Winning Space Race with Data Science

Leo Deer
2/11/23

Outline

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

Introduction

- Project background and context
  - SpaceX's Flacon 9 rocket is able to provide relatively cheap commercial space travel by reusing the first stage of the rocket process.
  - The first stage is the part of the rocket which boosts the payload of the rocket into space.
- Problems you want to find answers
  - While the Falcon 9 can reuse this first stage, sometimes they are unable to because the first stage crashes.
  - We want to analyze and draw insights on when the first stage will land.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Requested rocket launch data from SpaceX API
  - Web scraped from Wikipedia
- Perform data wrangling
  - Added column to specify when a landing was successful
- Perform exploratory data analysis (EDA) using Matplotlib visualizations and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models with SciKit-Learn
  - Used Logistic Regression, SVM, Decision Trees, and K – Nearest Neighbor

Data Collection Process – SpaceX API

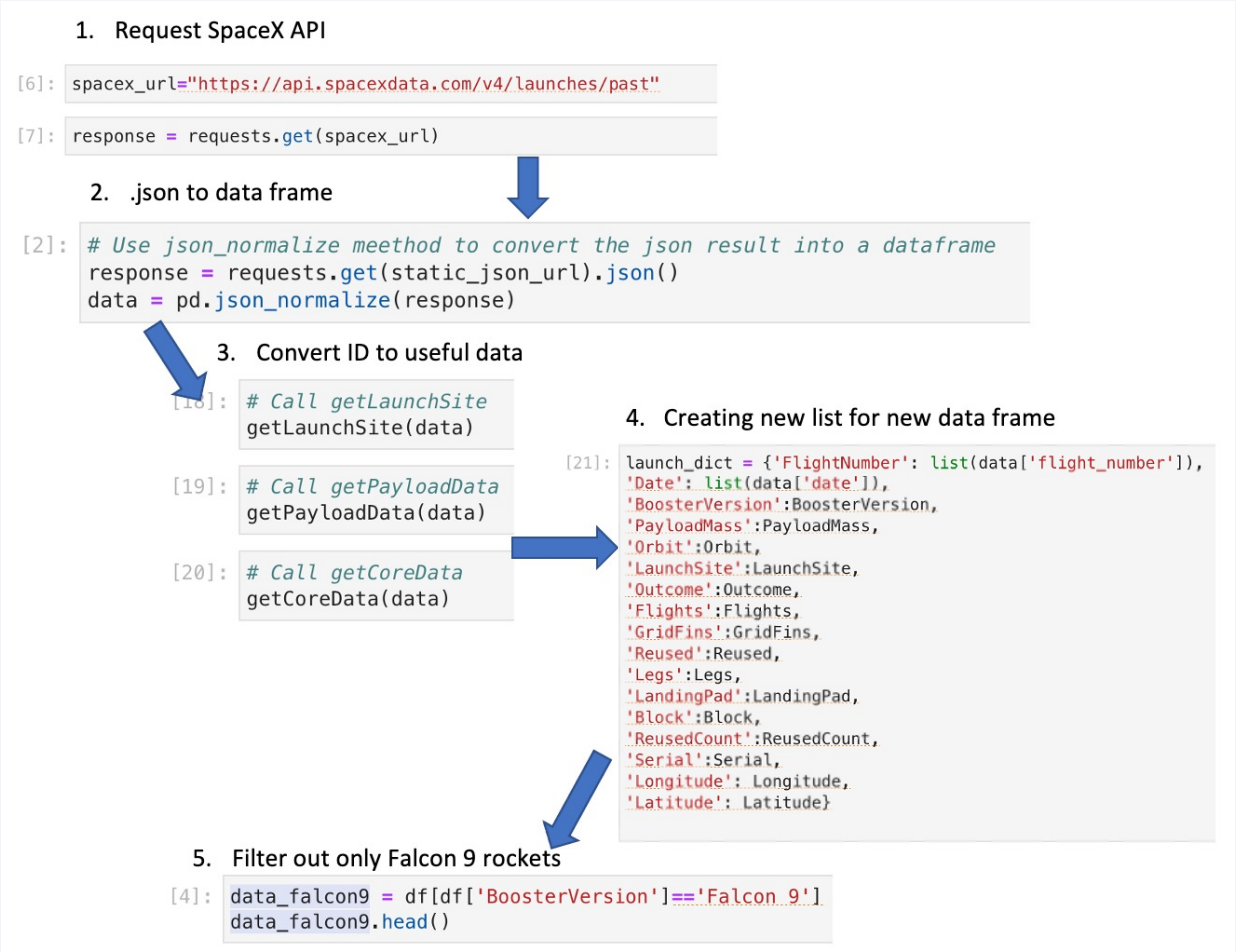
1. We requested the Space X rocket data from the Space X API.
2. We decoded the response as a .json file and turned it into a Pandas dataframe.
3. Used the API again to convert ID numbers into data we can understand.
    - Got booster name, payload mass, orbit type, name and coordinates of launch pad, landing information
4. Assign these new lists to our dataframe.
5. We filtered out all rocket information that wasn't from a Falcon 9 rocket launches.

Data Collection Process – SpaceX API

- Here we have a flowchart of the process for collecting the data from the SpaceX API.

- Here is a link to a GitHub repo containing this code:

  https://github.com/LeoDeer-99/IBM-Data-Science-Certification/blob/main/10Capstone/1_1_DataCollection.ipynb



Data Collection Process – Web scraping Wikipedia

1. Made a get request for the Wikipedia article on SpaceX launches.
2. Created a beautiful soup object from this get request.
3. Collected all the tables from the Wikipedia site.
4. Created a dataframe from the data in the tables.

Data Collection Process – Web scraping Wikipedia

- Here we have a flowchart of the process for scraping data from Wikipedia

- Here is a link to a GitHub repo containing this code:
  https://github.com/LeoDeer-99/IBM-Data-Science-Certification/blob/main/10Capstone/1_2_DataWebscraping.ipynb

Data Wrangling

- To format the data properly we first filled in missing values for PayloadMass with the mean PayloadMass value.

```
[47]: # Calculate the mean value of PayloadMass column
      payloadMassMean = data_falcon9['PayloadMass'].mean()

      # Replace the np.nan values with its mean value
      data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, payloadMassMean)

      data_falcon9.isnull().sum()
```

- From here we wanted to add another column to our dataframe indicating if the first stage successfully landed.
    1. First, we found each landing outcome category.
    2. Then separated all the failed landings.
    3. Finally, we populated a new list of successful (value = 1) and unsuccessful (value = 0) outcomes and added this list to our dataframe as a column.

