IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Christopher D.
December 1st, 2025

Outline

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

Executive Summary

Background:

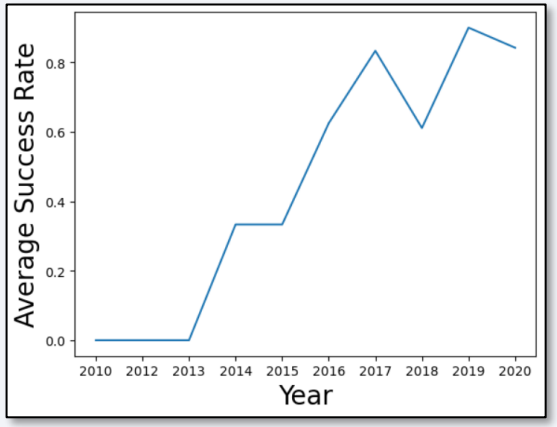
- **Predicting whether the first stage of a SpaceX rocket will land is key to estimating launch costs.**
- Using publicly available data, I conducted analysis, built dashboards and trained and evaluated machine learning models to predict landing outcomes for SpaceX rockets.

Methodology:

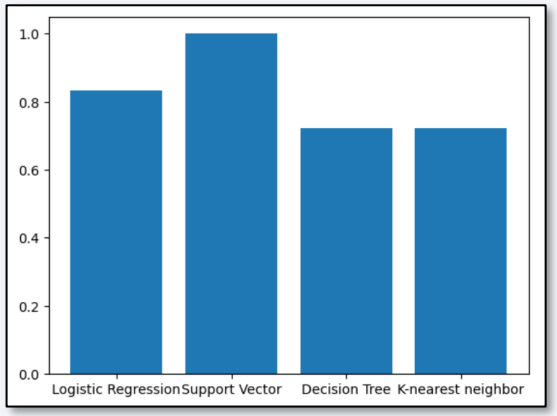
- Data Collection:
  - API Connector
  - Web Scraping
  - SQLite
  - Seaborn
- Data Wrangling / Exploratory Data Analysis:
  - Pandas
  - NumPy
- Data Visual Analytics & Dashboards:
  - Folium
  - Plotly + Dash
- Machine Learning Predictive Analysis and Model Evaluation:
  - Preprocessing Pipeline
  - ML Classification Models: Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbor
  - Cross-Validation Grid Search optimization

Results:

- Exploratory data analysis: The date of launch highly correlates with successful landing outcomes.



- Predictive analysis: The SVM Machine Learning model predicted landing outcomes with **100% accuracy** on test data.



Introduction

- The commercial space age is making space travel more affordable, with companies like Virgin Galactic, Rocket Lab, SpaceX, and Blue Origin.
- SpaceX is the most successful commercial space company. SpaceX achievements include ISS missions, Starlink satellite internet, and manned spaceflights, enabled by relatively low launch costs.
- SpaceX's Falcon 9 launches cost $62M versus $165M+ for competitors, largely due to **reusable rocket first stages**.
- **Predicting whether the first stage of a SpaceX rocket will land is key to estimating launch costs.** Sometimes it crashes or is sacrificed based on mission requirements.
- In this project, I acted as a data scientist for Space Y, a fictional competitor to SpaceX. Using publicly available data, I conducted analysis, built dashboards and trained and evaluated several machine learning models to predict landing outcomes for SpaceX rockets.

Section 1

Methodology

Methodology

Executive Summary

- Data Collection:
  - REST API Connector
  - Web Scraping with Beautiful Soup
- Data Wrangling / Exploratory Data Analysis:
  - Pandas
  - NumPy
  - SQLite
  - Seaborn
- Data Visual Analytics & Dashboards:
  - Folium
  - Plotly + Dash
- Machine Learning Predictive Analysis and Model Evaluation:
  - Preprocessing Pipeline: StandardScaler (numeric values), OneHotEncoder (categorical values), TestTrainSplit (20% test data, stratified output values).
  - ML Classification Models: Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbor
  - Cross-Validation Grid Search to optimize classification model parameters

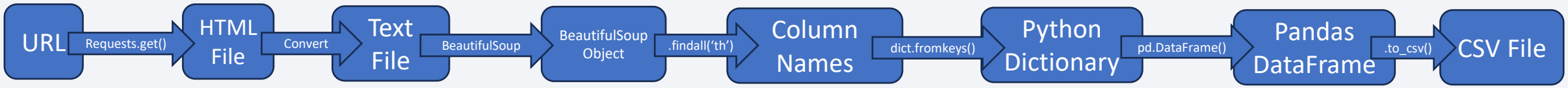
Data Collection

- The dataset was collected two different ways:
  - The SpaceX Data REST API: https://api.spacexdata.com/v4/launches/past
  - Web-Scraping: https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
- SpaceX REST API:
  - Returned data in .JSON format.
  - Converted to Pandas DataFrame, transformed data, and saved as a .CSV file.
- Web Scraping:
  - Returned data in .HTML format.
  - Converted to BeautifulSoup object, then organized into a Pandas DataFrame, transformed, and saved as a .CSV file.

Data Collection – SpaceX API

- API Call to URL: requests.get('url')
- Normalize JSON data to Pandas DataFrame: pd.json_normalize(response.json())
- Define Helper Functions to grab data from DF: List.append(response['column'])
- Execute Helper Functions: HelperFunction(data) → Lists
- Create Dictionary to associate desired column names with lists: {('column' : List), …}
- Create new DataFrame using dictionary: new_df = pd.DataFrame(dictionary)
- Filter DataFrame to include only Falcon 9 boosters: falcon9_df = new_df[new_df['BoosterVersion'] != 'Falcon 1']
- Calculate Mean PayloadMass, replace Null values with Mean: .describe['mean'], .fillna(mean)
- Save final DataFrame to CSV File: df.to_csv('file_name', index=False)
- Notebook GitHub URL: https://github.com/chrisdeitz/CourseraAssignments/blob/main/01_jupyter-labs-spacex-data-collection-api.ipynb

