it as a JSONfile and normalized it into a pandas dataframe. Then I filtered the dataframe to only see the falcon 9 launches. Then I found the mean payload of the falcon 9 rocket launches and used the mean to replace empty values

- github: https://github.com/RyanHernandezz/IBM_Data_Science_Certificate/blob/main/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:

In [40]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_sp
acex_api.json'
```

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

In [41]:
```
response.status_code
```

Out[41]:
```
200
```

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

In [51]:
```
# Use json_normalize meethod to convert the json result into a dataframe
from pandas.io.json import json_normalize
import json

data = pd.json_normalize(response.json())
```

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

[93]:
```
# Hint data['BoosterVersion']!='Falcon 1'

data_falcon9 = launch_dict_df.loc[launch_dict_df['BoosterVersion']!='Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

[94]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Data Wrangling

We can see below that some of the rows are missing values in our dataset.

In [95]:
```
data_falcon9.isnull().sum()
```

Out[95]:
```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        5
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

Before we can continue we must deal with these missing values. The `LandingPad` column will retain None values to represent when landing pads were not used.

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [100]:
```
# Calculate the mean value of PayloadMass column
MeanPayloadMass = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan,MeanPayloadMass,inplace=True)
print(MeanPayloadMass)
data_falcon9.head()
```

```
6123.547647058824
/opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages/pandas/core/series.py:4509: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy
  return super().replace(
```

Out[100]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0

You should see the number of missing values of the `PayLoadMass` change to zero.

Now we should have no missing values in our dataset except for in `LandingPad`.

Data Collection - Scraping

- First, I took this wikipedia article and made it into a static url. Then I performed a http get method upon the url and turned the http response into a beautiful soup object. Then I extracted all the columns and headers from the beautiful soup object. Finally, I parsed the html tables into a pandas dataframe for future use.
- 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 [6]: # use requests.get() method with the provided static_url
        # assign the response to a object
        import requests
        getresponse = requests.get(static_url)
        html = getresponse.text
```

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

```
In [7]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(html, 'html.parser')
```

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

```
In [8]: # 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 `BeautifulSou`
this lab

```
# 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.

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

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
[ ]: 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']=[]
```

Data Wrangling

- I counted the number of launches that happened on each site, calculated the number and occurrence of the different orbits, and then calculated the number and occurrence of mission outcome per orbit type. Finally, I created a class column that classified mission outcomes as 1(some sort of success) or 0 (some sort of failure).
- 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:

In [40]:
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Out[40]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, 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`

In [41]:
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

Out[41]:
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
GEO       1
HEO       1
ES-L1     1
SO        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 [59]:
```
# landing_outcomes = values on Outcome column
df['Outcome'].value_counts() == landing_outcomes
```

Out[59]:
```
True ASDS      True
None None      True
True RTLS      True
False ASDS     True
True Ocean     True
False Ocean    True
None ASDS      True
False RTLS     True
Name: Outcome, dtype: bool
```

EDA with Data Visualization

- First a scatter plot was drawn showing payload mass and flight number the see that payload mass generally went up over time, then a flight number and launch site scatter plot was drawn to see how larger flight numbers suceeded more often, then a scatter plot between payload mass and launch site showed how the extremes of payload mass seemed to have effected outcome and the launch site, Then there was a bar chart to compare orbits and average success rate, then a scatter plot showing orbit and flightnumber showed how flight number was correlated with which orbit it had. Then a payload and orbit scatter plot showed the effect of heavy payloads on orbit, finally a line graph showed the mission succes rate over the years and the most influential categorical variables of the dataframe were one hot encoded.
- Github

EDA with SQL

- Found distinct launch sites
- Found 5 rows with "CCA" in the launch site
- Found the sum of payloads launched for NASA(CRS)
- Found the average payload mass for the falcon 9 v1.1
- Found the date for the first succesful ground pad landing
- Found the boosters who have landed on a drone ship with a payload mass between 4000 and 6000
- Found total number of succesfull and failure mission outcomes
- Found boosters which have carried the maximum payload with a subquery
- Found booster names and launch sites for failed drone ship landings in 2015
- Ranked the count of number of times each outcome happened between 2010-06-04 and 2017-03-20 in descending order

Github

Build an Interactive Map with Folium

- Added circles on the launch sites to show them. Also added to labels to the launch sites. Along with this clusters of green and red markers on the launch sites representing the success and failure mission outcomes from that launch site. Along with this some lines showing the proximity the launch sites have to things like coast and railroads. These markers all give more nuanced visual information about the launch sites.
- Github

Build a Dashboard with Plotly Dash

- There is an interaction at the top that lets the user select if they want to see a specific launch site or all of them. Then there is a pie chart showing the percentage of succesfull and failed outcomes for each individual launch site and when all sites are selected the overall percent of succesfull outcomes that belong to each individual launch site are shown in the pie chart. Below that is a scatter plot showing payload and success/failure and there are interactions to only show certain data points like data points of only certain boosters and or data points within a certain payload range. All these interactions let the user explore the data at their own pace without being overwhelmed by too much data at once. Also, the pie charts show what the best and most used launch site is (KSC LC 39-A)
- github

Predictive Analysis (Classification)

- Standardized data set and then did a 80/20 train test split
- Created gridsearchcv objects with a cross validation score of 10 and fit logistic regression, support vector machine, decision tree classifier, and k nearest neighbors objects into the gridsearchcv object and took the resulting object and fit it to the training data before finally testing their accuracies on the testing data
- Resulting models all had their own flaws but shared a similar trait in that their accuracy to the test data was about the same
- github

Results

- Exploratory data analysis found correlations between orbit, launch site, flight number, booster version, and more with rocket success
- Interactive analytics demo in screenshots shows the criteria we should utilize to select launch sites (proximity to equator, ocean, and transportation)
- Predictive analysis results confirmed previous information but not much else due to the models having the same success as each other

Section 2

Insights drawn from EDA

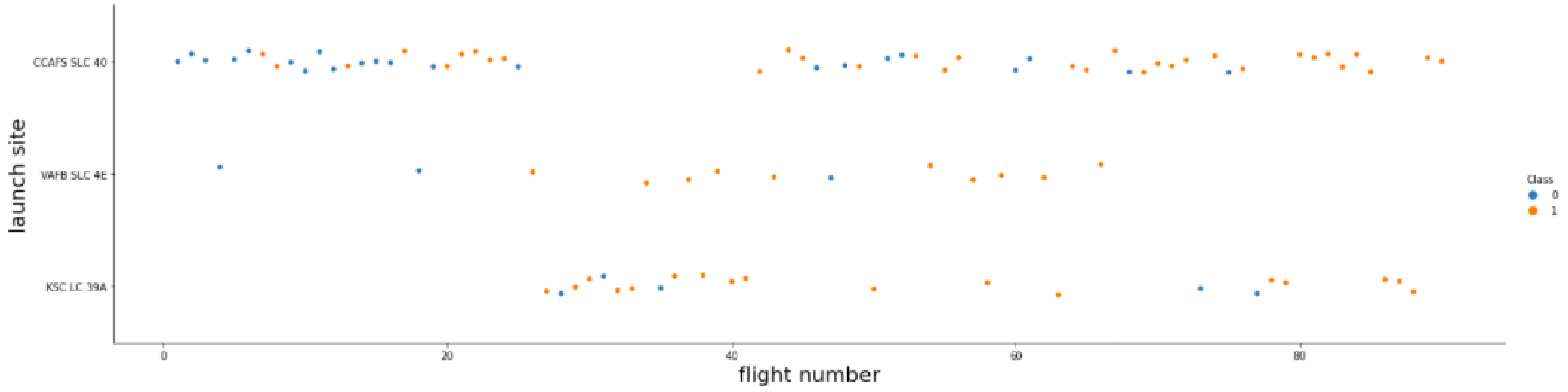
Flight Number vs. Launch Site

- As we can see as time went on and the flight number increased spacex switched back and forth be launch sites and overall had more success with later flights

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

5]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y='LaunchSite', x='FlightNumber', hue='Class', data=df, aspect = 4)
plt.xlabel('flight number', fontsize=20)
plt.ylabel('launch site', fontsize=20)
plt.show()
```