Data Wrangling – Landing Column Flowchart

- Here we have a flowchart of the process for data wrangling

- Here is a link to a GitHub repo containing this code:
  https://github.com/LeoDeer-99/IBM-Data-Science-Certification/blob/main/10Capstone/1_3_DataWrangling.ipynb

1. Find mission outcome category

```
[9]: # landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

2. Separated failed missions

```
[11]: bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

3. Added failed missions to list

```
[12]: landing_class = df['Outcome'].copy()
for i, outcome in enumerate(landing_class):
    if outcome in bad_outcomes:
        landing_class[i] = 0
    else:
        landing_class[i] = 1
landing_class
```

5. Added list to dataframe

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

Predictive Analysis

- Here's how we modeled our data using SciKit-Learn
    1. Loaded the dataframe.
    2. Converted the success column of the data frame to a numpy array.
    3. Standardized the other part of the dataframe to another array.
    4. Sectioned off training and testing data from these two arrays.
    5. Created a model and GridSearchCV objects and then fit the model to find best the model parameters.
    6. Scored the model and found a confusion matrix.
    7. Repeated this for Logistic Regression models, Support Vector Machine, Decision Tree, and K– Nearest Neighbor.
- Here is a link to a GitHub repo containing this code: https://github.com/LeoDeer-99/IBM-Data-Science-Certification/blob/main/10Capstone/4_1_LandingPredictiveModeling.ipynb

Predictive Analysis - Flowchart

- Here we have a flowchart of the process for Logistic Regression case

1. Convert success column to an array

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

2. Standardize other part of dataframe

```
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

3. Sectioned off training and testing data

```
[16]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

4. Created Model and GridSearchCV object and fit model

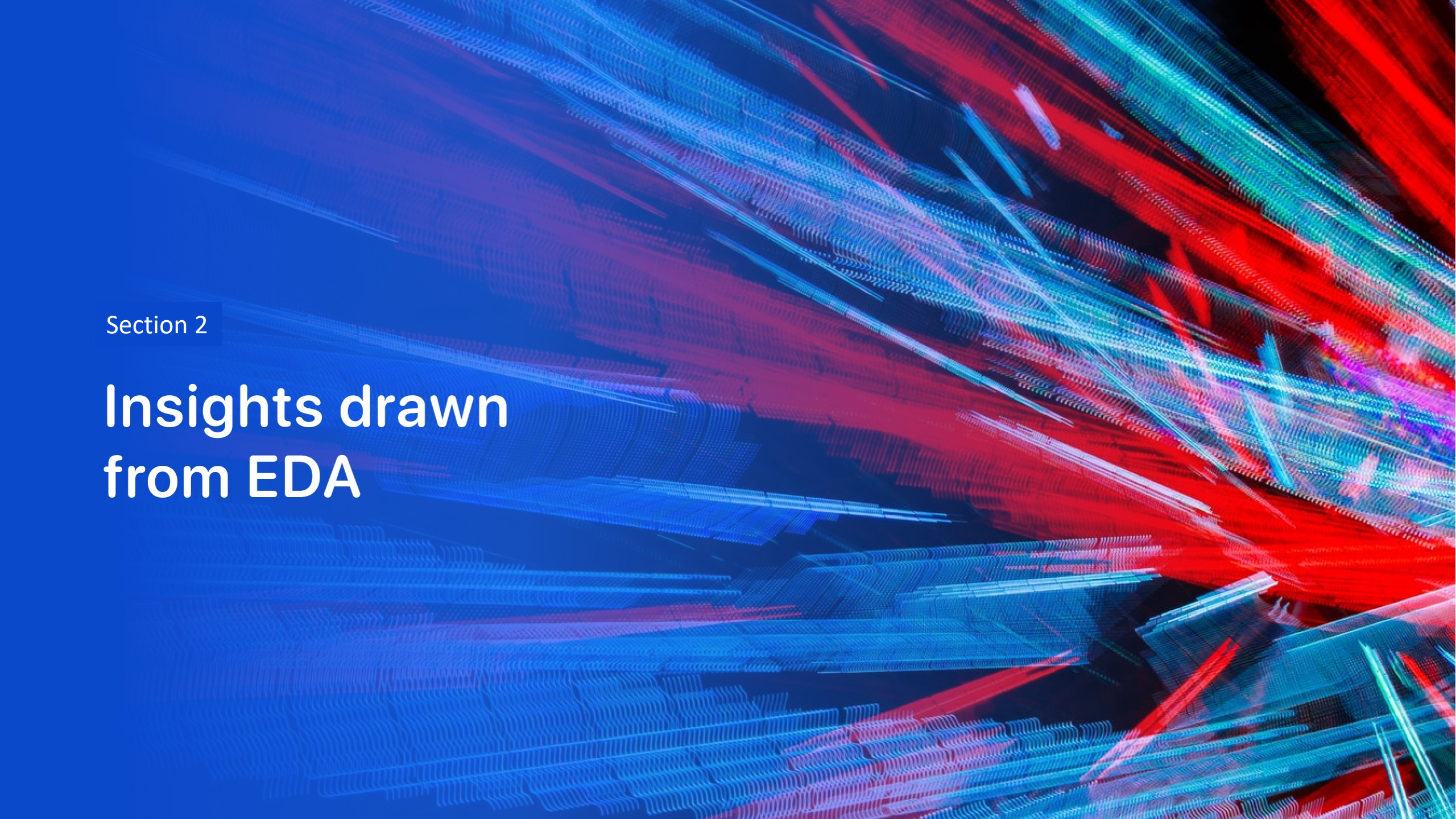
```
[26]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train,Y_train)
```

5. Score Model

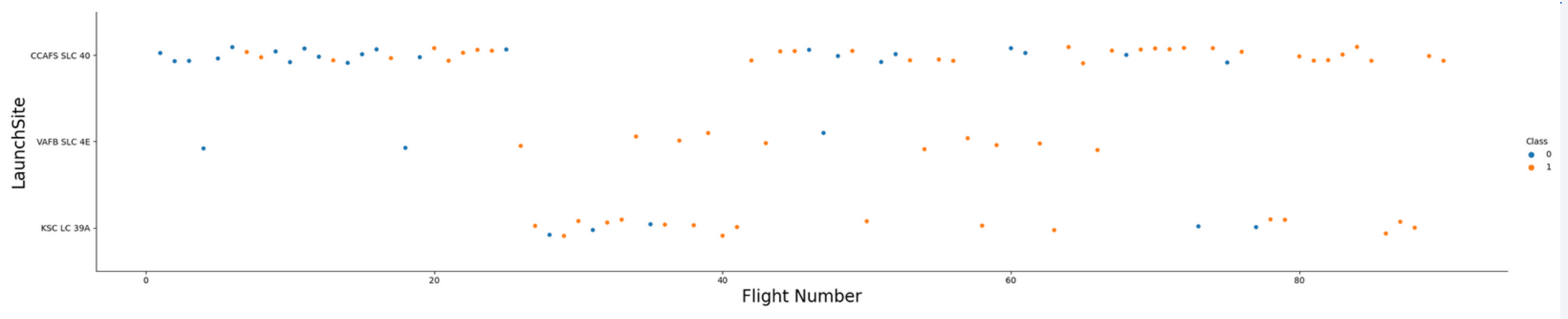
```
[28]: logreg_cv.score(X_train, Y_train)
```