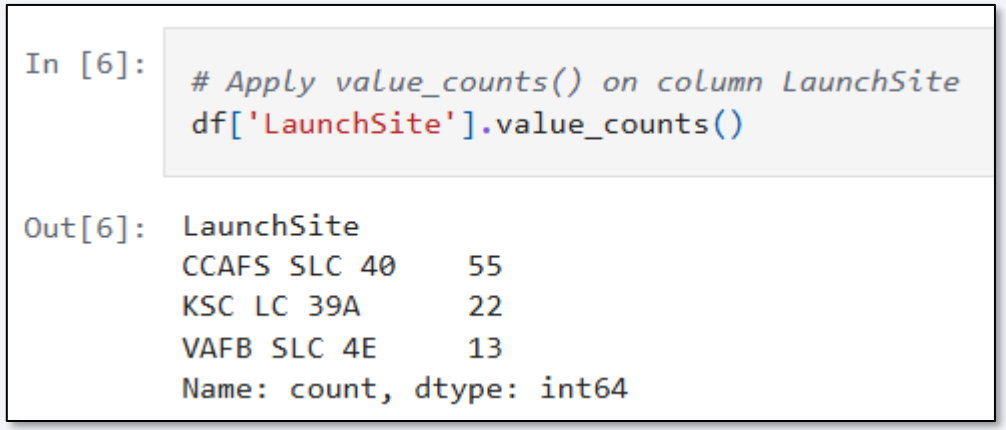
Data Collection – Web Scraping



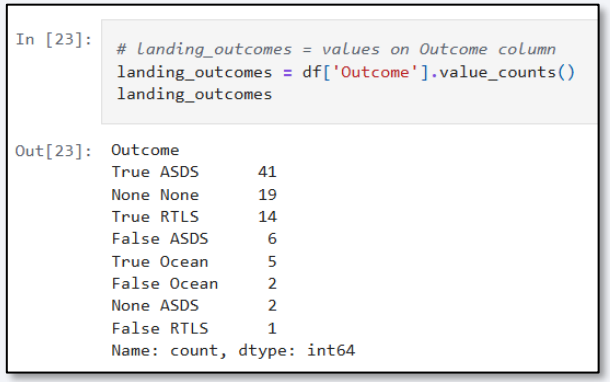
- An HTML GET request to the website URL downloads the raw HTML file: requests.get('url')
- Convert HTML file to plain text file: page_data.text
- Convert text file to a Beautiful Soup object, which parses (organizes) the HTML code: BeautifulSoup(data, 'html_parser')
- The Beautiful Soup library also helps find content by type (e.g. table headers): data.find_all('th')
- Using a FOR loop, we iterate through the table headers and create a Python dictionary associating the table headers with the desired column names for the Pandas DataFrame we will create: dict.fromkeys(column_names)
- Some user-defined Python functions to filter and clean data, combined with another FOR loop iterates through the HTML table content, populating Python Lists with data: dictionary['Column Name'].append(data)
- The Lists are then combined into a Pandas DataFrame: pd.DataFrame({key:pd.Series(value) for key, value in dictionary.items()})
- The DataFrame is then saved as a CSV file for analysis:
- GitHub URL: https://github.com/chrisdeitz/CourseraAssignments/blob/main/02_jupyter-labs-webscraping.ipynb

Data Wrangling

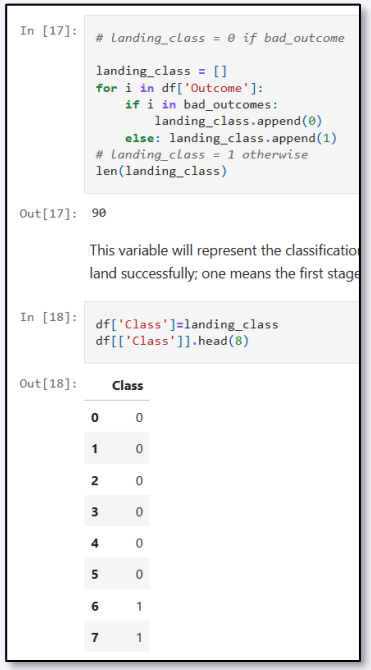
- The dataset was loaded to a Pandas DataFrame from a CSV file: pd.read_csv('url')
- Null values were identified: df.isnull().sum()
- Column data types were identified: df.dtypes
- The number of launches from each site, the number of launches per orbit type, and the number of each landing outcome type were calculated: df['Column_Name'].value_counts()
- The different types of outcomes are extracted...: bad_outcomes = set(landing_outcomes.keys()[[1,3,5,6,7]])
- … and, using a FOR loop with an if/else statement, replaced with a binary value classifying the outcome as either successful (1) or unsuccessful (0): if i in bad_outcomes: landing_class.append(0) // else: landing_class.append(1)
- The overall success rate can now be easily calculated: df['Class'].mean()
- The transformed DataFrame can now be saved again to a CSV file: df.to_csv('file_name', index=False)
- GitHub URL: https://github.com/chrisdeitz/CourseraAssignments/blob/main/03_labs-jupyter-spacex-Data%20wrangling.ipynb



```
In [6]:  # Apply value_counts() on column LaunchSite
         df['LaunchSite'].value_counts()

Out[6]:  LaunchSite
         CCAFS SLC 40    55
         KSC LC 39A      22
         VAFB SLC 4E     13
         Name: count, dtype: int64
```



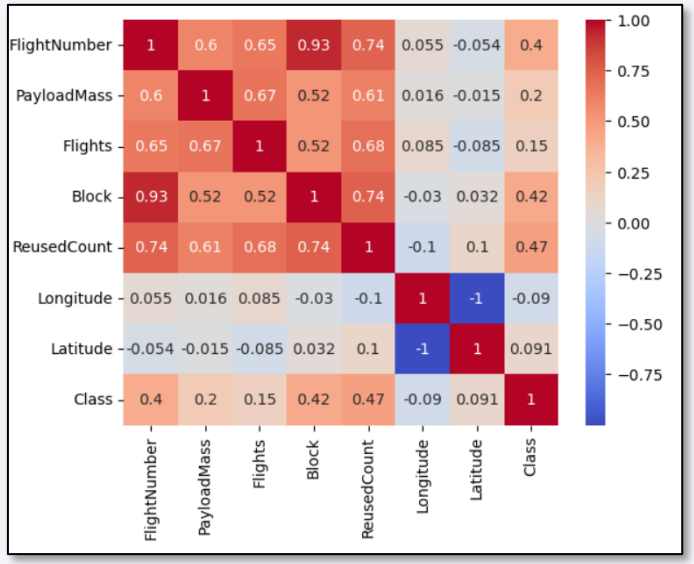
```
In [23]: # landing_outcomes = values on Outcome column
         landing_outcomes = df['Outcome'].value_counts()
         landing_outcomes

Out[23]: Outcome
         True ASDS       41
         None None       19
         True RTLS       14
         False ASDS       6
         True Ocean       5
         False Ocean      2
         None ASDS        2
         False RTLS       1
         Name: count, dtype: int64
```



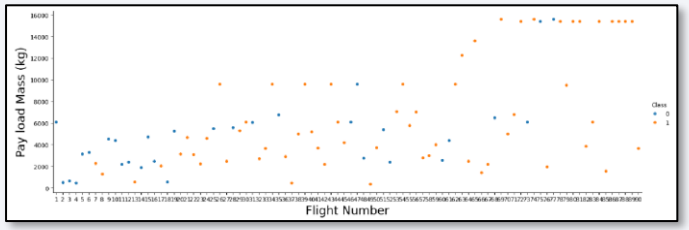
```
In [17]: # landing_class = 0 if bad_outcome

         landing_class = []
         for i in df['Outcome']:
             if i in bad_outcomes:
                 landing_class.append(0)
             else: landing_class.append(1)
         # landing_class = 1 otherwise
         len(landing_class)

Out[17]: 90

         This variable will represent the classificatio
         land successfully; one means the first stage

In [18]: df['Class']=landing_class
         df[['Class']].head(8)

Out[18]:
```

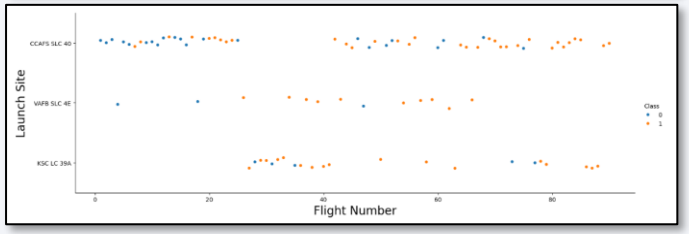
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

