IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

KORUKONDA JAYANTH KUMAR
19 July 2024

Outline

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

Executive Summary

- Summary of methodologies

  Data Collection through API

  Data Collection with Web Scraping

  Data Wrangling

  Exploratory Data Analysis with SQL

  Exploratory Data Analysis with Data Visualization

  Interactive Visual Analytics withFolium

  Machine Learning Prediction

- Summary of all results

  Exploratory Data Analysis result

  Interactive analytics in screenshots

  Predictive Analytics result

Introduction

- Project background and context

  Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

What factors determine if the rocket will land successfully?

The interaction amongst various features that determine the success rate of a successful landing.

What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

The data was collected using various methods

Data collection was done using get request to the SpaceX API.

Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().

We then cleaned the data, checked for missing values and fill in missing values where necessary.

In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.

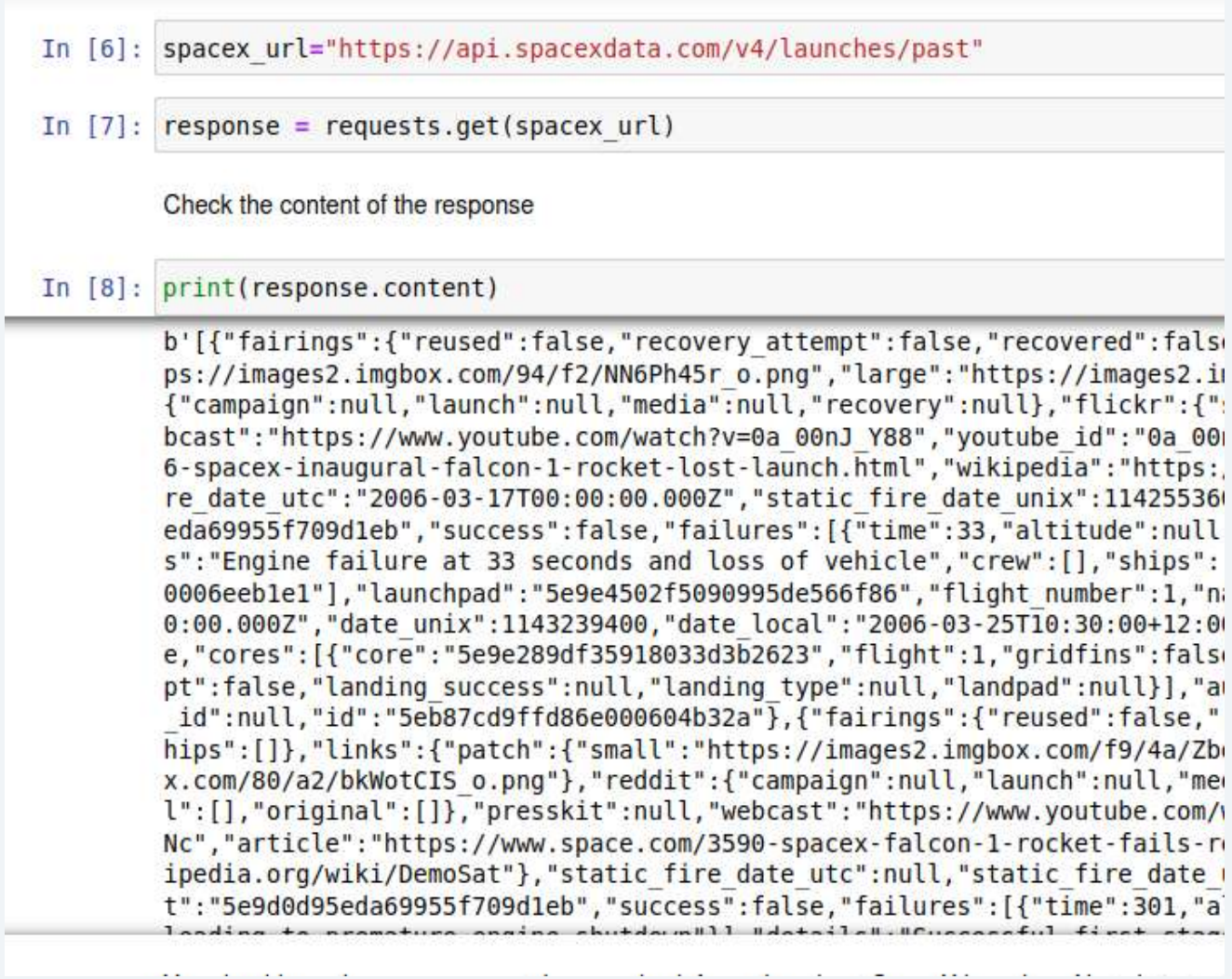
The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.

- The GitHub URL

https://github.com/KJayanth-10/IDM-Data-Science/blob/main/SpaceX-Data-Collection-API.ipynb



