IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

王彦祺 Andy yanqi WANG

5th Jan. 2024

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context
- Problems you want to find answers

Section 1

Methodology

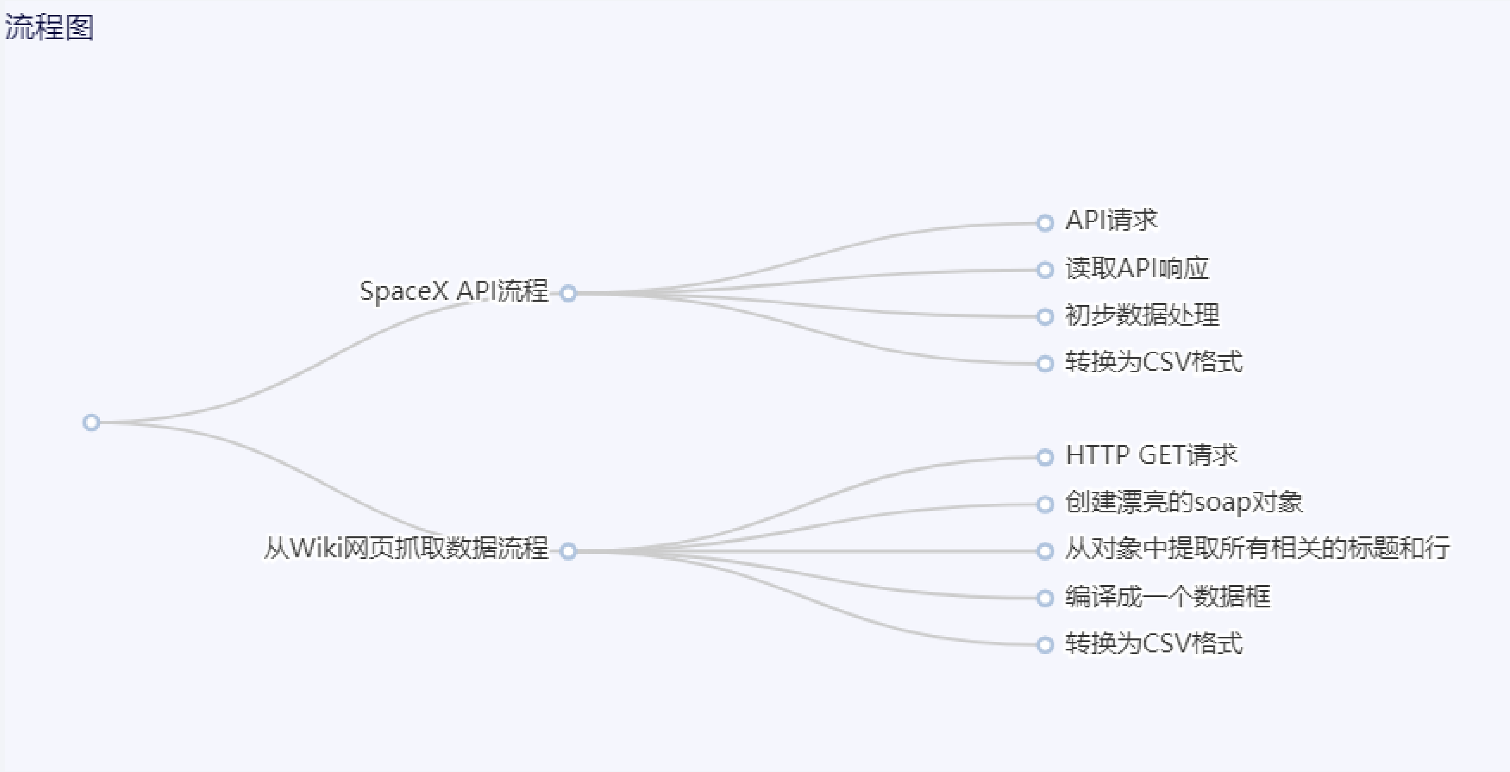
Methodology

Executive Summary

- Data collection methodology:
  - Web scrap Falcon 9 and Falcon Heavy launch records from Wikipedia
  - 维基百科上的“猎鹰9号”和“猎鹰重型”发射记录
- Perform data wrangling
  - By converting the outcomes of tasks into training labels to determine the labels for the supervised training model.
  - 通过将任务结果转换为训练标签来确定训练监督模型的标签
- 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

- 数据收集是从现有来源收集数据的过程。此数据可以是结构化、非结构化或半结构化。对于该项目，数据是通过SpaceX API和Web废弃相关发射数据的Wiki页面。
- Data collection is the process of gathering data from existing sources. This data can be structured, unstructured, or semi-structured. For this project, data was obtained through the SpaceX API and web-scraped related launch data from Wiki pages.



Data Collection – SpaceX API

1. **API Request and read response into DF**：首先，通过API请求获取数据，并将返回的结果读入到一个DataFrame中。DataFrame是一种常见的数据结构，用于存储表格型数据。
2. **Declare global variables**：在第二步中，声明一些全局变量。这些变量将在后续的步骤中被使用，可能是为了存储一些中间结果或者配置信息。
3. **Call helper functions with API calls to populate global vars**：第三步是调用一些辅助函数，通过API调用来填充之前声明的全局变量。这些辅助函数可能是为了获取更详细的信息或者处理特定的任务。
4. **Construct data using dictionary**：第四步是使用字典来构造数据。字典是一种Python内置的数据类型，用于存储键值对。在这个步骤中，可能会将之前获取的数据进行整理和重组，以便于后续的分析和处理。
5. **Convert Dict to Dataframe, filter for Falcon9 launches, covert to CSV**：最后一步是将字典转换为DataFrame，筛选出与Falcon9发射相关的信息，并将结果保存为CSV文件。CSV文件是一种常见的数据交换格式，便于在不同程序之间共享数据。

In [21]:
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

In [14]:
```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

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

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

In [10]:
```
response.status_code
```

Out[10]: 200

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

In [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

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

GIT

Data Collection - Scraping

- **CHINESE：**

1. 执行HTTP GET请求以获取HTML页面。
2. 创建BeautifulSoup对象，用于解析HTML文档。
3. 从HTML表头中提取列名。
4. 使用提取的列名作为键创建字典。
5. 调用辅助函数填充字典中的发射记录。
6. 将字典转换为数据框。

- **ENG：**

1. Execute an HTTP GET request to obtain the HTML page.
2. Create a BeautifulSoup object to parse the HTML document.
3. Extract the column names from the HTML table headers.
4. Create a dictionary using the extracted column names as keys.
5. Call a helper function to populate the launch records in the dictionary.
6. Convert the dictionary into a dataframe.

```python
def date_time(table_cells):
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

   # column_name = ' '.join(row.contents)
```

```python
In [ ]: column_names = []
#对于第一个发射表应用find_all()函数与th元素遍历每个th元素并应用提供的extract_column_from_header()函数以获取列名如果名称不为空
for x in range (len(colnames)):
    name2 = extract_column_from_header(colnames[x])
    if (name2 is not None and len(name2) > 3):
        column_names.append(name2)
```

Check the extracted column names

Data Wrangling

- 1. Load dataset to a Dataframe
- 1.将数据集加载到数据帧中
- 2. Value counts
- 2. 数值计算
- 3. Create landing outcome label
- 3.创建landing结果标签

[GIT]

Load Space X dataset, from last section.

