Winning Space Race with Data Science

Alicia Gonzalez Martinez
22 April 2022

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 with Folium
- Machine Learning Prediction

Summary of all results

- Exploratory Data Analysis result
- Interactive analytics in screenshots
- Predictive Analytics result

Introduction

Our goal in this project is to predict if 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.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- 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

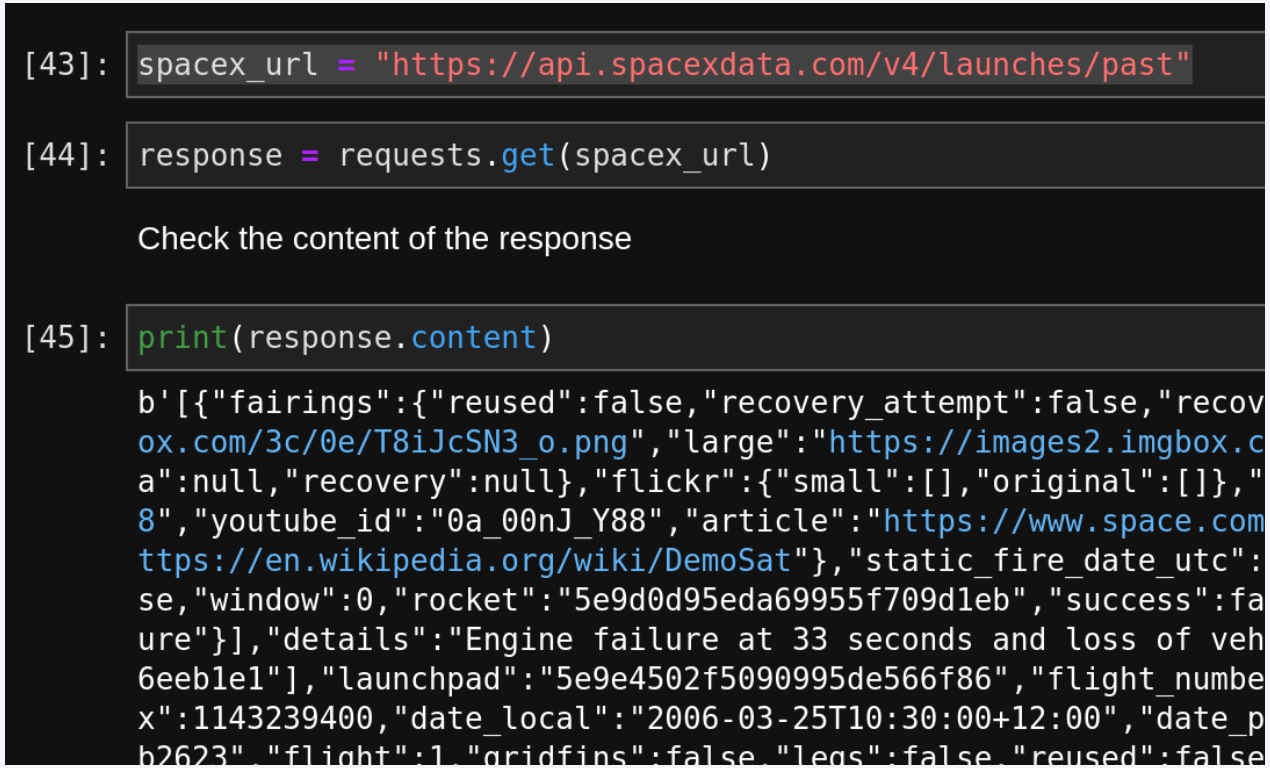
We collected data from the SpaceX Web API, https://api.spacexdata.com

We also took data from the wikipedia using web scraping

https://github.com/kabikaj/AppliedDataScienceCapstone

Data Collection – SpaceX API

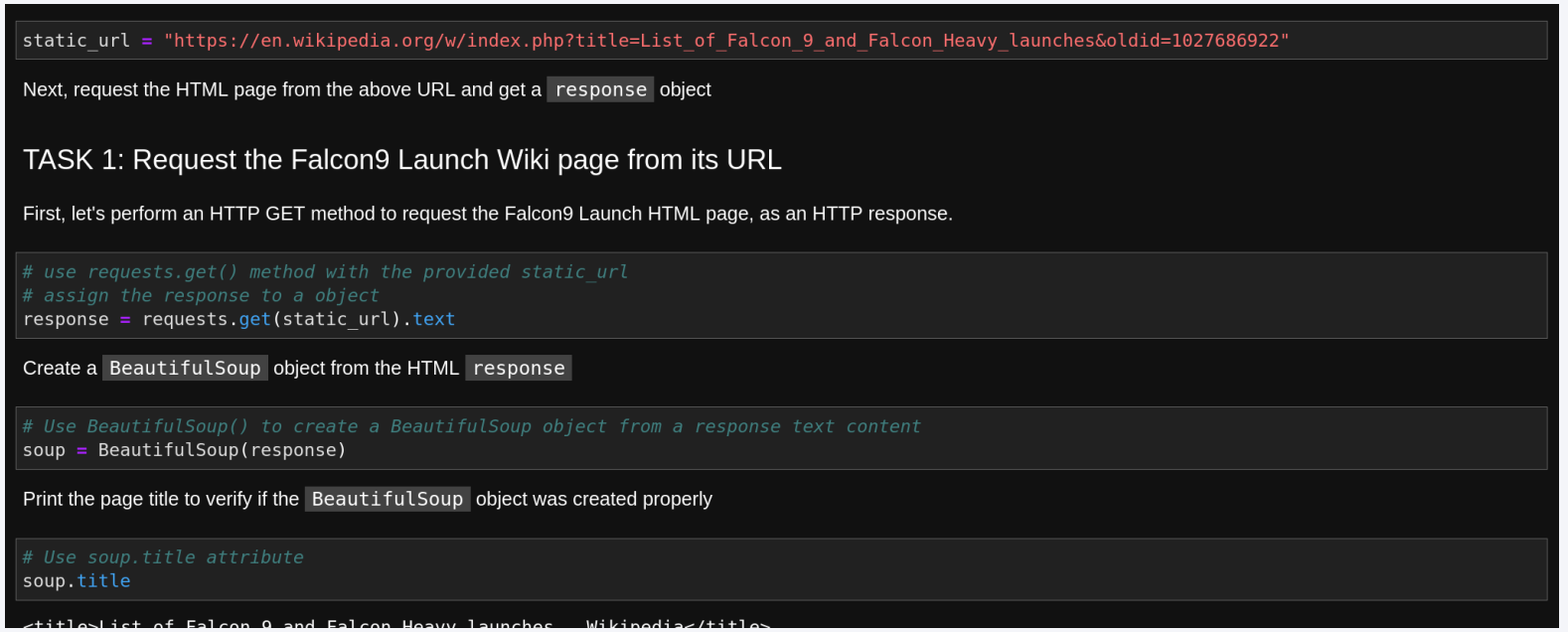
We made a REST call to the API of SpaceX to get the data we needed to process. The output of the response was a json object that we processed so that we could create a dataframe with our data.



https://github.com/kabikaj/AppliedDataScienceCapstone

Data Collection - Scraping

We used web scraping and applied a get request to the a wikipedia page to get the data needed for the project and then transform it using beautifulsoup.