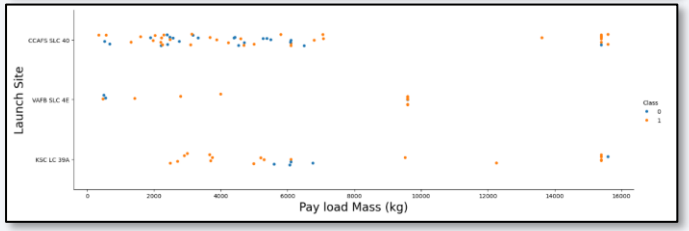
EDA with Data Visualization

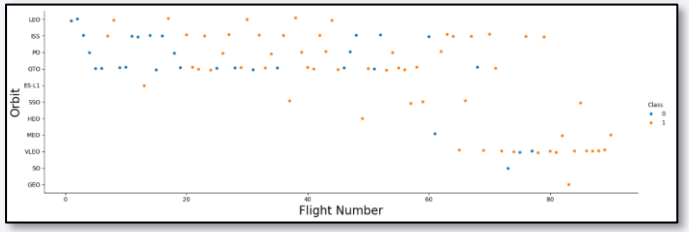


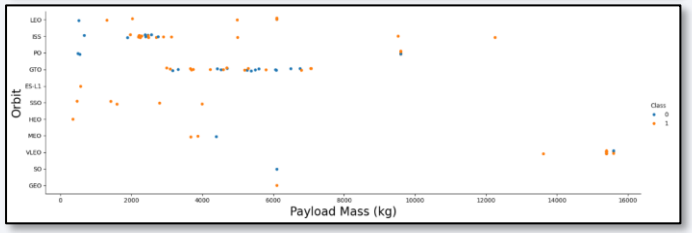
- **A Correlation Matrix** provided a quick look at which numeric variables correlate the most with "Class." **"ReusedCount"** had the highest correlation.
- Scatter Plots provided more detail of how well each variable correlated with "Class."
- A bar chart illustrates the average success rate of each Orbit type.
- A line chart illustrates the trend in average success rates over time.
- GitHub URL: https://github.com/chrisdeitz/CourseraAssignments/blob/main/05_edadataviz.ipynb

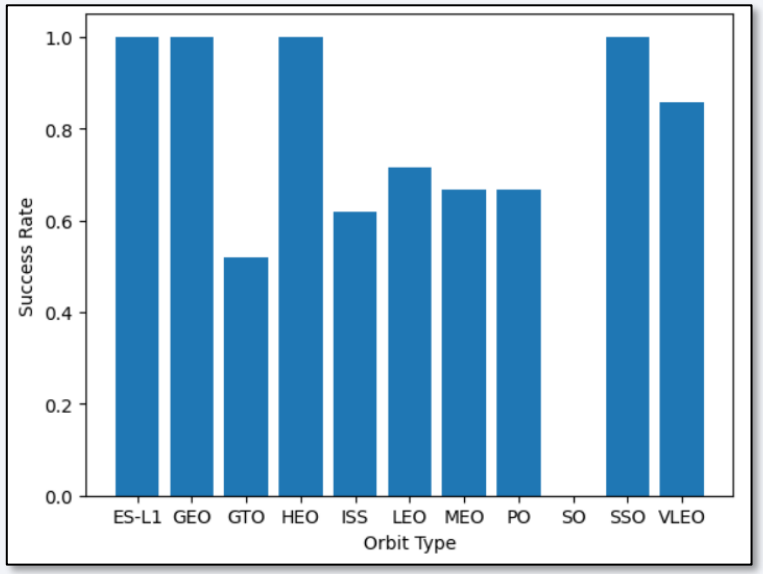


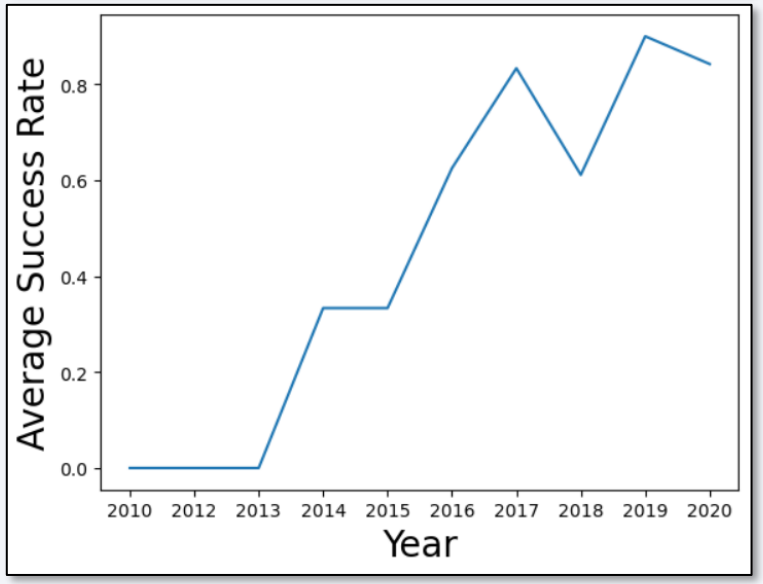












EDA with SQL

- SQLite was installed and used to query the data using Structured Query Language (SQL).
- A connector was created to store the dataset in a database: sqlite3.connect('database.db')
- The dataset was loaded from a CSV file into a Pandas DataFrame: pd.read_csv('url')
- The DataFrame was then converted to a database table and saved to the database: df.to_sql('TABLENAME', db_connector, …)
- Unique launch sites were displayed: SELECT DISTINCT Column_Name FROM TABLE;
- The first five records with a launch site beginning with 'CCA' were displayed: SELECT * FROM TABLENAME WHERE Column_Name LIKE 'STR%' LIMIT 5;
- The total payload mass was calculated: SELECT SUM(Column_Name) FROM TABLENAME;
- The average payload was was calculated too: SELECT AVG(Column_Name) FROM TABLENAME;
- The date of the first successful ground pad landing was identified: SELECT Date FROM TABLE WHERE Landing_Outcome = 'Success (ground pad)' ORDER BY Date LIMIT 1;

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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 (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

```
%sql SELECT DISTINCT Landing_Outcome FROM SPACEXTABLE;
```

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

Landing_Outcome
Failure (parachute)
No attempt
Uncontrolled (ocean)
Controlled (ocean)
Failure (drone ship)
Precluded (drone ship)
Success (ground pad)
Success (drone ship)
Success
Failure
No attempt

```
%%sql
SELECT SUBSTR(Date,6,2), SUBSTR(DATE,0,5), Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTABLE
WHERE Date LIKE '%2015%';
```

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

SUBSTR(Date,6,2)	SUBSTR(DATE,0,5)	Landing_Outcome	Booster_Version	Launch_Site
01	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
02	2015	Controlled (ocean)	F9 v1.1 B1013	CCAFS LC-40
03	2015	No attempt	F9 v1.1 B1014	CCAFS LC-40
04	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
04	2015	No attempt	F9 v1.1 B1016	CCAFS LC-40
06	2015	Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40
12	2015	Success (ground pad)	F9 FT B1019	CCAFS LC-40

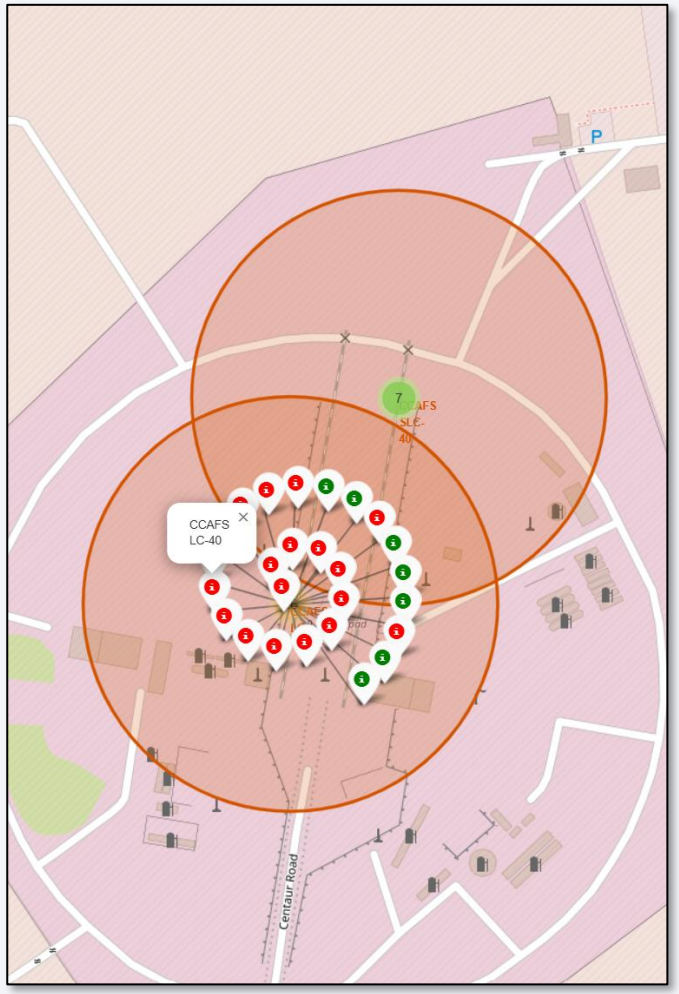
EDA with SQL (continued)