```
In [2]: df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv
        df.head(10)
```

Out[2]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011

launches on each site:

```
In [5]: df['LaunchSite'].value_counts()

Out[5]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dtype: int64
```

column Orbit

```
In [6]: df['Orbit'].value_counts()

Out[6]: GTO      27
        ISS      21
        VLEO     14
        PO        9
        LEO       7
        SSO       5
        MEO       3
        GEO       1
        HEO       1
        SO        1
        ES-L1     1
        Name: Orbit, dtype: int64
```

EDA with Data Visualization

- 1.Scatter plot（散点图）：

- 散点图是一种常用的数据可视化图表，它通过在直角坐标系中显示两个变量的值来描绘数据点。在散点图中，每个点代表了数据集中的一个观察值，其中一个变量的值确定了点的水平位置（X轴），另一个变量的值确定了点的垂直位置（Y轴）。这种图表非常适合展示和分析两个连续变量之间的关系或相关性。
- 在探索性数据分析（EDA）中，散点图有以下用途：

  发现变量之间的关系,观察数据分布模式,识别异常值,评估变量间的相关性强度,多变量分析。

- A scatter plot is a commonly used data visualization chart that depicts data points by displaying the values of two variables in a Cartesian coordinate system. In a scatter plot, each point represents an observation from the dataset, with the value of one variable determining the point's horizontal position (X-axis) and the value of the other variable determining its vertical position (Y-axis). This type of chart is particularly suited for showing and analyzing the relationship or correlation between two continuous variables.
- In Exploratory Data Analysis (EDA), scatter plots serve the following purposes:
- Identifying relationships between variables - Observing patterns of data distribution - Detecting outliers - Assessing the strength of correlation between variables - Multivariate analysis.

EDA with Data Visualization

- 2.Bar chart（柱状图）：

- 柱状图是一种数据可视化图表，通过使用垂直或水平的条形来表示数据。每个条形的长度或高度与它所代表的数值成正比，这使得对不同类别或时间序列中的数值大小进行快速比较变得简单直观。柱状图通常用于展示分类数据的分布情况，或者比较不同组之间的数值差异。
- 在探索性数据分析（EDA）中，柱状图有以下几个用途：
- 比较类别，显示频率或计数，识别数据分布特征，时间序列分析。
- A bar chart is a data visualization tool that represents data using vertical or horizontal bars. The length or height of each bar is proportional to the value it represents, making it easy and intuitive to compare the magnitudes of values across different categories or over time. Bar charts are commonly used to show the distribution of categorical data, or to compare numeric values across different groups.
- In Exploratory Data Analysis (EDA), bar charts are useful for:
- Comparing Categories，Showing Frequency or Counts，Identifying Distribution Characteristics，Time Series Analysis.

EDA with Data Visualization

- 3.Line chart（折线图）：

- 折线图是一种通过连接数据点上的直线来展示信息变化趋势的图表。在直角坐标系中，通常以时间序列为横轴（X轴），数据值为纵轴（Y轴），通过折线连接各个数据点，形象地展示数据随时间或其他变量的变化情况。折线图特别适合用于展示数据随时间的变化趋势，因此在时间序列分析中非常常见。
- 在探索性数据分析（EDA）中，折线图有以下几个用途：
- 趋势分析,比较不同数据序列,高亮异常值,预测未来趋势
- A line chart is a type of chart that displays information as a series of data points connected by straight lines. It is typically used in a Cartesian coordinate system with the horizontal axis (X-axis) often representing time, and the vertical axis (Y-axis) representing the values of the data. By connecting the data points with lines, it visually shows how data changes over time or across different variables. Line charts are particularly useful for showing trends over time, making them common in time series analysis.
- In Exploratory Data Analysis (EDA), line charts are useful for:
- Trend Analysis,Comparing Different Data Series,Highlighting Anomalies,Forecasting Future Trends

EDA with SQL

- 1. Display the names of the unique launch sites in the space mission
- 2. Display 5 records where launch sites begin with the string 'CCA'
- 3. Display the total payload mass carried by boosters launched by NASA (CRS)
- 4. Display average payload mass carried by booster version F9 v1.1
- 5. List the date when the first successful landing outcome in ground pad was achieved.
- 6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- 7. List the total number of successful and failure mission outcomes
- 8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- 9. List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 10.Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Build an Interactive Map with Folium

1.Folium interactive map helps analyze geospatial data to perform more interactive visual analytics and better understand factors such location and proximity of launch sites that impact launch success rate.

Following map object were created and added to the map:

Mark all launch sites on the map. This allowed to visually see the launch sites on the map.

Added ‘folium.circle’ and ‘folium.marker’ to highlight circle area with a text label over each launch site.

2.Added a ‘MarkerCluster()’ to show launch success (green) and failure (red) markers for each launch site.

3.Calculated distances between a launch site to its proximities (e.g., coastline, railroad, highway, city)

Added ‘MousePosition() to get coordinate for a mouse position over a point on the map

Added ‘folium.Marker()’ to display distance (in KM) on the point on the map (e.g., coastline, railroad, highway, city)

Added ‘folium.Polyline()’ to draw a line between the point on the map and the launch site

Repeated steps above to add markers and draw lines between launch sites and proximities – coastline, railroad, highway, city)

Build a Dashboard with Plotly Dash

- Built a Plotly Dash web application to perform interactive visual analytics on SpaceX launch
- data in real-time. Added Launch Site Drop-down, Pie Chart, Payload range slide, and a
- Scatter chart to the Dashboard.
- 1. Added a Launch Site Drop-down Input component to the dashboard to provide an ability to filter
- Dashboard visual by all launch sites or a particular launch site
- 2. Added a Pie Chart to the Dashboard to show total success launches when ‘All Sites’ is selected and
- show success and failed counts when a particular site is selected
- 3. Added a Payload range slider to the Dashboard to easily select different payload ranges to identify
- visual patterns
- 4. Added a Scatter chart to observe how payload may be correlated with mission outcomes for selected
- site(s). The color-lab

Predictive Analysis (Classification)

CHN：

- 1. 将数据集读入Dataframe，并创建一个“类别”数组。
- 2. 对数据进行标准化处理。
- 3. 将数据划分为训练集和测试集。
- 4. 创建并优化模型。
- 5. 找到表现最佳的模型。

ENG：

- 1. Read dataset into Dataframe and create a 'Class' array.
- 2. Standardize the data.
- 3. Train/Test/Split data into training and test data sets.
- 4. Create and refine models.
- 5. Find the best performing Model.

Load the dataframe

Load the data

```
In [3]: data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_p
data.head()
```

Out[3]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004

```
In [4]: X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part
X.head(100)
```

Out[4]:

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS	...	Serial_B1058	Serial_B1059	Serial_E
0	1.0	6104.959412	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
1	2.0	525.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
2	3.0	677.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	
3	4.0	500.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
4	5.0	3170.000000	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
85	86.0	15400.000000	2.0	5.0	2.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
86	87.0	15400.000000	3.0	5.0	2.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	
87	88.0	15400.000000	6.0	5.0	5.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
88	89.0	15400.000000	3.0	5.0	2.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	
89	90.0	3681.000000	1.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	

90 rows × 83 columns

```
In [5]: Y = data['Class'].to_numpy()
```

```
In [6]: # students get this
X= preprocessing.StandardScaler().fit(X).transform(X)

In [7]: X[0:5]
```

```
In [8]: # Split data for training and testing data sets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)

Train set: (72, 83) (72,)
Test set: (18, 83) (18,)
```

we can see we only have 18 test samples.