6. Find Confusion Matrix

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

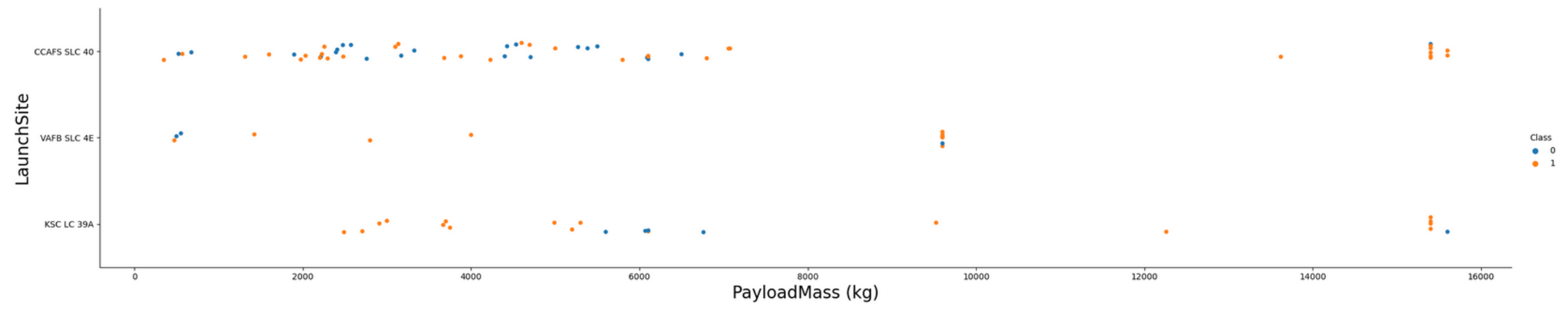

Section 2
Insights drawn from EDA

Plot - Flight Number vs. Launch Site



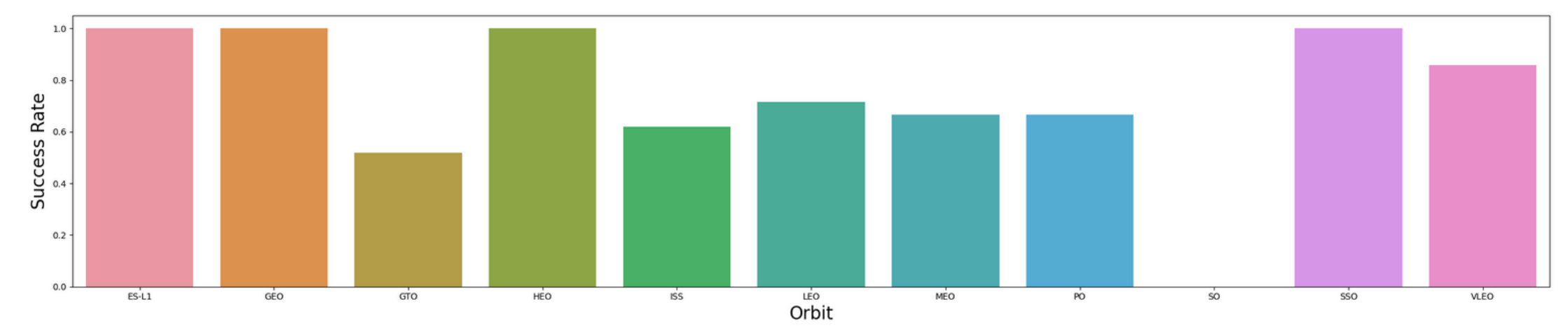
- In the table, class = 1 indicates a successful landing and class = 0 indicates an unsuccessful landing.
- It seems that at each sight, as flight number increases, more flights became successful.

Plot - Payload vs. Launch Site



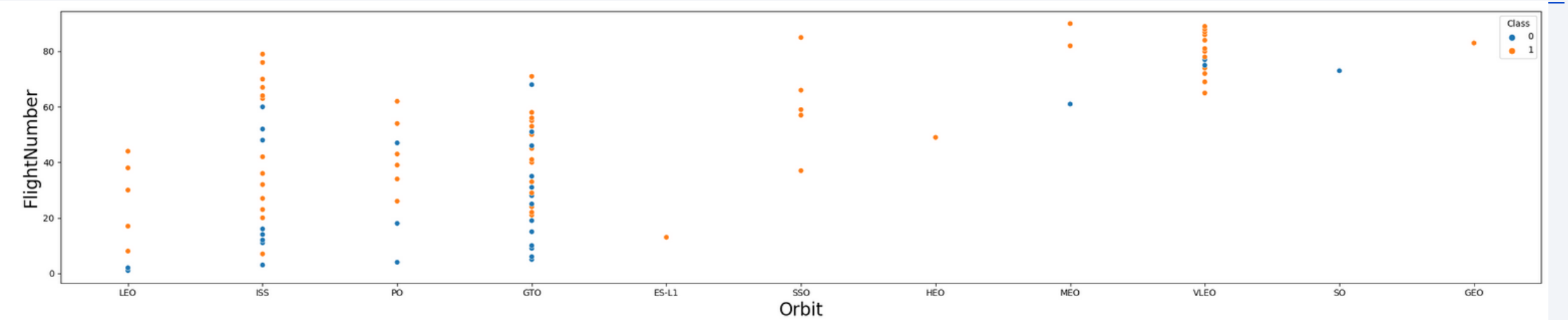
- It seems that at each sight, as PayloadMass increases, more flights became successful.
- So the rocket specifications used for large payloads may lead to successful first stage landings.

Plot - Success Rate vs. Orbit Type



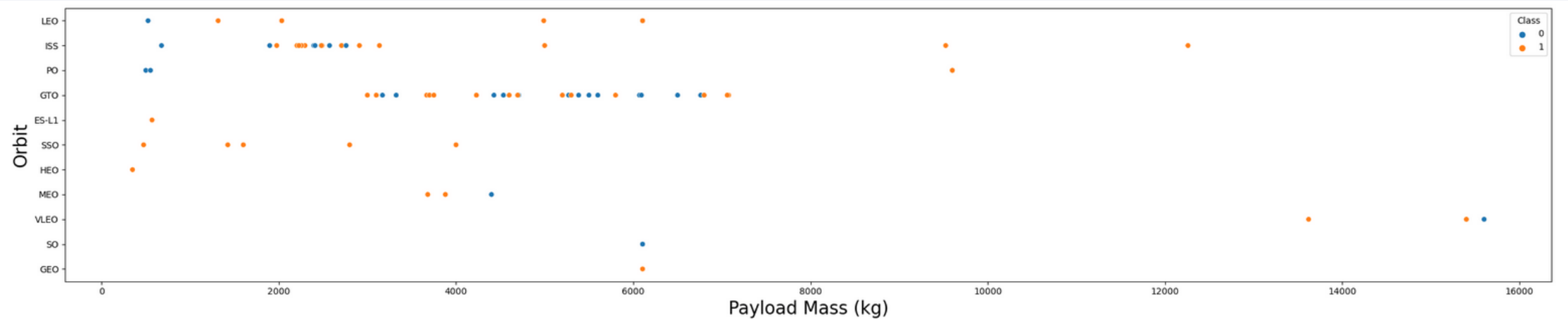
- We see that ES-L1, GEO, HEO and SSO are the most successful orbits

Plot - Flight Number vs. Orbit Type



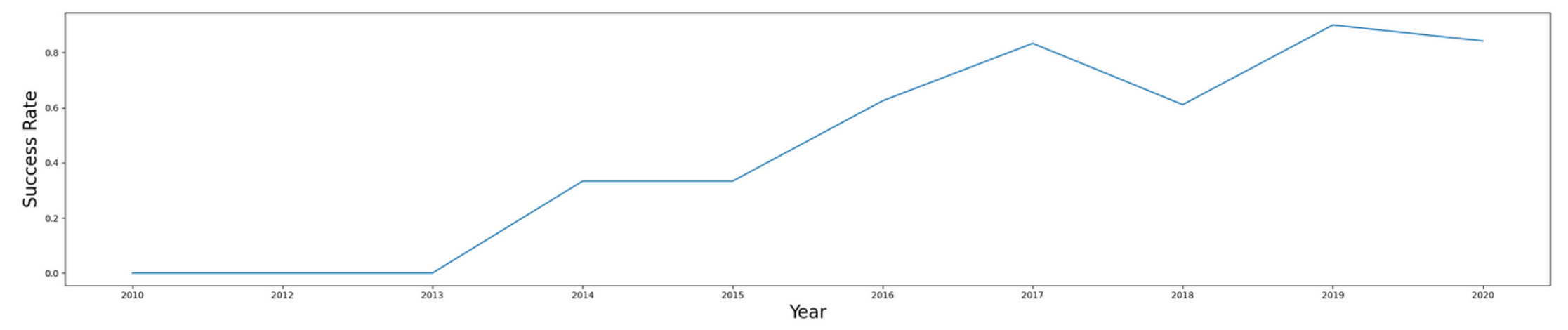
- In almost all orbit types (except GTO), we find that as flight number increases, more flights are successful.
- This is not the case with GTO which means SpaceX has not built technology suitable for the first stage of the rocket to return when propelling a payload to a GTO orbit.

