IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

MOOC Student
August

Outline

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

Executive Summary

Methodologies

- Data Collection via Web scraping
- Data Wrangling
- Complete the EDA with Visualization
- Dashboard

All results

- Predict first stage of the Falcon 9 lands successfully

Introduction

Project background and context

- Space Y wants to launch rockets with minimal costs

Problems you want to find answers

- Determine the price of each launch
- Determine if SpaceX will reuse the first stage

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Using SpaceX API and Web scraping
- Perform data wrangling
    - Use flowcharts and key phrases
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - Built, evaluated, improved, and found the best classification model

Data Collection

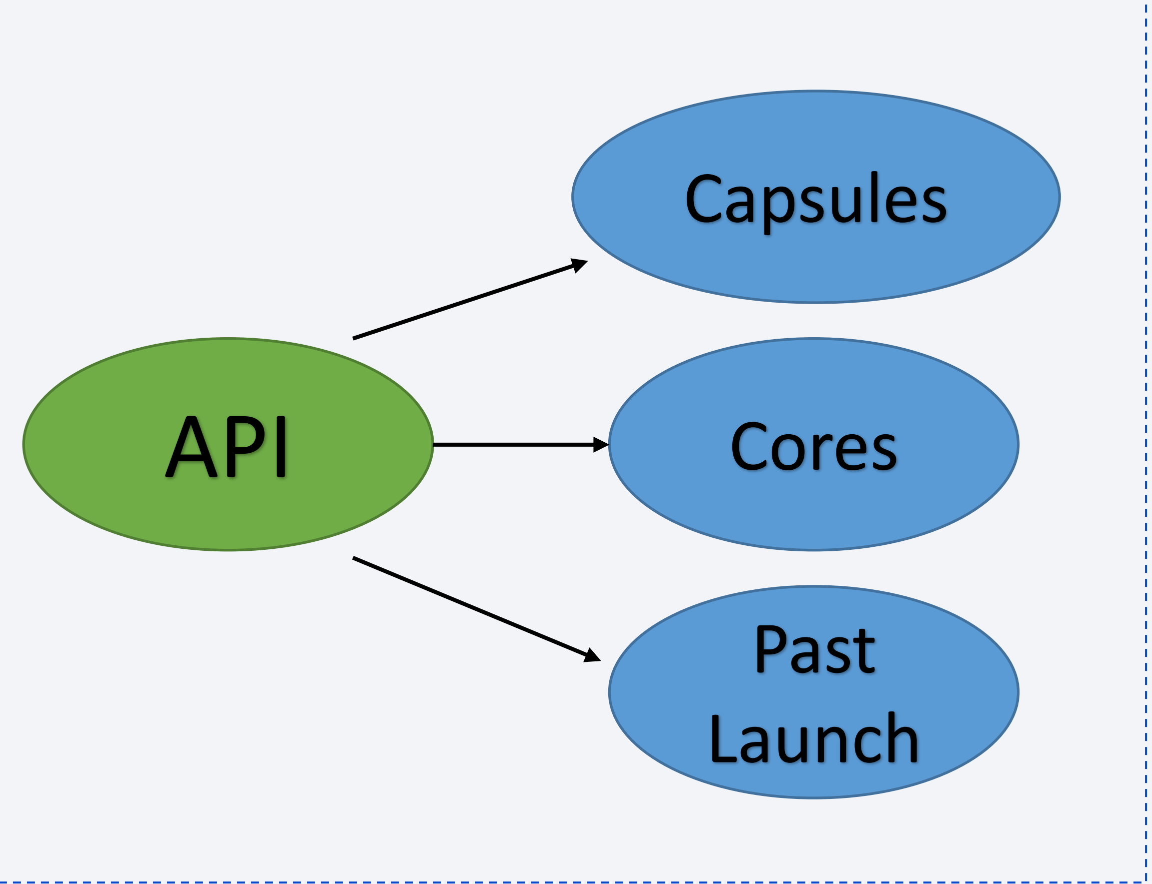
Describe how data sets were collected

- API will give us data about launches,including information about the rocket used, payload delivered, launch specifications,landing specifications, and landing outcome
- Web scrape some HTML tables that contain valuable Falcon 9 launch records