```
In [9]: Y_test.shape
Out[9]: (18,)
```

```
]: # Create a DF for algorithm type and respective best scores

Model_Performance_df = pd.DataFrame({'Algo Type': ['Logistic Regression', 'SVM','Decision Tree','KNN'],
'Accuracy Score': [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_],
'Test Data Accuracy Score': [logreg_cv.score(X_test, Y_test), svm_cv.score(X_test, Y_test),
tree_cv.score(X_test, Y_test), knn_cv.score(X_test, Y_test)]})
```

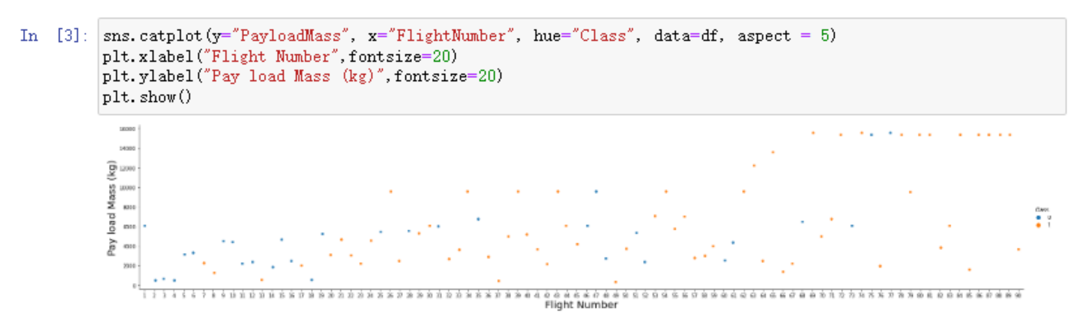
```
]: Model_Performance_df.sort_values(['Accuracy Score'], ascending = False, inplace=True)
```

```
]: Model_Performance_df
```

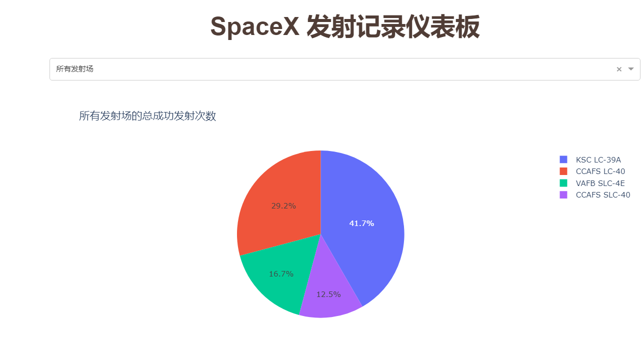
]:

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

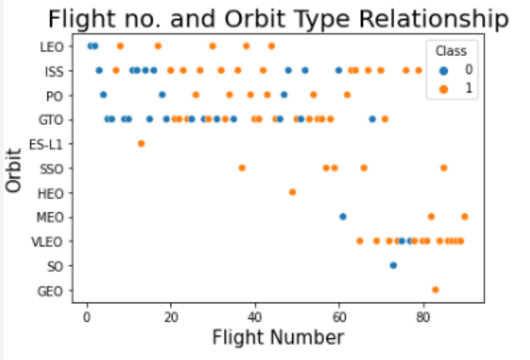
Results



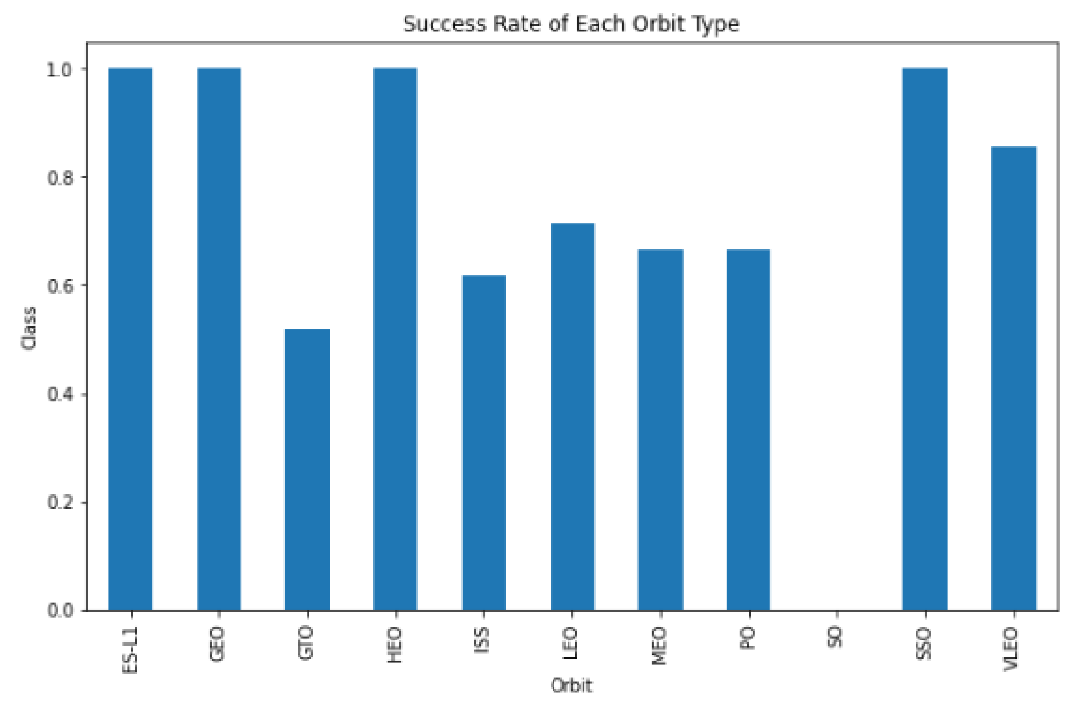
In [3]:
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
Pay load Mass (kg)
Flight Number



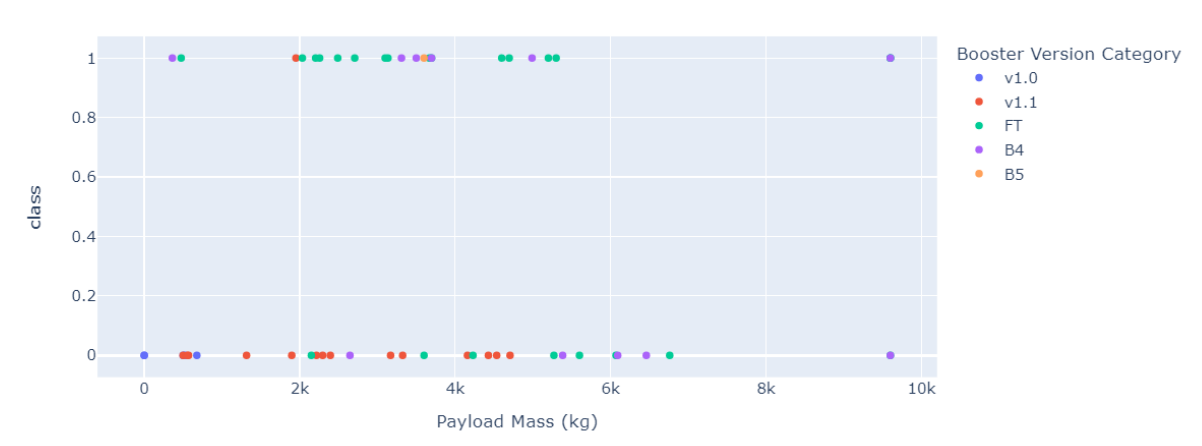
SpaceX 发射记录仪表板
所有发射场
所有发射场的总成功发射次数
KSC LC-39A
CCAFS LC-40
VAFB SLC-4E
CCAFS SLC-40
41.7%
29.2%
16.7%
12.5%



Flight no. and Orbit Type Relationship
Orbit
LEO
ISS
PO
GTO
ES-L1
SSO
HEO
MEO
VLEO
SO
GEO
Class
0
1
0
20
40
60
80
Flight Number



Success Rate of Each Orbit Type
Class
1.0
0.8
0.6
0.4
0.2
0.0
ES-L1
GEO
GTO
HEO
ISS
LEO
MEO
PO
SO
SSO
VLEO
Orbit



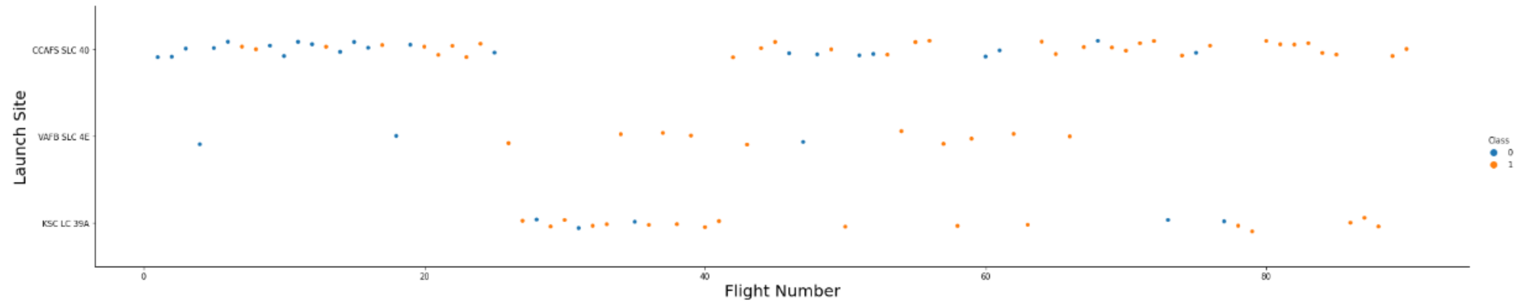
class
1
0.8
0.6
0.4
0.2
0
0
2k
4k
6k
8k
10k
Payload Mass (kg)
Booster Version Category
v1.0
v1.1
FT
B4
B5

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