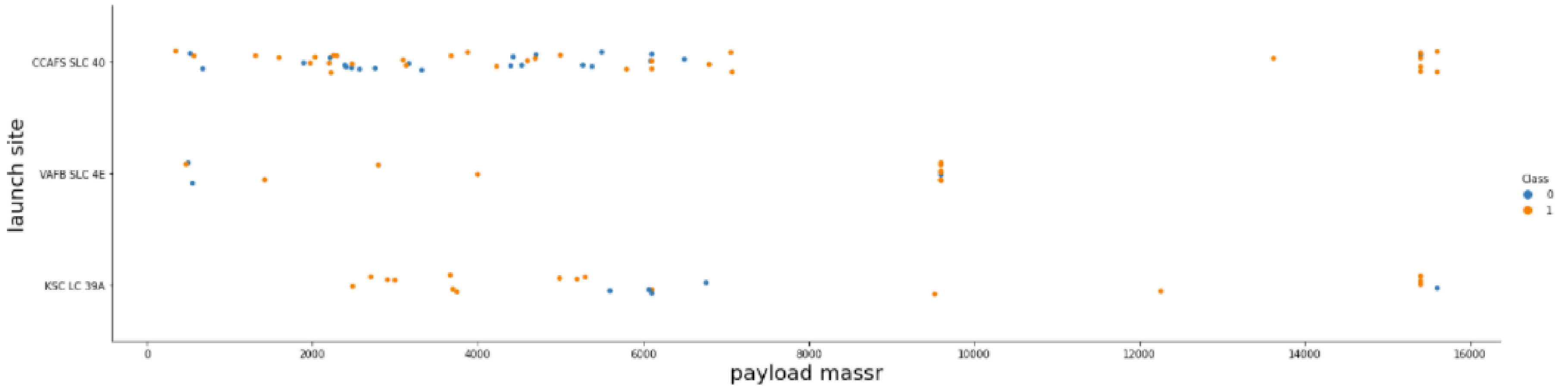
Payload vs. Launch Site

- As we can see different payload amounts called for different launch sites and there is a correlation between larger payloads and success

TASK 2: Visualize the relationship between Payload and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

