SpaceX - Falcon 9

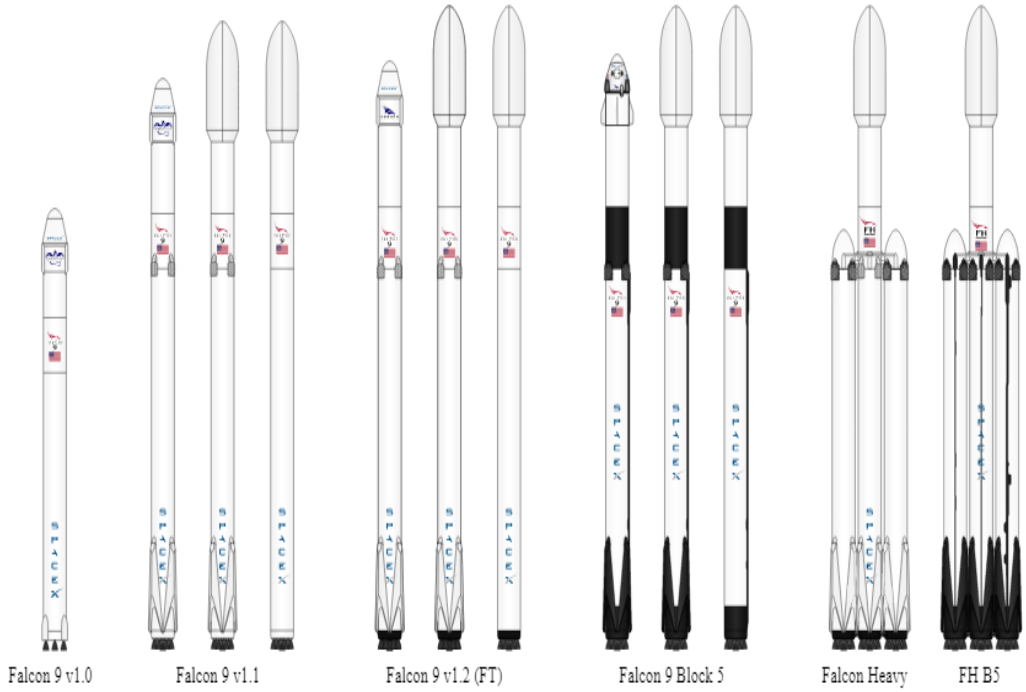


Table of Contents

1. Introduction

Space Exploration Technologies Corp. (**SpaceX**) is an American aerospace manufacturer, space transportation services and communications company headquartered in Hawthorne, California. SpaceX was founded in 2002 by Elon Musk with the goal of reducing space transportation costs to enable the colonization of Mars. SpaceX manufactures the Falcon 9 and Falcon Heavy launch vehicles, several rocket engines, Dragon cargo, crew spacecraft and Star_link communications satellites. SpaceX's achievements include the first privately funded liquid-propellant rocket to reach orbit (Falcon 1 in 2008), the first private company to successfully launch, orbit, and recover a spacecraft (Dragon in 2010), the first private company to send a spacecraft to the International Space Station (Dragon in 2012).

The first vertical take-off and vertical propulsive landing for an orbital rocket (Falcon 9 in 2015), the first reuse of an orbital rocket (Falcon 9 in 2017), and the first private company to send astronauts to orbit and to the International Space Station (SpaceX Crew Dragon Demo-2 in 2020). SpaceX has flown and reflown the Falcon 9 series of rockets over one hundred times. SpaceX is developing a satellite mega constellation named Star_link to provide commercial internet service. In January 2020 the Star_link constellation became the largest satellite constellation in the world. SpaceX is also developing Star ship, a privately funded, fully reusable, super heavy-lift launch system for interplanetary spaceflight. Star ship is intended to become the primary SpaceX orbital vehicle once operational, supplanting the existing Falcon 9, Falcon Heavy and Dragon fleet. Star ship will be fully reusable and will have the highest payload capacity of any orbital rocket ever on its debut, scheduled for the early 2020s.

2. Problem

The task should addressed using URL to target a specific endpoint of the API to get past launch data. Utilizing that it perform a get request to obtain the launch data, which we will use to get the data from the API and found few problem should addressed like…

- Data acquisition and handling Data
- Cleaning Data and Data analysis for Performing Task
- Important columns handling such Flight Number, Date, Booster version, Payload mass, Orbit, Launch Site, and Outcome utilised to obtain prediction
- The model with the best accuracy should be developed using any one technique like Logistic Regression, Support Vector machines, Decision Tree Classifier, and K-nearest neighbours
- The developed model is utilized gather information for Prediction
- This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

3. Target Audience:

The business person who want to determine if the first stage will land or not, it determine the cost of a launch. 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.