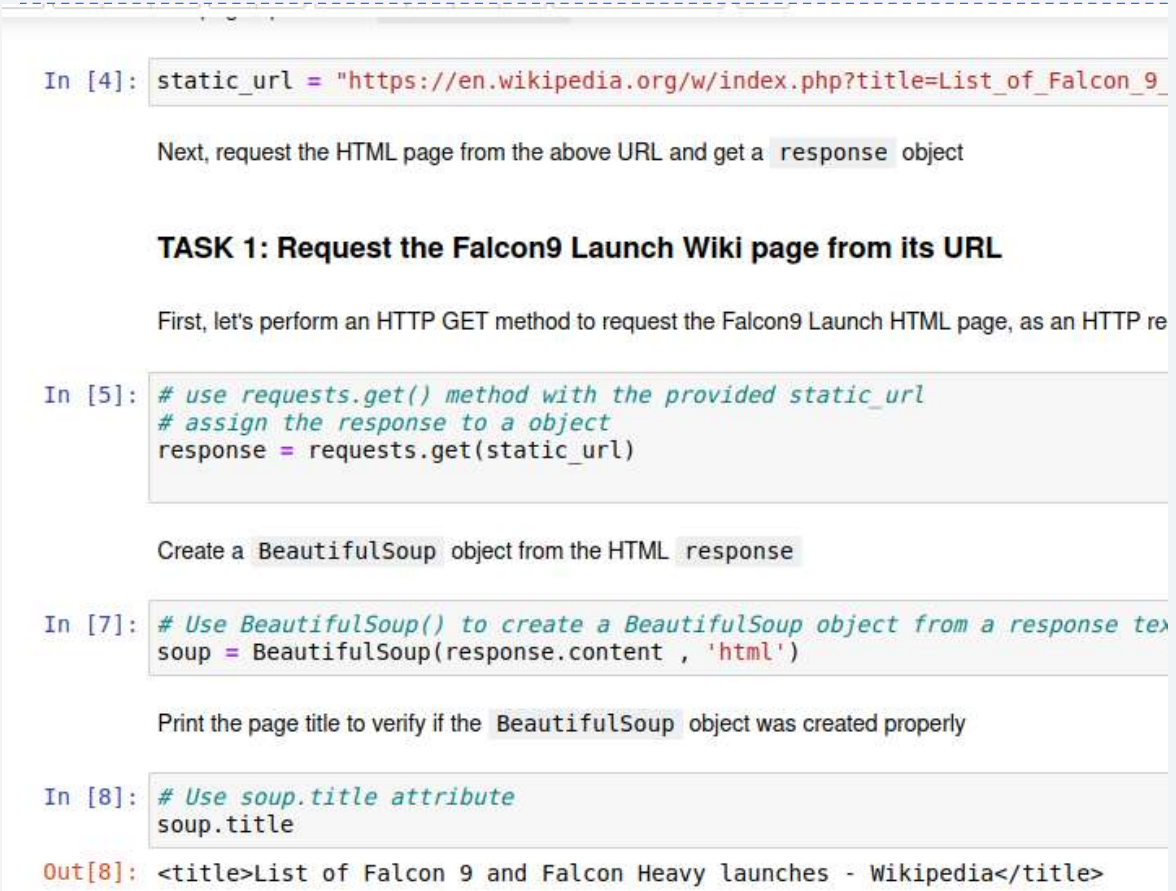
Data Collection - Scraping

We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup

We parsed the table and converted it into a pandas dataframe

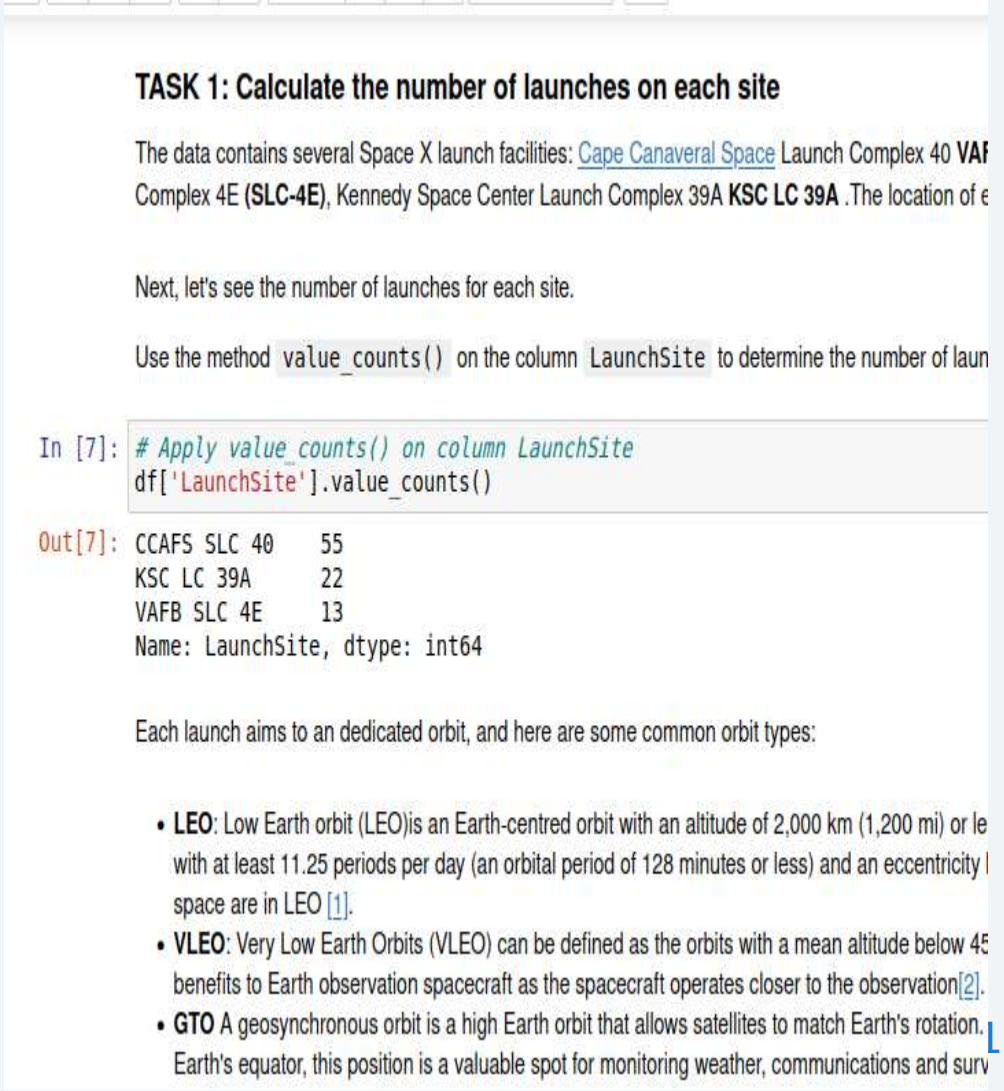
The Git Hub URL

https://github.com/KJayanth-10/IDM-Data-Science/blob/main/SpaceX-Webscraping.ipynb