```
In [4]: # 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("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



1.The success rate (Category = 1) increases with the number of launches.

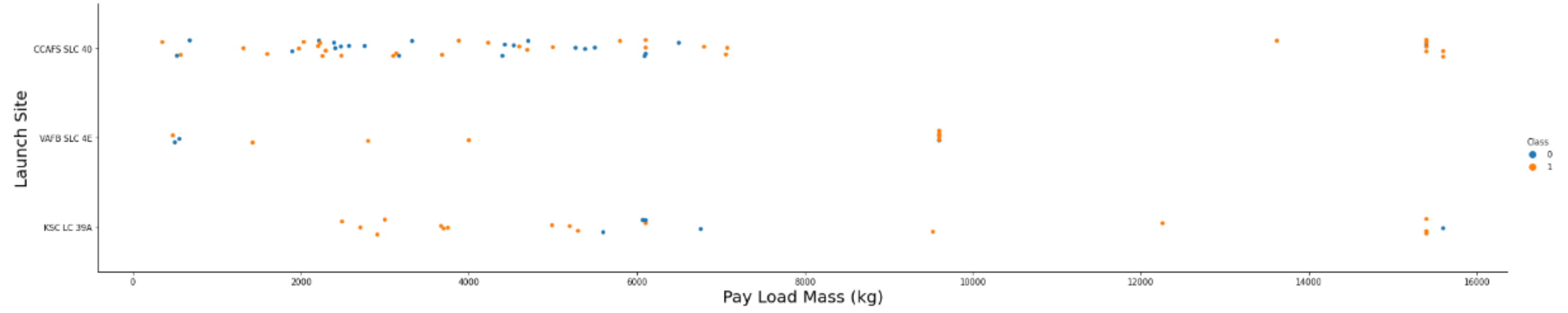
2.For the launch site "KSC LC 39A", at least about 25 launches are required to achieve the first successful launch.

•成功率（类别=1）随着飞行次数的增加而提高。

•对于发射场“KSC LC 39A”，至少需要大约25次发射才能实现首次成功发射。

Payload vs. Launch Site

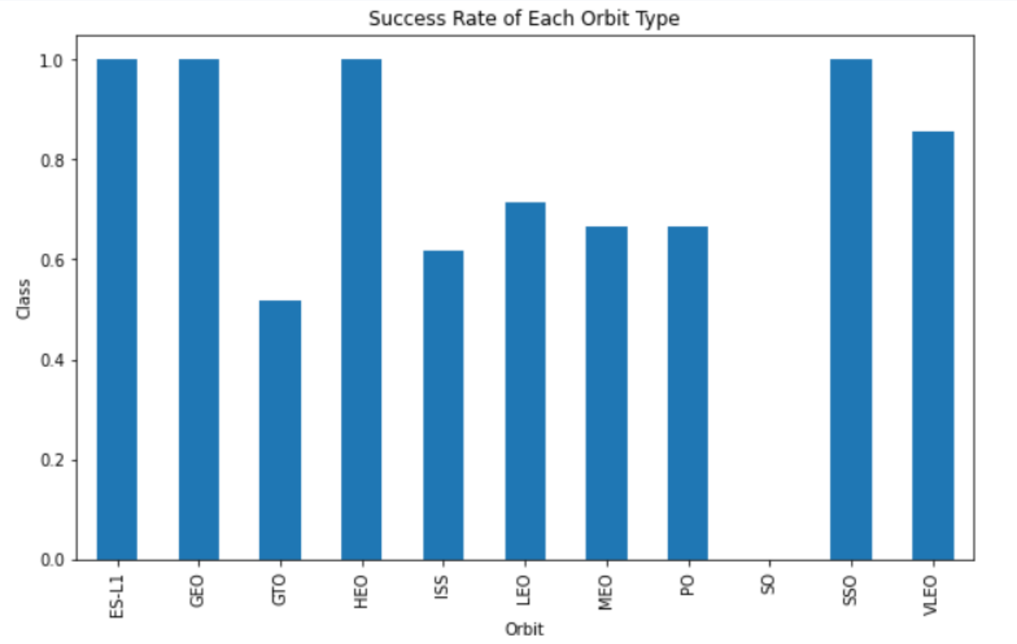
```
In [5]: sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
        plt.xlabel("Pay Load Mass (kg)",fontsize=20)
        plt.ylabel("Launch Site",fontsize=20)
        plt.show()
```



1.For the launch site "VAFB SLC 4E," no rockets have been launched with a payload greater than 10,000 kg.
2.The percentage of successful launches (Class=1) at the "VAFB SLC 4E" launch site increases with the increase in payload mass.
3.There is no clear correlation or pattern between the launch site and the mass of the payload.

1.对于“VAFB SLC 4E”发射场，没有发射有效载荷超过10000公斤的火箭。
2.“VAFB SLC 4E”发射场成功发射（1级）的百分比随着有效载荷质量的增加而增加。
3.发射场和有效载荷的质量之间没有明确的相关性或模式。

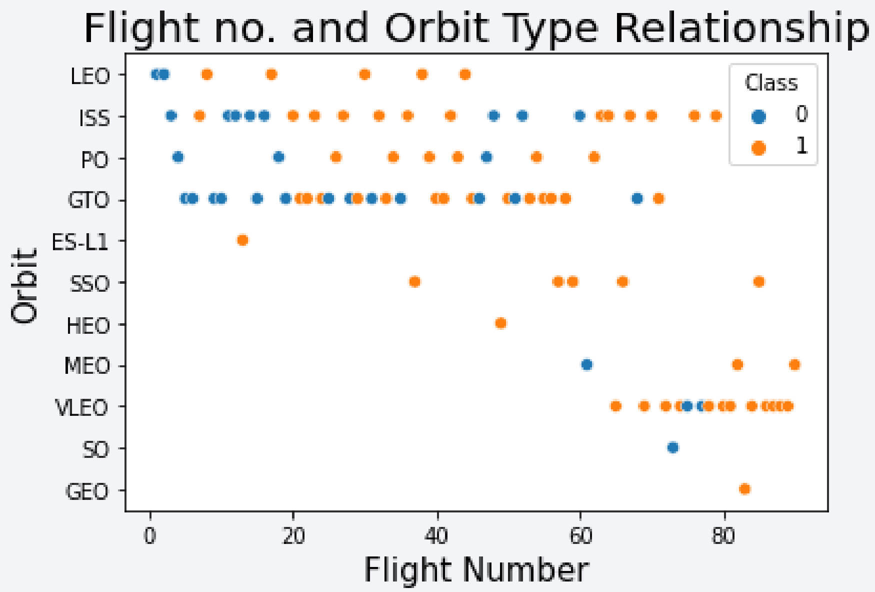
Success Rate vs. Orbit Type



1.Orbits ES-LI, GEO, HEO, and SSO have the highest success rates
2. GTO orbit has the lowest success rate