Therefore 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. In this lab, we will collect and make sure the data is in the correct format from an API.

4. Data Overview

The SpaceX launch data that is gathered from an API, specifically the SpaceX REST API. This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome. Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not. The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/. The different end points, for example: /capsules and /cores in our study will be working with the endpoint api.spacexdata.com/v4/launches/past.

In project will be using this URL to target a specific endpoint of the API to get past launch data. We will perform a get request using the requests library to obtain the launch data, which we will use to get the data from the API. This result can be viewed by calling the .json() method. Our response will be in the form of a JSON, specifically a list of JSON objects. Since we are using an API, you will notice in the lab that when we get a response

it is in the form of a JSON. Specifically, we have a list of JSON objects which each represent a launch. To convert this JSON to a dataframe, we can use the json_normalize function.

This function will allow us to "normalize" the structured json data into a flat table. This is what your JSON will look like in a table form. Another popular data source for obtaining Falcon 9 Launch data is web scraping related Wiki pages. In our work, will be using the Python Beautiful Soup package to web scrape some HTML tables that contain valuable Falcon 9 launch records. Then we need to parse the data from those tables and convert them into a Pandas data frame for further visualization and analysis. We want to transform this raw data into a clean dataset which provides meaningful data on the situation we are trying to address:

1. Wrangling Data using an API,
2. Sampling Data, and
3. Dealing with Nulls.

The data will be stored in lists and will be used to create our dataset. Another issue we have is that the launch data we have includes data for the Falcon 1 booster. Whereas we only want falcon 9. We use to figure out, how to filter/sample the data to remove Falcon 1 launches.

Finally, not all gathered data is perfect. We may end up with data that contains NULL values. We must sometimes deal with these null values in order to make the dataset viable for analysis. In this case, we will deal with the NULL values inside the Payload_Mass and the process must figure out a way to calculate the mean of the Payload_Mass data and then replace the null values in Payload_Mass with the mean. This will be dealt with using one required process to address the issue.

5. Methodology:

Over the study and reviewing some of the attributes like: Flight Number, Date, Booster version, Payload mass, Orbit, Launch Site, and Outcome: this is the status of the first stage Flights, Grid Fins: these help with landing Reused, Legs: used in landing Landing pad, Block, Reused count, Serial, Longitude and latitude of launch.

I have used following step in order move towards the solution of the problem:

- I have imported required libraries such as numpy, pandas, requests matplotlib. pyplot, folium, kmean, json_normalize, Logistic Regression, Support Vector machines, Decision Tree Classifier, and K-nearest neighbours, etc.…
- Data acquisition:
- Imported the data from the above given link using pandas.read_csv and analyzed the data.
- Cleaning Data
- Next step I had only selected important columns such Flight Number, Date, Booster version, Payload mass, Orbit, Launch Site, and Outcome.

Data acquisition:

Let us take a look at some of these attributes. The column “LaunchSite” contains the different launch sites, including: Vandenberg AFB Space Launch, Kennedy Space Center, and CCAFS SLC 40. The column orbits are the different orbits of the payload. For Example: LEO: Low Earth orbit (LEO) is an Earth-centred orbit with an altitude of 2,000 km GTO A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. It is located at 22,236 miles (35,786 kilometres) above Earth's equator. The column Outcome indicates if the first stage successfully landed. There are 8 of them, for example. True ASDS means the booster successfully landed to a drone ship. False ASDS means the mission outcome was unsuccessfully landed to a drone ship. Outcome would like landing outcomes to be converted to Classes (either 0 or 1).

0 is a bad outcome, that is, the booster did not land.

1 is a good outcome, that is, the booster did land.

```
[ ] # Create a data from launch_dict
    import pandas as pd
    data=pd.DataFrame(launch_dict)
```

Show the summary of the dataframe

```
[ ] # Show the head of the dataframe
    data.head()
```

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude
1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129
2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129
4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129
5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129
6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366

```
[ ] data_falcon9 = data_falcon9[data_falcon9['BoosterVersion'] == True]
```

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

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

```
    BoosterVersion  FlightNumber
4             True             1
5             True             2
6             True             3
7             True             4
8             True             5
..             ...           ...
89            True            86
90            True            87
91            True            88
92            True            89
93            True            90

[90 rows x 2 columns]
```