Data Wrangling

We performed exploratory data analysis and determined the training labels.
We calculated the number of launches at each site, and the number and occurrence of each orbits
We created landing outcome label from outcome column and exported the results to csv.
TheGITHUB URL https://github.com/KJayanth-10/IDM-Data-Science/blob/main/SpaceX-Data_Wrangling.jupyterlite.ipynb



TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAF** Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of e

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 laun

```
In [7]: # Apply value_counts() on column LaunchSite
        df['LaunchSite'].value_counts()

Out[7]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dtype: int64
```

Each launch aims to an dedicated orbit, and here are some common orbit types:

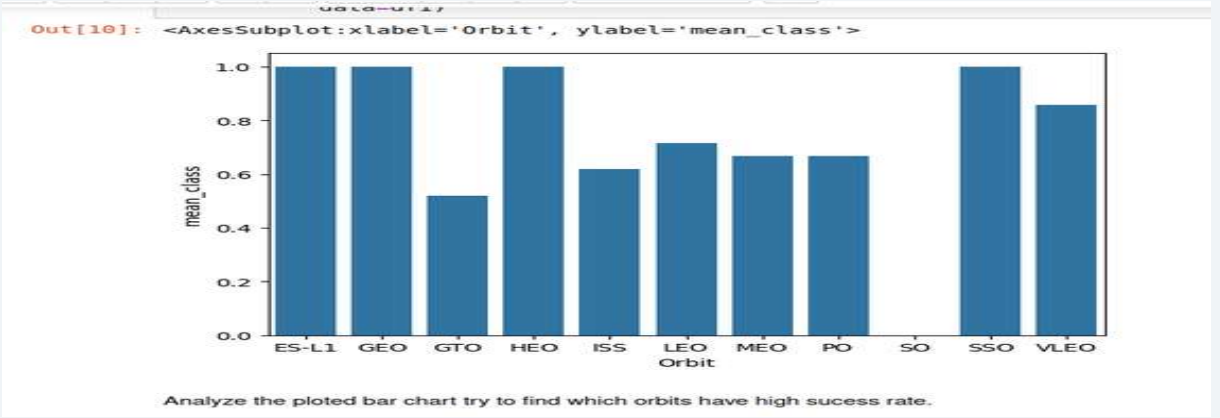
- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or le with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity l space are in LEO [1].
- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 45 benefits to Earth observation spacecraft as the spacecraft operates closer to the observation[2].
- **GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Earth's equator, this position is a valuable spot for monitoring weather, communications and surv

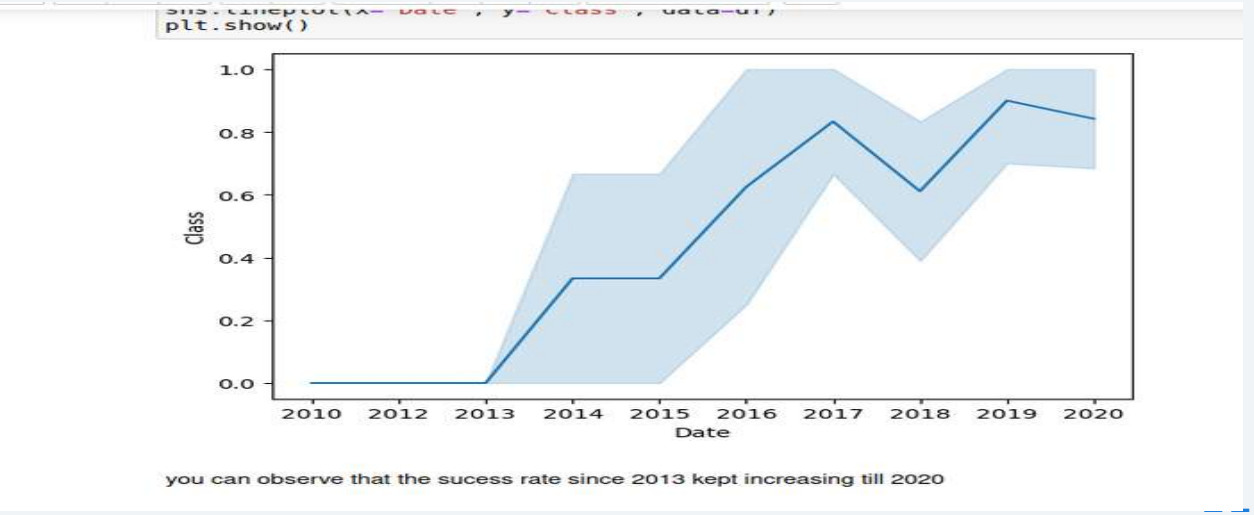
EDA with Data Visualization

We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.

The Git Hub Link

https://github.com/KJayanth-10/IDM-Data-Science/blob/main/SpaceX-EDA-DataViz.ipynb.jupyterlite.ipynb





EDA with SQL

We loaded the SpaceX dataset into a SQL database without leaving the jupyter notebook.