Data Collection – SpaceX API

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

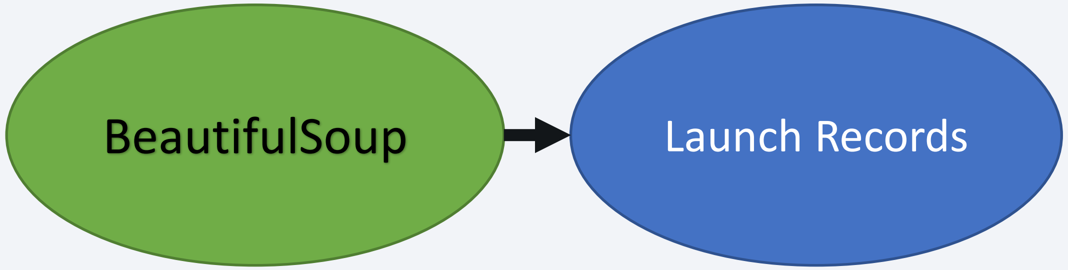
- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose



Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose



Data Wrangling

Describe how data were processed

- Calculate the number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate the number and occurrence of mission outcome per orbit type
- Create a landing outcome label from outcome column
- You need to present your data wrangling process using key phrases and flowcharts
- Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

EDA with Data Visualization

Summarize what charts were plotted and why you used those charts

- Visualize the relationship between Flight Number and Launch Site
- Visualize the relationship between Payload and Launch Site
- Visualize the relationship between success rate of each orbit type
- Visualize the relationship between FlightNumber and Orbit type
- Visualize the relationship between Payload and Orbit type
- Visualize the launch success yearly trend
- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

EDA with SQL

Summarize the SQL queries you performed

- Retrieve the most recent date from the SpaceX table
- Display the minimum payload mass
- Total payload_mass_kg carried by the booster versions
- Display 5 records launched on Friday
- Unique launch sites

Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

Build an Interactive Map with Folium

Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map

- Mark all launch sites on a map
- Mark the success/failed launches for each site
- Calculate the distances between a launch site to its proximities

Explain why you added those objects

Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

Build a Dashboard with Plotly Dash

Summarize what plots/graphs and interactions you have added to a dashboard

- Analyzing launch site geo and proximities
- Choose an optimal launch site

Explain why you added those plots and interactions

Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

Summarize how you built, evaluated, improved, and found the best performing classification model

Create 2 classes

Standardize the data, create a logistic regression object then create a GridSearchCV.

You need present your model development process using key phrases and flowchart

Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

Results

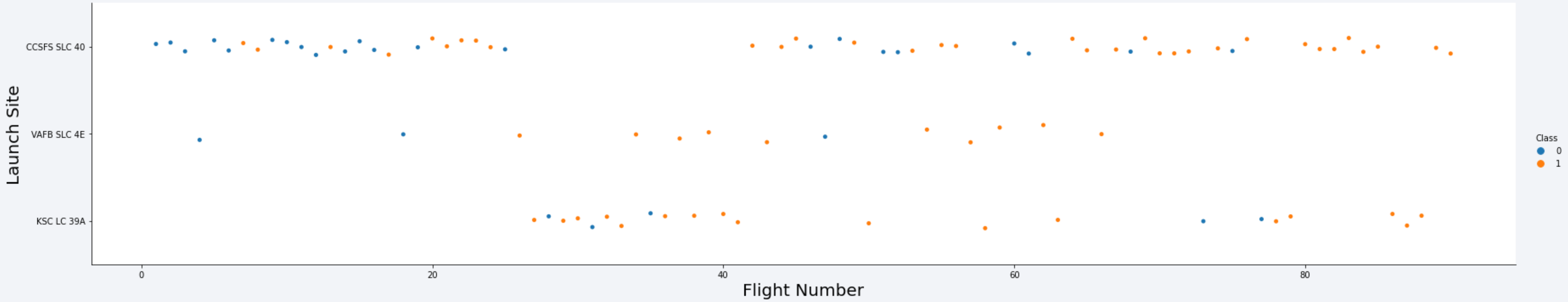
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

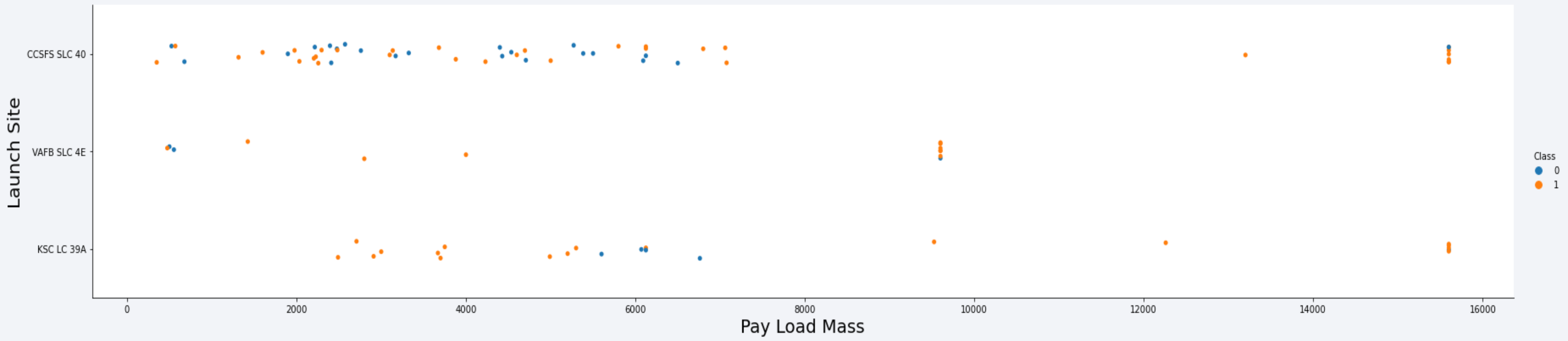
- Show a scatter plot of Flight Number vs. Launch Site