Data Wrangling

```
[ ] # Calculate the mean value of PayloadMass column
    meanvalue_PM = data['PayloadMass'].mean()
    print('the mean value of PayloadMass column =', meanvalue_PM)
    # Replace the np.nan values with its mean value
    data['PayloadMass'].fillna(value=0, inplace=True)
    data['PayloadMass']
```

```
the mean value of PayloadMass column = 5541.346276595745
0        20.0
1         0.0
2       165.0
3       200.0
4         0.0
       ...
89    15600.0
90    15600.0
91    15600.0
92    15600.0
93     3681.0
Name: PayloadMass, Length: 94, dtype: float64
```

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

Exploratory Data Analysis:

Exploratory Data Analysis is the first step of any data science project. We will perform some Exploratory Data Analysis using a database and will see if the data can be used to automatically determine if the Falcon 9's second stage will land. Some attributes can be used to determine if the first stage can be reused. Also we can then use these features with machine learning to automatically predict if the first stage can land successfully, for example. We can observe that the success rate since 2013 has improved. We can incorporate this as a feature via launch Number. We see that different launch sites have different success rates. As a result, they can be used to help determine if the first stage will land successfully. CCAFS LC-40 has a success rate of 60%, while KSC LC-39A and VAFB SLC 4E have a success rate of around 77%. Combining attributes also gives us more information. If we overlay the result of the landing outcomes as a colour we see that CCAFS LC-40, has a success rate of 60%, but if the mass is above 10,000 kg the success rate is 100%. Therefore, we will combine multiple features. In the work it will determine what attributes are correlated with successful landings. The categorical variables will be converted using one hot encoding, preparing the data for a machine learning model that will predict if the first stage will successfully land.

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

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

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

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.

```
[ ] # use requests.get() method with the provided static_url
    # assign the response to a object
    response=requests.get(static_url);
    response.status_code
```

```
200
```

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

```
[ ] # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
    soup = BeautifulSoup(response.text)
```

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

```
# Use soup.title attribute
title = soup.title
print(title.text)
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

Task 1

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

```
%sql SELECT DISTINCT("LAUNCH_SITE") as " unique launch sites"  FROM SPACEX
```

```
 * ibm_db_sa://zhr81900:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

unique launch sites
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Task 2

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

```
[ ] %sql select * from SPACEX WHERE "LAUNCH_SITE" = 'CCAFS SLC-40' LIMIT 5;
```

```
 * ibm_db_sa://zhr81900:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-12-15	15:36:00	F9 FT B1035.2	CCAFS SLC-40	SpaceX CRS-13	2205	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2018-01-08	01:00:00	F9 B4 B1043.1	CCAFS SLC-40	Zuma	5000	LEO	Northrop Grumman	Success (payload status unclear)	Success (ground pad)
2018-01-31	21:25:00	F9 FT B1032.2	CCAFS SLC-40	GovSat-1 / SES-16	4230	GTO	SES	Success	Controlled (ocean)
2018-03-06	05:33:00	F9 B4 B1044	CCAFS SLC-40	Hispasat 30W-6 PODSat	6092	GTO	Hispasat NovaWurks	Success	No attempt
2018-04-02	20:30:00	F9 B4 B1039.2	CCAFS SLC-40	SpaceX CRS-14	2647	LEO (ISS)	NASA (CRS)	Success	No attempt

Task 2

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

```
[ ] %sql select * from SPACEX WHERE "LAUNCH_SITE" = 'CCAFS SLC-40' LIMIT 5;
```

```
 * ibm_db_sa://zhr81900:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-12-15	15:36:00	F9 FT B1035.2	CCAFS SLC-40	SpaceX CRS-13	2205	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2018-01-08	01:00:00	F9 B4 B1043.1	CCAFS SLC-40	Zuma	5000	LEO	Northrop Grumman	Success (payload status unclear)	Success (ground pad)
2018-01-31	21:25:00	F9 FT B1032.2	CCAFS SLC-40	GovSat-1 / SES-16	4230	GTO	SES	Success	Controlled (ocean)
2018-03-06	05:33:00	F9 B4 B1044	CCAFS SLC-40	Hispasat 30W-6 PODSat	6092	GTO	Hispasat NovaWurks	Success	No attempt
2018-04-02	20:30:00	F9 B4 B1039.2	CCAFS SLC-40	SpaceX CRS-14	2647	LEO (ISS)	NASA (CRS)	Success	No attempt

Task 3

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

```
[ ] %sql SELECT SUM("PAYLOAD_MASS__KG_") as "total payload mass carried by boosters launched by NASA (CRS)" FROM SPACEX  where "CUSTOMER"='NASA (CRS)'
```

```
 * ibm_db_sa://zhr81900:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
```