[6]:
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y='LaunchSite', x='PayloadMass', hue='Class', data=df, aspect = 4)
plt.xlabel('payload massr', fontsize=20)
plt.ylabel('launch site', fontsize=20)
plt.show()
```



Now try to explain any patterns you found in the Payload Vs. Launch Site scatter point chart.

when the payload is on the lower extreme it is never ksc lc 39a and when it is on the upper extreme it is never vafb slc 4e also the upper extreme is usually class 1

Success Rate vs. Orbit Type

- The ES-L1, SSO Orbit, HEO, and Geo orbits all had perfect success rates and SO had no success showing that there are obvious choices for which orbit you should or shouldn't the launch the rocket into

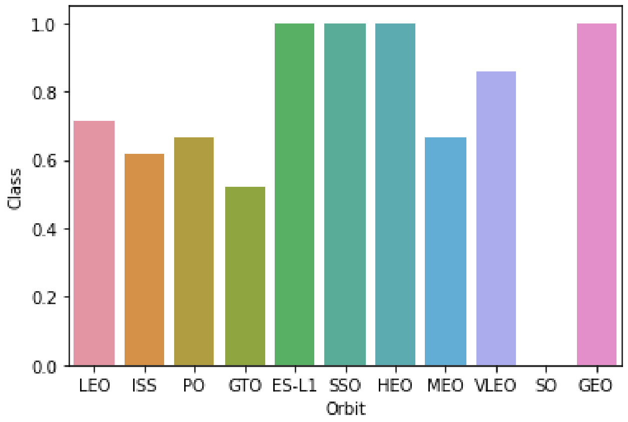
TASK 3: Visualize the relationship between success rate of each orbit type

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

In [7]:
```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_successrate = df[['Orbit','Class']]
orbit_successrate.groupby('Orbit').mean()
sns.barplot(x='Orbit', y='Class', data=orbit_successrate, ci=None)
```

Out[7]: `<AxesSubplot:xlabel='Orbit', ylabel='Class'>`



Analyze the ploted bar chart try to find which orbits have high sucess rate.

orbits ES-L1, SSO Orbit, HEO, and GEO all have perfect success rates

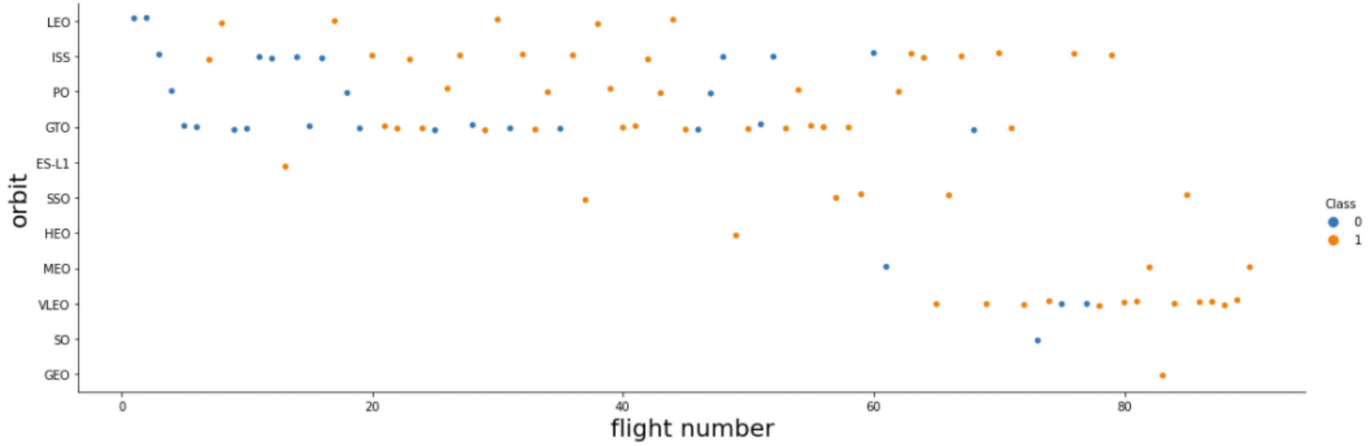
Flight Number vs. Orbit Type

- As time went on certain orbits were abandoned and certain orbits were adopted as shown by the switch in orbit selection for the later flight numbers
- This is supported by the previous slide showing varying success rates for the different orbits

TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [8]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y='Orbit', x='FlightNumber', hue='Class', data=df, aspect = 3)
plt.xlabel('flight number', fontsize=20)
plt.ylabel('orbit', fontsize=20)
plt.show()
```



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

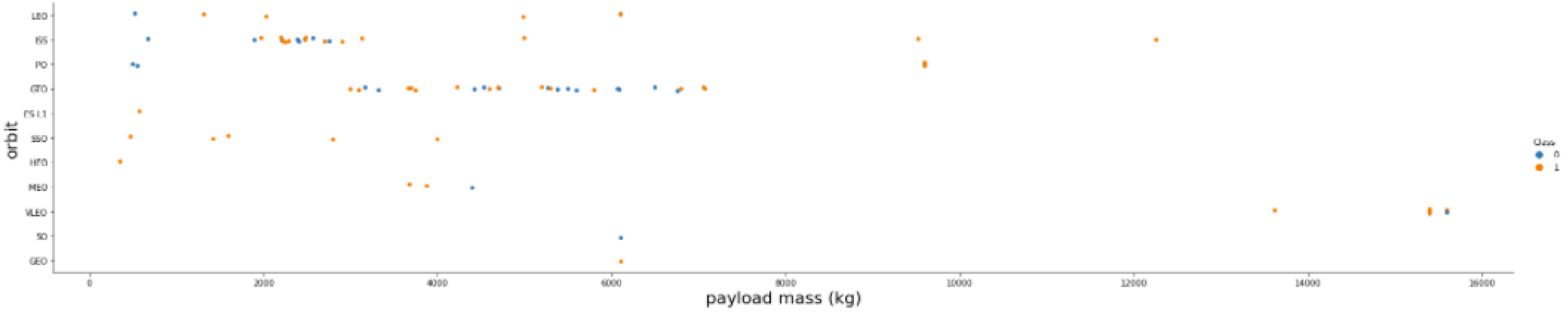
Payload vs. Orbit Type

- It seems certain orbits are more commonly chosen for certain payloads for example ISS for small payloads and VLEO for large ones

TASK 5: Visualize the relationship between Payload and Orbit type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

In [9]:
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y='Orbit', x='PayloadMass', hue='Class', data=df, aspect = 5)
plt.xlabel('payload mass (kg)', fontsize=20)
plt.ylabel('orbit', fontsize=20)
plt.show()
```



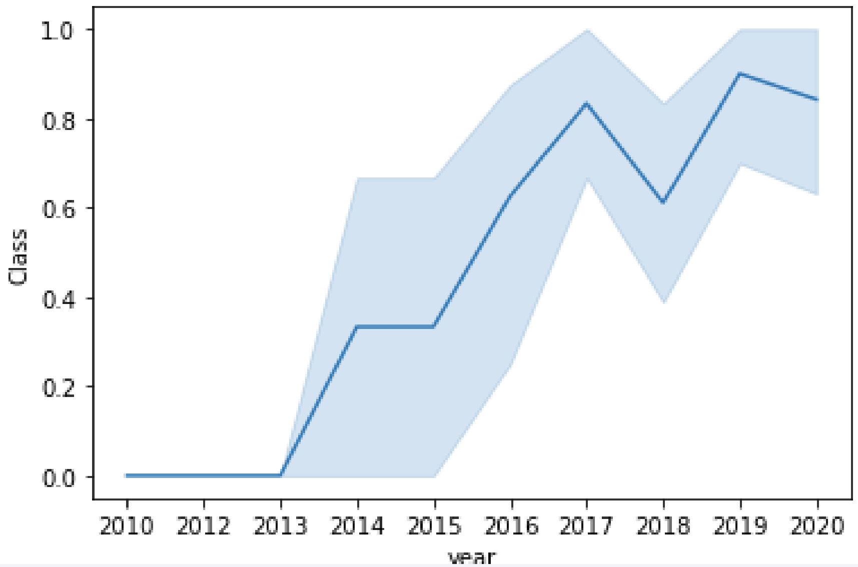
You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

Launch Success Yearly Trend

- As time went on spacex got more data and learned from their past launches so they improved their success rates.