Plot - Payload vs. Orbit Type



- Excluding GTO, it seems that as PayloadMass increases, most orbit lead to more successful missions. This is consistent with the plot on slide 16.

Plot - Launch Success Yearly Trend



- We find that since 2013, the Success Rate has been trending up.
- The rate is declining in the newest data, but we need more data to conclude that the Success Rate is falling.
- The rate at which Success Rate is increasing seems to be linear when averaged over time.

SQL Query - All Launch Site Names

[21]:
```
%sql select Launch_Site as 'Launch Site', count(Launch_Site) as 'Lauches' \
        from SPACEXTBL group by Launch_Site
```

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

[21]:

Launch Site	Lauches
CCAFS LC-40	26
CCAFS SLC-40	34
KSC LC-39A	25
VAFB SLC-4E	16

SQL Query - Total Payload Mass

```
[120]: %sql select sum(PAYLOAD_MASS__KG_) as 'Total Payload Mass (kg)' \
           from SPACEXTBL where Customer = 'NASA (CRS)'
```

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

[120]:

Total Payload Mass (kg)
45596

- Thus, a total of 45596 kg worth of payloads have been launched by SpaceX for NASA.

SQL Query - Average Payload Mass by F9 v1.1

Task 4

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

```
[121]: %sql select avg(PAYLOAD_MASS__KG_) as 'Average Payload Mass (kg)' \
           from SPACEXTBL where Booster_Version like 'F9 v1.1%'
```

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

[121]:

Average Payload Mass (kg)
2534.6666666666665

- Thus, the F9 v1.1 booster launches an average of 2535 kg worth of payload.

SQL Query - First Successful Ground Landing Date

```
[122]: %sql select Date from SPACEXTBL where "Landing _Outcome" like '%ground pad%' limit 1;
 * sqlite:///my_data1.db
Done.
[122]:
```

Date
22-12-2015

- Thus, the first successful ground landing was on December 22, 2015.

SQL Query - Successful Drone Ship Landing with Payload between 4000 kg and 6000 kg

```
[123]: %sql select Booster_Version from SPACEXTBL \
       where "Landing _Outcome" like 'Success (drone ship)' and\
       (PAYLOAD_MASS__KG_ between 4000 and 6000)
```

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

[123]:

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

- These are the boosters that have successful drone ship landings of the first stage which carried a payload between 4000 kg and 6000 kg

SQL Query - Total Number of Successful and Failure Mission Outcomes

```
[124…] %sql select Mission_Outcome, count(*) from SPACEXTBL group by Mission_Outcome
```

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

[124]:

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

- In total, 100 successful and 1 failed mission.

SQL Query - Boosters Carried Maximum Payload

```
[125]: %sql select Booster_Version as 'Booster Version', max(PAYLOAD_MASS__KG_) as 'Maximum Payload Mass' \
       from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)

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

[125]:

Booster Version	Maximum Payload Mass
F9 B5 B1048.4	15600

- Thus, the F9 B5 B1048.4 Booster carried the heaviest payload with a mass of 15600 kg

SQL Query - 2015 Launch Records

```
[127]: %sql select substr(Date, 4, 2), "Landing _Outcome", Booster_Version, launch_site \
from SPACEXTBL \
where "Landing _Outcome" = 'Failure (drone ship)' and \
(substr(Date,7,4)='2015')
```

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

[127]:

substr(Date, 4, 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

- Here are all the failed drone ship landings including their booster version and launch site from 2015.

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