Show the screenshot of the scatter plot with explanations

Payload vs. Launch Site

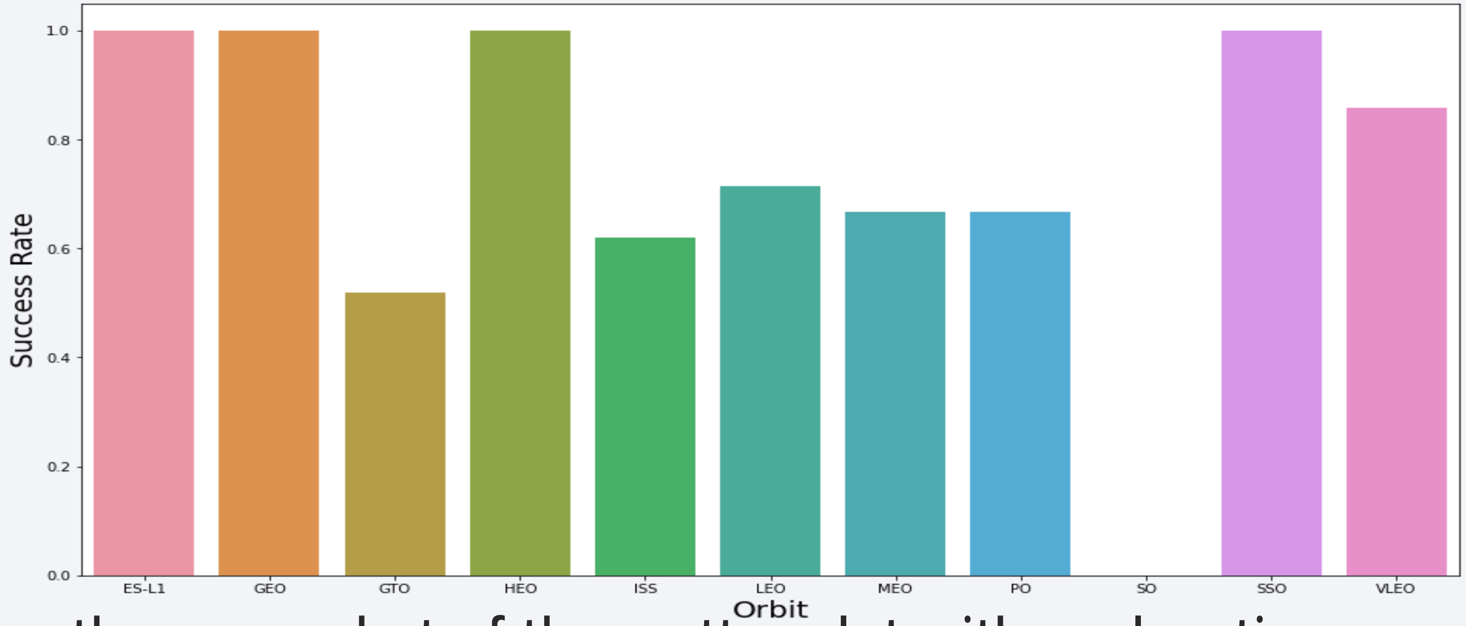
- Show a scatter plot of Payload vs. Launch Site