1.ES-LI、GEO、HEO和SSO轨道的成功率最高
2.GTO轨道的成功率最低

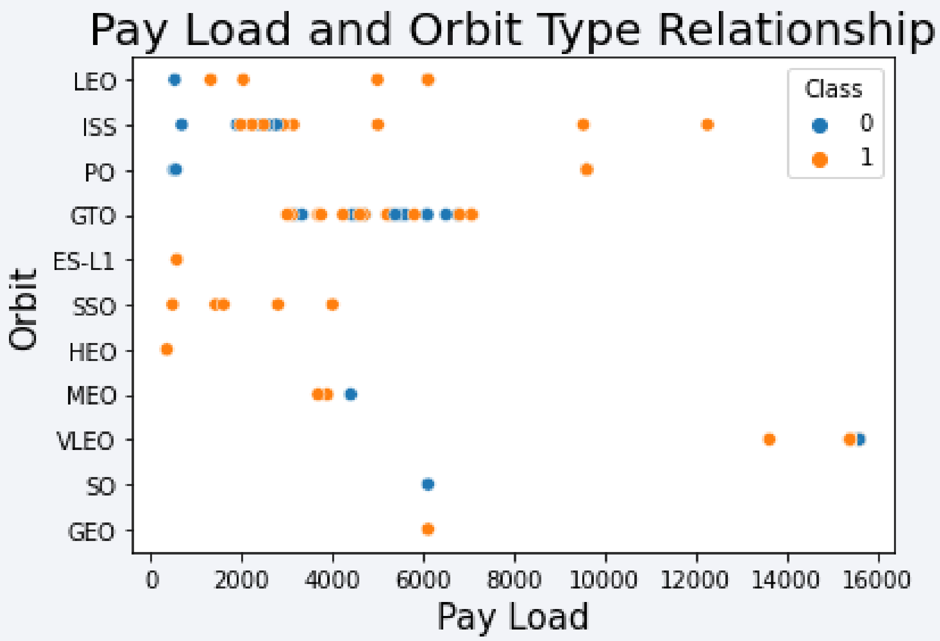
Flight Number vs. Orbit Type



- For VLEO orbits, the first successful landing (Category 1) occurred after more than 60 flight instances.
- For most orbits (LEO, ISS, PO, SSO, MEO, and VLEO), the success rate of landings appears to increase as the number of flights increases.
- For GTO, there is no relationship between the number of flights and the orbit.

- 对于VLEO轨道，首次成功着陆（类别为1）发生在60多的飞行次数之后
- 对于大多数轨道（LEO、ISS、PO、SSO、MEO、VLEO），成功着陆率似乎随着飞行次数的增加而增加
- 对于GTO，飞行次数与轨道之间没有关系

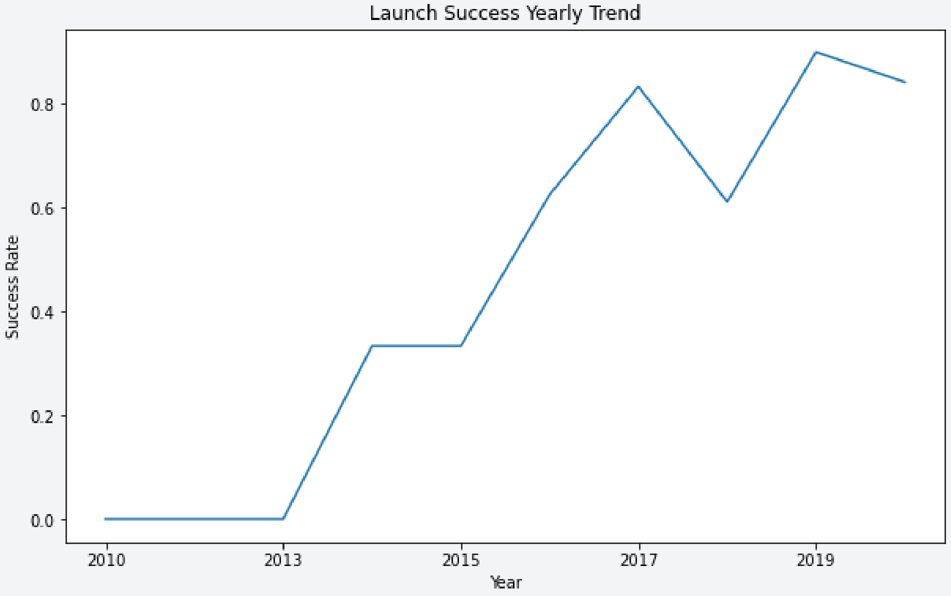
Payload vs. Orbit Type



With this graph we can further associate the success rate per orbit type to the relatively low payload used for the highest performance (ES-L1, GEO, HEO, SSO)

通过这张图表，我们可以进一步将各轨道类型的成功率与用于最高性能（ES-L1、GEO、HEO、SSO）的相对低有效载荷联系起来

Launch Success Yearly Trend



1.Success rate (Class=1) increased by about 80% between 2013 and 2020
2. Success rates remained the same between 2010 and 2013 and between 2014 and 2015
3.Success rates decreased between 2017 and 2018 and between 2019 and 2020

1.成功率（Class=1）在2013年至2020年间增加了约80%。
2.在2010年至2013年以及2014年至2015年间，成功率保持不变。
3.在2017年至2018年以及2019年至2020年间，成功率有所下降。

All Launch Site Names

```
%%sql
select DISTINCT(LAUNCH_SITE) from SPACEX
```

```
 * ibm_db_sa://bqn92294:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

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

In [10]:
```
%%sql

select * from spacextbl where Launch_Site LIKE 'CCA%' limit 5;
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.clogj3sd0tgtu01qde00.databases.appdomain.cloud:30426/bludb
Done.
```

Out[10]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Task 3

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

```
In [11]: %%sql

select sum(PAYLOAD_MASS__KG_) from spacextbl where Customer = 'NASA (CRS)'

 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.clogj3sd0tgtu01qde00.databases.appdomain.cloud:30426/bludb
Done.
```

Out[11]:

1
45596

Average Payload Mass by F9 v1.1

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

In [15]:
```
%%sql

select avg(PAYLOAD_MASS__KG_) from spacextbl where Booster_Version LIKE 'F9 v1.1';
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde0
0.databases.appdomain.cloud:30426/bludb
Done.
```

Out[15]:

1
2928

First Successful Ground Landing Date

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

Hint:Use min function

In [27]:
```
%%sql

select min(Date) as min_date from spacextbl where Landing__Outcome = 'Success (ground
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.clogj3sd0tgtu0lqde0
0.databases.appdomain.cloud:30426/bludb
Done.
```

Out[27]:

min_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

In [29]:
```
%%sql

select Booster_Version from spacextbl where (PAYLOAD_MASS__KG_> 4000 and PAYLOAD_MASS__KG_ < 6000)
and (Landing__Outcome = 'Success (drone ship)');
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

Out[29]:

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

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [6]:
```
%%sql

select Mission_Outcome, count(Mission_Outcome) as counts from spacextbl group by Mission_Outcome;
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

Out[6]:

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

Boosters Carried Maximum Payload

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

In [32]:
```
%%sql

select Booster_Version, PAYLOAD_MASS__KG_ from spacextbl where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextbl);
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

