IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Kaden Chan
July 25, 2025

Outline

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

Executive Summary

Several methodologies were implemented to capture key insights for whether SpaceX's Falcon 9 first stage will land successfully. Data Collection and Wrangling implemented REST API and webscraping and plotted them using charts and plots. Plotly's dashboard provided dynamic visualization of launch success rates according to payload mass and launch site along with Follium's maps. Model Training using different classification models yielded accurate predictions.

The data visualization and predictive attributes identify payload mass, launch site, and booster version as key attributes that affect landing success. Using these attributes to predict a successful launch will allow SpaceX to lower costs.

Introduction

Space X Falcon 9 rocket launches cost of 62 million dollars compared to competitors whose launches cost 165 million dollars each. Savings are due to Space X can reusing the first stage. Determining if the first stage will land can estimate the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch.

The question that needs to be tackled is whether Space X's first stage can be predicted to land successfully. In determining the landing success rate, key attributes that impact success need to be found and tested to prove accurate.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Import data from a csv into a pandas data-frame for clean up
- Perform data wrangling
  - Replace null data with mean values and classified success and failure
- 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

- Data sets were fetched from the SpaceX API and scraped from Wikipedia's Falcon 9 Launch List.
- Several tools in python assisted collecting data such as BeautifulSoup and get requests to extract data into a pandas dataframe.
- Nasa coordinates and launch site locations were also extracted.

Data Collection – SpaceX API

- 52-54 fetch spacex data
- 68-69 filter data
- 85 replace missing data
- https://github.com/ShadowShark99/Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

```
In [52]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

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

```
In [53]: response=requests.get(static_json_url)

In [54]: response.status_code

Out[54]: 200
```

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

```
In [68]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']

         # Verify the result
         print(data_falcon9['BoosterVersion'].value_counts())

Falcon 9    90
Name: BoosterVersion, dtype: int64
```

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

```
In [69]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
         data_falcon9
```

```
In [85]: # Calculate the mean value of PayloadMass column
         payload_mean = data_falcon9['PayloadMass'].mean()
         payload_mean
         # Replace the np.nan values with its mean value
         data_falcon9['PayloadMass'].replace(np.nan, payload_mean,inplace=True)
         data_falcon9.head()
         data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

- 8-10 get html, create soup
- 11-12 track table w/ info
- 21 extract columns

- https://github.com/ShadowShark99/Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

In [8]:
```python
# use requests.get() method with the provided static_url
# assign the response to a object
response= requests.get(static_url)
```

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

In [9]:
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, 'html.parser')
```

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

In [10]:
```python
# Use soup.title attribute
soup.title
```

Out[10]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

In [11]:
```python
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

In [12]:
```python
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">U
TC</a>)
</th>
```