```
In [15]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
         sns.lineplot(x=df1['year'],
                      y=df1['Class'], data=df1)

Out[15]: <AxesSubplot:xlabel='year', ylabel='Class'>
```



All Launch Site Names

- This query just checks for the distinct names in the launch site column

Now write and execute SQL queries to solve the assignment tasks.

Task 1

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

In [49]:
```
%%sql
select DISTINCT(launch_site) from SPACEX
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e6
Done.
```

Out[49]:

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

Launch Site Names Begin with 'CCA'

- This query uses the sql wildcard to search for rows with entries like 'CCA' in the launch site column with a limit of 5 results

Task 2

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

[50]:
```
%%sql
SELECT * FROM SPACEX
WHERE launch_site LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

[50]:

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

- This query uses the sql sum method to add up all payload mass values for rows where the customer is NASA (CRS)

Task 3

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

[52]:
```
%%sql
SELECT SUM(payload_mass__kg_) from spacex where customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

[52]:

1
45596

Average Payload Mass by F9 v1.1

- This query uses the sql avg method to average up the payload mass values in rows where the booster version is F9 v1.1

Task 4

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

```
[53]: %%sql
select AVG(payload_mass__kg_) from spacex WHERE booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

[53]:

1
2928

First Successful Ground Landing Date

- This query uses the min method to find the oldest date where the mission outcome column has the entry of Success

Task 5

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

Hint:Use min function

[54]:
```
%%sql
SELECT MIN(DATE) from spacex where mission_outcome = 'Success'
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd
Done.
```

[54]:

1
2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

- This query finds the distinct booster versions where the payload mass is between 4000 and 6000 and the landing outcome was a succesfull drone ship landing. (It should be between 4001 and 5999 but it doesn't change the results in this case)

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

5]:
```
%%sql
SELECT DISTINCT(booster_version) FROM spacex WHERE (payload_mass__kg_ BETWEEN 4000 AND 6000) and "Landing _Outcome" = 'Success (dro
ne ship)';
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

5]:

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

Total Number of Successful and Failure Mission Outcomes

- This query counts the number of rows which have an entry like 'Success' in the mission outcome column and then does it again but checking how many rows have an entry like 'Failure'

Task 7

List the total number of successful and failure mission outcomes

[56]:
```
%%sql
select COUNT(mission_outcome) as outcomes from spacex where mission_outcome like '%Success%' /* second row */
union
select COUNT(mission_outcome) from spacex where mission_outcome like '%Failure%' /* first row */
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

[56]:

outcomes
1
100

[57]:
```
%sql select COUNT(mission_outcome) from spacex where mission_outcome like '%Failure%'
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

[57]:

1
1

Boosters Carried Maximum Payload

- This query checks the distinct booster version entries where the row also has the max payload mass value

Task 8

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

[58]:
```
%%sql
select distinct(booster_version) from spacex where payload_mass__kg_ in (select MAX(payload_mass__kg_) from spacex)
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:3273
Done.
```

[58]:

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

- The query grabs the landing outcome, booster version, and launch site of rows where the date is in 2015 and the landing outcome was a failed drone ship landing

Task 9

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

```
%%sql
select "Landing _Outcome", booster_version, launch_site from spacex where DATE like '%2015%' and "Landing _Outcome" like '%Failure (drone ship)%'
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

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

Looking back, I don’t know why I didn't use a loop here, but I guess this gets the job done despite it being hard and unpleasing to read. This query counts the number of times each distinct landing outcome occurred and puts them in descending order.