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

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

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

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

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

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

```
# Use soup.title attribute
soup.title
```

<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>

https://github.com/kabikaj/AppliedDataScienceCapstone

Data Wrangling

The data was converted into a dataframe in order to operate with it. Based on the information available in the data about successes and failures of lauches, we created an additional column in our data, called „class“ and set it to 0 when the lauch was a failure and to 1 when the launch was a success.

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [(0 if x in bad_outcomes else 1) for x in df.Outcome]
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
df['Class'] = landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

https://github.com/kabikaj/AppliedDataScienceCapstone

EDA with Data Visualization

We created an extensive amount of different plots and charts iin order to be able to understand our data better. We also perform a one-hot encoding on several fields.

```
# A function to Extract years from the date
year=[]
def Extract_year(date):
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
```

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df_trend = pd.DataFrame(Extract_year(df['Date']), columns=['Year'])
df_trend['Class'] = df['Class']
#df_trend.head()

sns.lineplot(data=df_trend,
             x=np.unique(Extract_year(df['Date'])),
             y=df_trend.groupby('Year')['Class'].mean()
)
```

https://github.com/kabikaj/AppliedDataScienceCapstone

EDA with SQL

We perform an extensive amount of SQL queries in order to explore the data, such as displaying the average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEX WHERE Booster_Version LIKE '%F9 v1.1%';
```

```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

Task 5

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

Hint:Use min function

```
%sql SELECT Date FROM SPACEX WHERE Landing_Outcome LIKE '%ground%' LIMIT 1;
```

```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

Date
2015-12-22 00:00:00

Task 6

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

```
%sql SELECT Booster_Version FROM SPACEX WHERE Mission_Outcome = 'Success' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

Booster_Version
F9 v1.1

https://github.com/kabikaj/AppliedDataScienceCapstone

Build an Interactive Map with Folium

The interactive map we created with Folium gave us the possibility to visualise the data from a different perspective, having into account relevant geographical implications. Below you can see a screenshot of part of the code.

```
# Create and add a folium.Marker on your selected closest coastline point on the map
# Display the distance between coastline point and launch site using the icon property
coordinate = [coastline_lat, coastline_lon]
distance_marker = folium.Marker(
    coordinate,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color: #d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_coastline),
    )
)
```

TODO: Draw a `PolyLine` between a launch site to the selected coastline point

```
# Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
lines = folium.PolyLine(
    locations=[[launch_site_lat, launch_site_lon], [coastline_lat, coastline_lon]],
    weight=1
)
site_map.add_child(lines)
site_map
```

https://github.com/kabikaj/AppliedDataScienceCapstone

Build a Dashboard with Plotly Dash

With plotly dah we were able to create an interactive dashboard with which we could see different charts by selecting values from menus. Here you can see part of the code.

```
# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                dcc.Dropdown(id='site-dropdown',
                                             options=[{'label': 'All Sites', 'value': 'ALL'}]+[{'label': x, 'value': x} for x in spacex_df['Launch Site'].unique()],
                                             value='ALL',
                                             placeholder="place holder here",
                                             searchable=True
                                ),

                                html.Br(),

                                # TASK 2: Add a pie chart to show the total successful launches count for all sites
                                # If a specific launch site was selected, show the Success vs. Failed counts for the site
                                html.Div(dcc.Graph(id='success-pie-chart')),
                                html.Br(),

                                html.P("Payload range (Kg):"),
                                # TASK 3: Add a slider to select payload range
                                #dcc.RangeSlider(id='payload-slider',...)
                                dcc.RangeSlider(id='payload-slider',
                                                min=0,
                                                max=10000,
                                                step=1000,
                                                value=[min_payload, max_payload]
                                ),

                                # TASK 4: Add a scatter chart to show the correlation between payload and launch success
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])
```

https://github.com/kabikaj/AppliedDataScienceCapstone

Predictive Analysis (Classification)

The last part of the project consisted of applying differet classification algorithms in order to see if they could predict well the success rates of the launches and which of them was better.