```
[138…] %sql select "Landing _Outcome" as 'Landing Outcome', count("Landing _Outcome") as 'Frequency' \
from SPACEXTBL \
where Date between '04-06-2010' and '20-03-2017' and \
("Landing _Outcome" like '%Success%') \
group by "Landing _Outcome" \
order by count("Landing _Outcome") desc
```

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

[138]:

Landing Outcome	Frequency
Success	20
Success (drone ship)	8
Success (ground pad)	6

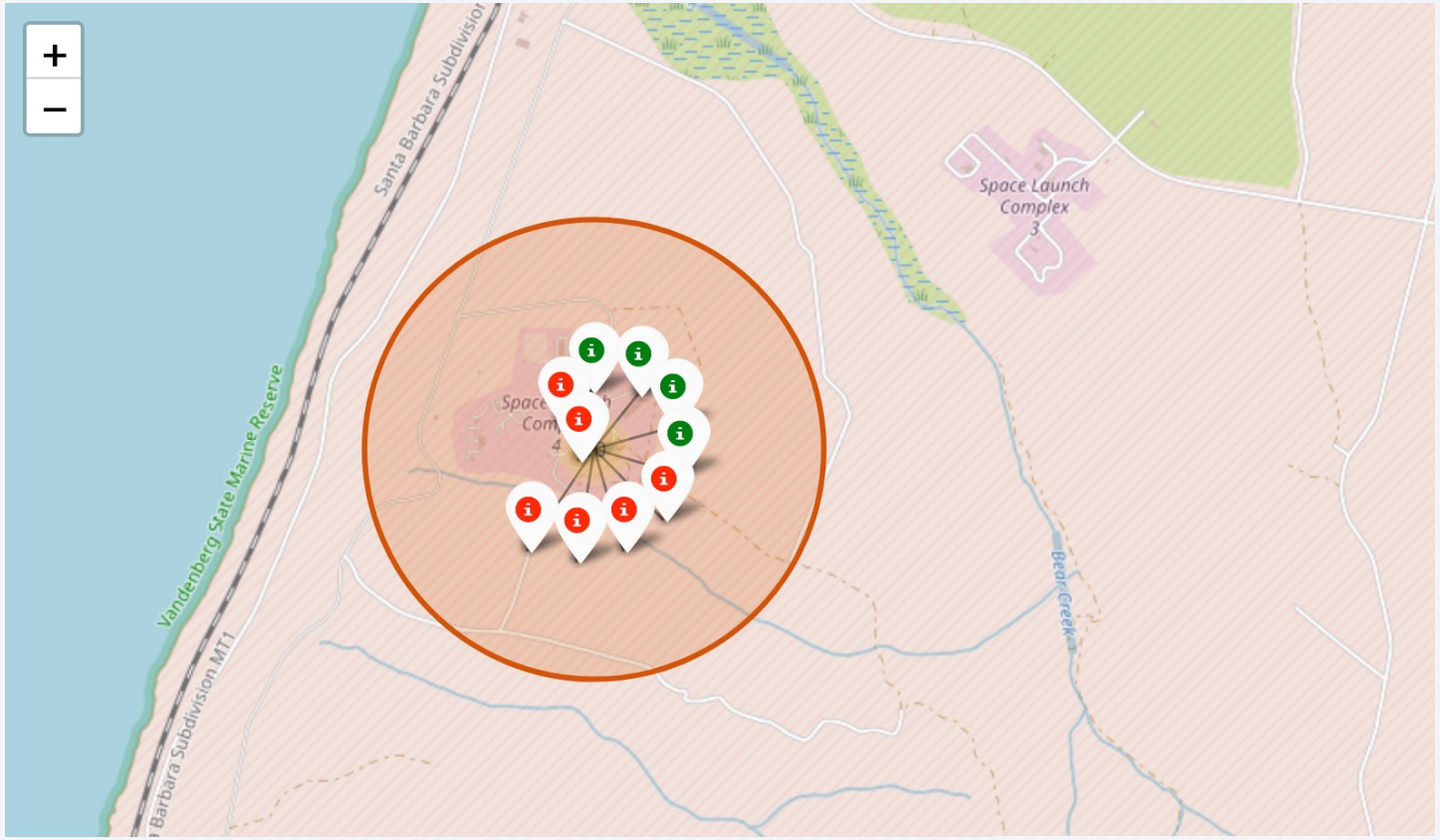
- Here we see that drone ship successes were more common than ground pad successes


Section 3
Launch Sites
Proximities Analysis

Launch Locations in in the United States

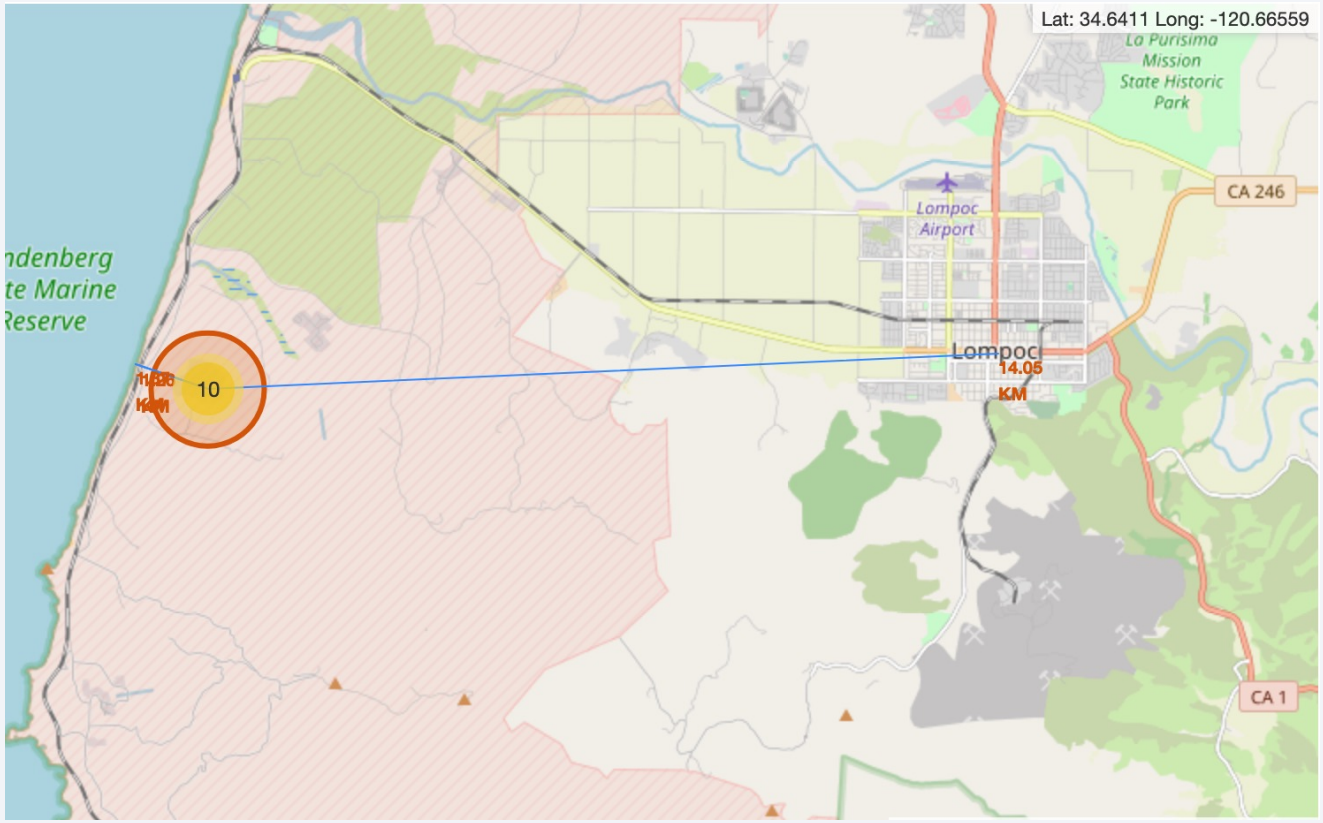
- From this map, we see that 46 launches have taken place in Florida and 10 in California.

Launches in California Zoomed in

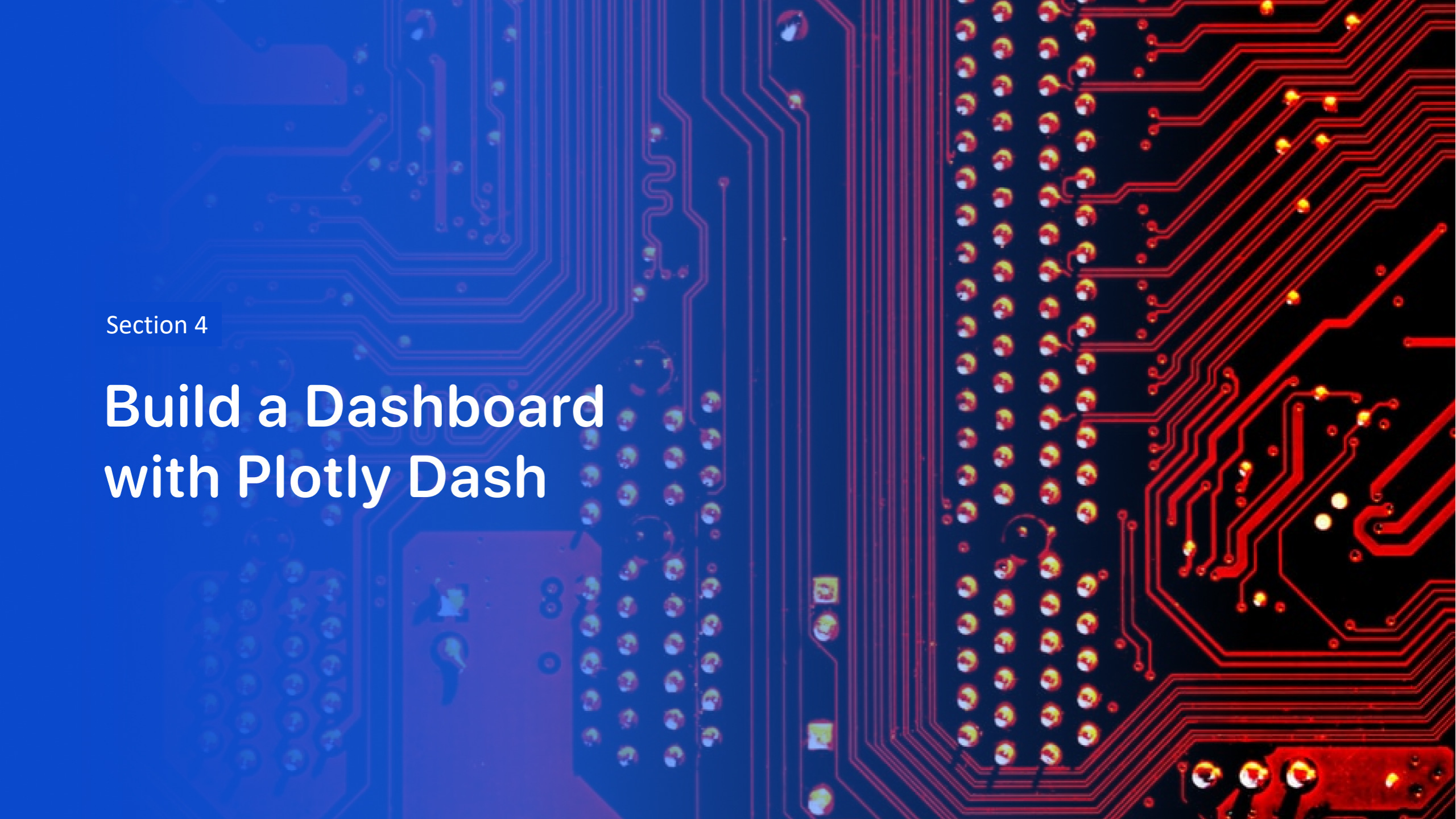


- Here we see the 10 launches from California. The green launches had a successful landing of the first stage, whereas the red launches were unsuccessful.