Task 5

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

Hint:Use min function

```
[ ] %sql SELECT * FROM SPACEX  where  "LANDING__OUTCOME"= 'Success';
```

```
 * ibm_db_sa://zhr81900:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2018-07-22	05:50:00	F9 B5B1047.1	CCAFS SLC-40	Telstar 19V	7075	GTO	Telesat	Success	Success
2018-07-25	11:39:00	F9 B5B1048.1	VAFB SLC-4E	Iridium NEXT-7	9600	Polar LEO	Iridium Communications	Success	Success
2018-08-07	05:18:00	F9 B5 B1046.2	CCAFS SLC-40	Merah Putih	5800	GTO	Telkom Indonesia	Success	Success
2018-09-10	04:45:00	F9 B5B1049.1	CCAFS SLC-40	Telstar 18V / Apstar-5C	7060	GTO	Telesat	Success	Success
2018-10-08	02:22:00	F9 B5 B1048.2	VAFB SLC-4E	SAOCOM 1A	3000	SSO	CONAE	Success	Success
2018-11-15	20:46:00	F9 B5 B1047.2	KSC LC-39A	Es hail 2	5300	GTO	Es hailSat	Success	Success
2018-12-03	18:34:05	F9 B5 B1046.3	VAFB SLC-4E	SSO-A	4000	SSO	Spaceflight Industries	Success	Success
2019-	15:31:00	F9 B5 B1049.2	VAFB SLC-	Iridium NEXT-8	9600	Polar	Iridium Communications	Success	Success

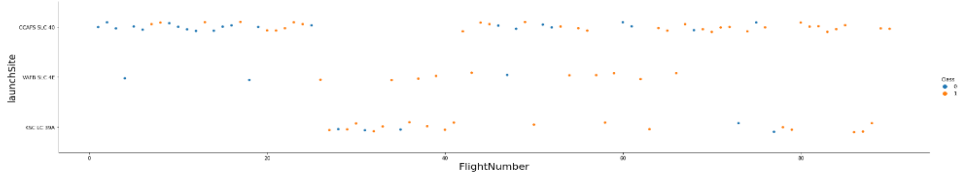
Interactive Visual Analytics

The Interactive Visual Analytics and Dashboard module. Will be used to build a Dashboard for stakeholders. Interactive visual analytics enables users to explore and manipulate data in an interactive and real-time way. Common interactions including zoom-in and zoom-out, pan, filter, search, and link. With interactive visual analytics, users could find visual patterns faster and more effectively. Instead of presenting your findings in static graphs, interactive data visualization, or dashboarding, can always tell a more appealing story. In this module, we will be using Folium and Plotly Dash to build an interactive map and dashboard to perform interactive visual analytics. The first part of this module will be focused on analysing launch site geo and proximities with Folium. We will first mark the launch site locations and their close proximities on an interactive map. Then, we can explore the map with those markers and try to discover any patterns from them. Finally, we will be building a dashboard application with the Python Plotly Dash package. This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart. Can be use it to find more insights from the SpaceX dataset more easily than with static graphs.

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'`

In [6]:
```
# 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 = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("launchSite",fontsize=20)
plt.show()
```



Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

TASK 2: Visualize the relationship between Payload and Launch Site

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
df.groupby(by=["Orbit"], dropna=False).mean("Class")
```

Out[7]:

Orbit	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Longitude	Latitude	Class
ES-L1	13.000000	570.000000	1.000000	1.000000	0.000000	1.000000	1.000000	0.000000	-80.577366	28.561857	1.000000
GEO	83.000000	6104.959412	2.000000	1.000000	1.000000	1.000000	5.000000	2.000000	-80.577366	28.561857	1.000000
GTO	35.037037	5011.994444	1.407407	0.629630	0.333333	0.629630	3.037037	0.962963	-80.586229	28.577258	0.518519
HEO	49.000000	350.000000	1.000000	1.000000	0.000000	1.000000	4.000000	1.000000	-80.577366	28.561857	1.000000
ISS	39.142857	3279.938095	1.238095	0.809524	0.238095	0.857143	3.142857	1.285714	-80.583697	28.572857	0.619048
LEO	20.000000	3882.839748	1.000000	0.571429	0.000000	0.714286	2.142857	0.428571	-80.584963	28.575058	0.714286
MEO	77.666667	3987.000000	1.000000	0.666667	0.000000	0.666667	5.000000	0.666667	-80.577366	28.561857	0.666667
PO	36.333333	7583.666667	1.333333	0.888889	0.333333	0.777778	3.222222	1.555556	-120.610829	34.632093	0.666667
SO	73.000000	6104.959412	4.000000	0.000000	1.000000	0.000000	5.000000	3.000000	-80.603956	28.608058	0.000000
SSO	60.800000	2060.000000	2.400000	1.000000	0.800000	1.000000	4.600000	3.200000	-112.604136	33.418046	1.000000
VLEO	78.928571	15315.714286	3.928571	1.000000	1.000000	1.000000	5.000000	3.928571	-80.586862	28.578358	0.857143