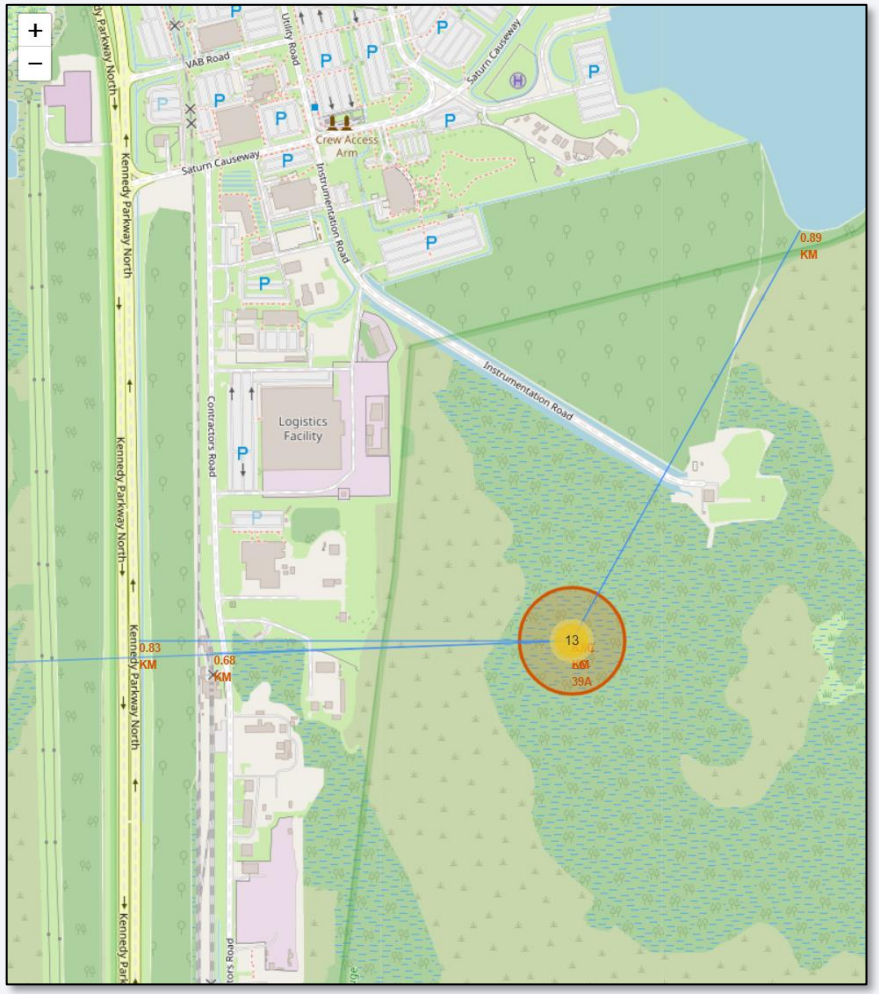
- The names of boosters with successful landings on a drone ship, with payload mass between 4,000 and 6,000kg was identified: SELECT DISTINCT Booster_Version, PAYLOAD_MASS__KG_, Landing_Outcome FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND Landing_Outcome = 'Success (drone ship)';
- The number of successful and failed mission outcomes was calculated: SELECT COUNT(Mission_Outcome) FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Failure%'; // '%Success%';
- All booster versions which carried the maximum payload mass were listed using a suq-query: SELECT Booster_Verions FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
- The Month, Booster Versions, and Launch Sites were displayed for all failed landings on drone ships in 2015: SELECT SUBSTR(Date, 6, 2), SUBSTR(DATE, 0, 5), Landing_Outcome, Booster_Verision, Launch_Site FROM SPACEXTABLE WHERE Date LIKE '%2015%';
- Landing outcomes were ranked based on count, highest to lowest, between June 4th, 2010 and March 20th, 2017: SELECT Landing_Outcome, COUNT(*) FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY COUNT(*) DESC;
- GitHub URL: https://github.com/chrisdeitz/CourseraAssignments/blob/main/04_jupyter-labs-eda-sql-coursera_sqllite.ipynb

Building an Interactive Map with Folium

Map Objects:

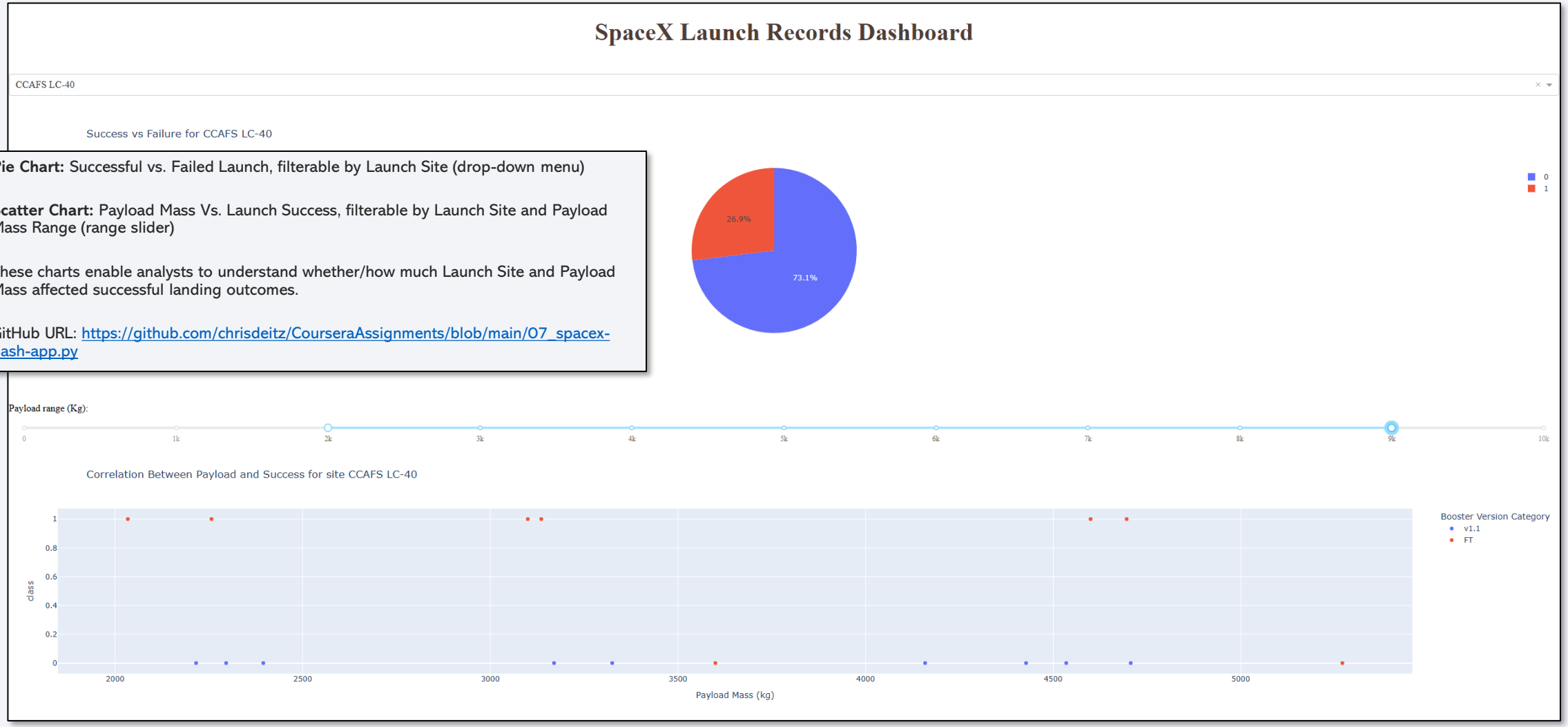
- Created a red circle with a label around each site, to make it easy to find and identify the sites on the map.
- Marker cluster with green and red markers denote successful and unsuccessful recoveries for rockets launched from each site.
- Lines with labels depict the distance from a launch site to the nearest railroad, highway, body of water, and city, illustrating what type of infrastructure launch sites need to be close or far away from.
- GitHub URL: https://github.com/chrisdeitz/CourseraAssignments/blob/main/06_lab_jupyter_launch_site_location.ipynb