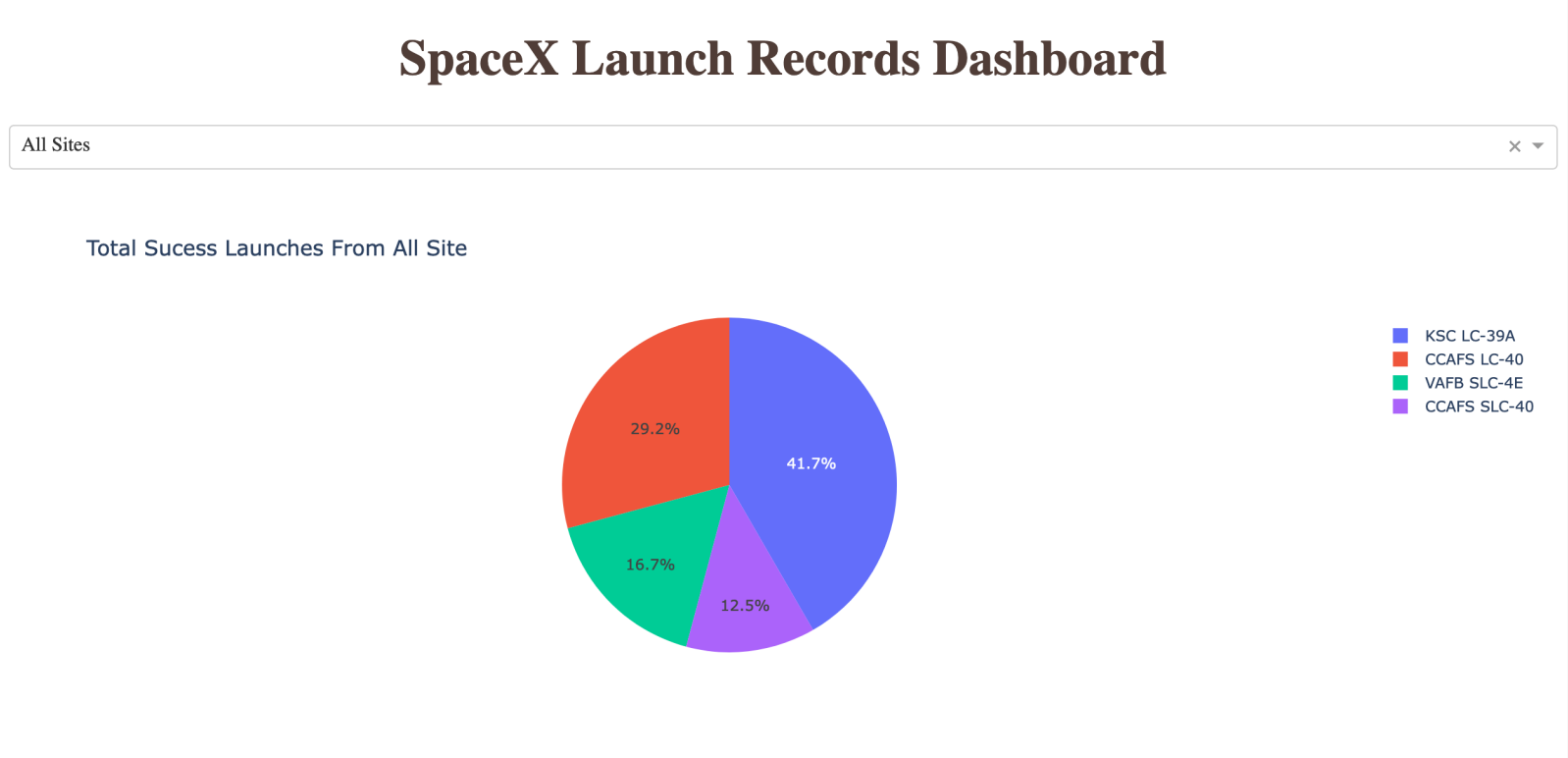
California Launches Proximity to Infrastructure



- Here we see that the 10 California launches took place 14.05 km from the nearest town Lompoc, 1.26 km from the coastal railroad, and 1.37km from the coastline.

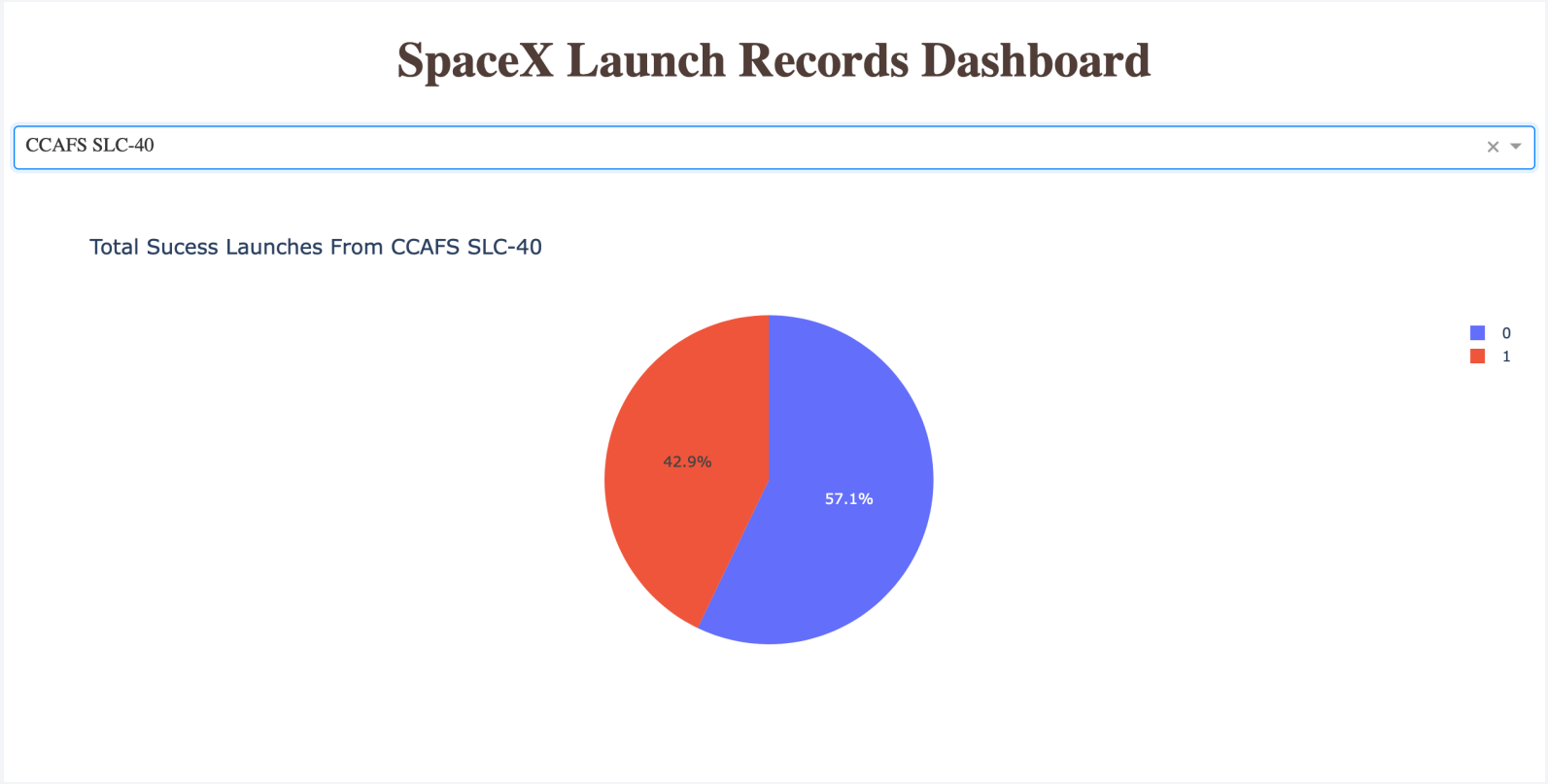

Section 4
Build a Dashboard with Plotly Dash

Plotly Dashboard Looking at All Sites



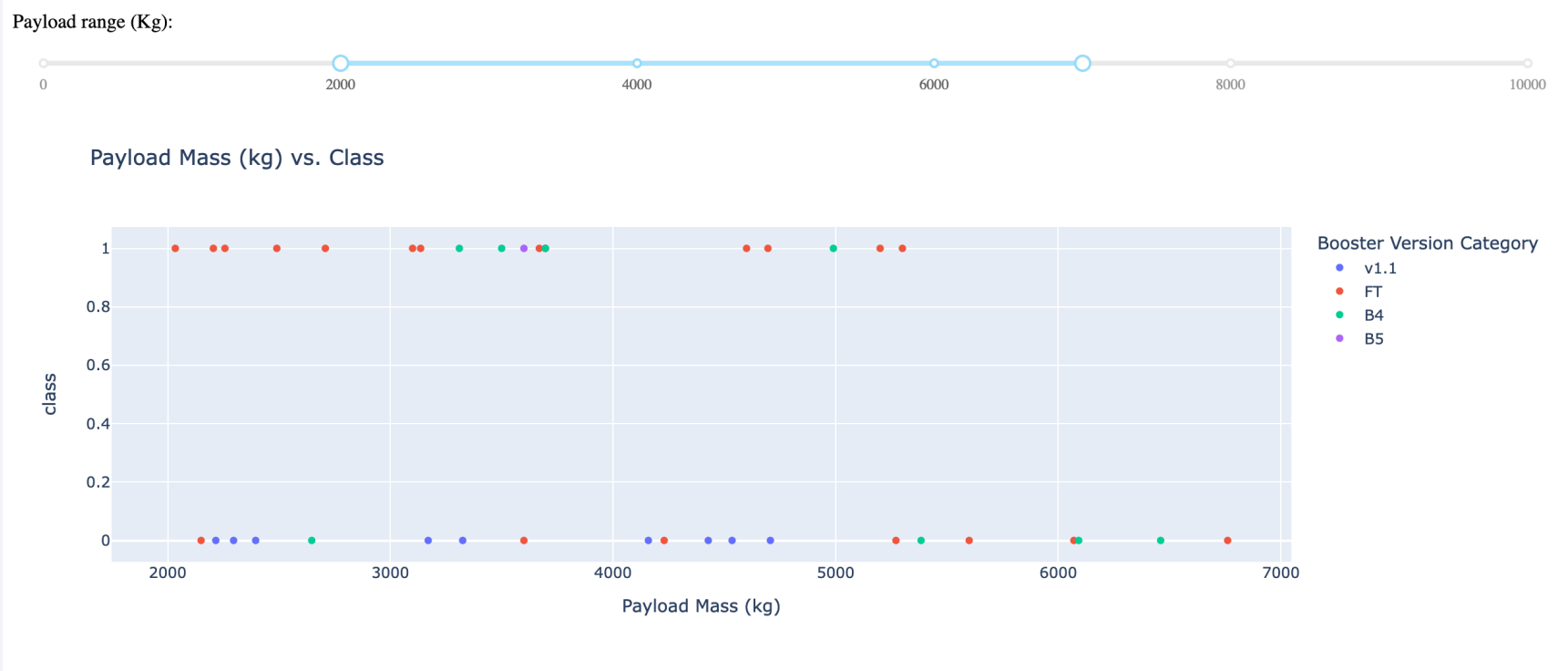
- From this we see that the KSC LC-39A has had the most successful launches out any launch site.

Plotly Dashboard Looking at the CCAFS SLC-40 Site



- We see that out of all the launches at the CCAFS SLC-40 launch site, most of them have not had a successful landing of the first stage.

Plotly Dashboard Payload vs Class Diagram



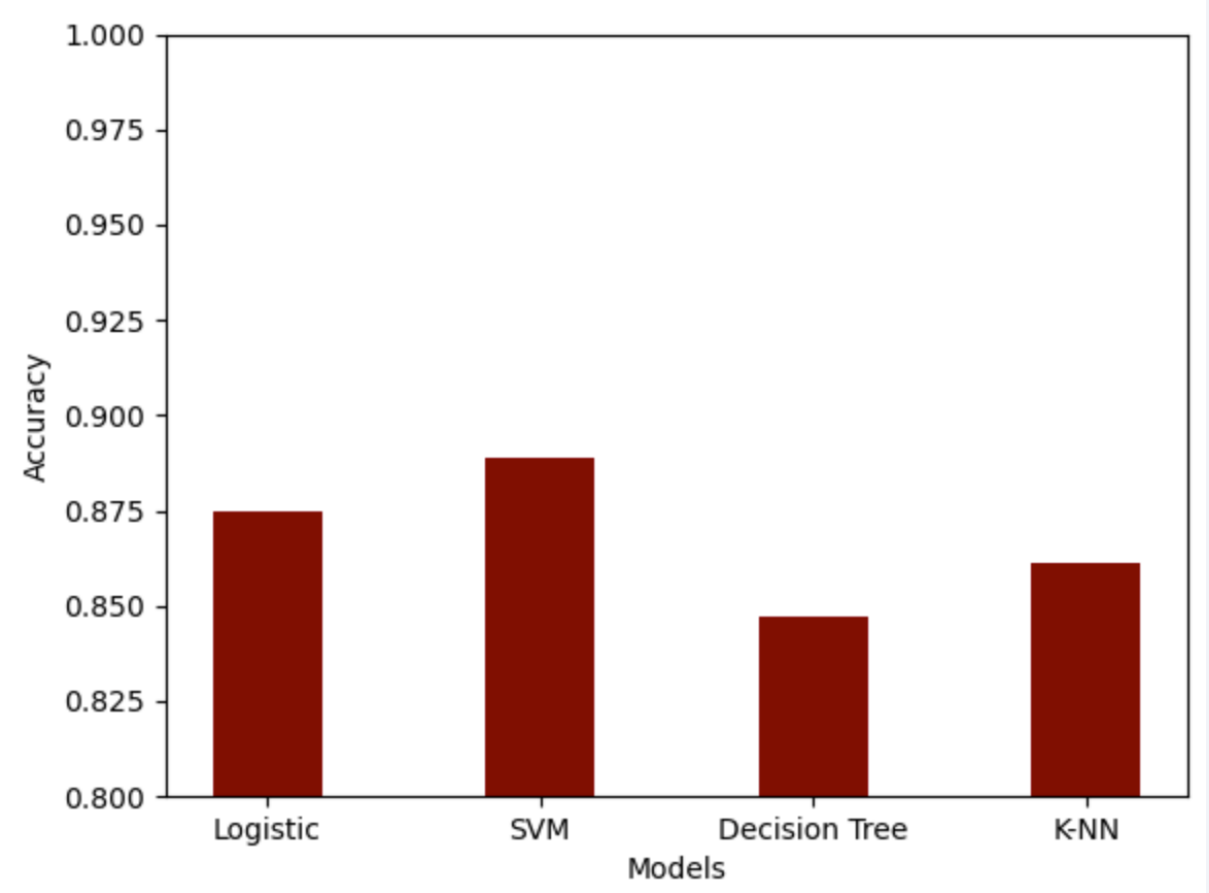
- We see that in this payload range, the successful landings have occurred with PayloadMass less than 5500 kg
- We see that the v1.1 booster has had no successful landings