- Show the screenshot of the scatter plot with explanations

Success Rate vs. Orbit Type

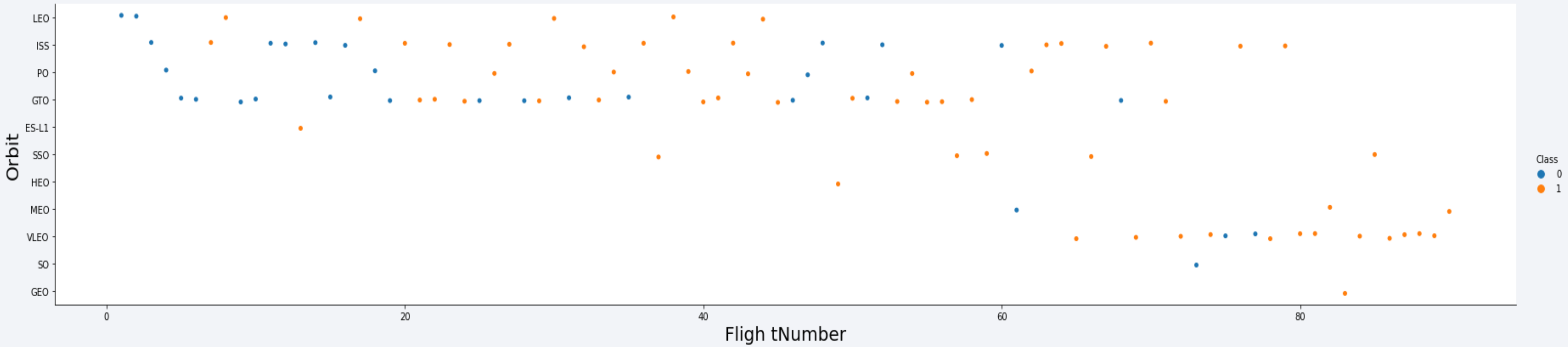
- Show a bar chart for the success rate of each orbit type



Show the screenshot of the scatter plot with explanations

Flight Number vs. Orbit Type

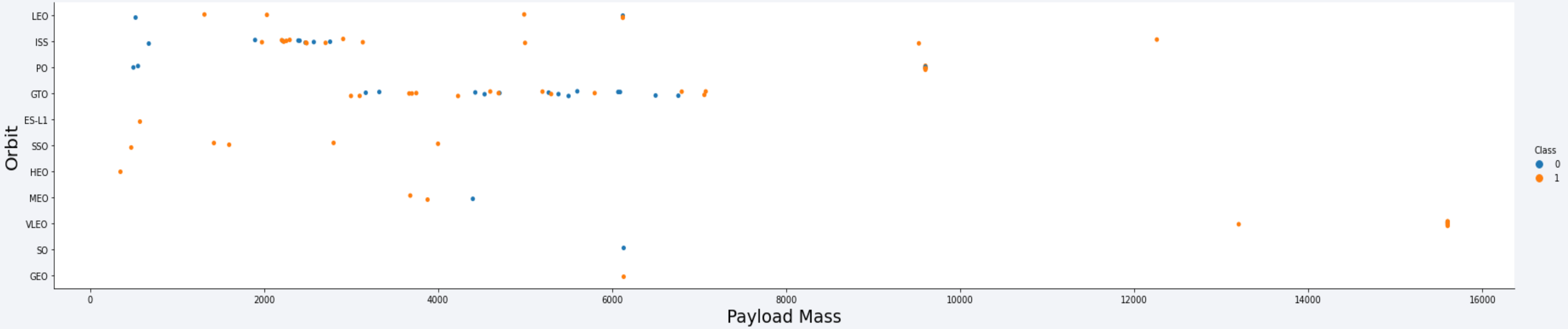
- Show a scatter point of Flight number vs. Orbit type



- Show the screenshot of the scatter plot with explanations

Payload vs. Orbit Type

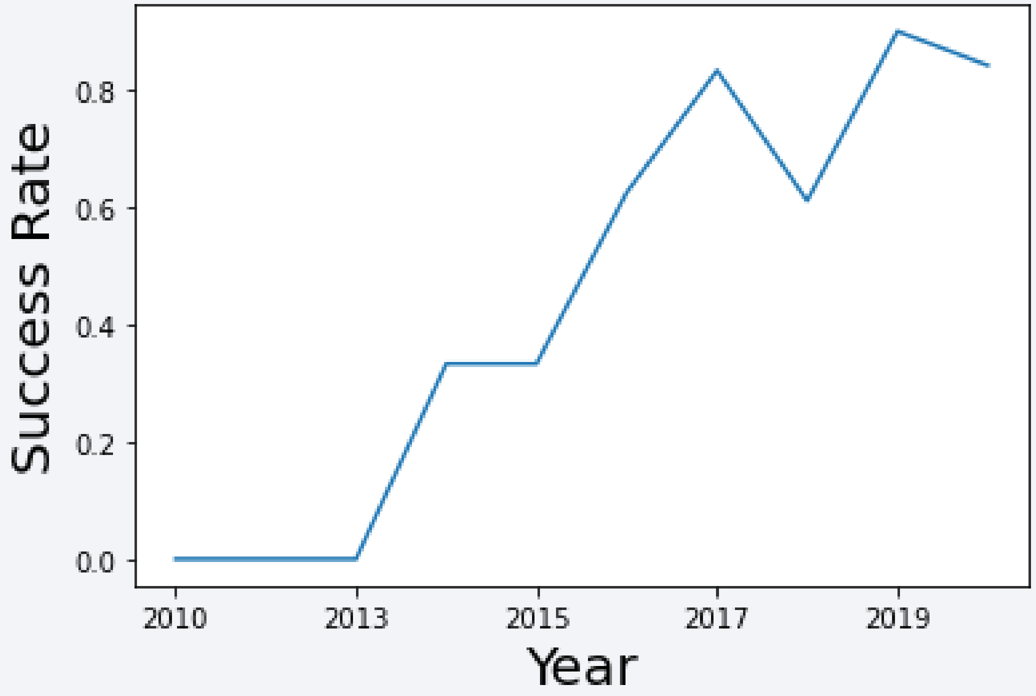
- Show a scatter point of payload vs. orbit type