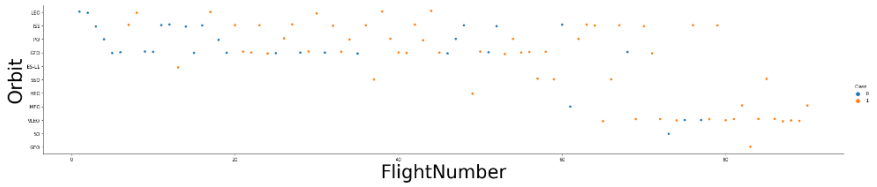
Analyze the ploted bar chart try to find which orbits have high sucess rate.

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

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 = 5)
plt.xlabel("FlightNumber",fontsize=40)
plt.ylabel("Orbit",fontsize=40)
plt.show()
```

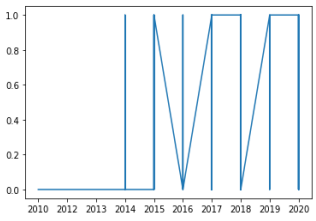


You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.


```
86  2020
87  2020
88  2020
89  2020

[90 rows x 1 columns]>
```

In [13]:
```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate

Y=df['Class']
plt.plot(X, Y)
```

Out[13]: `[<matplotlib.lines.Line2D at 0x7f201fec6e90>]`

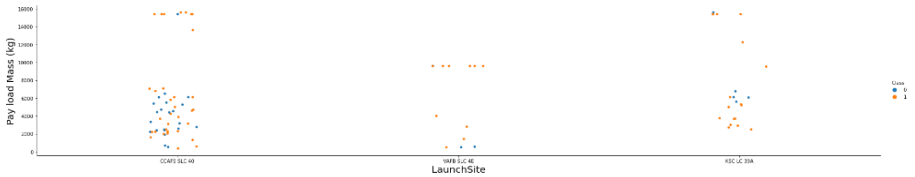


you can observe that the sucess rate since 2013 kept increasing till 2020

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.

In [ ]:
```
# 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 valu
e
sns.catplot(y="PayloadMass", x="LaunchSite", hue="Class", data=df, aspect = 5)
plt.xlabel("LaunchSite",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