Out[32]:

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

2015 Launch Records

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

In [37]:

```
%%sql

select Landing__Outcome, Booster_Version, Launch_Site from spacextbl where Landing__Outcome = 'Failure (drone ship)' and year(Date) = '2
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

Out[37]:

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

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

In [39]:
```
%%sql

select Landing__Outcome, count(*) as LandingCounts from spacextbl where Date between '2010-06-04' and '2017-03-20'
group by Landing__Outcome
order by count(*) desc;
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

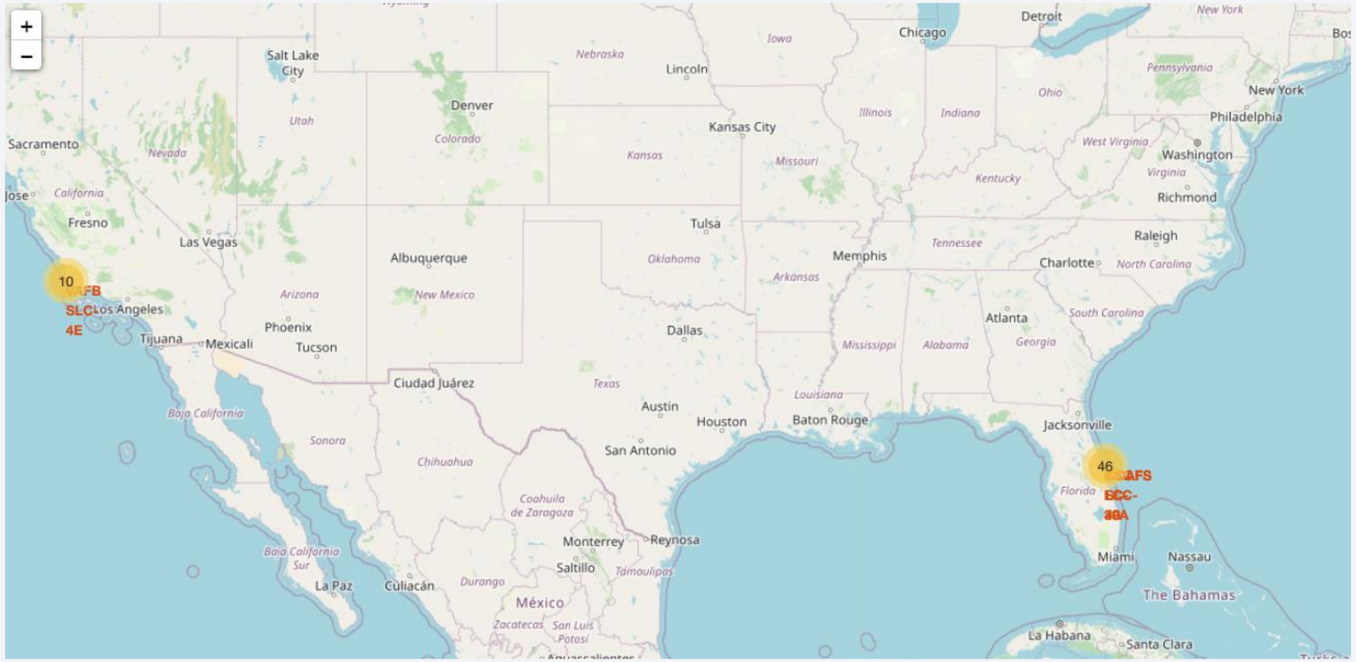
Out[39]:

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

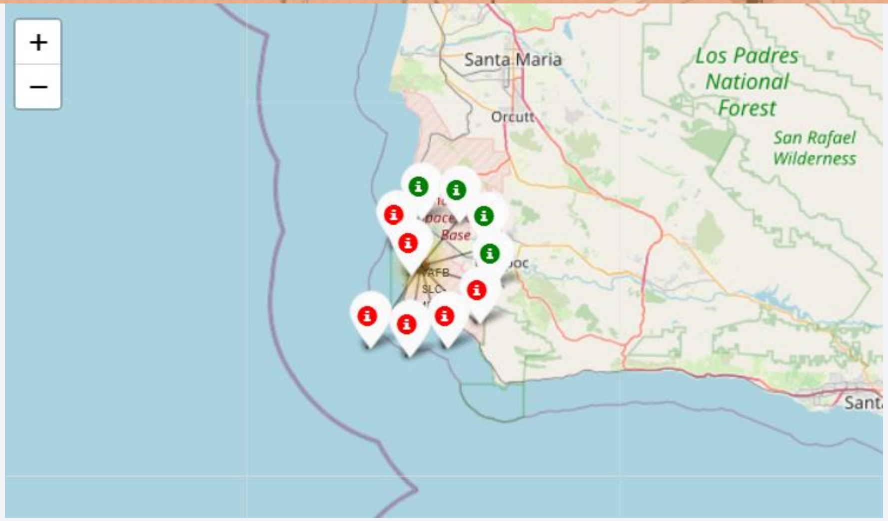
Section 3

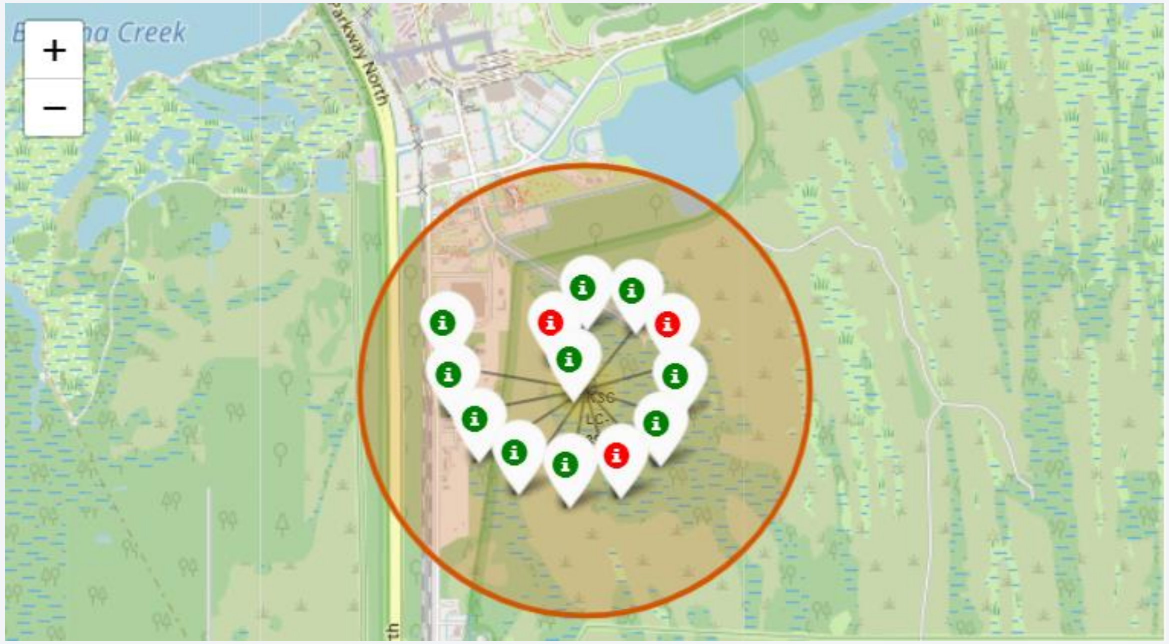
Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