```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

```
knn_cv = GridSearchCV(KNN, parameters, cv=10)
knn_cv.fit(X_train, Y_train)
```

```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```


https://github.com/kabikaj/AppliedDataScienceCapstone

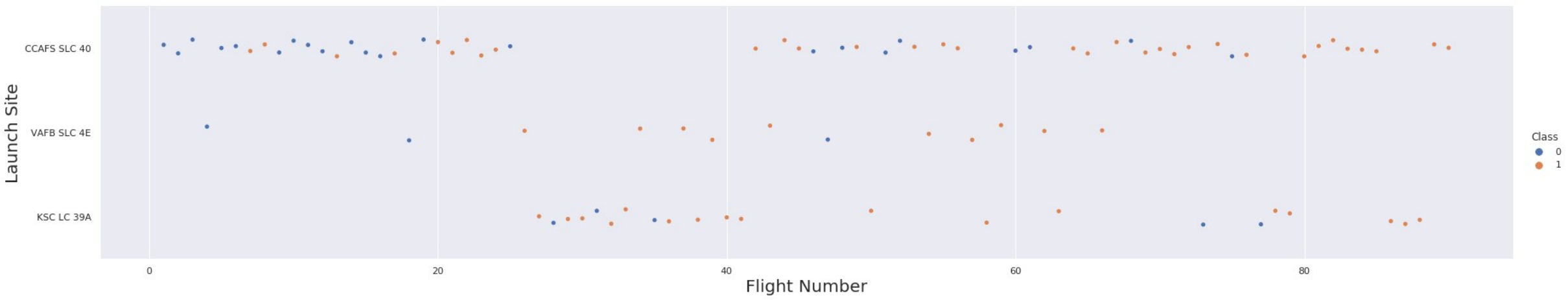
Results

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

Section 2

Insights drawn from EDA

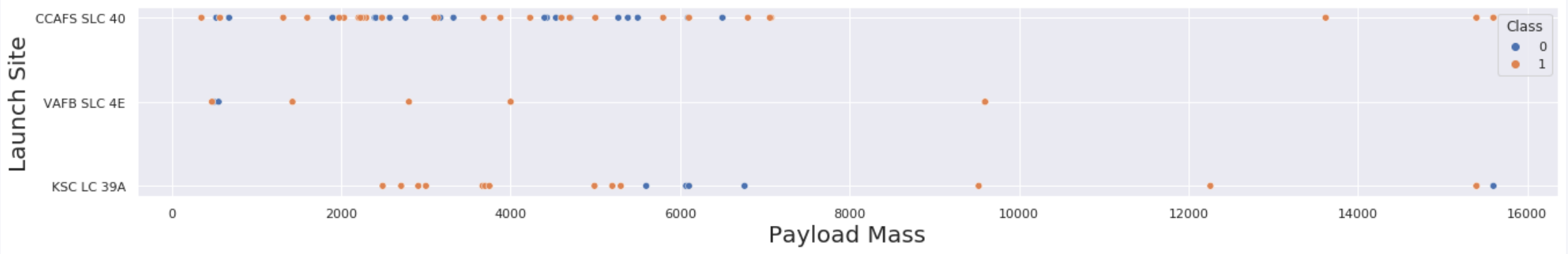
Flight Number vs. Launch Site



As we can see, if the flight numbers are very high we have more successes in all Launch sites

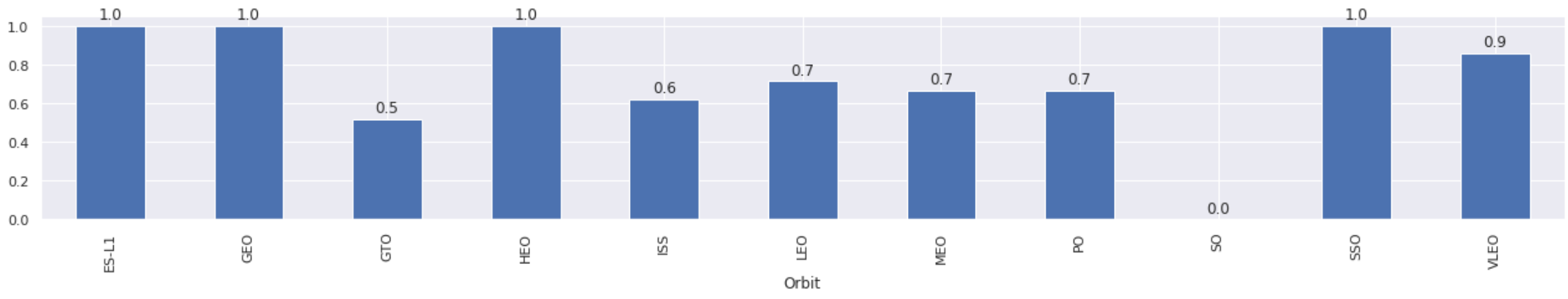
Payload vs. Launch Site

As we can see in the chart, after 8000 of payload mass all launches are succeses.



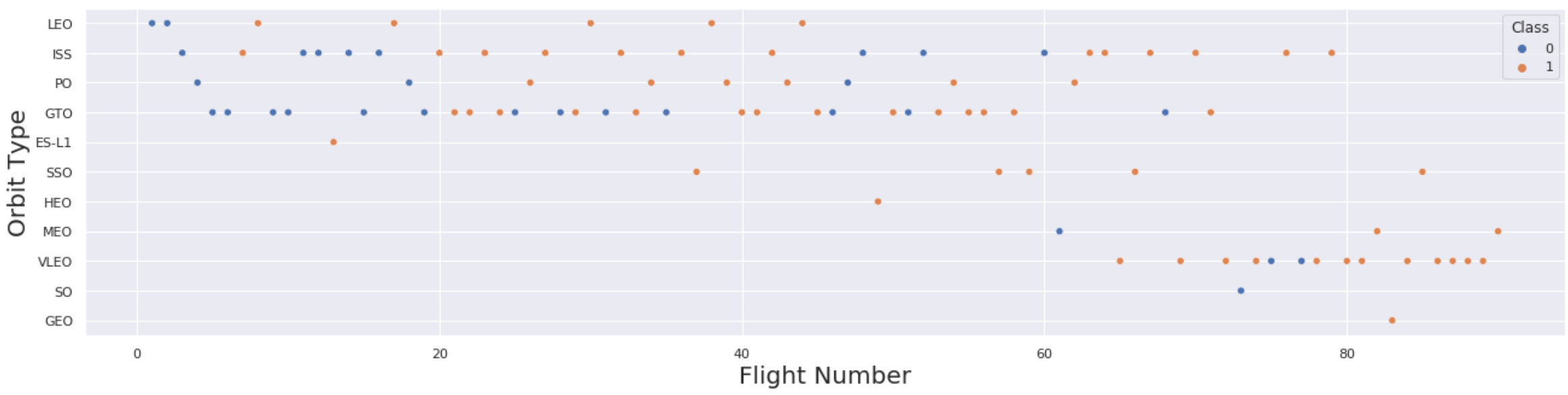
Success Rate vs. Orbit Type

As we can see in the chart below, ES-L1, GEO, HEO and SS0 orbit types have a sucess rate of 1.



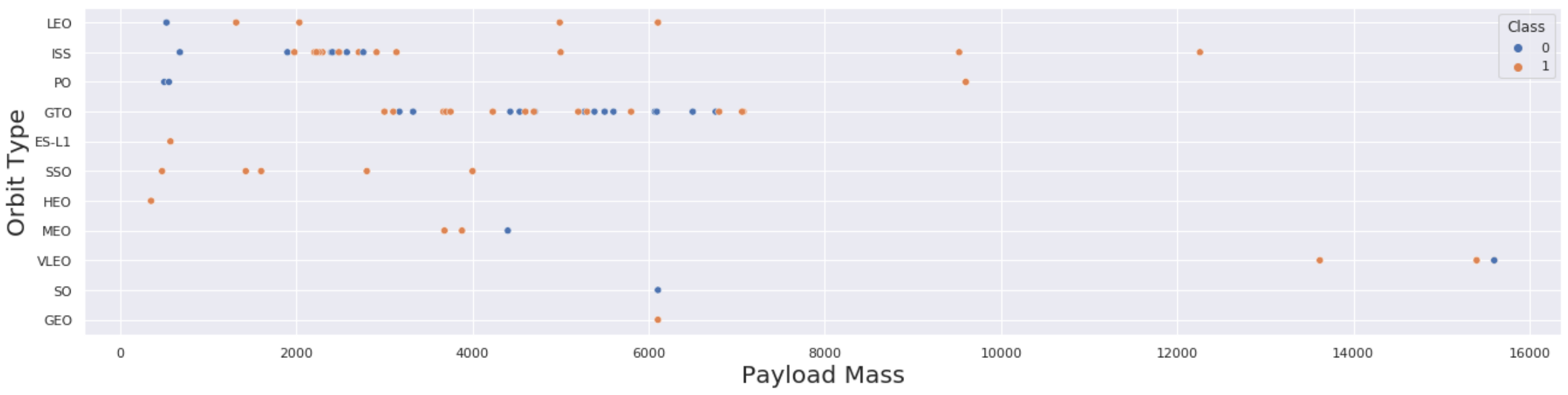
Flight Number vs. Orbit Type

In general we can see that the more flight numbers, the more the successes.



Payload vs. Orbit Type

We cannot see any remarkable tendency from this chart.



Launch Success Yearly Trend

The sucess rate since 2013 kept increasing till 2020

All Launch Site Names

We apply the distinct method to the Lauch_Site data to get the result.