We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:

The names of unique launch sites in the space mission.

The total payload mass carried by boosters launched by NASA (CRS)

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

The total number of successful and failure mission outcomes

The failed landing outcomes in drone ship, their booster version and launch site names.

The link to the notebook is https://github.com/KJayanth-10/IDM-Data-Science/blob/main/SpaceX-EDA-SQL-edX_sqllite.ipynb

Build an Interactive Map with Folium

We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.

We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.

Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.

We calculated the distances between a launch site to its proximities. We answered some question for instance:

Are launch sites near railways, highways and coastlines.

Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

We built an interactive dashboard with Plotly dash

We plotted pie charts showing the total launches by a certain sites

We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

Predictive Analysis (Classification)

We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.

We built different machine learning models and tune different hyperparameters using GridSearchCV.

We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.

We found the best performing classification model.

The link to the notebook is https://github.com/KJayanth-10/IDM-Data-Science/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite(1).ipynb

Results

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

Section 2

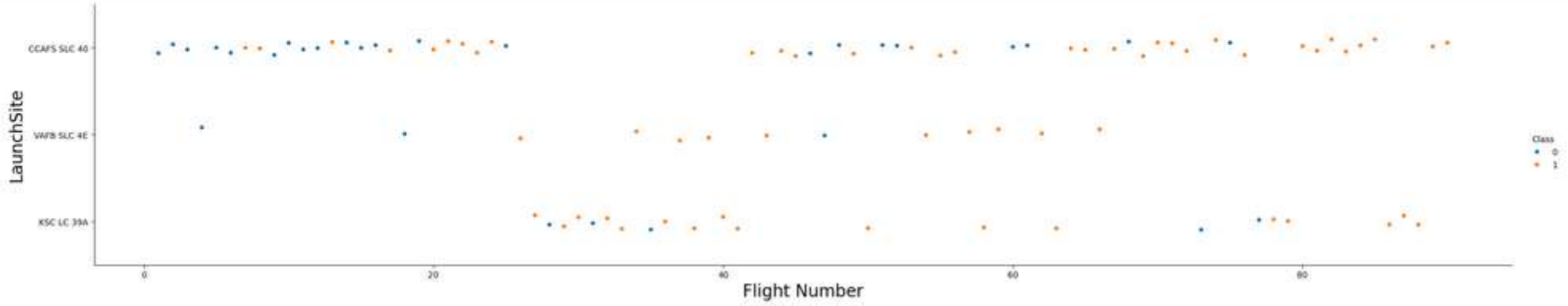
Insights drawn from EDA

Flight Number vs. Launch Site

From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a 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'`

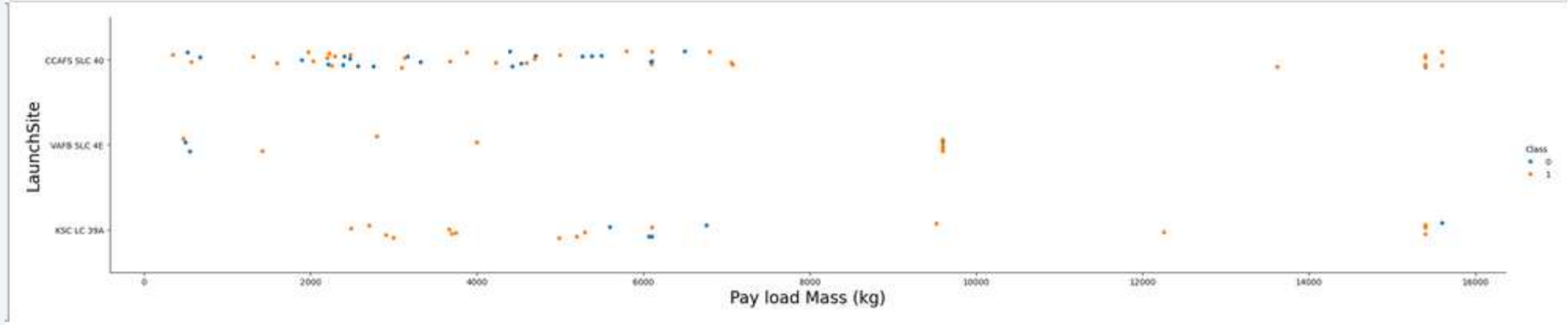
In [5]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the clas
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



Payload vs. Launch Site

- The greater the payload mass for launch site CCAFS SLC 40 ,The higher the Success