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

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

Now, you can take a look at what are the coordinates for each site.

```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

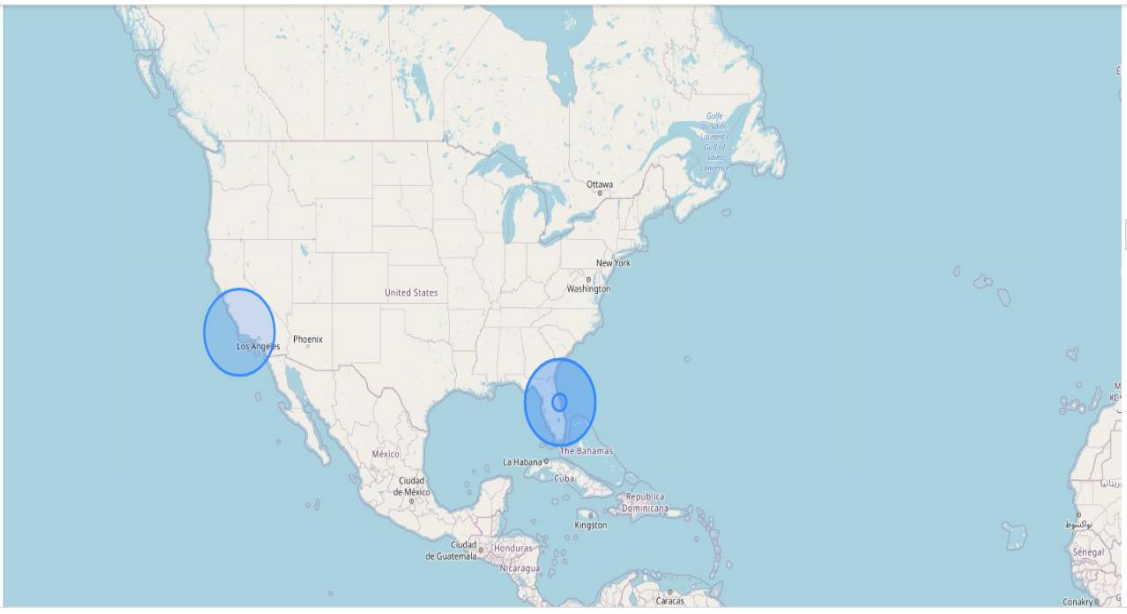
	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610746

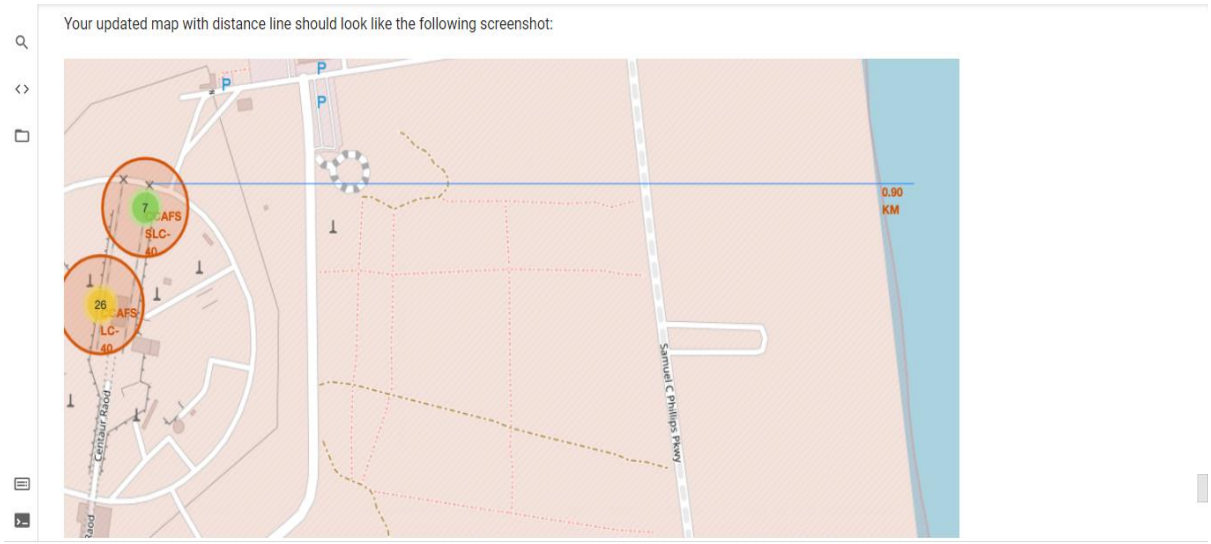
Above coordinates are just plain numbers that can not give you any intuitive insights about where are those launch sites. If you are very good at geography, you can interpret those numbers directly in your mind. If not, that's fine too. Let's visualize those locations by pinning them on a map.

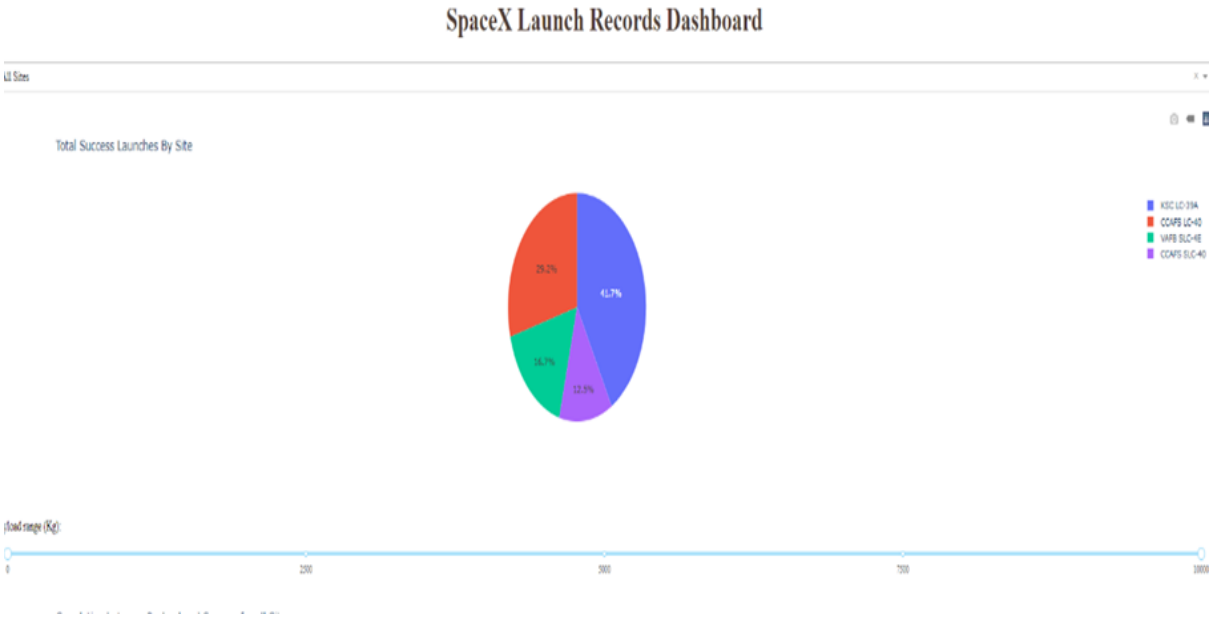
We first need to create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

[ ]
```
# Start location is NASA Johnson Space Center
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