```
%sql SELECT DISTINCT(Launch_Site) FROM SPACEX;
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

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

Launch Site Names Begin with 'CCA'

We used the where and like sql command and limit our results to 5.

```
%sql SELECT * FROM SPACEX WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

index	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
1	2010-06-04 00:00:00	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2	2010-12-08 00:00:00	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)
3	2012-05-22 00:00:00	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
4	2012-10-08 00:00:00	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
5	2013-03-01 00:00:00	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

For getting the total payload mass by NASA we filtered our results first and them applied the sum method.

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEX WHERE Customer LIKE '%NASA%';
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
SUM(PAYLOAD_MASS__KG_)
                107010
```

Average Payload Mass by F9 v1.1

In this case we used the method avg to get the average.

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEX WHERE Booster_Version LIKE '%F9 v1.1%';
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
AVG(PAYLOAD_MASS__KG_)
2534.6666666666665
```

First Successful Ground Landing Date

We filtered our data and limit the results to 1.

```
%sql SELECT Date FROM SPACEX WHERE Landing_Outcome LIKE '%ground%' LIMIT 1;
```

```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

Date
2015-12-22 00:00:00

Successful Drone Ship Landing with Payload between 4000 and 6000

We applied again several filters to get our result

```
%sql SELECT Booster_Version FROM SPACEX WHERE Mission_Outcome = 'Success' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 FT B1032.2
F9 B4 B1040.2
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

Total Number of Successful and Failure Mission Outcomes

We used the method count to count both cases.

```
%sql SELECT COUNT(*) AS 'Success' FROM SPACEX WHERE Mission_Outcome = 'Success';
```

```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

Success
98

```
%sql SELECT COUNT(*) AS 'Failure' FROM SPACEX WHERE Mission_Outcome LIKE 'Failure%';
```

```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

Failure
1

Boosters Carried Maximum Payload

In this case we used a subquery to get our results.

```
%sql SELECT Booster_Version FROM SPACEX WHERE PAYLOAD_MASS__KG_ IN (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

For this query, we had to apply a method to extract the year from the date value.

```
%sql SELECT Landing_Outcome, Booster_Version, Launch_Site FROM SPACEX WHERE Landing_Outcome = 'Failure (drone ship)' AND strftime('%Y', Date) = '2015';
```

```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

Landing_Outcome	Booster_Version	Launch_Site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

In this last query we used a group by and then ordered our results in descdnding order.