In [21]:
```python
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.

Data Wrangling

- Calculate number of launches per site
- Calculate number and occurrence of each orbit
- Calculate number and occurrence of mission outcomes of orbits
- Classify landing outcome label (1,0) from outcome column

- Translated categorical data to numeric data to quantify results
- https://github.com/ShadowShark99/Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Plot scatter, bar, and line charts to visualize correlation between attributes and mission outcomes.
- Attributes included Flight Number, Payload Mass, Launch Site, and Orbit
- Data analysis and feature engineering using Matplotlib
- https://github.com/ShadowShark99/Applied-Data-Science-Capstone/blob/main/edadataviz.ipynb

EDA with SQL

- Query sum, min date.
- Query successful Booster Versions with payload mass between 40k-60k
- Query count of success and failure mission out comes
- Query Booster Versions with max payload mass
- Query months with failed attempts in 2015
- Query ascending count of landing outcomes from 2010-6-04 to 2017-3-20
- https://github.com/ShadowShark99/Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Mark all launch sites using markers and circles to visualize site location
- Mark color to illustrate success or failure with green or red.
- Add Mouse position to get coordinates of sites
- Add Polyline to measure distance from site to coastline
- https://github.com/ShadowShark99/Applied-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

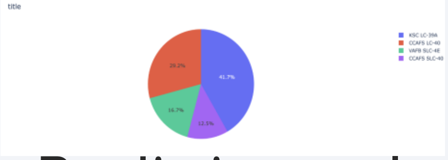
- Add drop downlist to choose all sites or a specific site graph
- Add pie chart to show total successful launch for all sites or success ratio of a specified launch site.
- Add slider to select payload range for the scatter plot to show correlation between payload and launch success
- https://github.com/ShadowShark99/Applied-Data-Science-Capstone/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

- Standardize attribute data (X) and Success data (Y)
- Split data into training and testing data
- Train regression, svm, tree, and knn models and evaluate their accuracy
- Plot confusion matrix to visualize test results