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



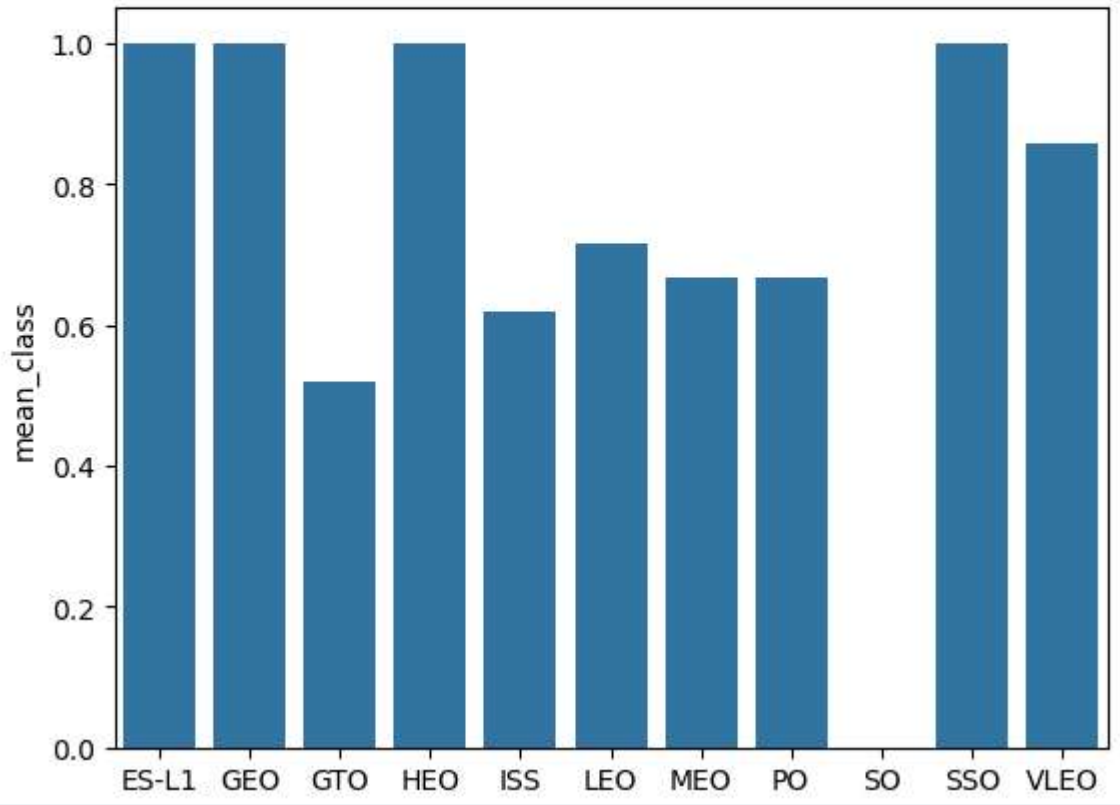
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

```
# plot barplot
sns.barplot(x='Orbit',
            y="mean_class",

            data=df1)
```

```
Out[10]: <AxesSubplot:xlabel='Orbit', ylabel='mean_class'>
```

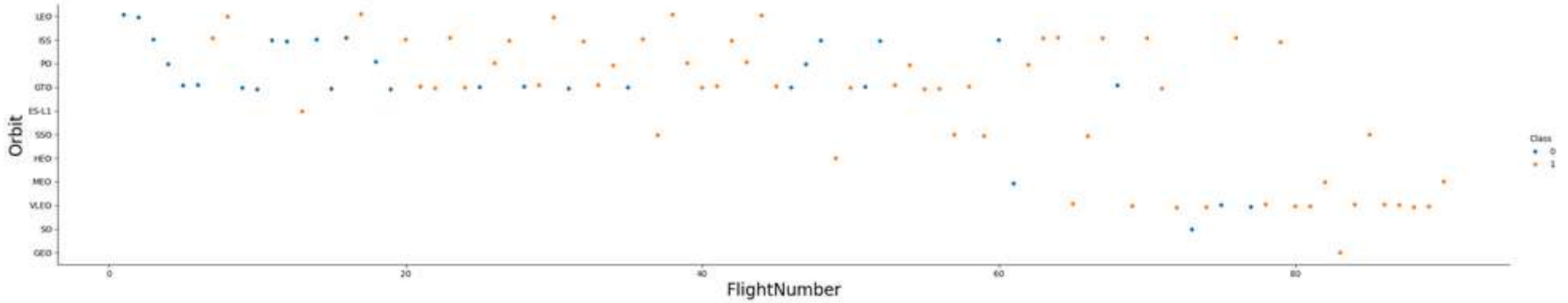


Flight Number vs. Orbit Type

The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

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

```
In [11]: # 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(x="FlightNumber", y="Orbit", hue="Class", data=df, aspect = 5)
plt.ylabel("Orbit",fontsize=20)
plt.xlabel("FlightNumber",fontsize=20)
plt.show()
```