```
%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) as Count FROM SPACEX WHERE Date BETWEEN "2010-06-04" AND "2017-03-20" GROUP BY Landing_Outcome ORDER BY Count DESC;
```

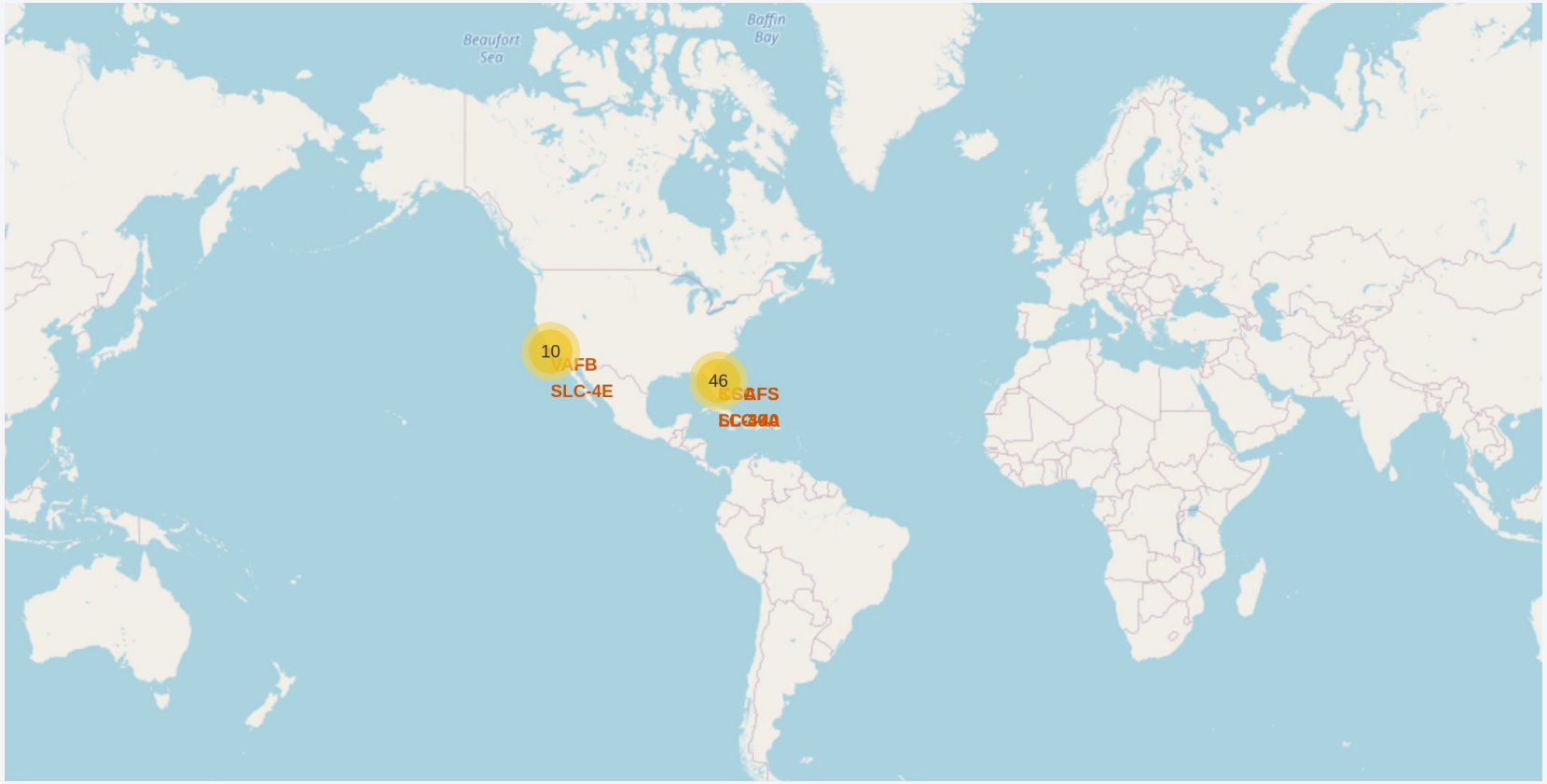
```
 * sqlite:////home/alicia/Dropbox/courses/data_science/AppliedDataScienceCapstone/Spacex.db
Done.
```

Landing_Outcome	Count
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

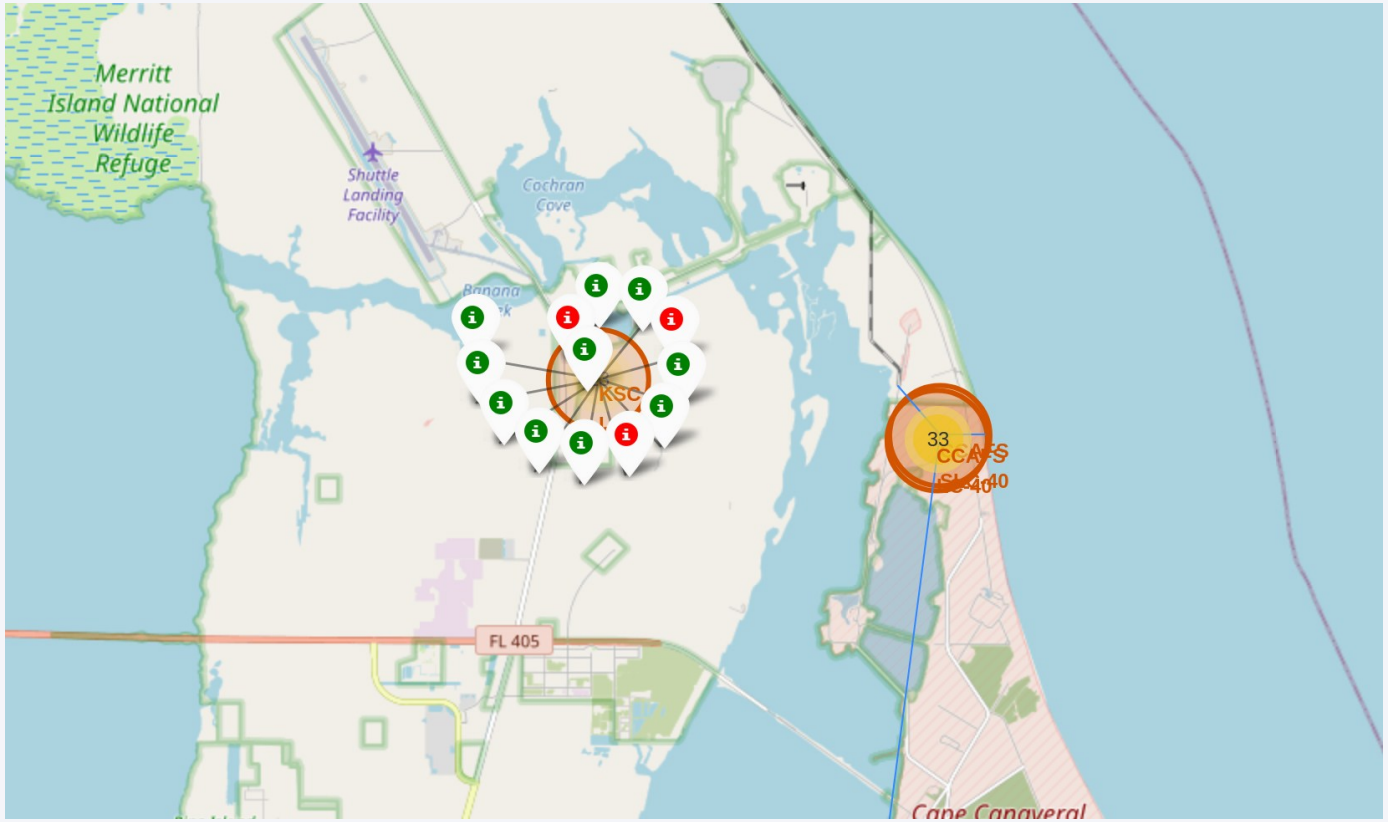
Folium Map: global map with all launch sites

Here we can see a global map including all launch sites’ location markers.



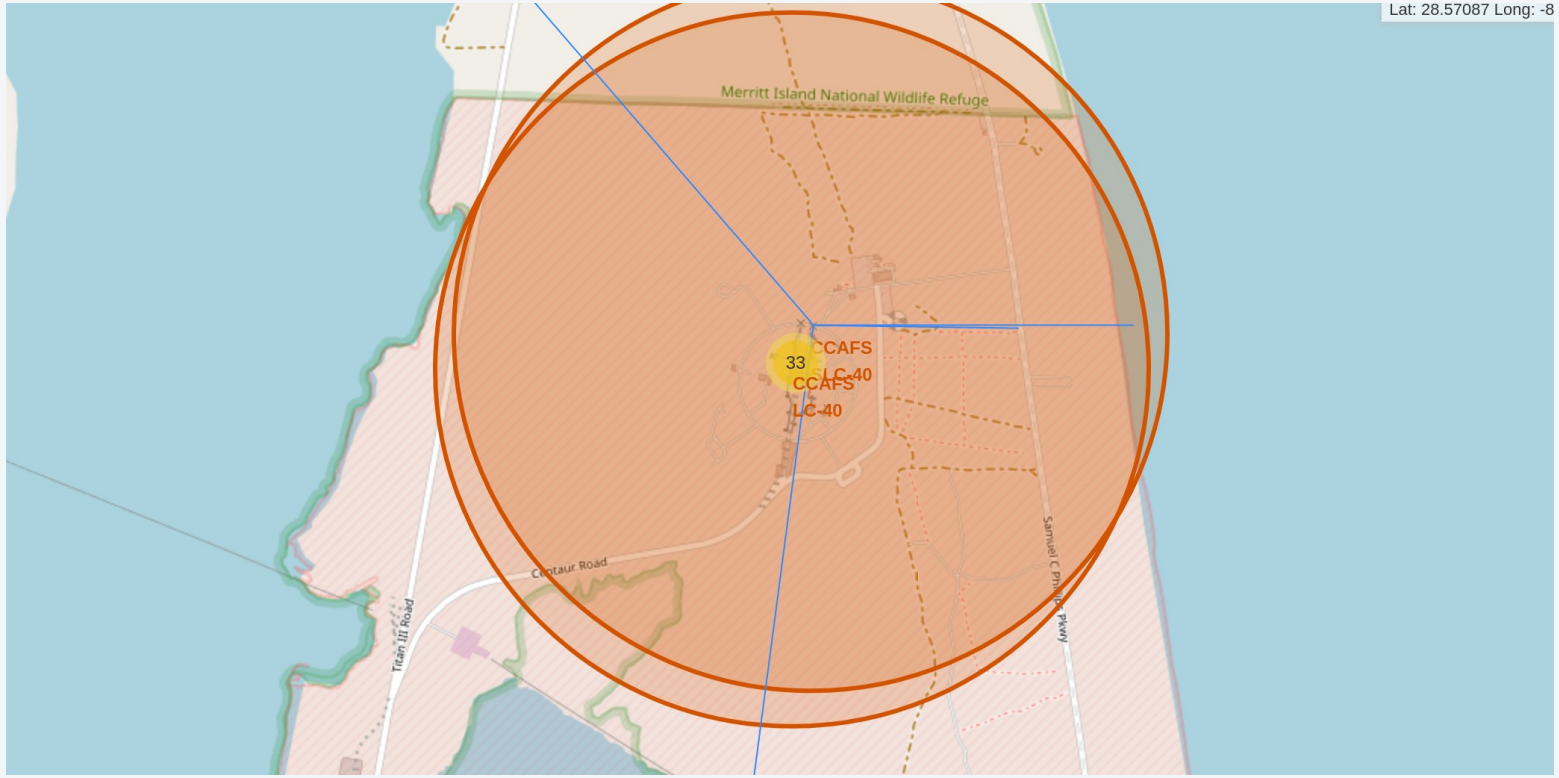