- https://github.com/ShadowShark99/Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

- Exploratory data analysis results: Payload mass, Year, and Launch site are found to affect mission outcome
- Interactive analytics demo in screenshots

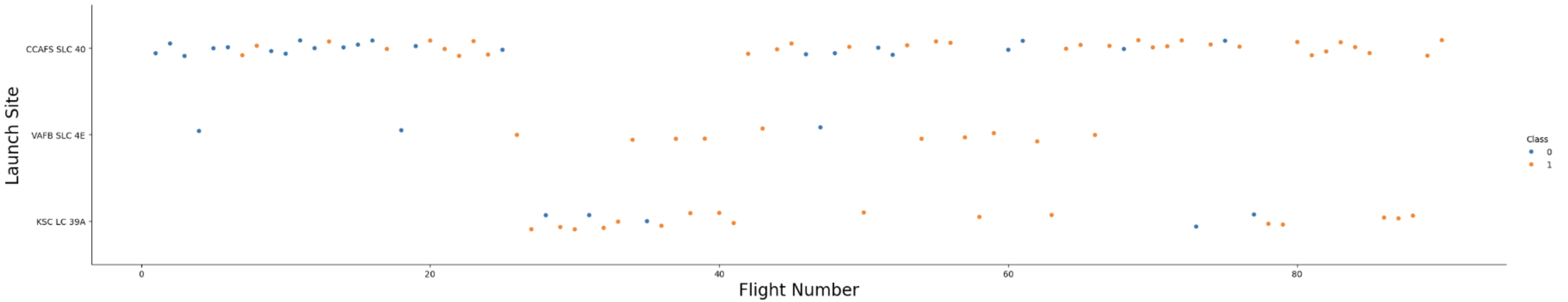


- Predictive analysis results: Tree model had highest accuracy

Section 2

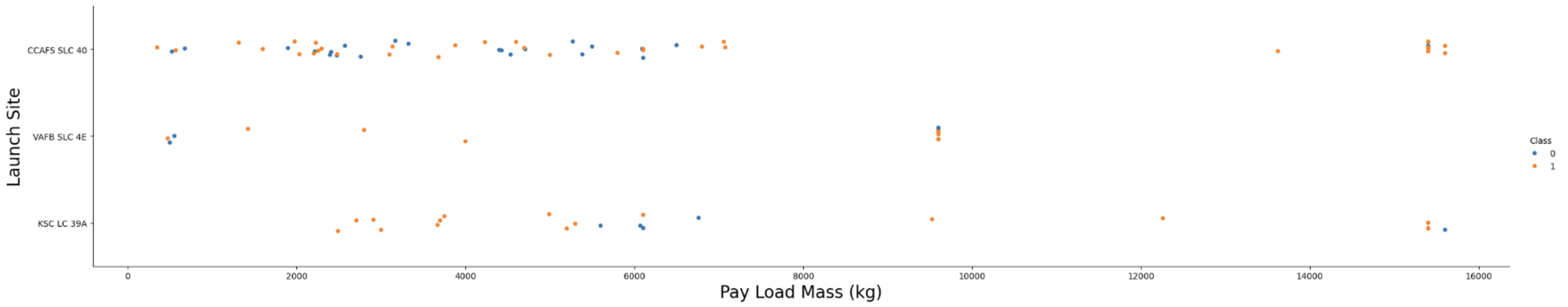
Insights drawn from EDA

Flight Number vs. Launch Site



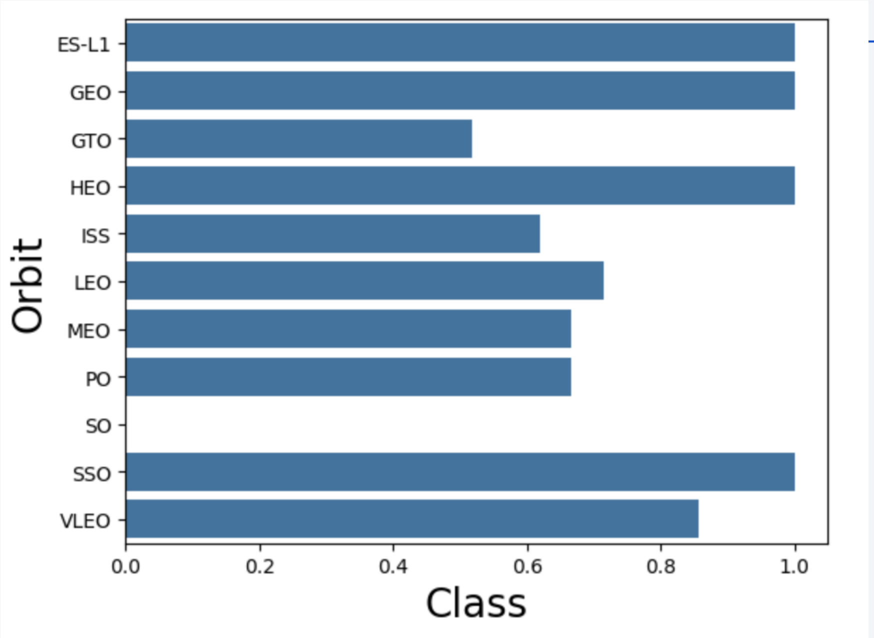
- Some Launch sites have higher success rate

Payload vs. Launch Site



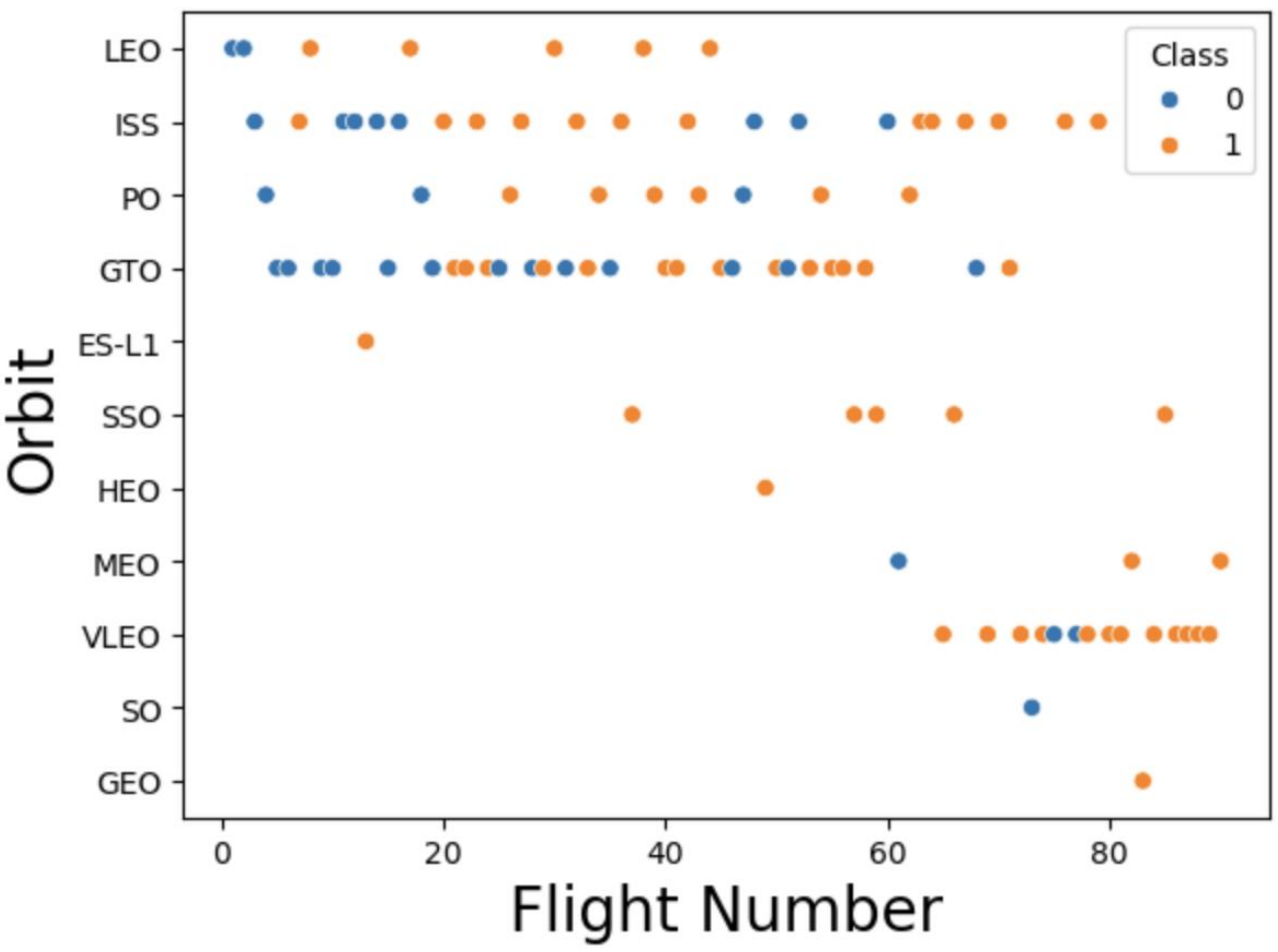
Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type



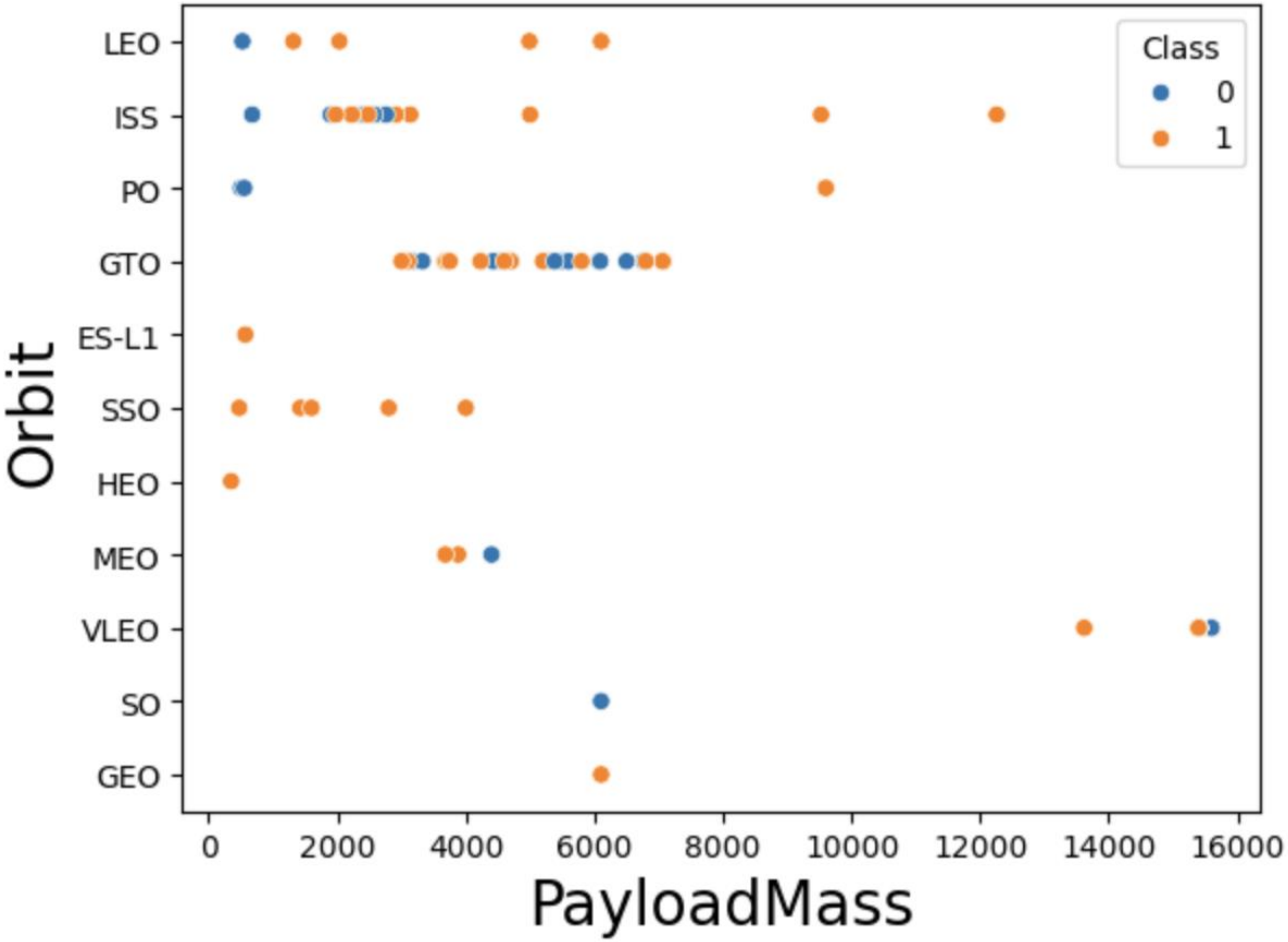
- ES-L1, GEO, HEO, and SSO have highest success rates

Flight Number vs. Orbit Type



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

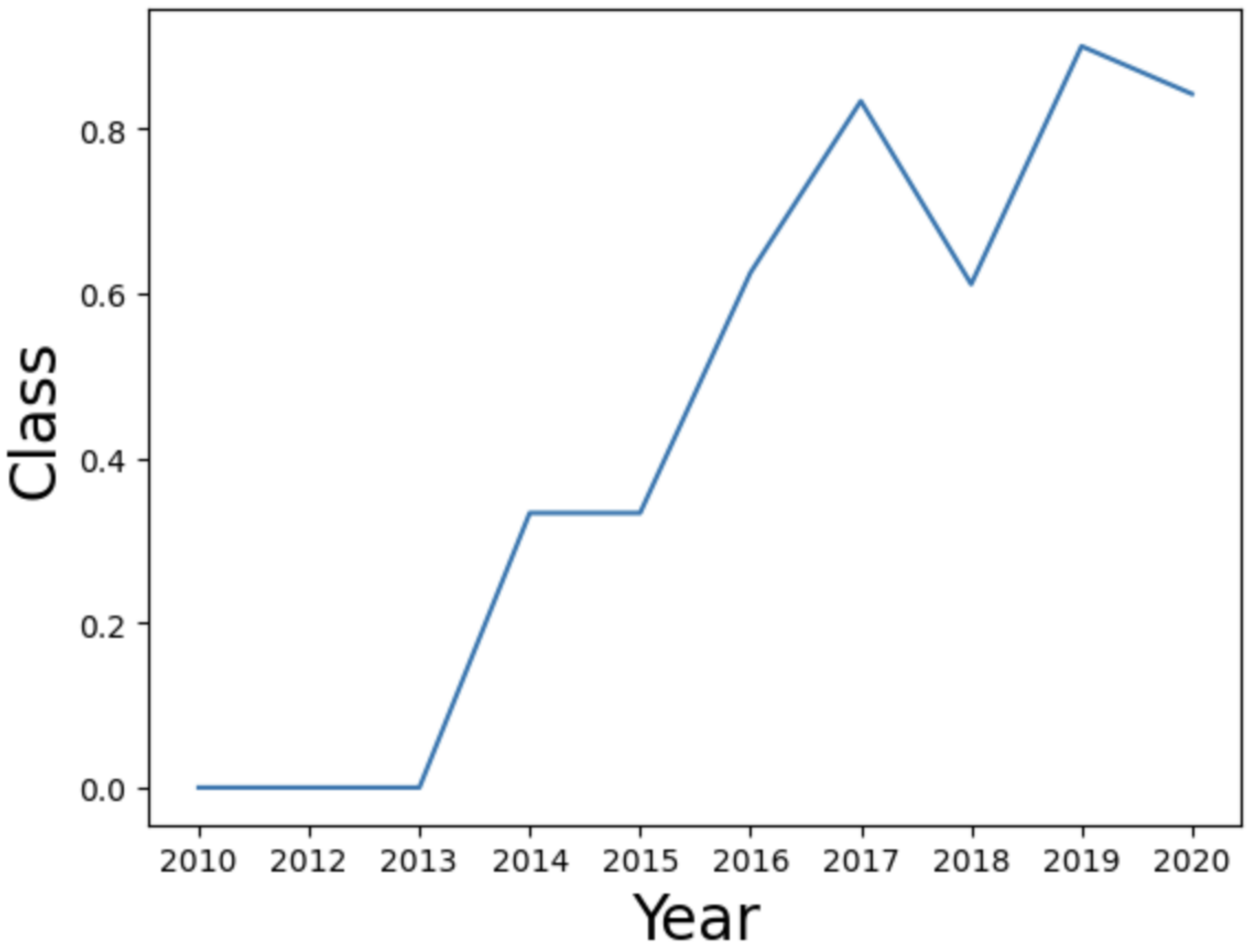
Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend



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

All Launch Site Names

- Select Distinct Launch Sites