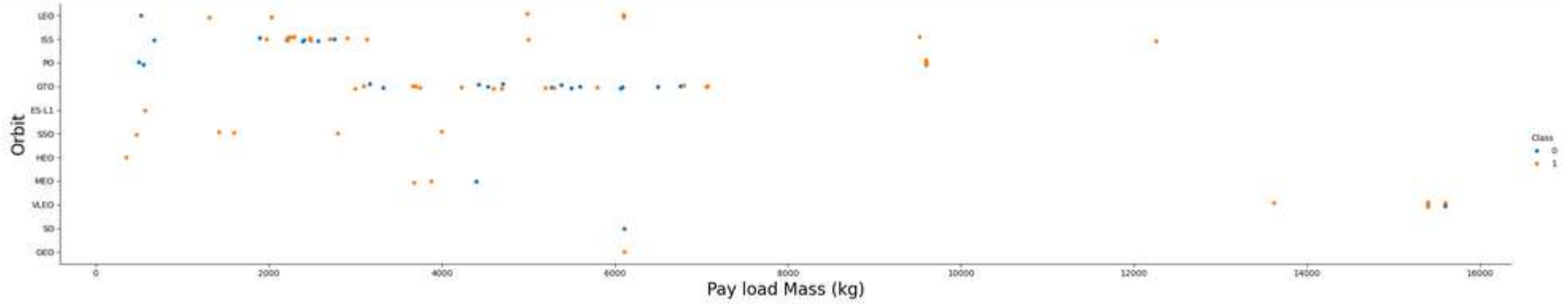


Payload vs. Orbit Type

We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits

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

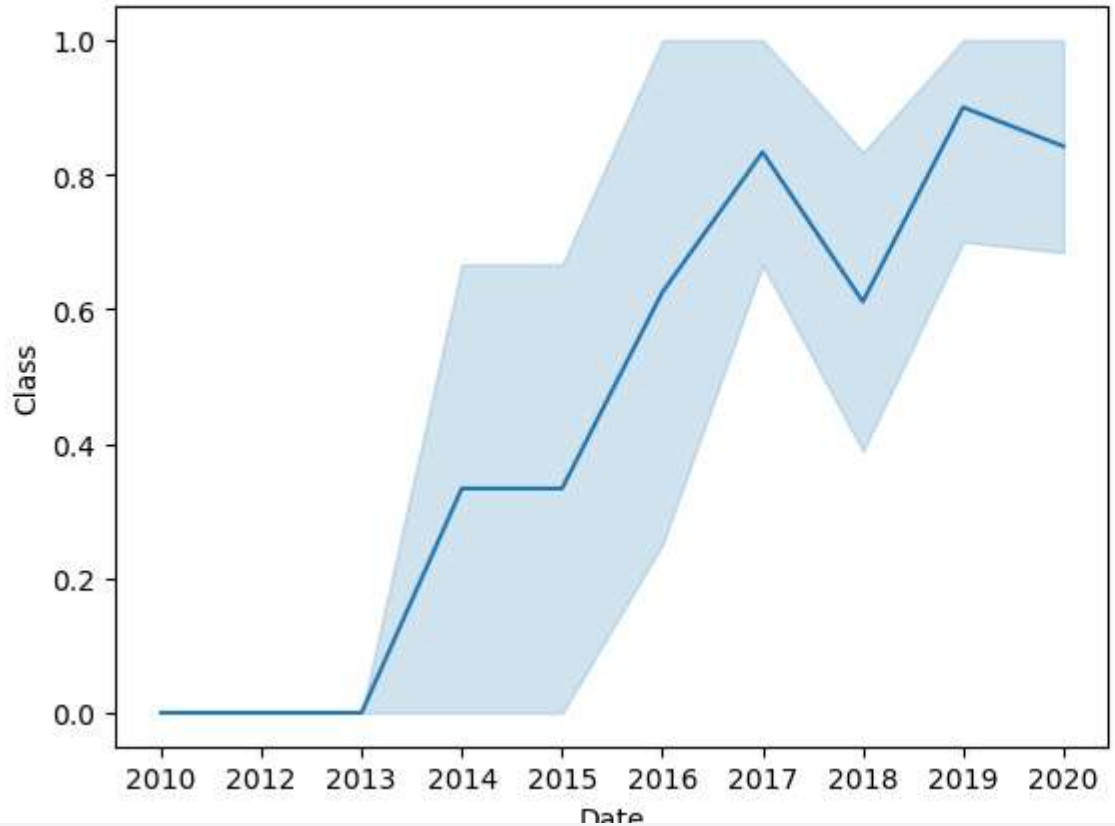
In [12]:
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value

sns.catplot(x="PayloadMass", y="Orbit", hue="Class", data=df, aspect = 5)
plt.ylabel("Orbit",fontsize=20)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.

```
sns.lineplot(x="Date", y="Class", data=df)
plt.show()
```



All Launch Site Names

We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

```
]: %sql select DISTINCT "Launch_Site" from SPACEXTABLE;
 * sqlite:///my_data1.db
Done.
```

]:

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

Launch Site Names Begin with 'KSC'

- We use like in query to retrieve launch sites names begin with ‘KSC’

```
In [10]: %sql select * from SPACEXTABLE where "Launch_Site" like "KSC%" limit 5;
 * sqlite:///my_data1.db
Done.
```

Out[10]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass

We calculated the total payload carried by boosters from NASA as 45596 using the query below

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

In [11]:
```
%sql select SUM("PAYLOAD_MASS__KG_") from SPACEXTABLE where "Customer" = "NASA (CRS)";
```

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

Out[11]:

SUM("PAYLOAD_MASS__KG_")
45596

Task 4

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

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

In [12]:
```
%sql select avg("PAYLOAD_MASS__KG_") from SPACEXTABLE where "Booster_Version" = "F9 v1.1";
```

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

Out[12]:

avg("PAYLOAD_MASS__KG_")
2928.4

Task 5 ¶

First Successful Ground Landing Date

- At 2016-04-08

Task 5

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

In [13]:
```
%sql select "Date" from SPACEXTABLE where "Landing_Outcome" = 'Success (drone ship)';
```

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

Out[13]:

Date
2016-04-08
2016-05-06
2016-05-27

Successful Drone Ship Landing with Payload between 4000 and 6000

We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

Task 6

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

In [14]:
```
%sql select "Booster_Version" from SPACEXTABLE where "Landing_Outcome" = 'Success (ground pad)' and "PAYLOAD_MASS__KG
```

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

Out[14]:

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

We used groupby to filter MissionOutcome was a success or a failure.

```
In [22]: %sql select "Mission_Outcome",count("Mission_Outcome") from SPACEXTABLE group by "Mission_Outcome";
 * sqlite:///my_data1.db
Done.
```

Out[22]:

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

Task 8

Boosters Carried Maximum Payload

We determined the booster that have carried the maximum payload using a subquery in the **FROM** clause and the ORDER BY **clause**.

```
In [27]: %sql select "PAYLOAD_MASS__KG_","Booster_Version" from SPACEXTABLE order by "PAYLOAD_MASS__KG_" DESC;
 * sqlite:///my_data1.db
Done.
```

Out[27]:

PAYLOAD_MASS__KG_	Booster_Version
15600	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

2015 Launch Records

We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2017

```
In [28]: %sql select "Booster_Version",Launch_Site from SPACEXTABLE where "Landing_Outcome" = "Success (ground pad)" and substr("Date",0,5) = '2017';
 * sqlite:///my_data1.db
Done.
```

Out[28]:

substr("Date",6,2)	Landing_Outcome	Booster_Version	Launch_Site
02	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
05	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
06	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
08	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
09	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

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

We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.