- Show the screenshot of the scatter plot with explanations

Launch Success Yearly Trend

- Show a line chart of yearly average success rate



- Show the screenshot of the scatter plot with explanations

All Launch Site Names

- Find the names of the unique launch sites

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

In [16]:
```
pd.read_sql_query("SELECT DISTINCT Launch_Site FROM SPACEX \
                   LIMIT 5;", db)
```

Out[16]:

	Launch_Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	CCAFS SLC-40

- Present your query result with a short explanation here

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

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

In [17]:

```
pd.read_sql_query("SELECT * FROM SPACEX \
                  WHERE Launch_Site LIKE 'CCA%' \
                  LIMIT 5;", db)
```

Out[17]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Present your query result with a short explanation here

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

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

In [18]:

```
pd.read_sql_query("SELECT SUM (PAYLOAD_MASS__KG_) FROM SPACEX \
                  WHERE Customer LIKE '%NASA (CRS)%';", db)
```

Out[18]:

	SUM (PAYLOAD_MASS__KG_)
0	48213

- Present your query result with a short explanation here

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

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

In [19]:
```
pd.read_sql_query("SELECT AVG (PAYLOAD_MASS__KG_) FROM SPACEX \
                   WHERE Booster_Version = 'F9 v1.1';", db)
```

Out[19]:

	AVG (PAYLOAD_MASS__KG_)
0	2928.4

- Present your query result with a short explanation here

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

In [21]:
```
df[df["Landing _Outcome"] == "Success (ground pad)"].head(3)
```

Out[21]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
19	22-12-2015	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)
26	18-07-2016	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
29	19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)

- Present your query result with a short explanation here

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

In [23]:
```
df2[(df2["PAYLOAD_MASS__KG_"] > 4000) & (df2["PAYLOAD_MASS__KG_"] <= 6000)]
```

Out[23]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
23	06-05-2016	05:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
27	14-08-2016	05:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
31	30-03-2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
42	11-10-2017	22:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200	GTO	SES EchoStar	Success	Success (drone ship)

- Present your query result with a short explanation here

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

List the total number of successful and failure mission outcomes

In [24]:
```
df["Mission_Outcome"].value_counts()
```

Out[24]:
```
Success                             98
Failure (in flight)                  1
Success (payload status unclear)     1
Success                              1
Name: Mission_Outcome, dtype: int64
```

- Present your query result with a short explanation here

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

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

In [25]:
```
df["PAYLOAD_MASS__KG_"].max()
```

Out[25]: 15600

In [26]:
```
df[df["PAYLOAD_MASS__KG_"] == 15600]
```

Out[26]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
74	11-11-2019	14:56:00	F9 B5 B1048.4	CCAFS SLC-40	Starlink 1 v1.0, SpaceX CRS-19	15600	LEO	SpaceX	Success	Success
77	07-01-2020	02:33:00	F9 B5 B1049.4	CCAFS SLC-40	Starlink 2 v1.0, Crew Dragon in-flight abort t...	15600	LEO	SpaceX	Success	Success
79	29-01-2020	14:07:00	F9 B5 B1051.3	CCAFS SLC-40	Starlink 3 v1.0, Starlink 4 v1.0	15600	LEO	SpaceX	Success	Success
80	17-02-2020	15:05:00	F9 B5 B1056.4	CCAFS SLC-40	Starlink 4 v1.0, SpaceX CRS-20	15600	LEO	SpaceX	Success	Failure
82	18-03-2020	12:16:00	F9 B5 B1048.5	KSC LC-39A	Starlink 5 v1.0, Starlink 6 v1.0	15600	LEO	SpaceX	Success	Failure
83	22-04-2020	19:30:00	F9 B5 B1051.4	KSC LC-39A	Starlink 6 v1.0, Crew Dragon Demo-2	15600	LEO	SpaceX	Success	Success
85	04-06-2020	01:25:00	F9 B5 B1049.5	CCAFS SLC-40	Starlink 7 v1.0, Starlink 8	15600	LEO	SpaceX, Planet Labs	Success	Success