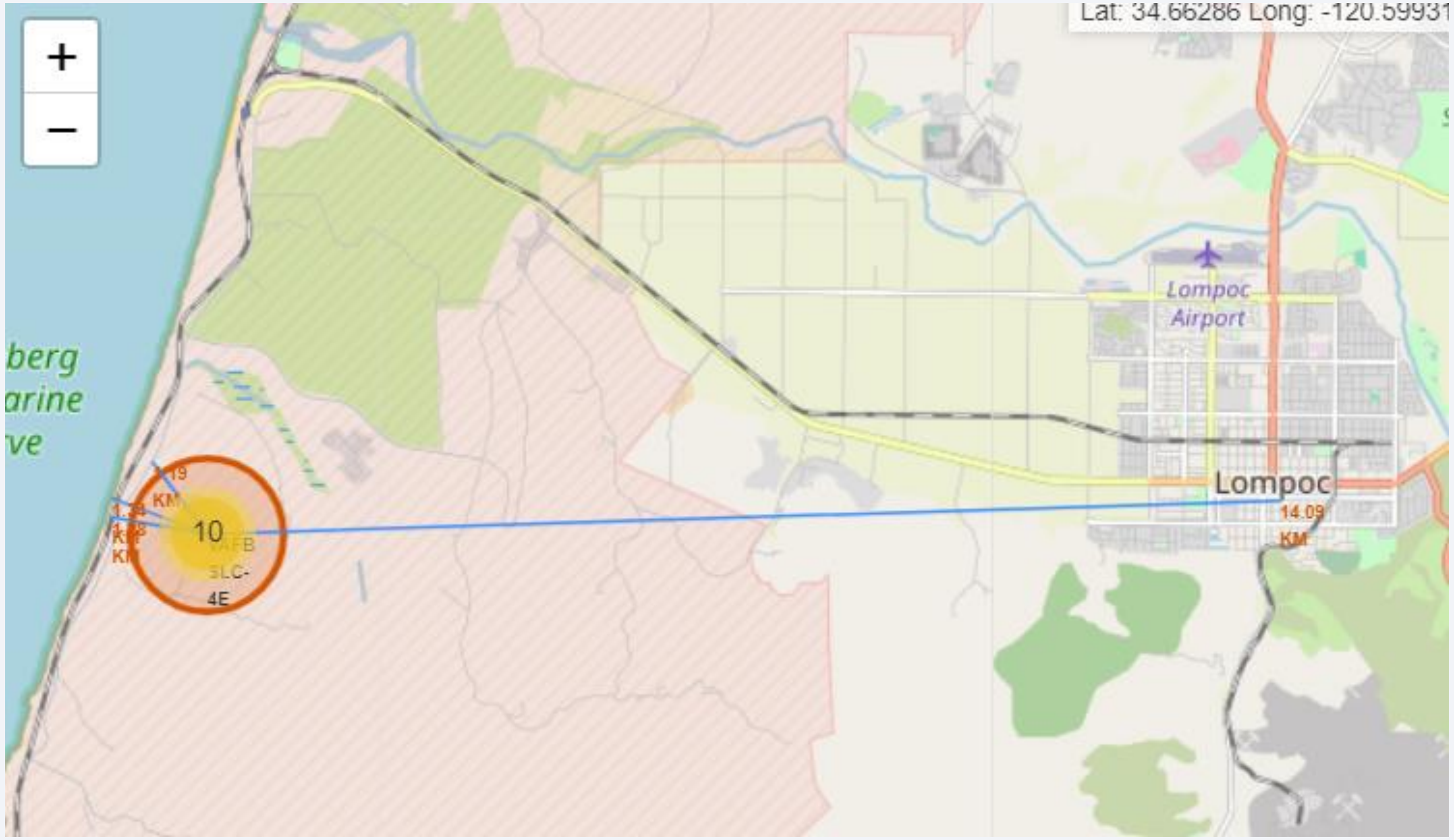


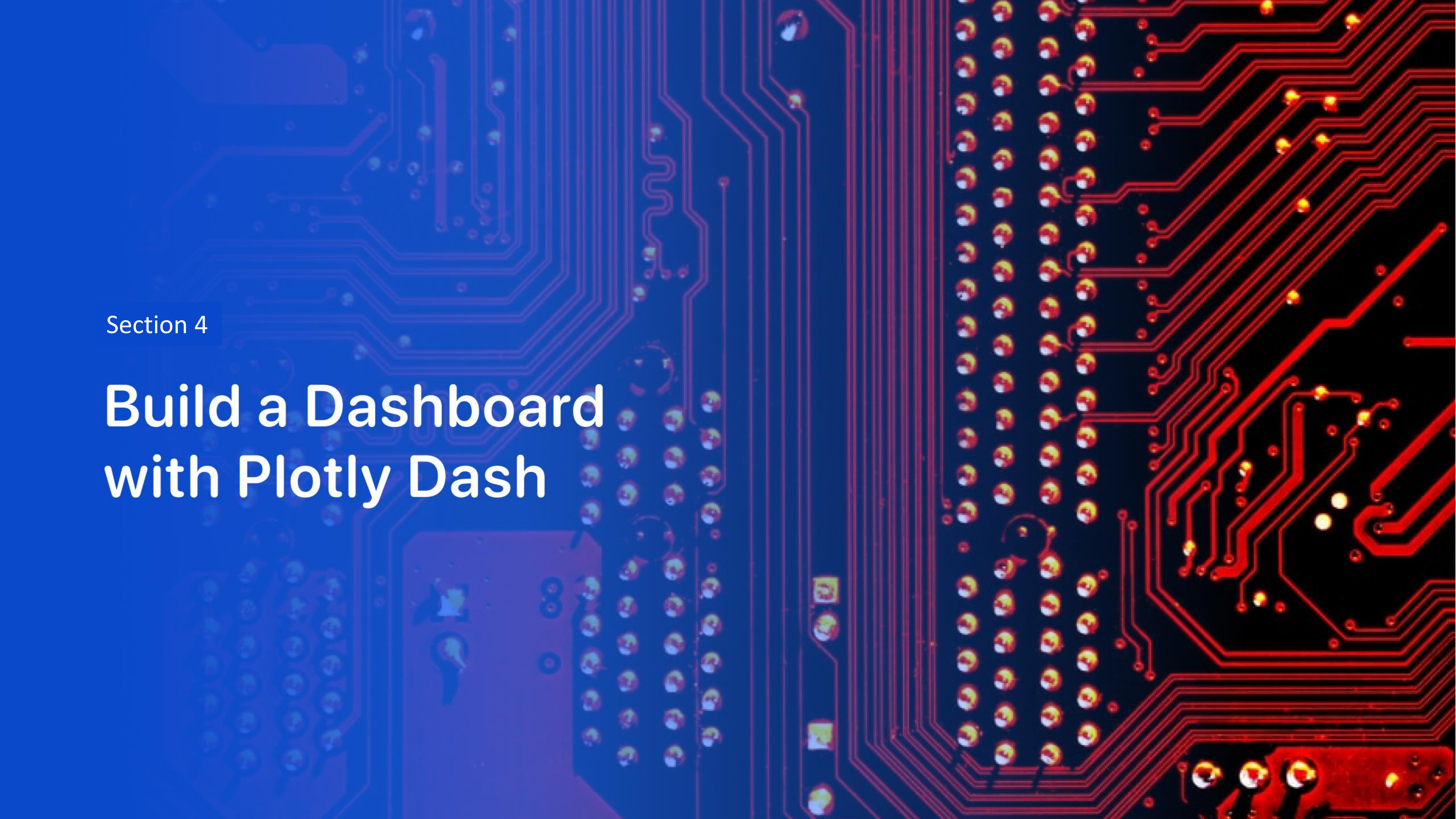
<Folium Map Screenshot 2>





<Folium Map Screenshot 3>

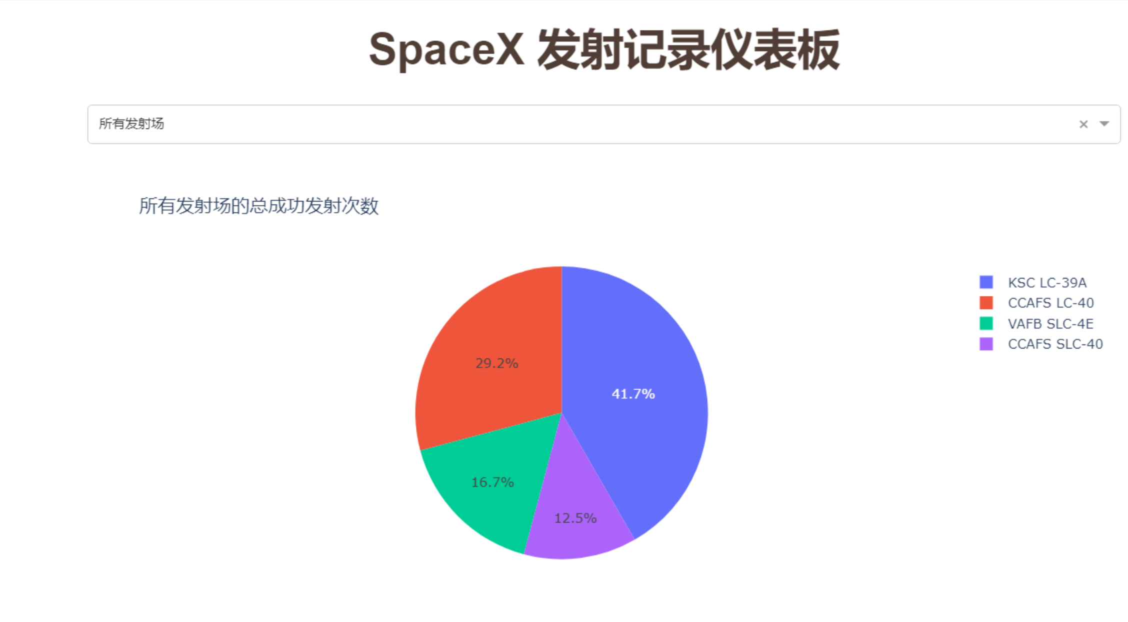




Section 4

Build a Dashboard with Plotly Dash

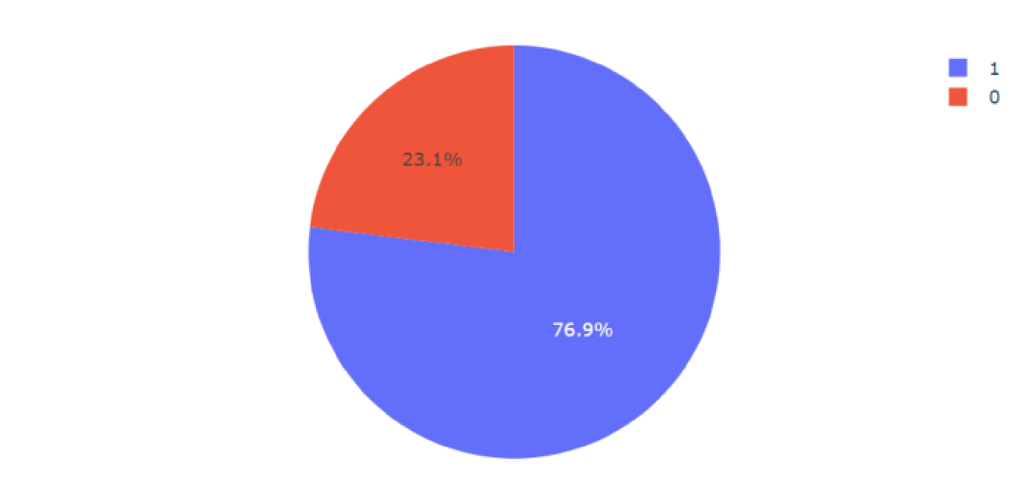
<Dashboard Screenshot 1>



1. Launch Site ‘KSC LC-39A’has the highest launch success rate.
2. Launch Site ‘CCAFS SLC40’ has the lowest launch success rate.

1.“KSC LC-39A”发射场的发射成功率最高
2.“CCAFS SLC40”发射场的发射成功率最低

<Dashboard Screenshot 2>



1.KSC LC-39A Launch Site has the highest launch success rate and count
2.Launch success rate is 76.9%
3. Launch success failure rate is 23.1%

1.肯尼迪航天中心LC-39A发射场拥有最高的发射成功率和发射次数
2.发射成功率为76.9%
3.发射失败率为23.1%

<Dashboard Screenshot 3>

1.最成功的发射的载荷范围通常在2000至约5500之间。
2.“FT”助推器版本类别的发射最为成功。
3.当载荷大于6k时，唯一成功发射的助推器是“B4”。

1.Most successful launches are in the payload range from 2000 to about 5500
2.Booster version category ‘FT’ has the most successful launches
3.Only booster with a success launch when payload is greater than 6k is ‘B4’

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Out[34]:

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

Decision Tree has the best accuracy(0.875)

决策树模型最优（0.875）

Confusion Matrix

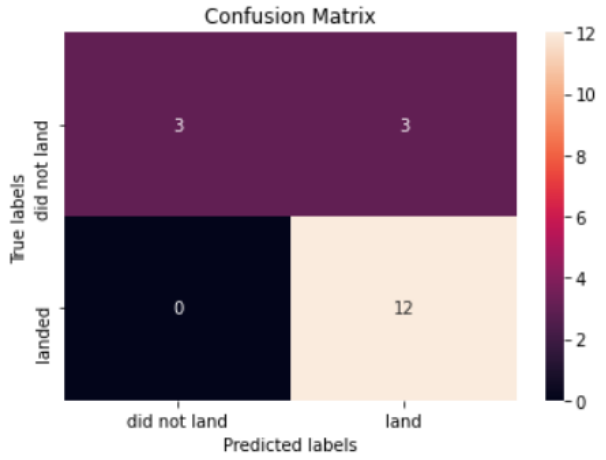
Calculate the accuracy on the test data using the method `score` :

In [13]:
```
print("Test data accuracy score - Logistic Regression: ", logreg_cv.score(X_test, Y_test))
```

```