We could use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate. For example,







Predictive Analysis:

The Predictive Analysis will build a machine learning pipeline to predict if the first stage of the Falcon 9 lands successfully. This will include: Pre-processing, allowing us to standardize our data, and Train_test_split, allowing us to split our data into training and testing data, then train the model and perform Grid Search, allowing us to find the hyper parameters that allow a given algorithm to perform best. Using the best hyper parameter values, we will determine the model with the best accuracy using the training data and test Logistic Regression, Support Vector machines, Decision Tree Classifier, and K-nearest neighbors. Finally, we will output the confusion matrix and result are shown in the figure.

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.drop(columns=['FlightNumber','Date','BoosterVersion','PayloadMass','Orbit','LaunchSite','Outcome','Flights','GridFins','Reused','Legs','LandingPad','Blc
y=data['Class']
y.to_numpy
y
```

```
0     0
1     0
2     0
3     0
4     0
     ..
85    1
86    1
87    1
88    1
89    1
Name: Class, Length: 90, 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)
print(X)
```

```
[[-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 4

Create a logistic regression object using 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()
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
logreg_cv = GridSearchCV(lr, parameters)
logreg_cv.fit(X_train, y_train)
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8166666666666668
```

TASK 5

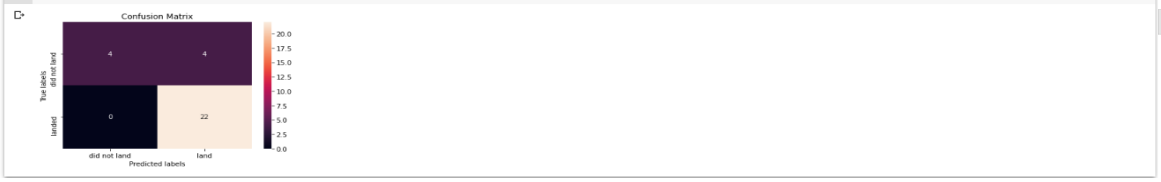
Calculate the accuracy on the test data using the method `score`:

```
print(logreg_cv.score(X_test,y_test))
```

```
0.8666666666666667
```

Lets look at the confusion matrix:

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



Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.

TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv - 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
[ ] parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
                  'C': np.logspace(-3, 3, 5),
                  'gamma':np.logspace(-3, 3, 5)}
    svm = SVC()
```

```
[ ] svm_cv=GridSearchCV(svm, parameters);
    svm_cv.fit(X_train, y_train);
```

```
[ ] print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
    print("accuracy :",svm_cv.best_score_)

    tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 1.0, 'kernel': 'sigmoid'}
    accuracy : 0.8166666666666667
```

TASK 7

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


```
[ ]
    tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 1.0, 'kernel': 'sigmoid'}
    accuracy : 0.8166666666666667
```

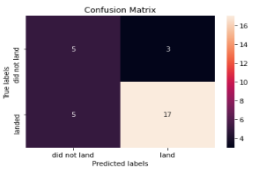
TASK 7

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


```
[ ] print(svm_cv.score(X_test, y_test))

    0.7333333333333333
```

We can plot the confusion matrix

```
    yhat=svm_cv.predict(X_test)
    plot_confusion_matrix(y_test,yhat)
```




```
[ ]
```

TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
[ ] parameters = {'criterion': ['gini', 'entropy'],
         'splitter': ['best', 'random'],
         'max_depth': [2*n for n in range(1,10)],
         'max_features': ['auto', 'sqrt'],
         'min_samples_leaf': [1, 2, 4],
         'min_samples_split': [2, 5, 10]}

    tree = DecisionTreeClassifier()
```

```
[ ] tree_cv=GridSearchCV(tree, parameters);
    tree_cv.fit(X_train, y_train);
```