```
%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 'CCA'

- Use Limit =5 and Like "CCA%" to filter results

```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" Like "CCA%" LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
2010-12-08	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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Total Payload Mass

- SUM function takes total of the PAYLOAD_MASS__KG_ column

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE "Customer" Like "NASA (CRS)";
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE "Booster_Version" Like "F9 v1.1";
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

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

```
%sql SELECT MIN("Date") FROM SPACEXTABLE WHERE "Mission_Outcome" Like "Success";
```

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

MIN("Date")
2010-06-04

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

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Mission_Outcome" Like "Success" AND PAYLOAD_MASS__KG_ > 400
```

```
 * sqlite:///my_data1.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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT(*) FROM SPACEXTABLE GROUP BY "Mission_Outcome"
```

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

Mission_Outcome	COUNT(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

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

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE (PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPA
```

```
* sqlite:///my_data1.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

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

```
%sql SELECT substr(Date, 6, 2), "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE substr
```

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

substr(Date, 6, 2)	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	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

```
%sql SELECT "Landing_Outcome",COUNT(*) FROM SPACEXTABLE WHERE "Date" > '2010-06-04' AN
```

```
 * sqlite:///my_data1.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
Precluded (drone ship)	1
Failure (parachute)	1

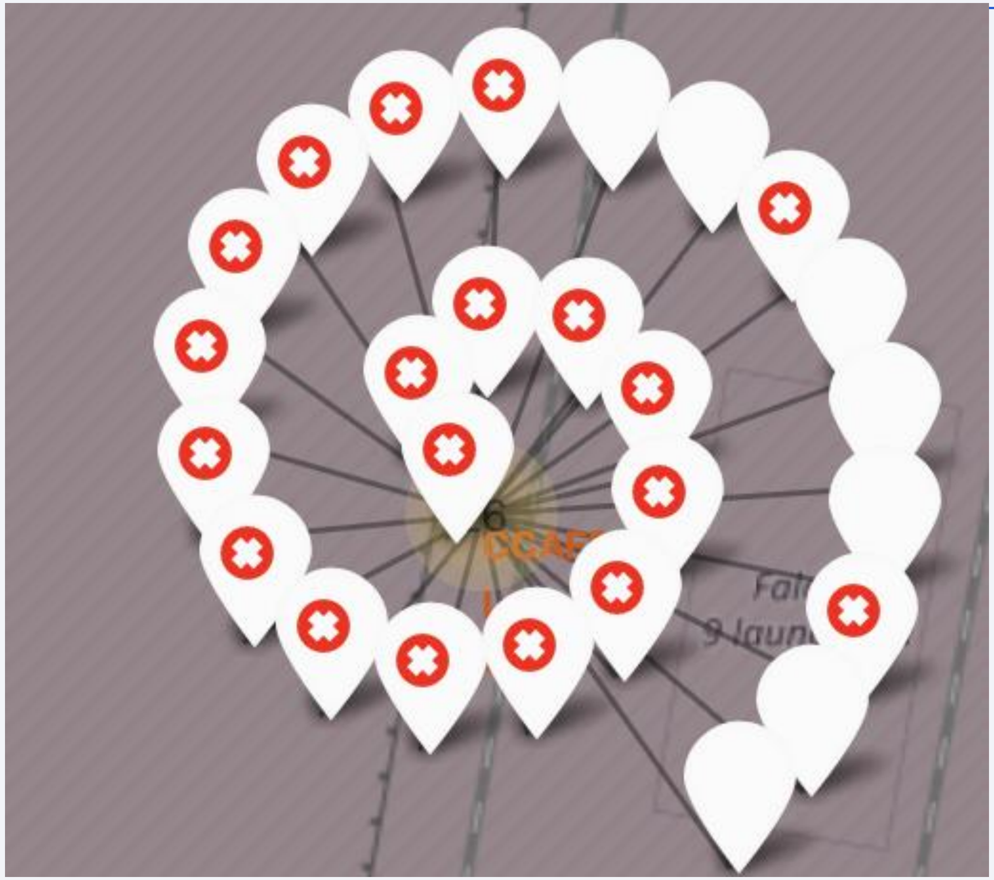
Section 3

Launch Sites Proximities Analysis

Launch Site Locations

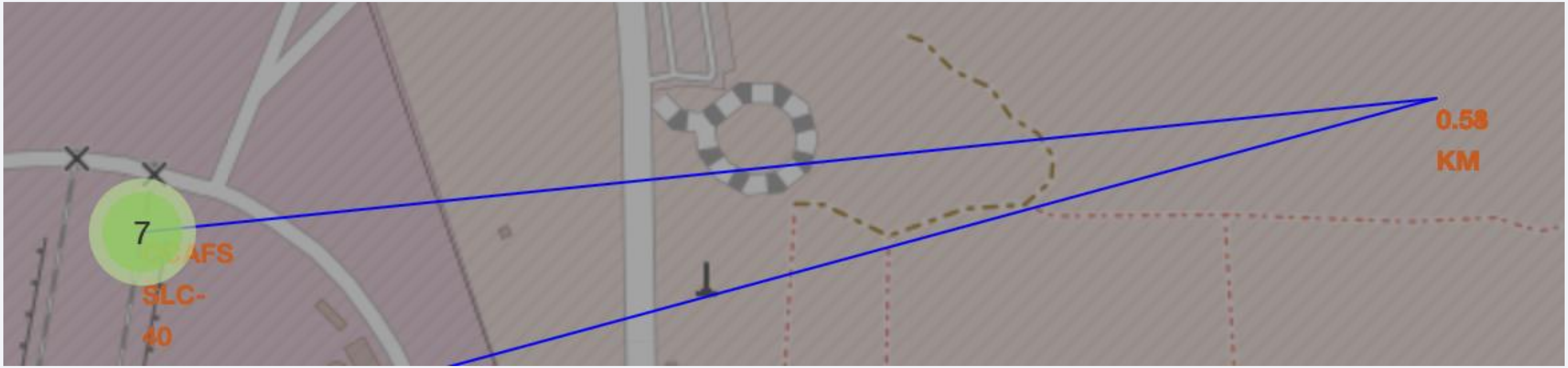
- Launch Sites are located in California(1) and Florida(3), marked in red.

Success and failure of Launch Results



- Successes are white and Failures are red.

Distance from site to Highway

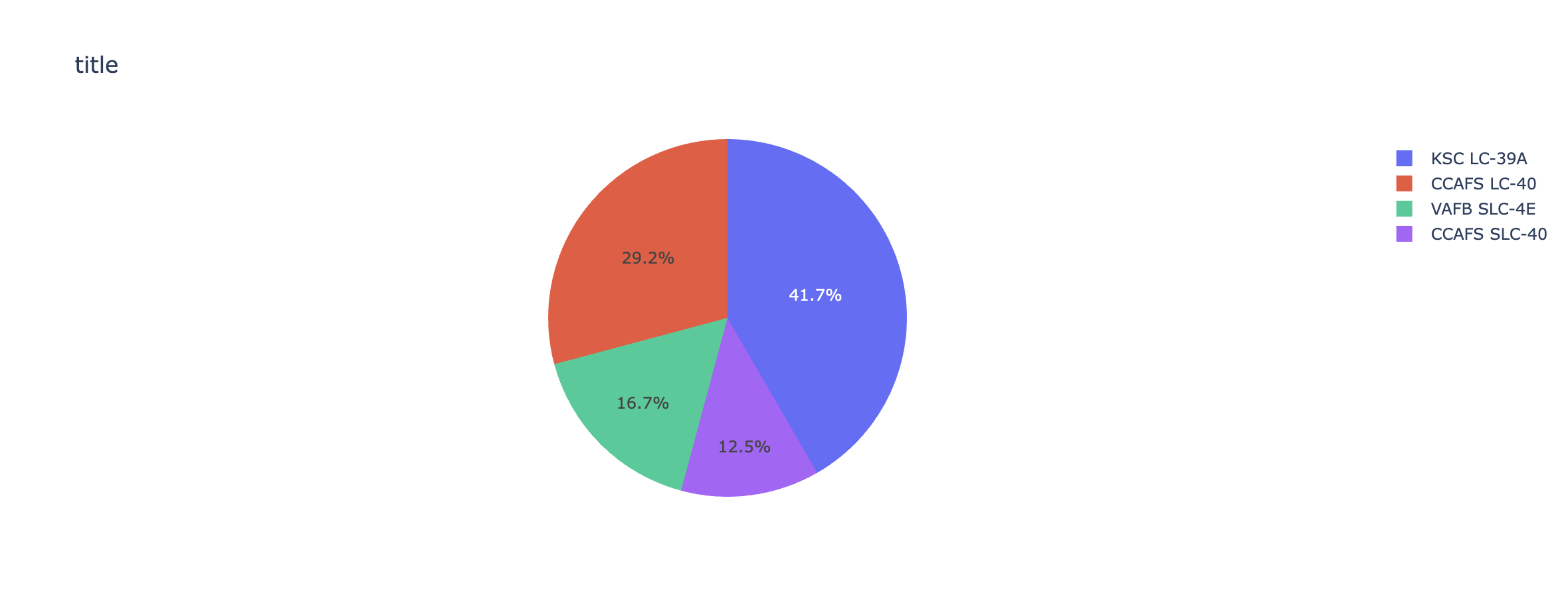


- This Launch site is .58 KM away from a highway

Section 4

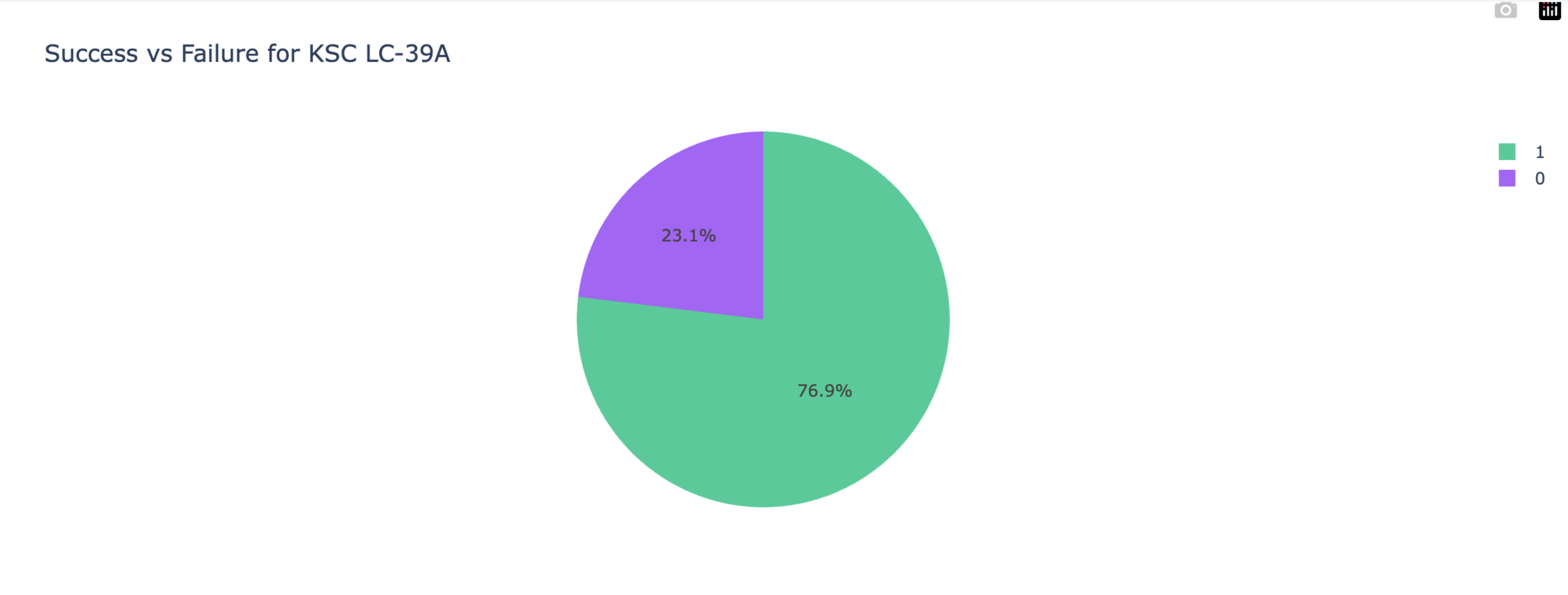
Build a Dashboard with Plotly Dash

Total Success Counts for all sites



- KSC LC-39A has the highest total success

Highest Success ratio pie chart