In [61]:

```
%%sql
select count("Landing _Outcome") as number_of_outcomes, "Landing _Outcome" as outcome from spacex where "Landing _Outcome" like '%C
ontrolled (ocean)' and DATE between '2010-06-04' and '2017-03-20' group by "Landing _Outcome" /* 3 */
union all
select count("Landing _Outcome") as number_of_outcomes, "Landing _Outcome" as outcome from spacex where "Landing _Outcome" like '%F
ailure (drone ship)' and DATE between '2010-06-04' and '2017-03-20' group by "Landing _Outcome"/* 5 */
union all
select count("Landing _Outcome") as number_of_outcomes, "Landing _Outcome" as outcome from spacex where "Landing _Outcome" like '%F
ailure (parachute)' and DATE between '2010-06-04' and '2017-03-20' group by "Landing _Outcome"/* 0 */
union all
select count("Landing _Outcome") as number_of_outcomes, "Landing _Outcome" as outcome from spacex where "Landing _Outcome" like '%N
o attempt' and DATE between '2010-06-04' and '2017-03-20' group by "Landing _Outcome"/* 0 */
union all
select count("Landing _Outcome") as number_of_outcomes, "Landing _Outcome" as outcome from spacex where "Landing _Outcome" like '%P
recluded (drone ship)' and DATE between '2010-06-04' and '2017-03-20' group by "Landing _Outcome"/* 0 */
union all
select count("Landing _Outcome") as number_of_outcomes, "Landing _Outcome" as outcome from spacex where "Landing _Outcome" like '%S
uccess (drone ship)' and DATE between '2010-06-04' and '2017-03-20' group by "Landing _Outcome"/* 0 */
union all