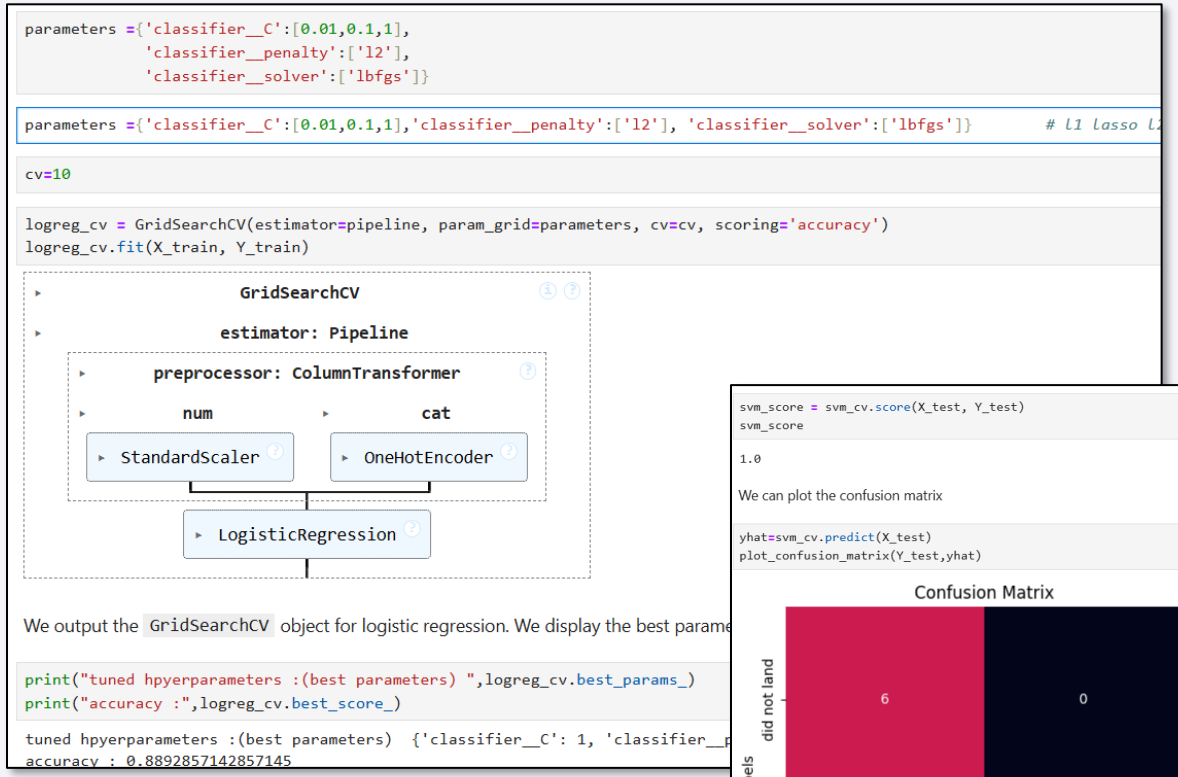
Building a Dashboard with Plotly Dash

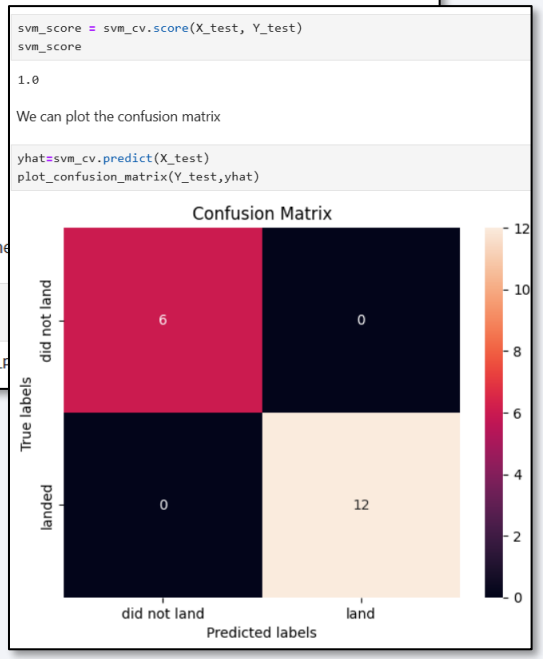
- **Pie Chart:** Successful vs. Failed Launch, filterable by Launch Site (drop-down menu)
- **Scatter Chart:** Payload Mass Vs. Launch Success, filterable by Launch Site and Payload Mass Range (range slider)
- These charts enable analysts to understand whether/how much Launch Site and Payload Mass affected successful landing outcomes.
- GitHub URL: https://github.com/chrisdeitz/CourseraAssignments/blob/main/07_spacex-dash-app.py



Predictive Analysis (ML Classification)

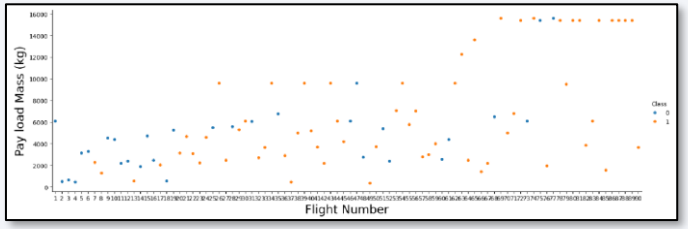
- Four different ML classification models were trained, tested, and evaluated to determine the most accurate model to predict landing outcomes.
- The dataset was split into training and testing sets. The test set comprised 20% of the data. The landing outcome data was stratified, to ensure adequate results of each type in both the training and test data.
- StandardScaler eliminated bias caused by numerical features with different scales.
- OneHotEncoder converted categorical features to numeric values, enabling the ML models to interpret them.
- Supervised ML Models trained, tested, and evaluated:
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree
  - K-Nearest Neighbor
- Grid Search Cross-Validation identified model parameters resulting in the highest predictive accuracy.
- The .score() function and Confusion Matrices evaluated the accuracy of each model.
- SVM resulted in the highest accuracy. Using the test data, this model predicted 100% accurate landing outcomes.
- GitHub URL: https://github.com/chrisdeitz/CourseraAssignments/blob/main/08_SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