- Present your query result with a short explanation here

2015 Launch Records

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

In [32]:
```
df3 = df[(df["Landing _Outcome"] == "Failure (drone ship)") & (df["Year"] == 2015)]
df3
```

Out[32]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome	Year
13	2015-10-01	09:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)	2015
16	2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)	2015

- Present your query result with a short explanation here

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

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 [33]: df4 = df[df["Date"].between("2010-04-06","2017-03-20")]
```

```
In [34]: df4["Landing _Outcome"].value_counts()
```

```
Out[34]: No attempt                10
         Failure (drone ship)       5
         Success (ground pad)       5
         Success (drone ship)       5
         Controlled (ocean)         3
         Failure (parachute)        2
         Uncontrolled (ocean)       2
         Precluded (drone ship)     1
         Name: Landing _Outcome, dtype: int64
```

- Present your query result with a short explanation here

Section 3

Launch Sites Proximities Analysis

Mark all launch sites on a map

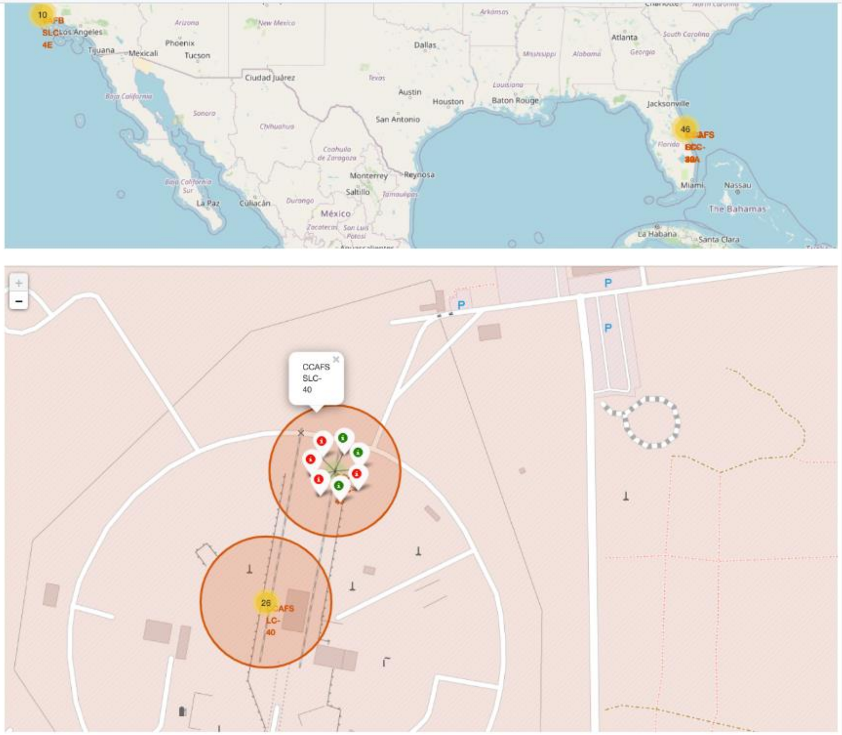
- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map

The generated map with marked launch sites should look similar to the following:

- Explain the important elements and findings on the screenshot

Mark the success/failed launches for each site on the map

- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map



From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

- Explain the important elements and findings on the screenshot

Calculate the distances between a launch site to its proximities

- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

Your updated map with distance line should look like the following screenshot:

TODO: Similarly, you can draw a line betwee a launch site to its closest city, railway, highway, etc. You need to use `MousePosition` to find the their coordinates on the map first

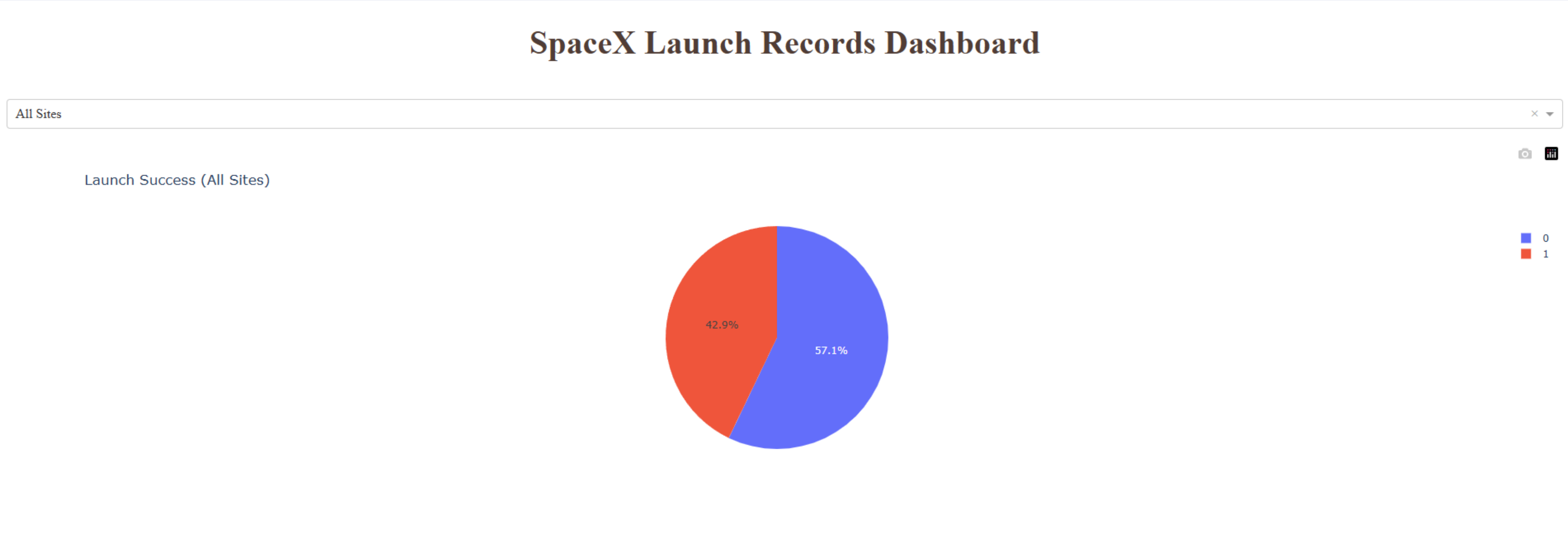
- Explain the important elements and findings on the screenshot

Section 4

Build a Dashboard with Plotly Dash

Launch success count for all sites

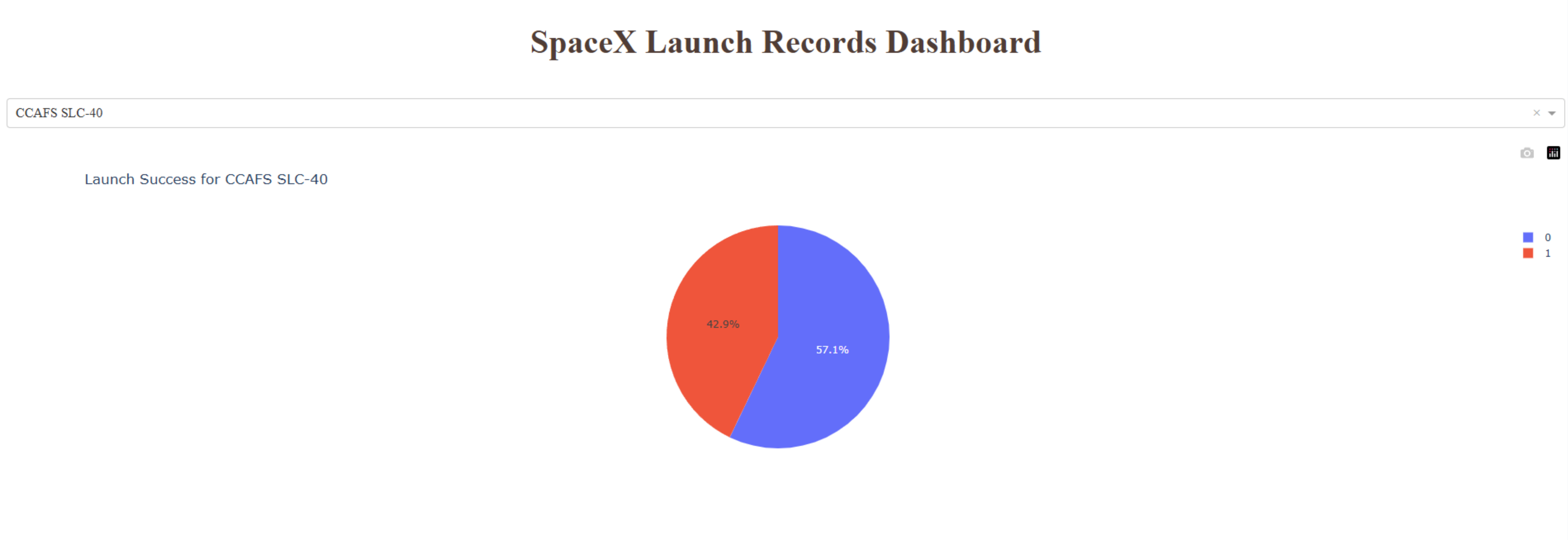
- Show the screenshot of launch success count for all sites, in a piechart



- Explain the important elements and findings on the screenshot

Launch site with highest launch success ratio

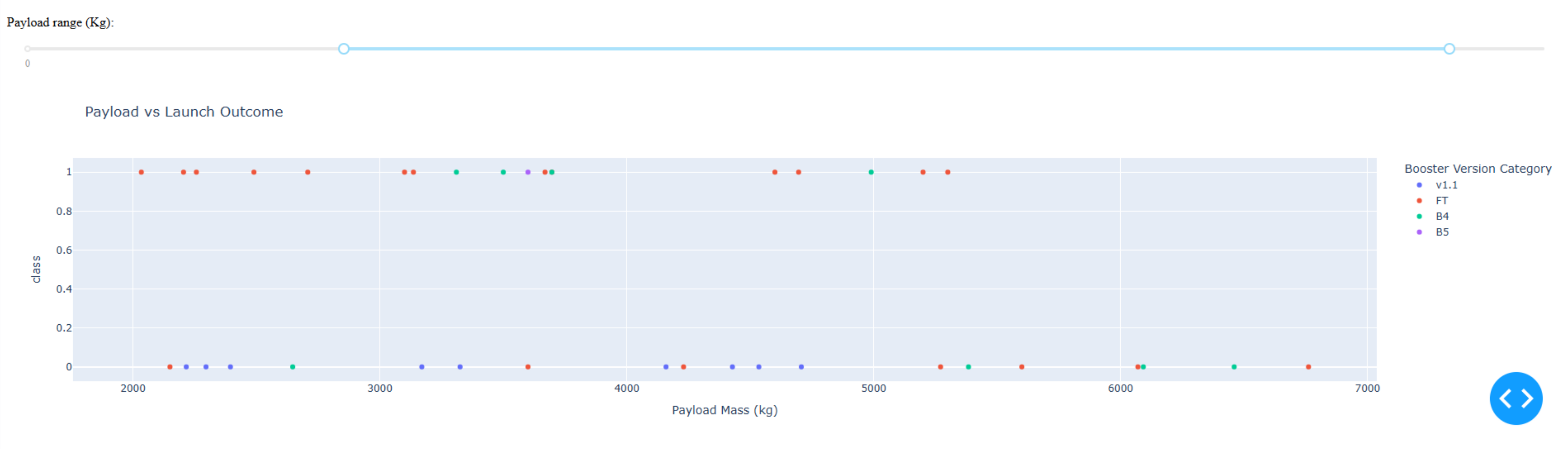
- Show the screenshot of the piechart for the launch site with highest launch success ratio



- Explain the important elements and findings on the screenshot

Payload vs. Launch Outcome scatter plot for all sites

- 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

- Visualize the built model accuracy for all built classification models, in a bar chart

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

In [24]:
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

In [25]:
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
```