select count("Landing _Outcome") as number_of_outcomes, "Landing _Outcome" as outcome from spacex where "Landing _Outcome" like '%S
uccess (ground pad)' and DATE between '2010-06-04' and '2017-03-20' group by "Landing _Outcome"/* 0 */
union all
select count("Landing _Outcome") as number_of_outcomes, "Landing _Outcome" as outcome from spacex where "Landing _Outcome" like '%U
ncontrolled (ocean)' and DATE between '2010-06-04' and '2017-03-20' group by "Landing _Outcome"/* 0 */
order by number_of_outcomes DESC;
```

```
 * ibm_db_sa://nws66943:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

Out[61]:

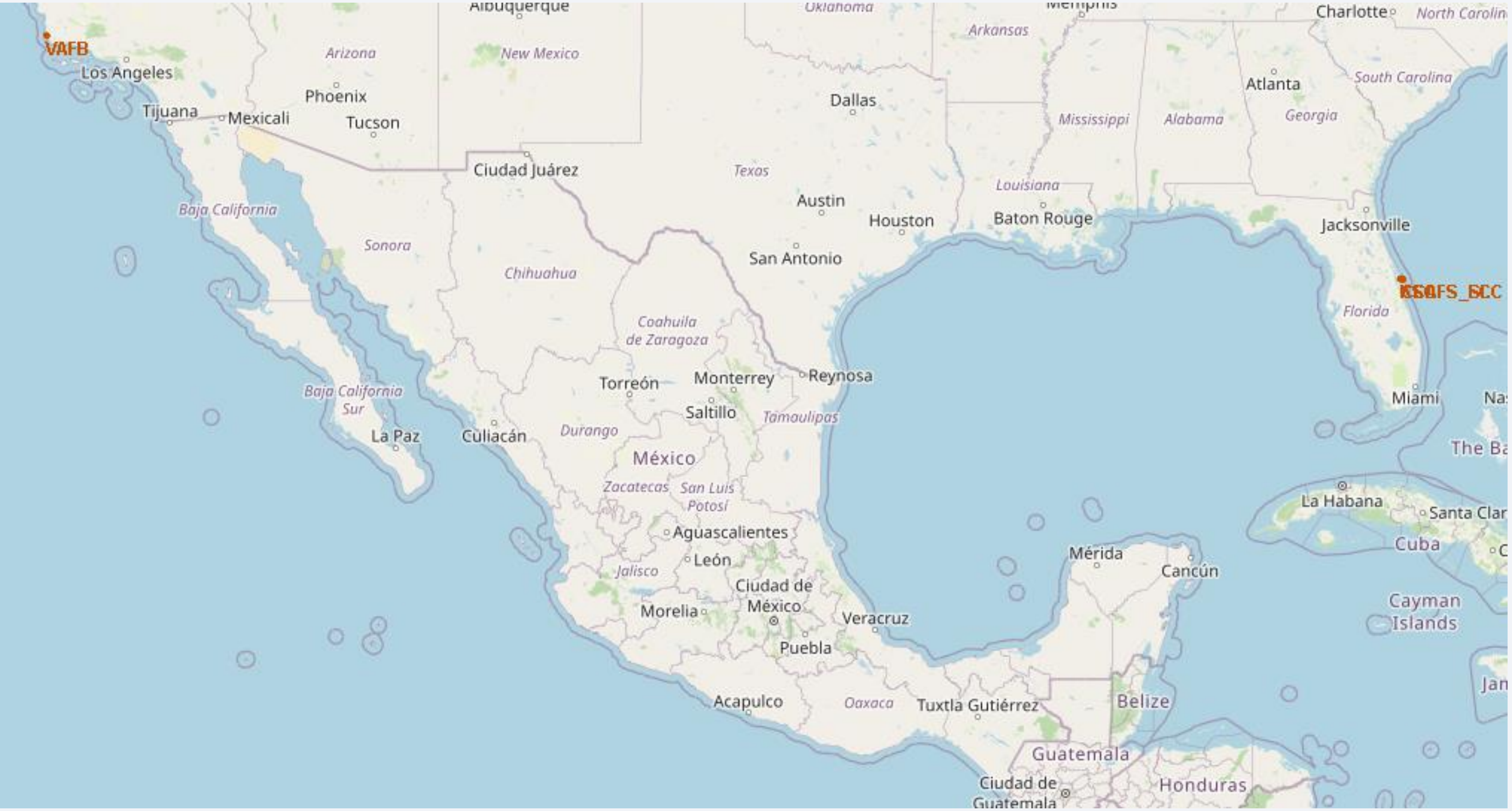
number_of_outcomes	outcome
10	No attempt
5	Failure (drone ship)
5	Success (drone ship)
3	Controlled (ocean)
3	Success (ground pad)
2	Failure (parachute)
2	Uncontrolled (ocean)
1	Precluded (drone ship)

Section 4

Launch Sites Proximities Analysis

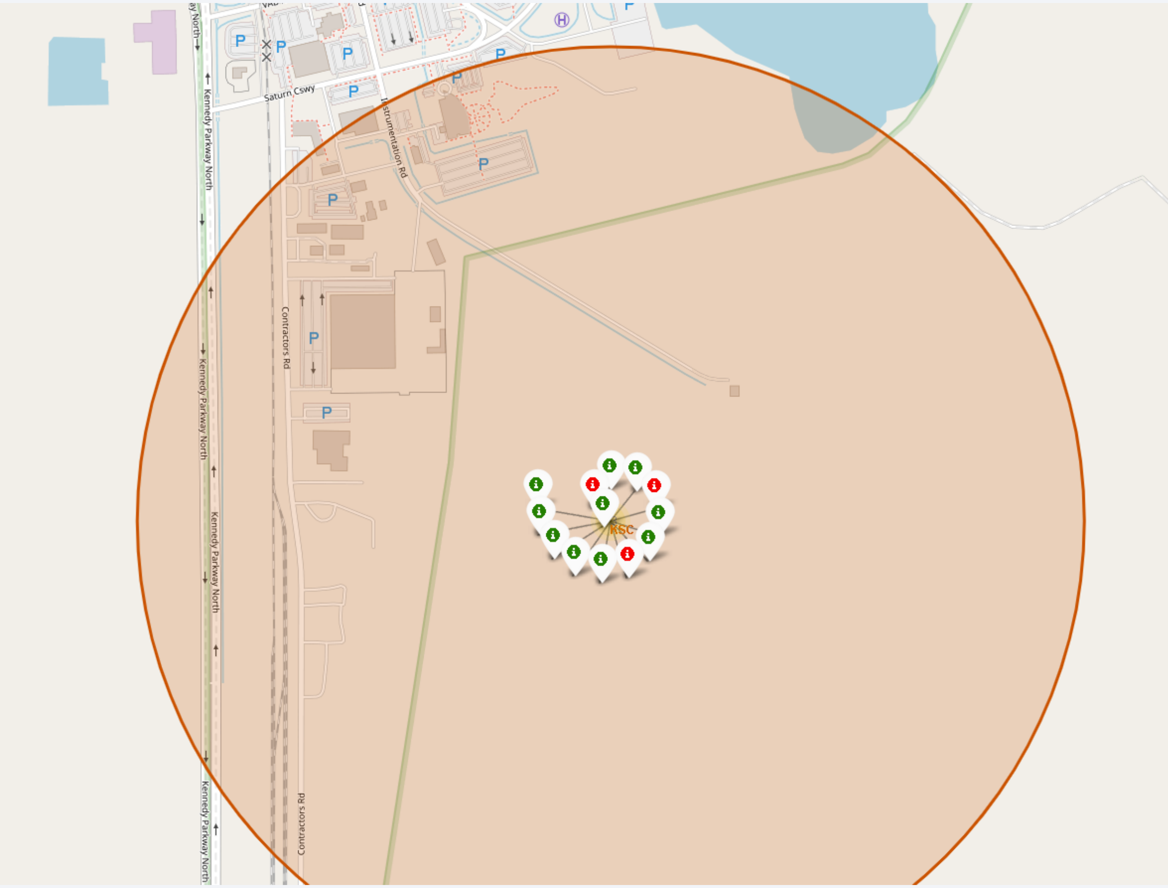
Launch sites map

- This broader international map of the launch sites shows their shared quality of proximity to the equator and the coast



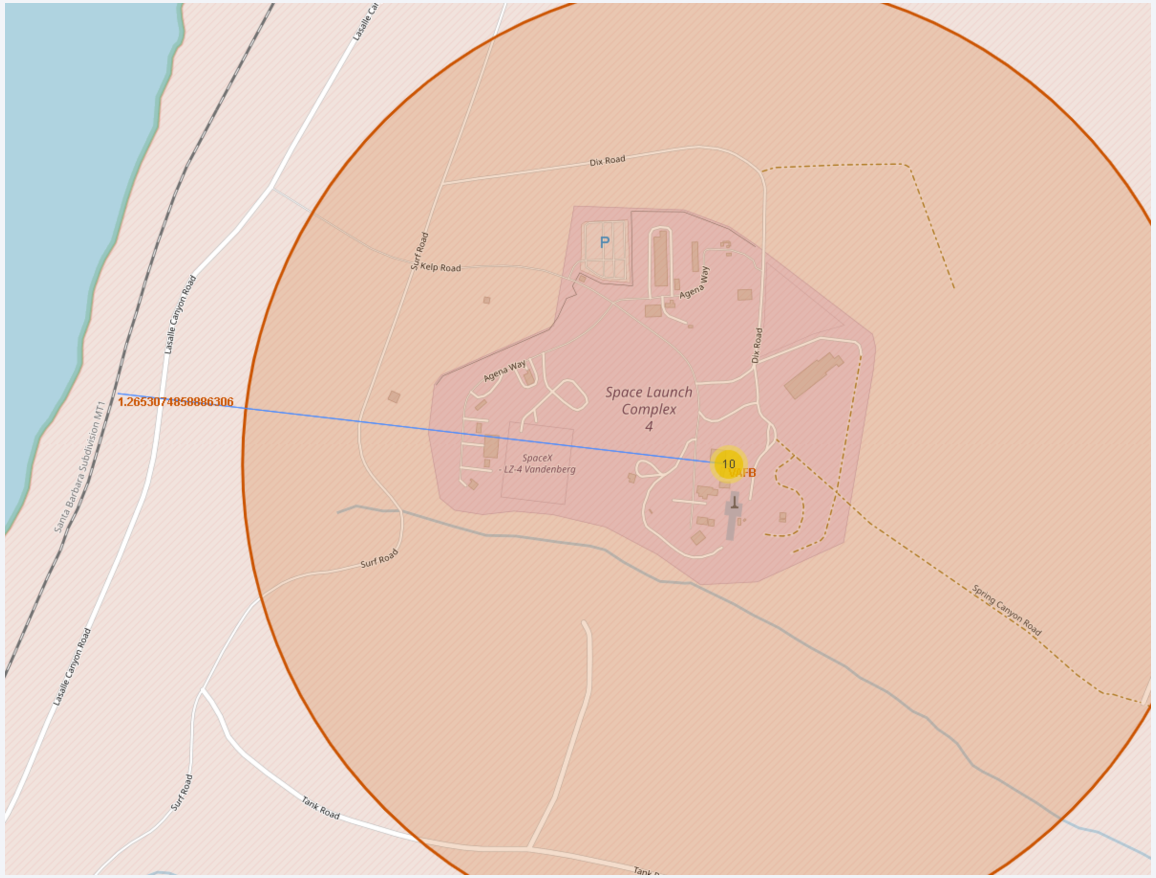
KSC LC-39A marked launch outcomes

- The abundance of the green markers goes to show why KSC LC-39A was the most heavily used by spacex since it had the most succesfull track record for them



VAFB SLC-4E nearby area

- As we can see there is a railroad very close to VAFB SLC-4E which is very convenient for transporting materials, people, and parts