We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

```
In [37]: %sql select "Landing_Outcome",count("Landing_Outcome") from SPACEXTABLE where "Date" between "2010-06-04" and '2017-0
```

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

Out[37]:

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

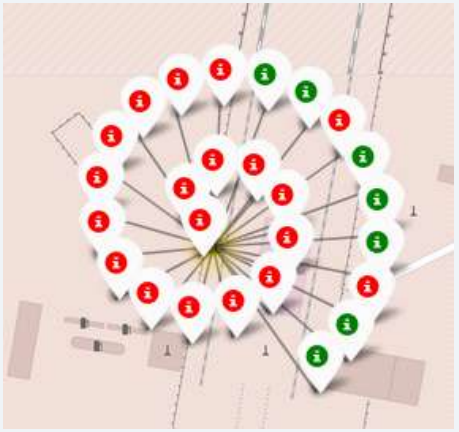
Section 3

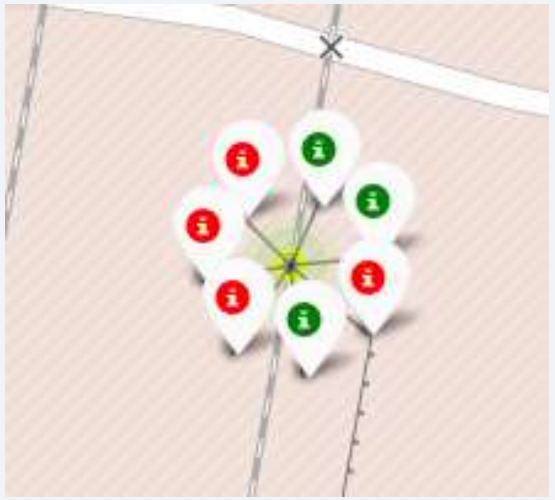
Launch Sites Proximities Analysis

Launching Sites

- All launchsites are close to equator

Launch Sites Success and Failures

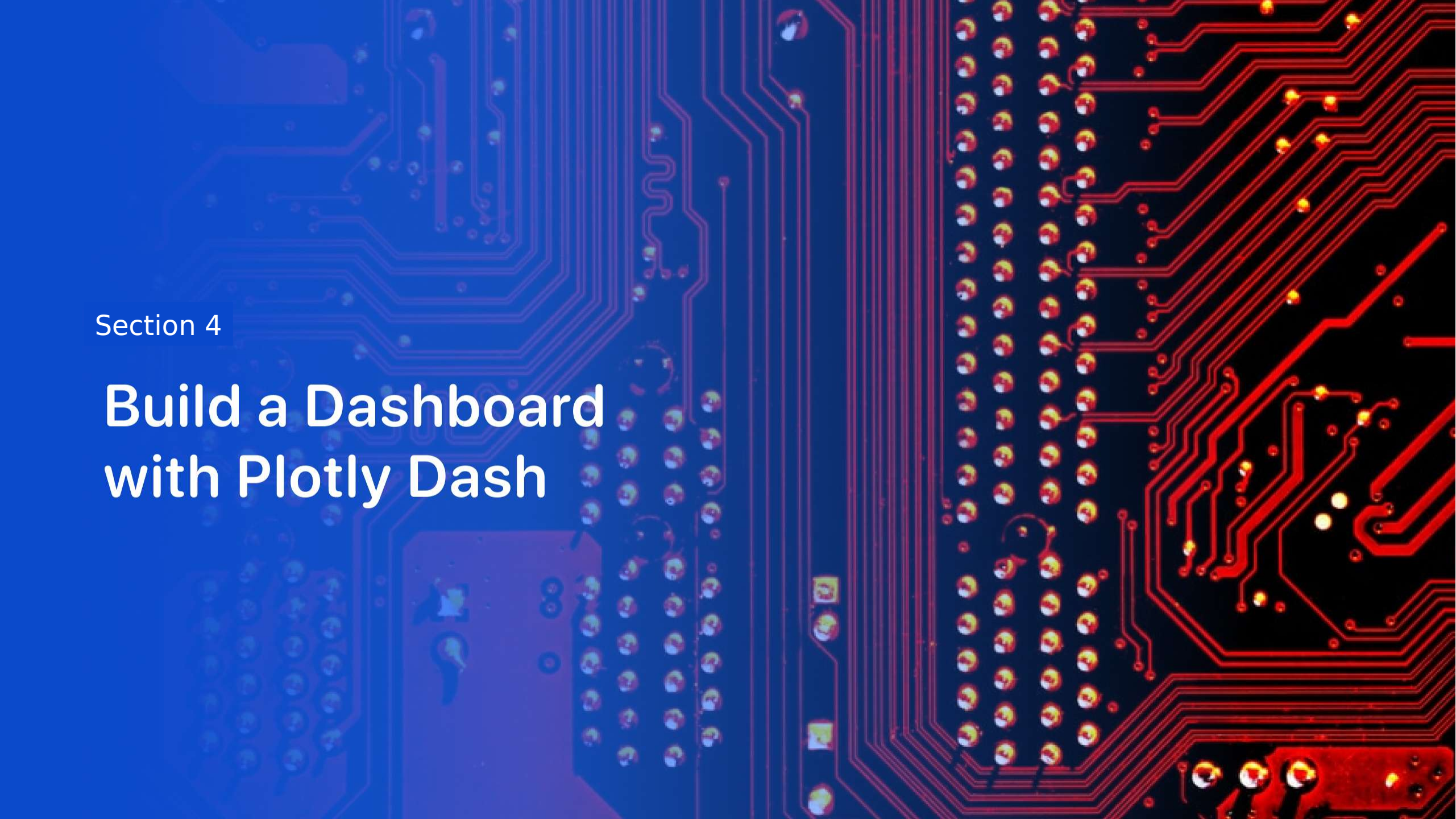




Launch site distances

They all away from certain distances from cities and they are neat to coastline

```
    )
lines = folium.PolyLine([[28.56319,-80.57682],[28.57138,-80.58538]],weight = 1)
site_map.add_child(lines)
site_map.add_child(distance_marker)
```


Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

There has been some issue with my port 8050 it didn’t open fully and show me half project so I cant project my findings please kindly

Excuse me

<Dashboard Screenshot 2>

Replace <Dashboard screenshot 2> title with an appropriate title

Show the screenshot of the piechart for the launch site with highest launch success ratio

Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

Replace <Dashboard screenshot 3> title with an appropriate title

Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Find the method performs best:

the methods that performs best is DecisionTreeClassifier with score 0.88

i dont know whether to write a code for checking jaccard_similarity_score or f1_score but i think the score would be sufficient to tell the performance if not please enlighten me in comment section

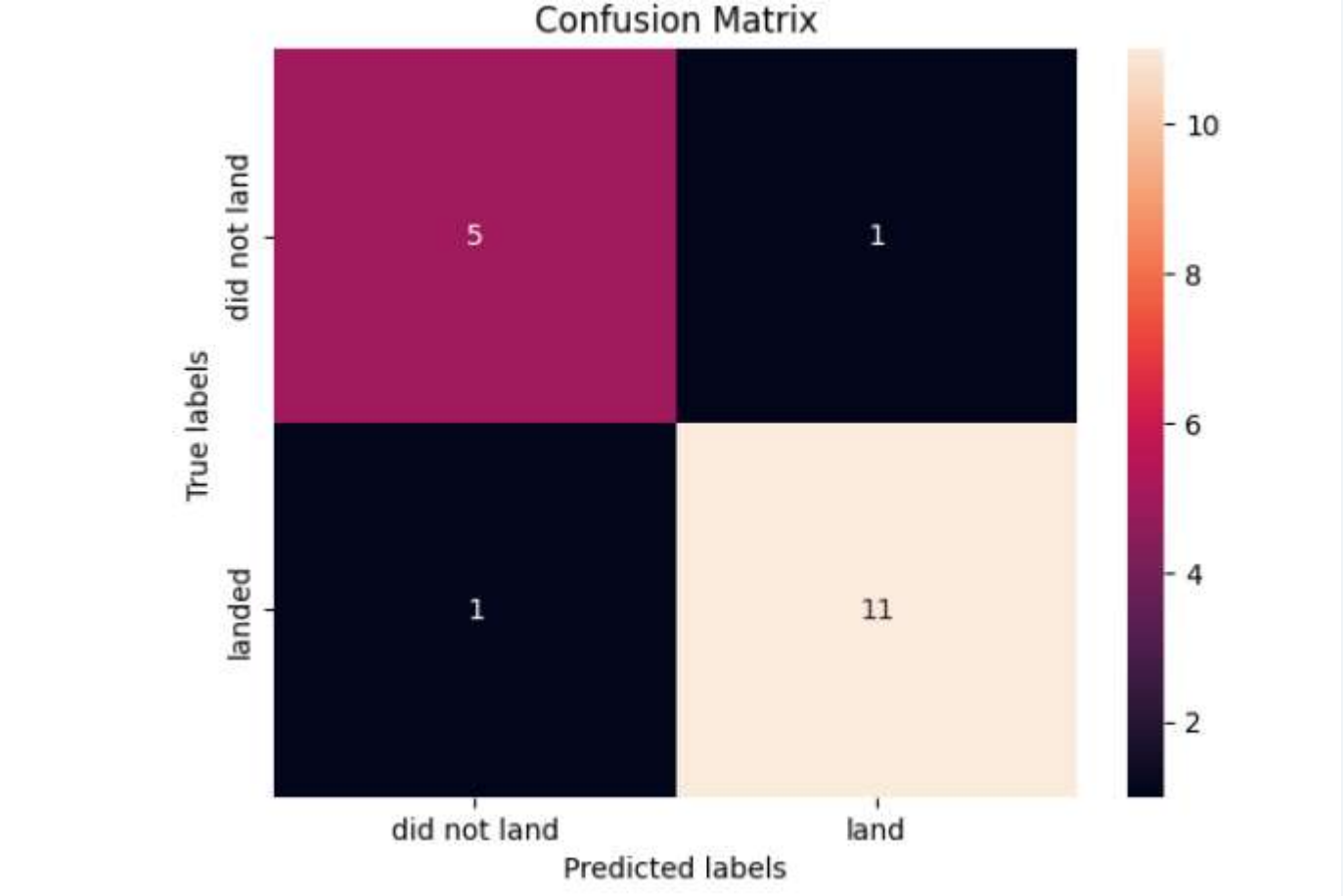
TASK 9

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

```
In [68]: tree_cv.score(X_test,Y_test)
Out[68]: 0.8888888888888888
```

Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

We can conclude that:

The larger the flight amount at a launch site, the greater the success rate at a launch site.

Launch success rate started to increase in 2013 till 2020.

Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

KSC LC-39A had the most successful launches of any sites.

The Decision tree classifier is the best machine learning algorithm for this task.

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!