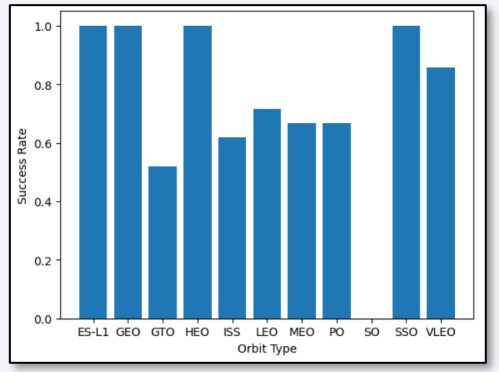


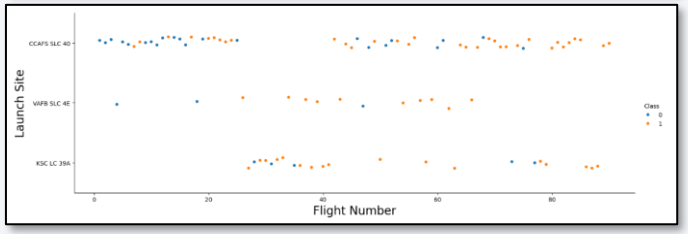


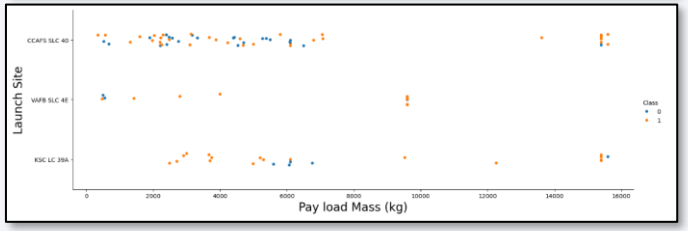
Results

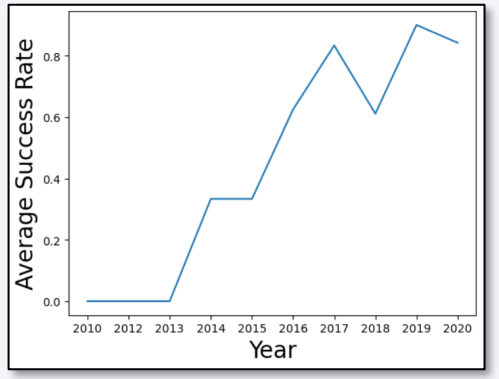
- Exploratory data analysis: The date of launch highly correlates with successful landing outcomes.
- Predictive analysis: The SVM Machine Learning model predicted landing outcomes with 100% accuracy on test data.