Folium Map: color-labeled launch outcomes

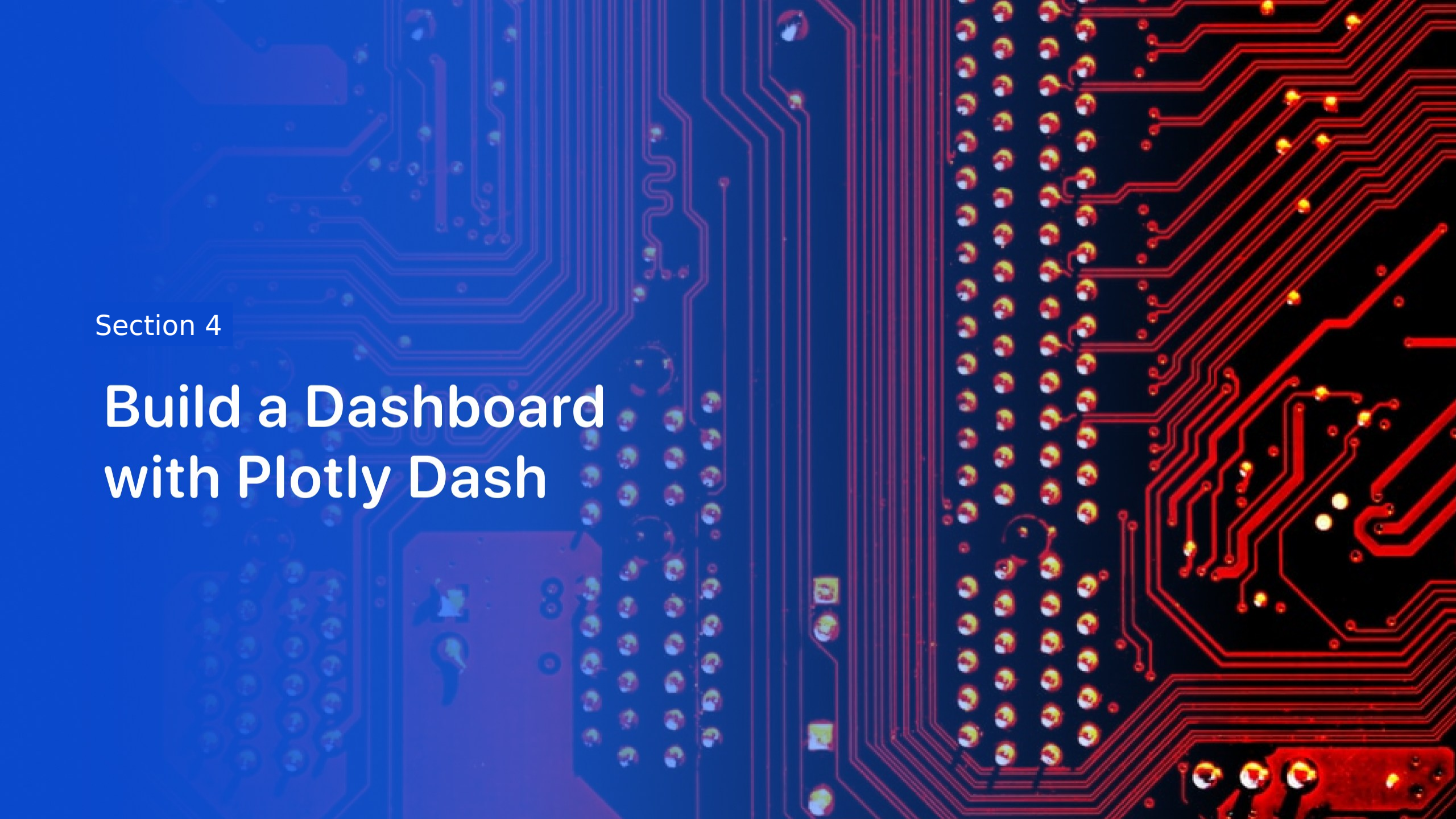
Here we can see the color-labeled launch outcomes in one of the sites.



Folium Map: proximities of a launch site

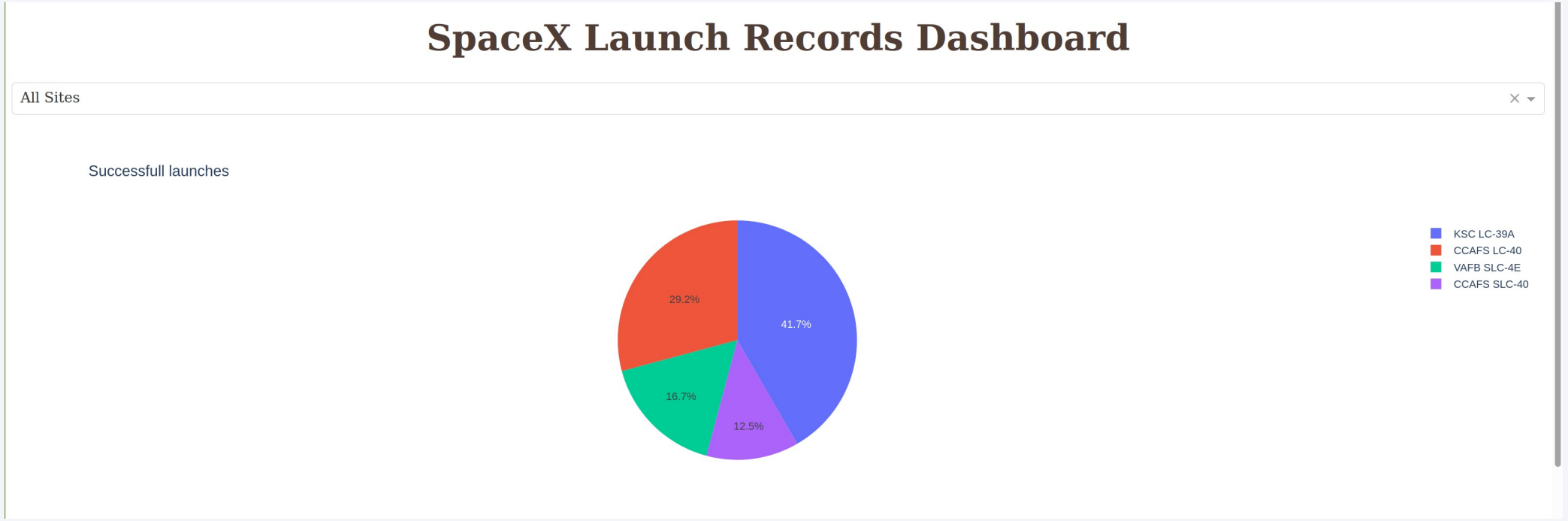
Here we can see a zoom in of a launch site.




Section 4
Build a Dashboard with Plotly Dash

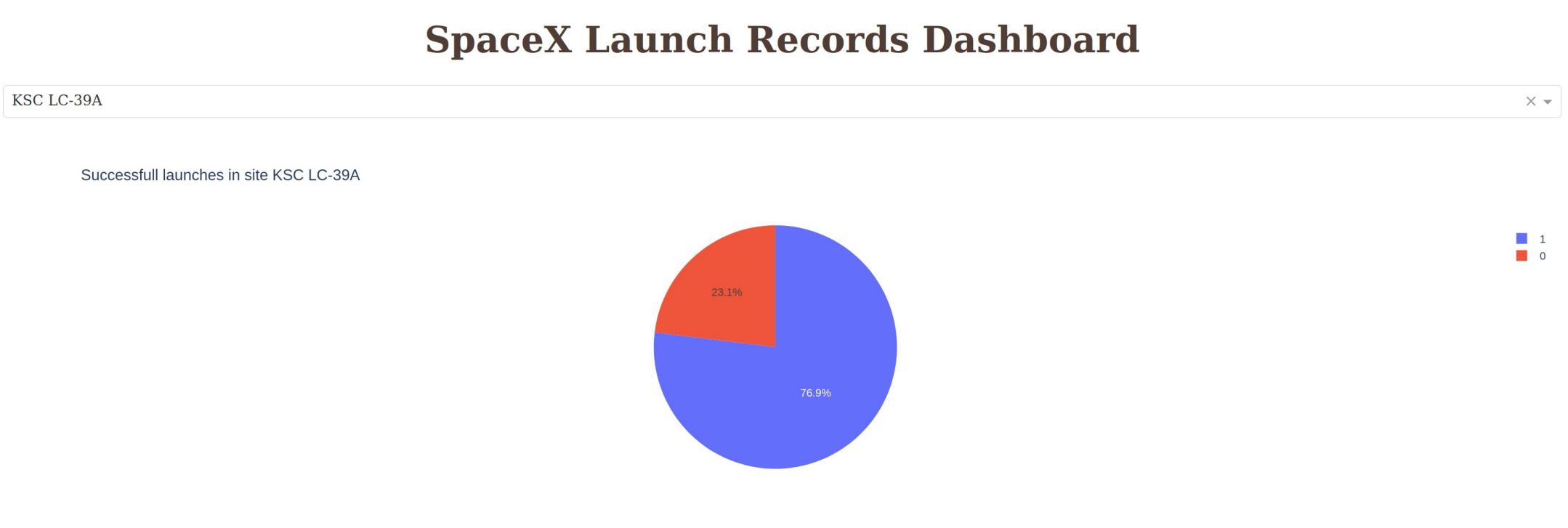
Dashboard: Succesfull launches

As we can see, the KSC LC-39A Site has the biggest amount of successfull launches.



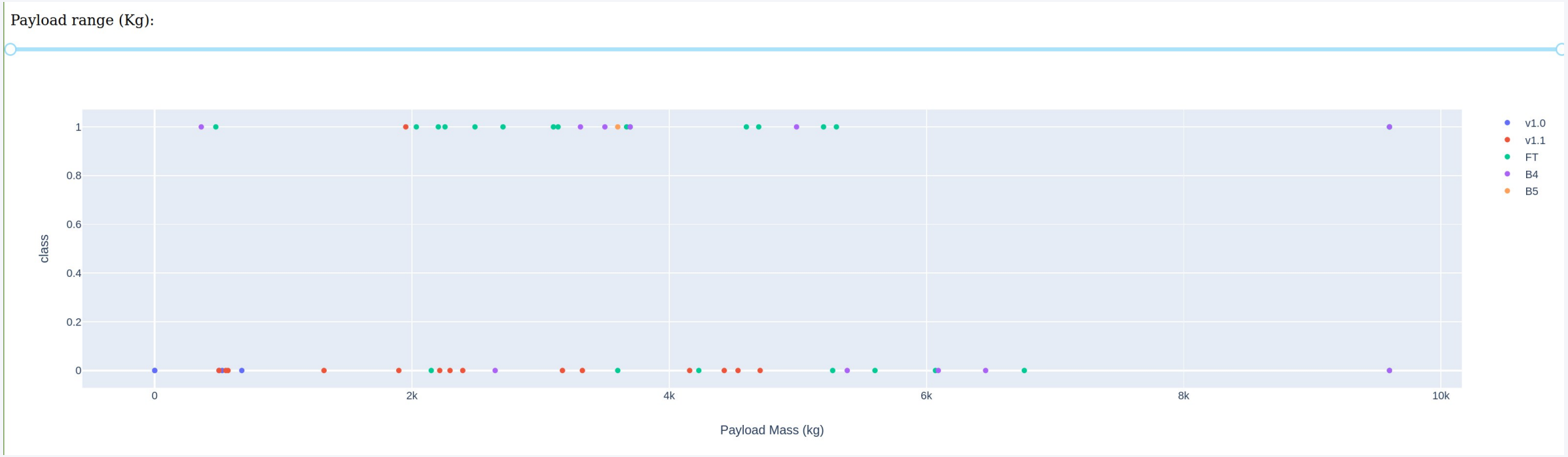
Dashboard: Launch site with highest launch success ratio