In [26]:
```
logreg_cv = GridSearchCV(estimator=lr, param_grid=parameters,cv=10)
```

In [27]:
```
logreg_cv.fit(X_train,y_train);
```

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_`.

In [28]:
```
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 : 0.8464285714285713
```

- Find which model has the highest classification accuracy

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation

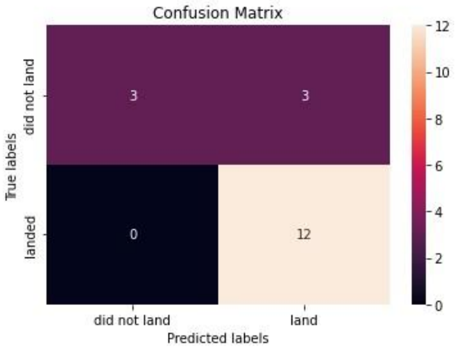
Calculate the accuracy on the test data using the method `score` :

In [29]:
```
logreg_cv.score(X=X_test, y=y_test)
```

Out[29]: 0.8333333333333334

Lets look at the confusion matrix:

In [30]:
```
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.

Conclusions

- Predicting Falcon 9's first-stage landing success can optimize SpaceX’s operations and reduce launch costs.
- By leveraging historical data, we can develop accurate models to forecast landing outcomes.
- This prediction model provides valuable insights into reusability, benefiting both SpaceX and its competitors.
- With continuous improvement, this model could revolutionize cost estimation and efficiency in spaceflight.

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!