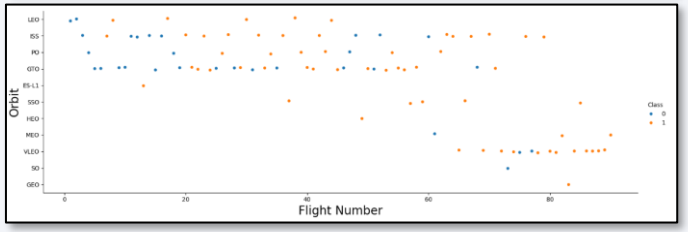


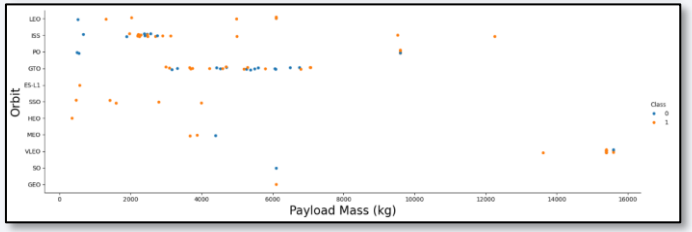


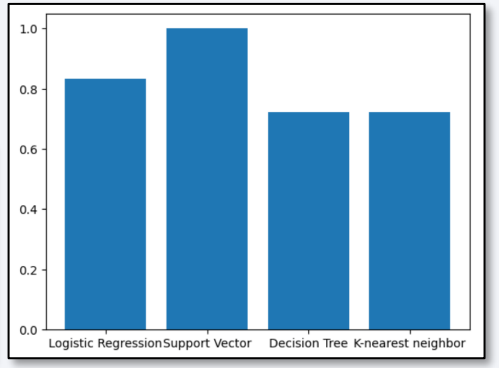


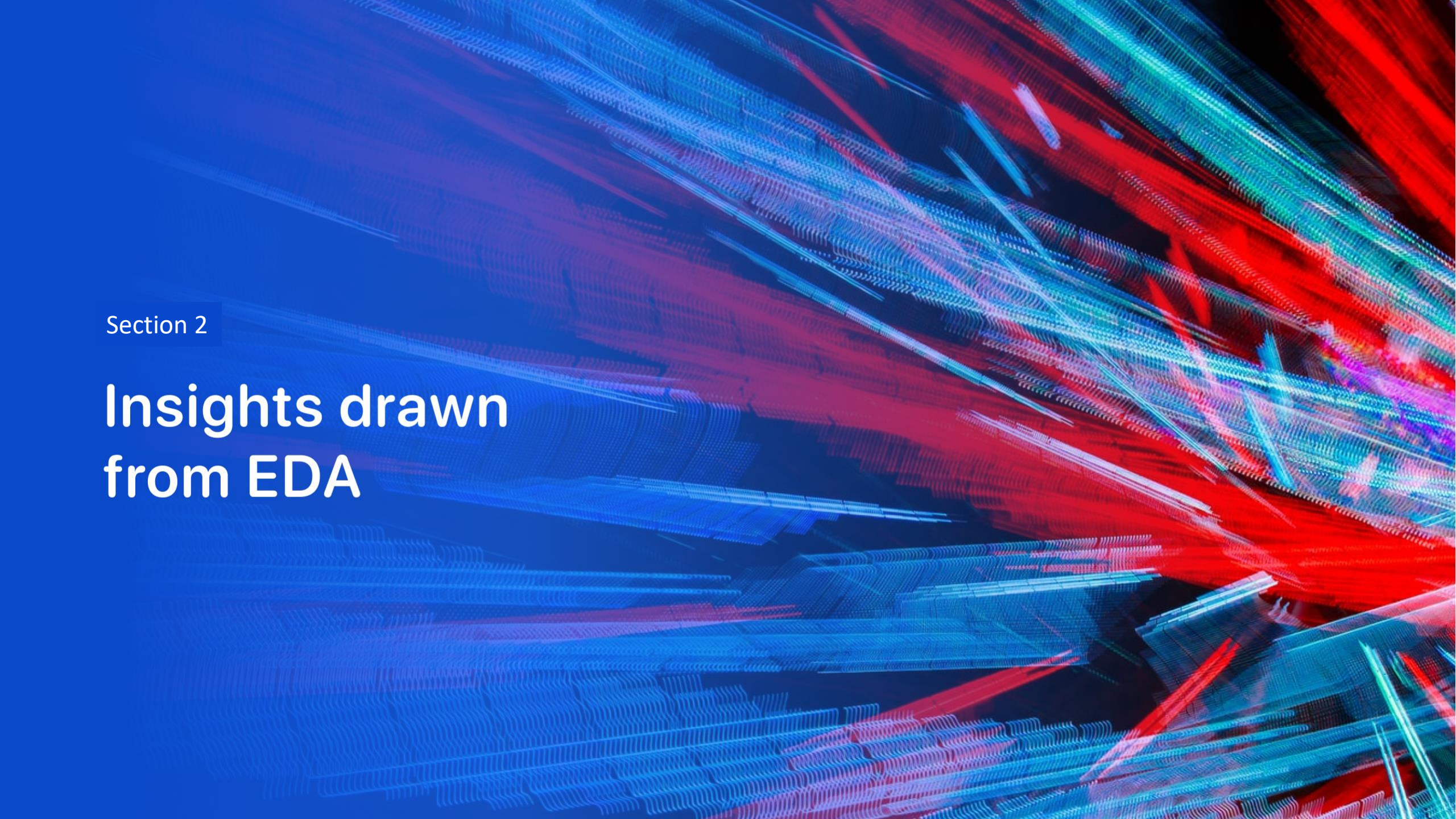




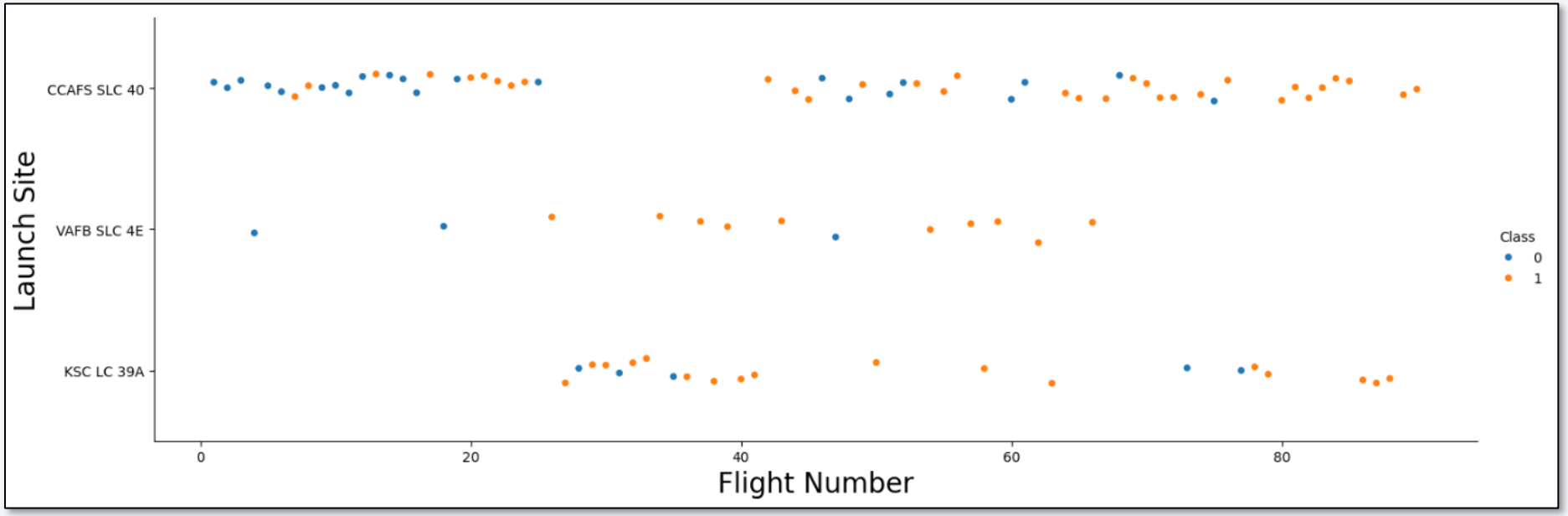






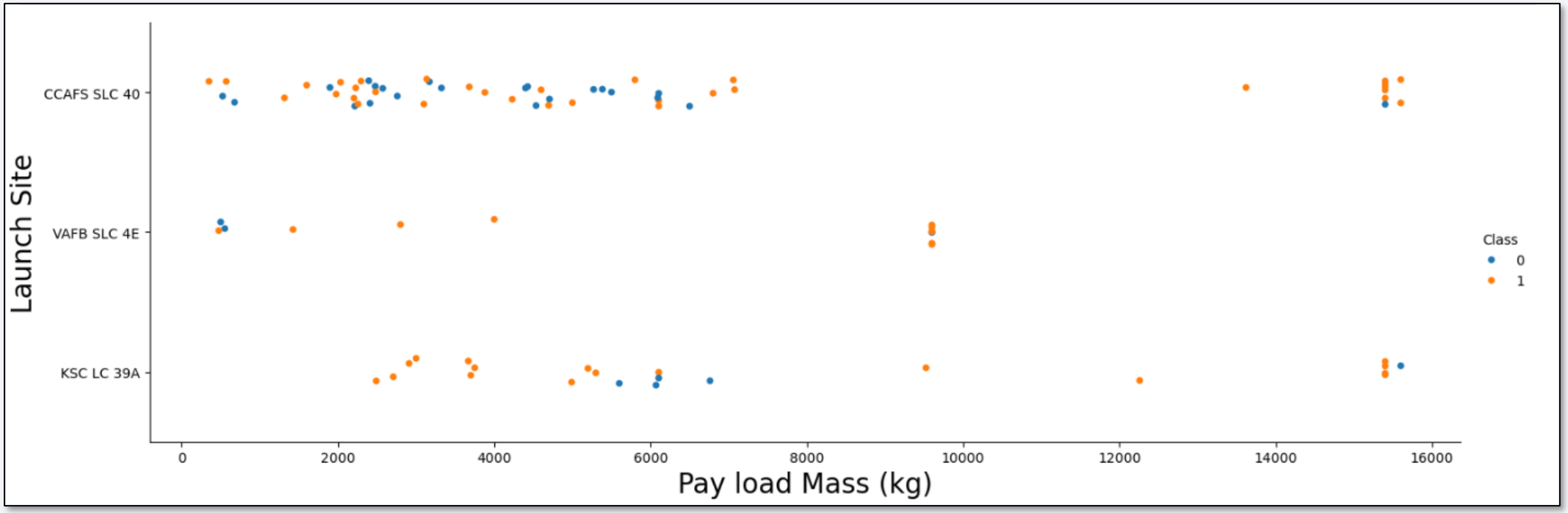

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



- This scatter plot shows whether flight number relates to launch site, and whether landing outcomes are associated with either.
- It appears that landing outcomes improved as the number of flights increased.