- KSC LC-39A has the highest Success Ratio

Payload Max Range Scatter Plot

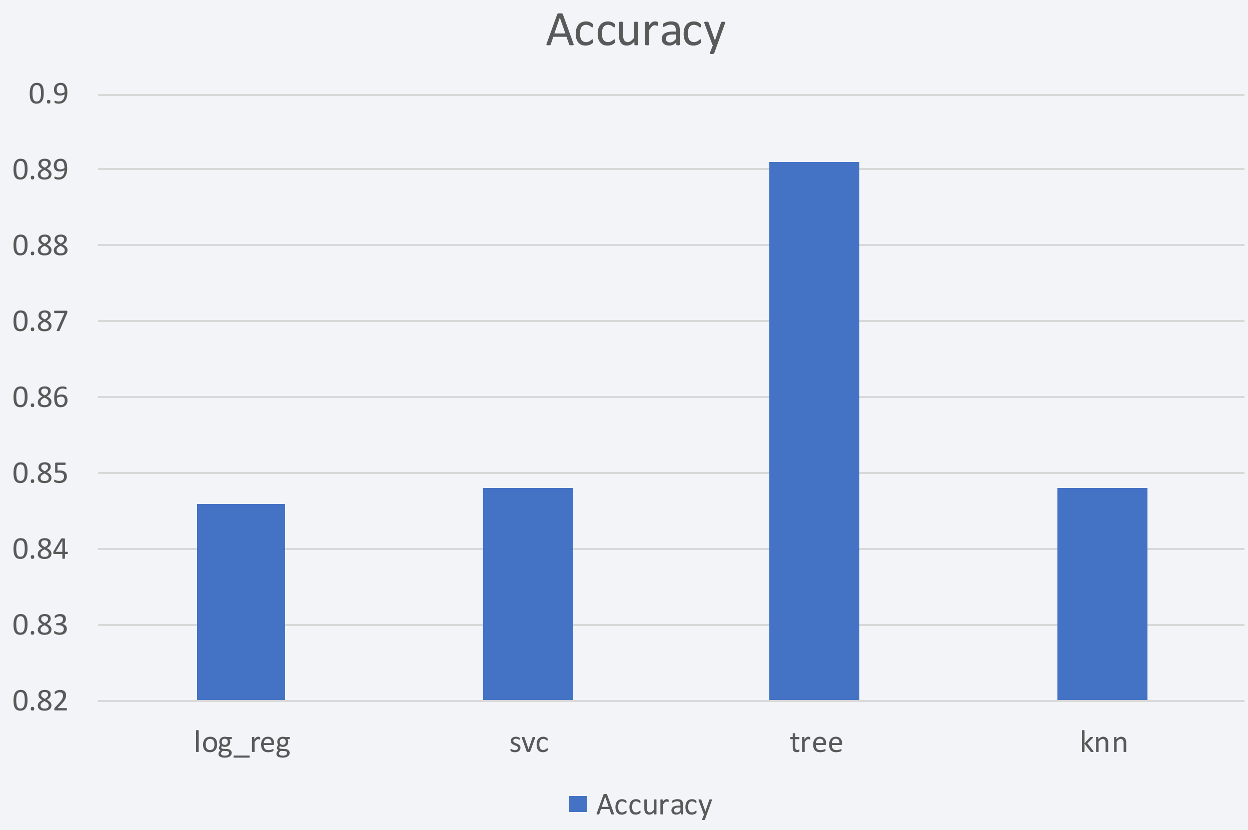
- Booster FT between 2k and 6k payload range has the highest success rate

Section 5

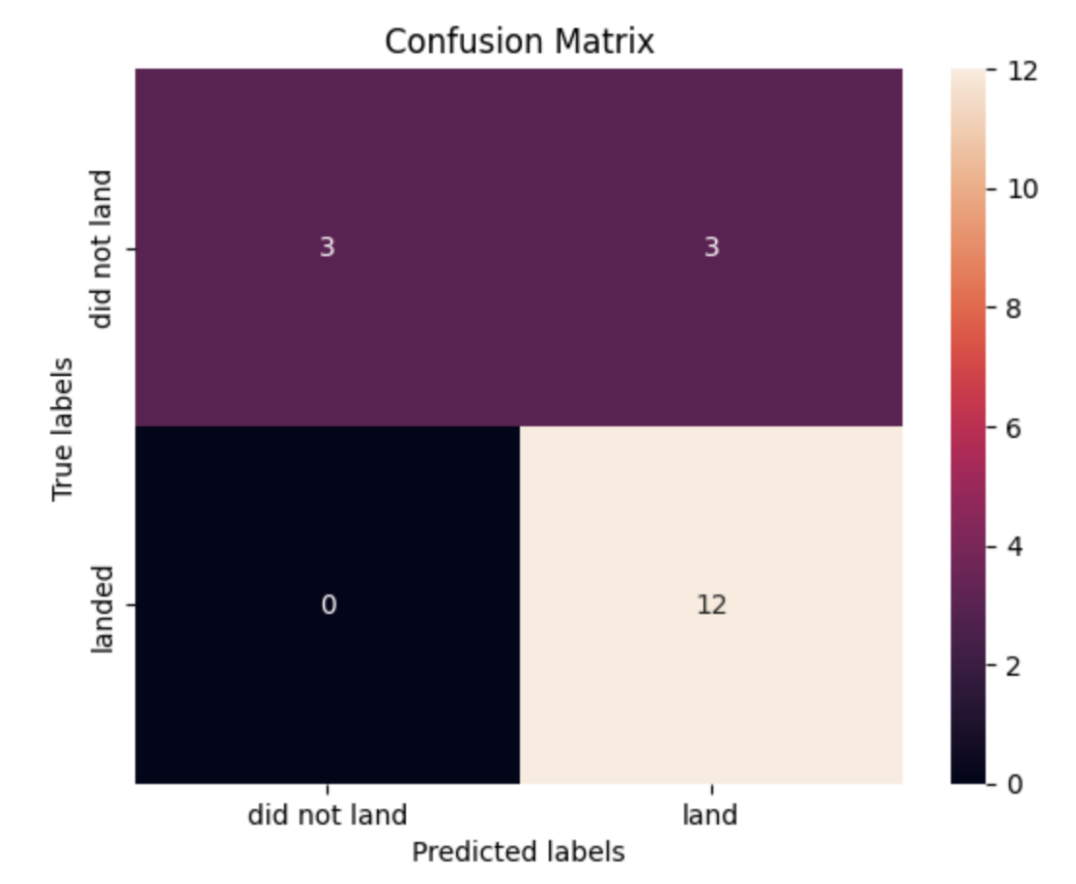
Predictive Analysis (Classification)

Classification Accuracy

- Decision Tree Classifier has the highest accuracy.



Confusion Matrix



- 12 True Positives, 3 false positives, 3 true negatives

Conclusions

- The Tree Classifier Model has the highest accuracy
- Best Parameters include entropy and max_depth 4.
- 89% accuracy for predicting landing is crucial to calculate cost

of launch.

- Using this model will save the most cost for first launch
- …

Appendix

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

▸ DecisionTreeClassifier

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 4, 'm
ax_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'ran
dom'}
accuracy : 0.8910714285714285
```

Thank you!