```
[ ] print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
    print("accuracy :",tree_cv.best_score_)

    tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 2, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
    accuracy : 0.8666666666666666
```

TASK 9

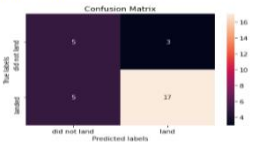
TASK 9

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

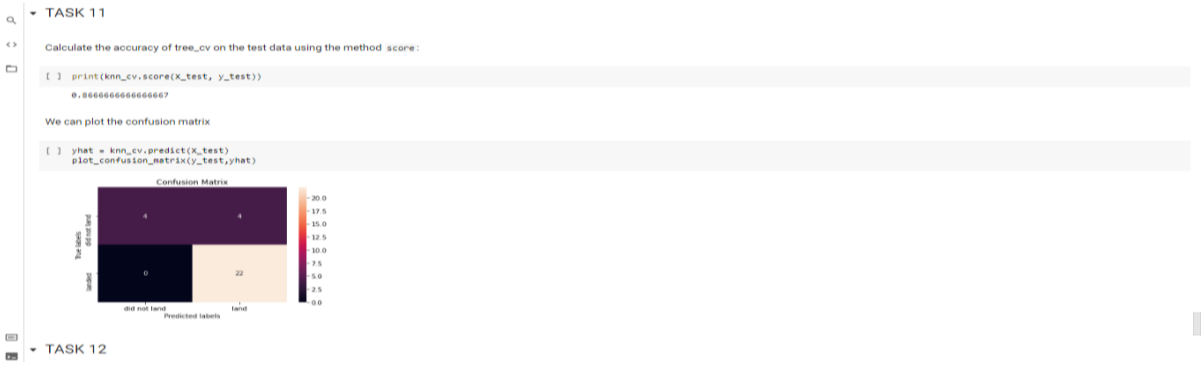
```
[ ] print(tree_cv.score(X_test, y_test))

    0.8333333333333334
```

We can plot the confusion matrix

```
[ ] yhat = svm_cv.predict(X_test)
    plot_confusion_matrix(y_test,yhat)
```



TASK 10



6. Discussion:

The depth and length for each element may vary depending on the audience and format of report. The first step in creating your report is properly creating an executive summary. This summary will briefly explain the details of the project and should be considered a stand-alone document. This information is taken from the main points of your report and while it is acceptable to repeat information, no new information is presented. By providing this interpretation of data, we are able to give a detailed explanation to the audience and how it relates to the problem that was stated in the introduction. Let's say we were conducting research for top programming languages for providing clear picture which would find the need to reach the conclusion of the report findings. This final section should reiterate the problem given in the introduction and gives an overall summary of the findings. It would also state the outcome of the analysis and if any other steps would be taken in the future.

After analysis we typically begin to compose a findings report that explains what was learned.
Depending on the stakeholders and how they receive the information, our report could vary in form. The findings report is a crucial part of data analysis, as it conveys what was discovered. The Predictive Analysis will build a machine learning pipeline to predict if the first stage of the Falcon 9 lands successfully. This will include: Pre-processing, allowing us to standardize our data, and Train_test_split, allowing us to split our data into training and testing data, then train the model and perform Grid Search, allowing us to find the hyper parameters that allow a given algorithm to perform best. Using the best hyper

parameter values, we will determine the model with the best accuracy using the training data and test Logistic Regression, Support Vector machines, Decision Tree Classifier, and K-nearest neighbours. Finally, we will output the confusion matrix and result are shown in the figures. The machine learning pipeline help to predict if the first stage will land given the data from the preceding labs

7. Conclusion

Coursera “IBM Data Science Professional certificate” course has provided us a best opportunity to learn data science by the end of this course we had an opportunity on a business problem, and it was tackled in a way that it was similar to how a genuine data scientist would do. We utilized numerous Python libraries to fetch the information, control the content and break down and visualize those datasets. Over the study and understanding it is found that, finding and cleaning data is an important first step in data analysis, a concept can be lost if you are not able to organize and represent the findings effectively to your audience. Understanding how to represent findings by focusing on specific elements to create a successful data findings report. After the data has been collected, cleaned, and organized the work of interpretation begins. To obtain a complete view of the data and hopefully, answer the questions that was formed after developing model and performing analysis. Few things listed below after working with project.

- Performing Request to the SpaceX API and Clean the requested data from collected data.
- Performing exploratory Data Analysis and determine using Data wrangling
- Dataset includes a record for each payload carried during a SpaceX mission into outer space. his information can be used if an alternate company wants to bid against SpaceX for a rocket launch, SQL helps to make analysis
- performing more interactive visual analytics using Folium help to get detailed picture
- Performing exploratory Data Analysis and Feature Engineering using Pandas and Matplotlib has performed
- Machine learning pipeline to predict if the first stage will land given the data from the preceding labs. KNN and Logistic Regression Model shows better result