- Along with this the coast is nearby aswell

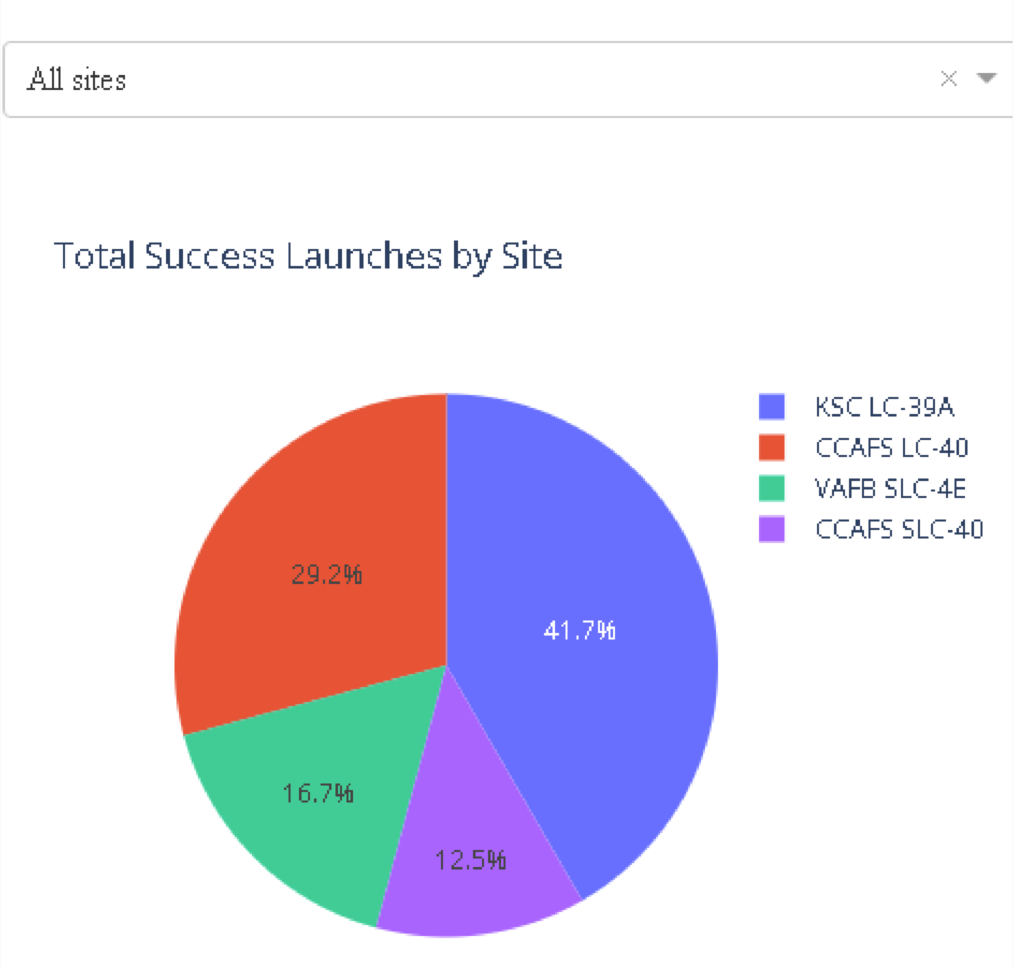


Section 5

Build a Dashboard with Plotly Dash

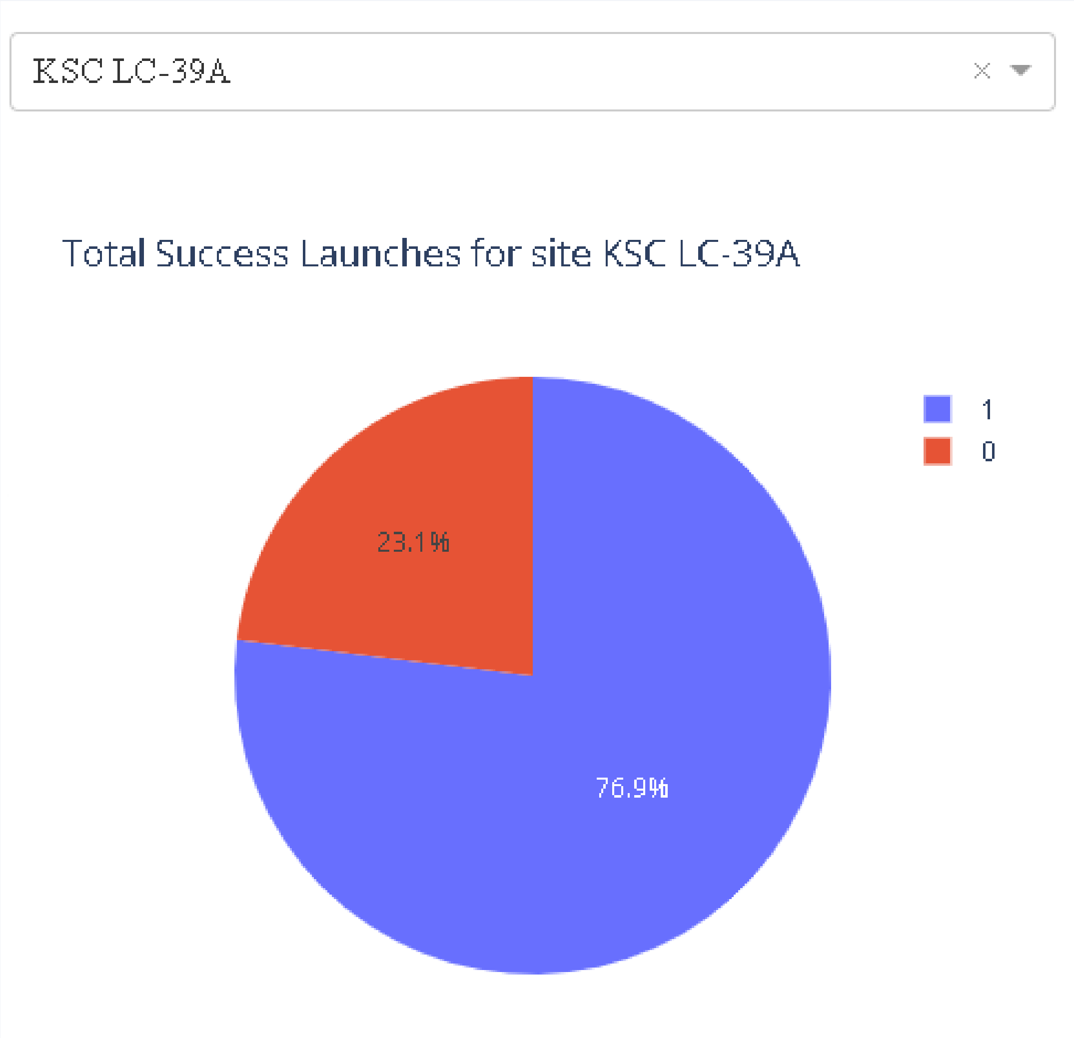
Total percent of Successful launches for each launch site

- This chart shows us that KSC LC-39A had the lion's share of succesfull launches with 41.7% while CCAFS LC-40 was successful with 29.2% and VAFB SLC-4E and CCAFS SLC-40 being either underused of having a bad track record as they only had 16.7% and 12.5% respectively



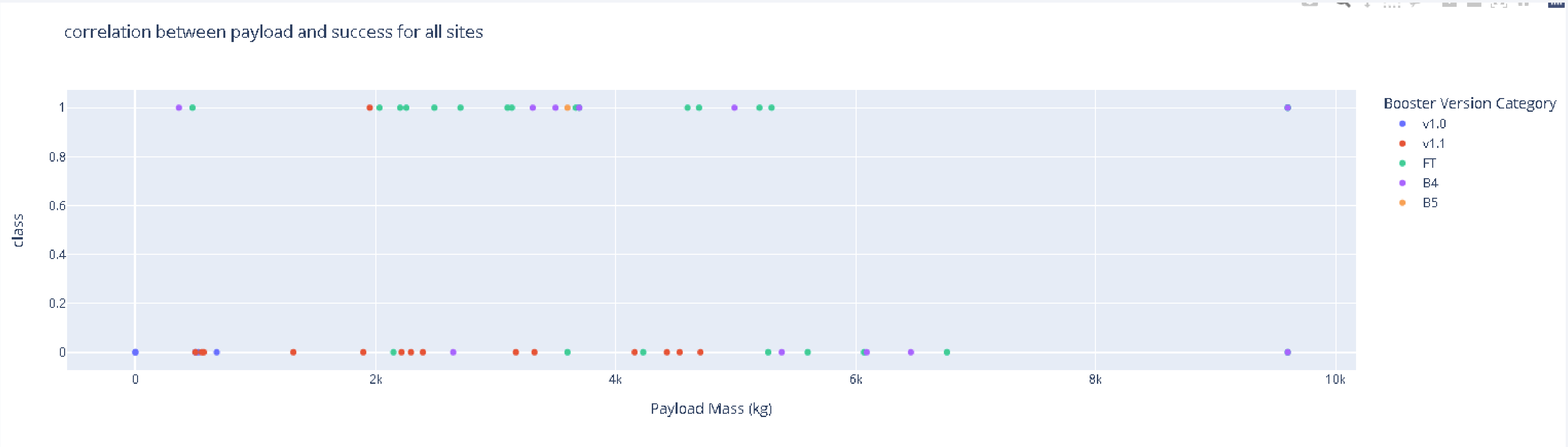
KSC LC-39A Launch Outcomes

- As we can see KSC LC-39A was used a lot and rightfully so as it had the highest success rate at 76.9%. This goes to show that it should be the go-to launch site due to its proven track record and other factors like proximity to the equator, coast, and transportation.



Correlation between payload and outcome

- This scatter plot doesn't show any notable correlation between payload and success when looking at all sites.



Section 6

Predictive Analysis (Classification)

Classification Accuracy

- All models had the same accuracy and confusion matrix with the exeption of logistic regression which was slightly worse for some reason.

TASK 12 ¶

Find the method performs best:

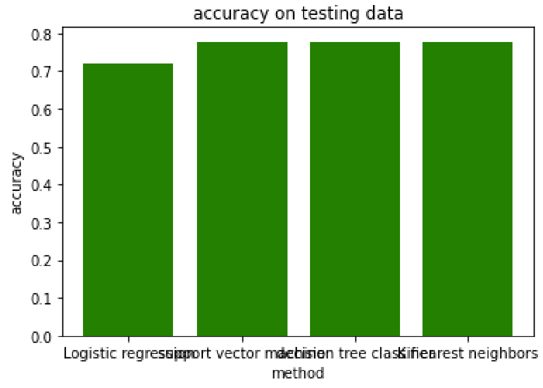
[34]:
```
x = ['Logistic regression', 'support vector machine', 'decision tree classifier', 'K nearest neighbors']
accuracy_values = [lr_accuracy, svm_accuracy, tree_accuracy, knn_accuracy]