Test data accuracy score - Logistic Regression:  0.8333333333333334
```

Lets look at the confusion matrix:

In [14]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- 随着航班数量的增加，第一阶段成功着陆的可能性更大
  •有效载荷增加时，成功率似乎上升，但有效载荷质量与成功率之间没有明确的关联
  •从2013年到2020年，发射成功率提高了约80%
  •发射场地“KSC LC-39A”的发射成功率最高，发射场地“CCAFS SLC40”的发射成功率最低
  •轨道ES-L1、GEO、HEO和SSO的发射成功率最高，轨道GTO最低
  •午餐地点位于远离城市、靠近海岸线、铁路和高速公路的战略位置
  •表现最好的机器学习分类模型是决策树，准确率约为87.5%。当模型在测试数据上进行评分时，所有模型的准确率得分约为83%。可能需要更多数据来进一步调整模型，并找到更好的潜在适应度。
- As the number of flights increases, the probability of successful landing in the first stage becomes higher.
- The success rate seems to increase with payload, but there is no clear correlation between payload mass and success rate.
- The launch success rate improved by about 80% from 2013 to 2020.
- The launch success rate is highest for the launch site "KSC LC-39A" and lowest for the launch site "CCAFS SLC40".
- The launch success rate is highest for orbits ES-L1, GEO, HEO, and SSO and lowest for orbit GTO.
- The lunch location is strategically positioned away from cities, near the coast, railway, and highways.
- The best-performing machine learning classification model was the decision tree, with an accuracy score of approximately 87.5%. When the models were scored on test data, all models had an accuracy score of around 83%. More data may be needed to further fine-tune the models and find better potential fitness.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!