- The FT booster seems like the most successful booster in this payload range

Section 5

Predictive Analysis (Classification)

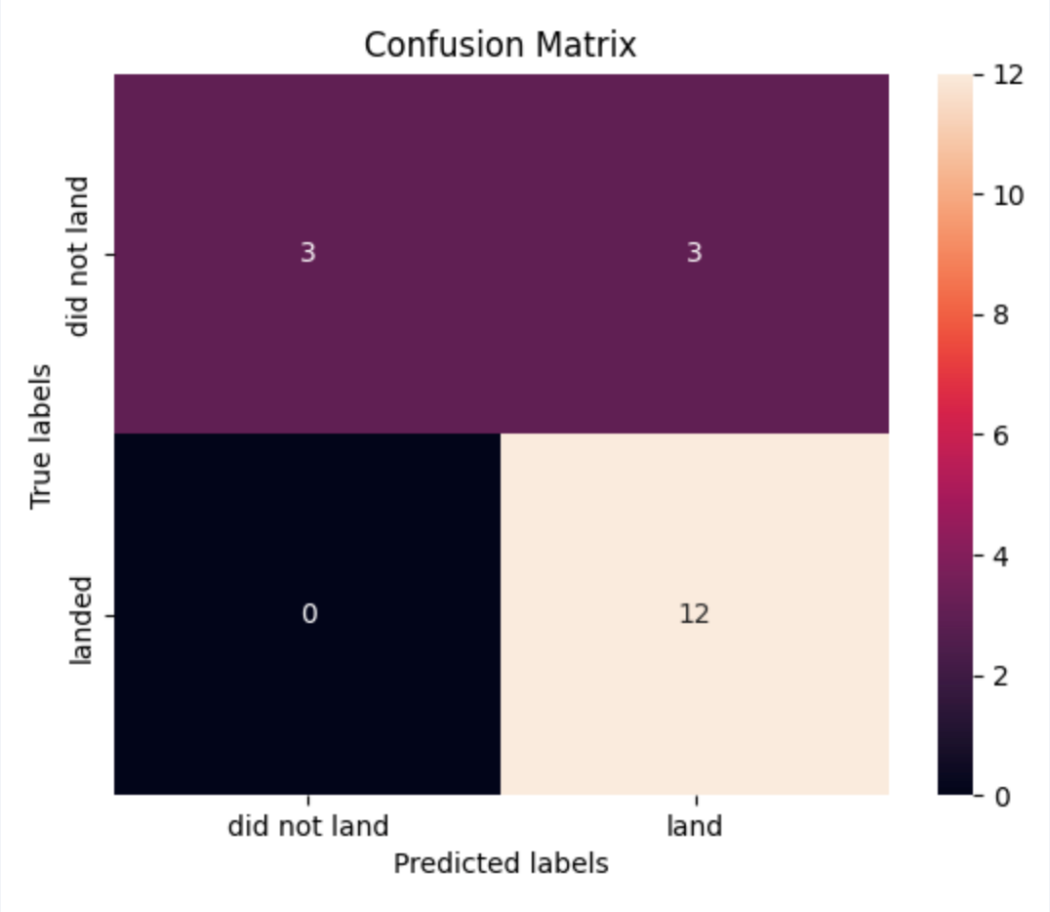
Classification Accuracy

- Here is a bar chart of all the models tested with their corresponding accuracy score.
- It seems Support Vector Machine is the most accurate with a score of approximately 89%



Confusion Matrix

- Here we see the confusion matrix for the Support Vector Machine model.
- We see that this model is very accurate at predicting when the first stage landed, but it is not accurate when predicting when the first stage did not land.



Conclusions

- Overall, the success rate has been increasing. We see this when flight numbers increase and as time has increased.
- We find that as PayloadMass increases, the Success Rate also increases. We also see that the F9 B5 B1048.4 booster has carried the heaviest payload and for that reason may be a marker of success. But we saw in slide 37 that this booster has been used infrequently, so we need more data to make a proper conclusion.
- Likewise, we found that the F9 v1.1 Booster had low success.
- In terms of orbits, most orbit types have been improving in success. But, for the GTO orbit, it still seems hard to predict whether the first stage will land.
- As far as launch sites, KSC LC-39A is the most successful.
- In the future I suggest analyzing other components of the rockets to try and find more trends in the rocket design that lead to success.

Appendix

- Data visualization code used to make plots: https://github.com/LeoDeer-99/IBM-Data-Science-Certification/blob/main/10Capstone/2_2_DataVisualization.ipynb
- Folium map code: https://github.com/LeoDeer-99/IBM-Data-Science-Certification/blob/main/10Capstone/3_1_LaunchSitesLocationsAnalysisFolium.ipynb
- Plotly Dashboard code: https://github.com/LeoDeer-99/IBM-Data-Science-Certification/blob/main/10Capstone/3_2_SpacexDashApp.py

Thank you!