x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, accuracy_values, color='green')
plt.xlabel("method")
plt.ylabel("accuracy")
plt.title("accuracy on testing data")

plt.xticks(x_pos, x)

plt.show()
```

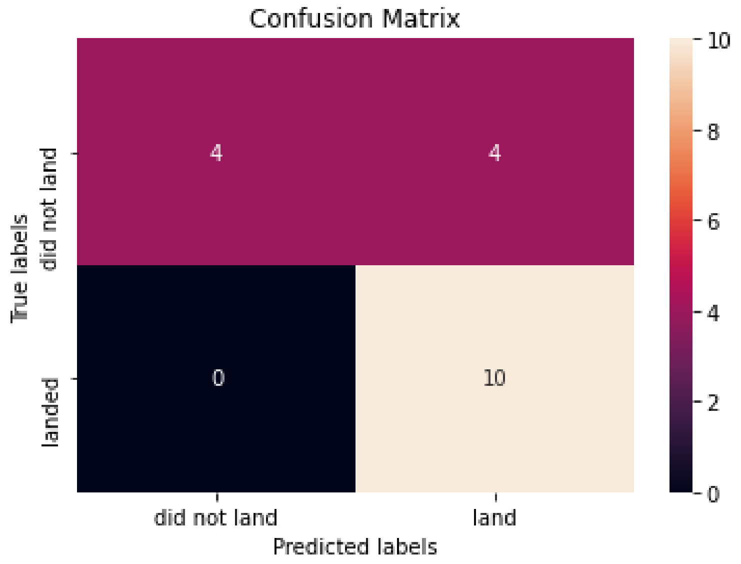


Confusion Matrix

The support vector machine, decision tree, and k nearest neighbors' models all had the exact same confusion matrix so there isn't a clear winner but heres the knn matrix for the visualization.

We can plot the confusion matrix

In [28]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(y_test,yhat)
```



Conclusions

- We should use KSC LC-39A as our launch site due to its stellar track record and proximity to the equator, coast, and transportation
- We should launch our rockets on certain orbits to increase success
- We should launch our rockets with certain boosters and payload combinations to increase success
- Later flight numbers are better developed and more likely to succeed
  - We should take our time to study SpaceX's success since they have increased their success rates over the years with data and trial and error

Appendix

- https://github.com/RyanHernandezz/IBM_Data_Science_Certificate
- Thank you IBM for the course
- Thank you Staff for maintaining the course and providing support
- Thank you Peers for reviewing my work and giving feedback

Thank you!