As we can see, KSC LC-39A has a success rate of 76.9%.



Dashboard: Payload vs Launch Outcome

As we can see, between 2k and 6k we have most of the values for FT and the majority of them belong to the success class.

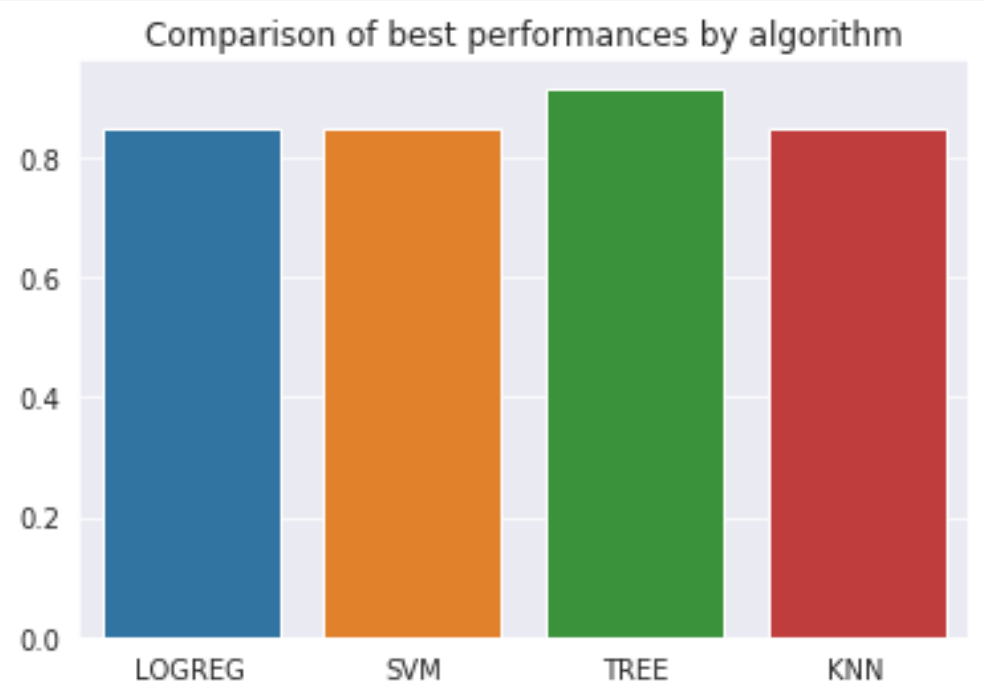


Section 5

Predictive Analysis (Classification)

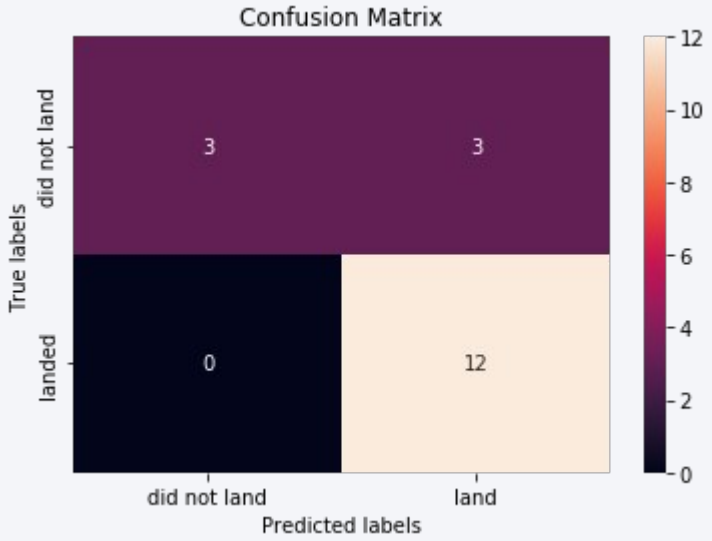
Classification Accuracy

As we can see, the decision tree model has slightly better accuracy than the rest



Confusion Matrix

All algorithms showed the same confussion matrix. As we can see, they predicted correctly all the cases where there was not landing, so there are no false positives. But we have cases of false negatives, so the algorithm predicted a landing but there was no.



Conclusions

We saw that if the flight numbers are very high we have more successes in all Launch sites

After 8000 of payload mass all launches are succeses.

ES-L1, GEO, HEO and SS0 orbit types have a sucess rate of 1.

All machine learning algorithms we applied perfomed well in predicting the right class, but the decision tree has slighly better accuracy

Appendix

All code can be find in the github project
https://github.com/kabikaj/AppliedDataScienceCapstone

Thank you!