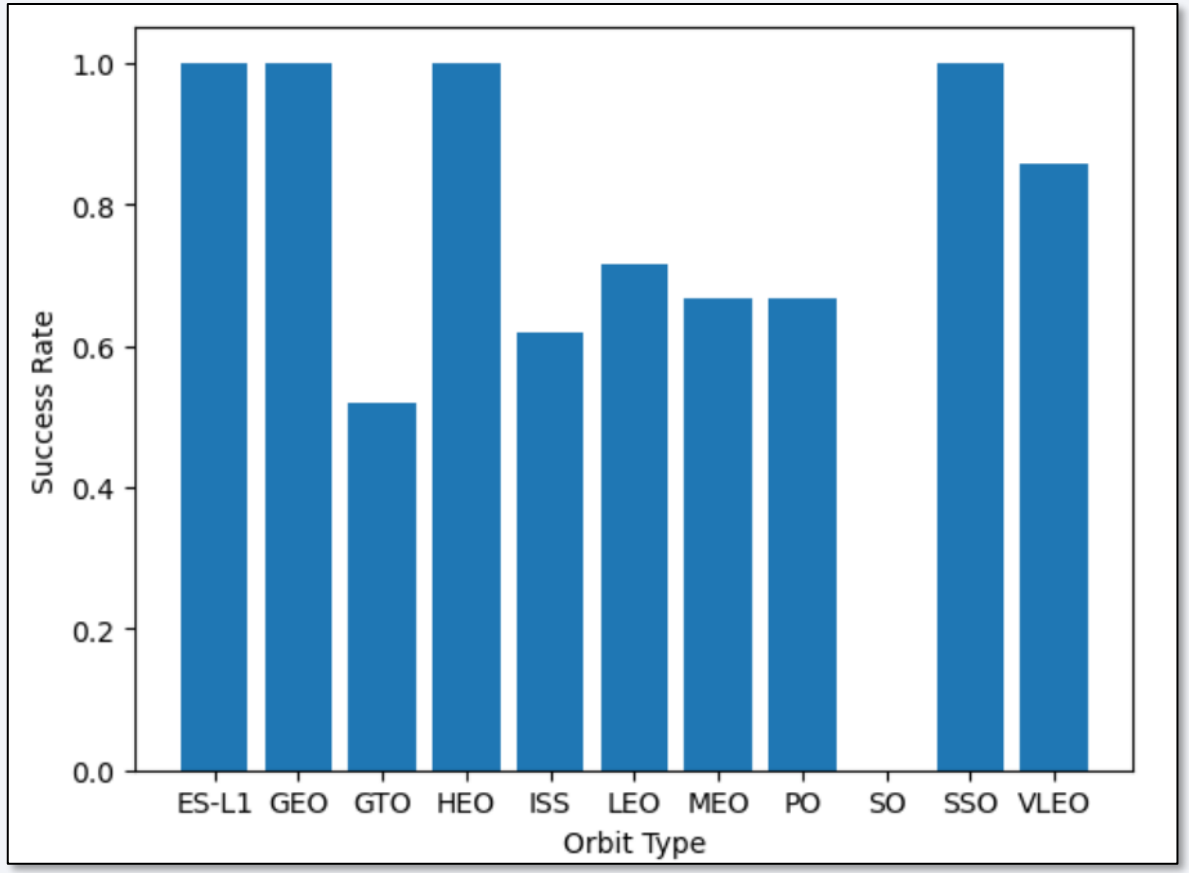
Payload vs. Launch Site



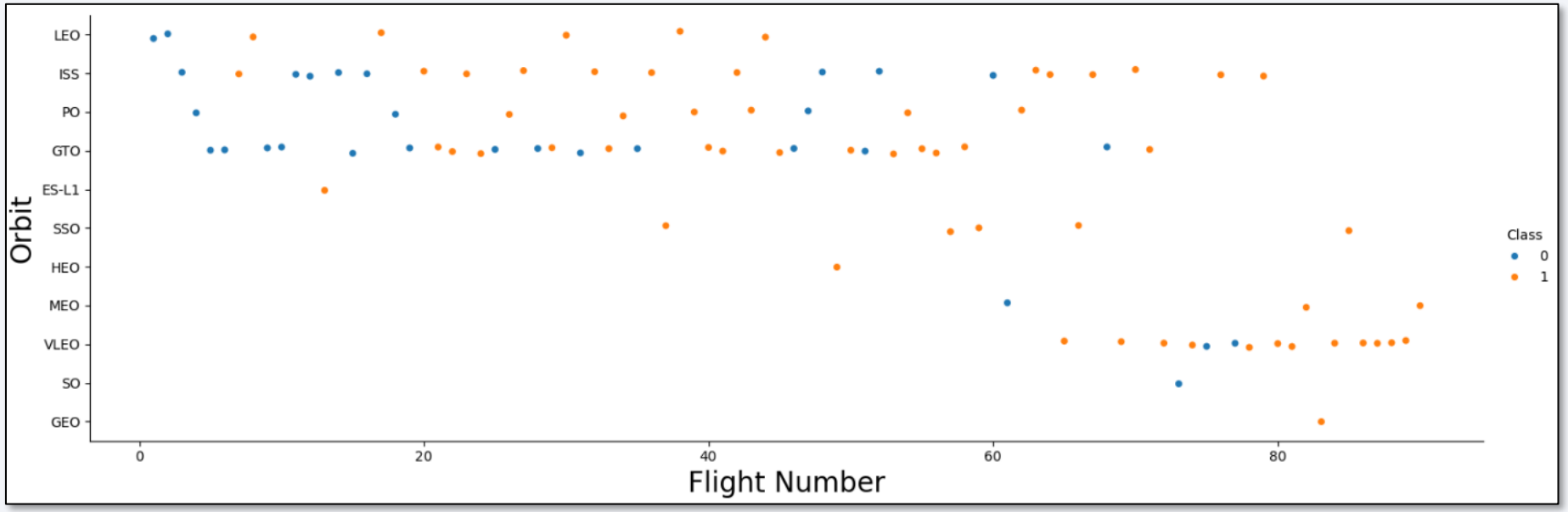
- This scatter chart explores whether Payload Mass is related to Launch Sites, and whether either of these affects landing outcomes.
- For the VAFB site, no rockets with a payload greater than 10,000kg were launched from that location.
- There does not appear to be a correlation between landing outcomes and either Launch Site or Payload Mass.

Success Rate vs. Orbit Type

- This bar chart shows the average landing success rate of rockets, broken out by payload orbit type.
- SO (Synchronous Orbit) appears to have an overall average success rate of zero, but this is misleading – SO is synonymous with SSO (sun-synchronous orbit), which has a 100% success rate.
- Further data wrangling should combine both categories to improve the accuracy of these two features.

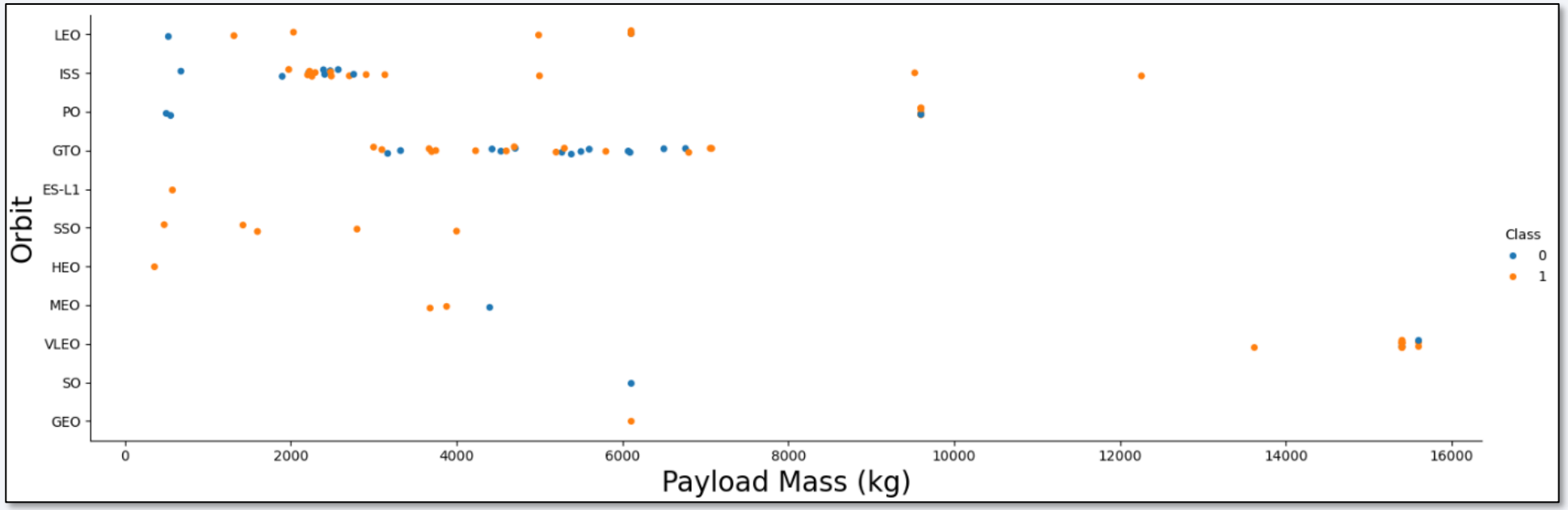


Flight Number vs. Orbit Type



- This scatter chart depicts rocket Flight Numbers vs. Orbit Types, and shows the Landing Outcome for each data point in orange (successful landings) and blue (unsuccessful landings).
- From this chart, you can see that certain Payload Orbit types have higher Flight Numbers, indicating a later launch date.
- For these Orbit types with higher Flight Numbers, there appears to be a correlation between Flight Number and Landing Outcome (class) success.

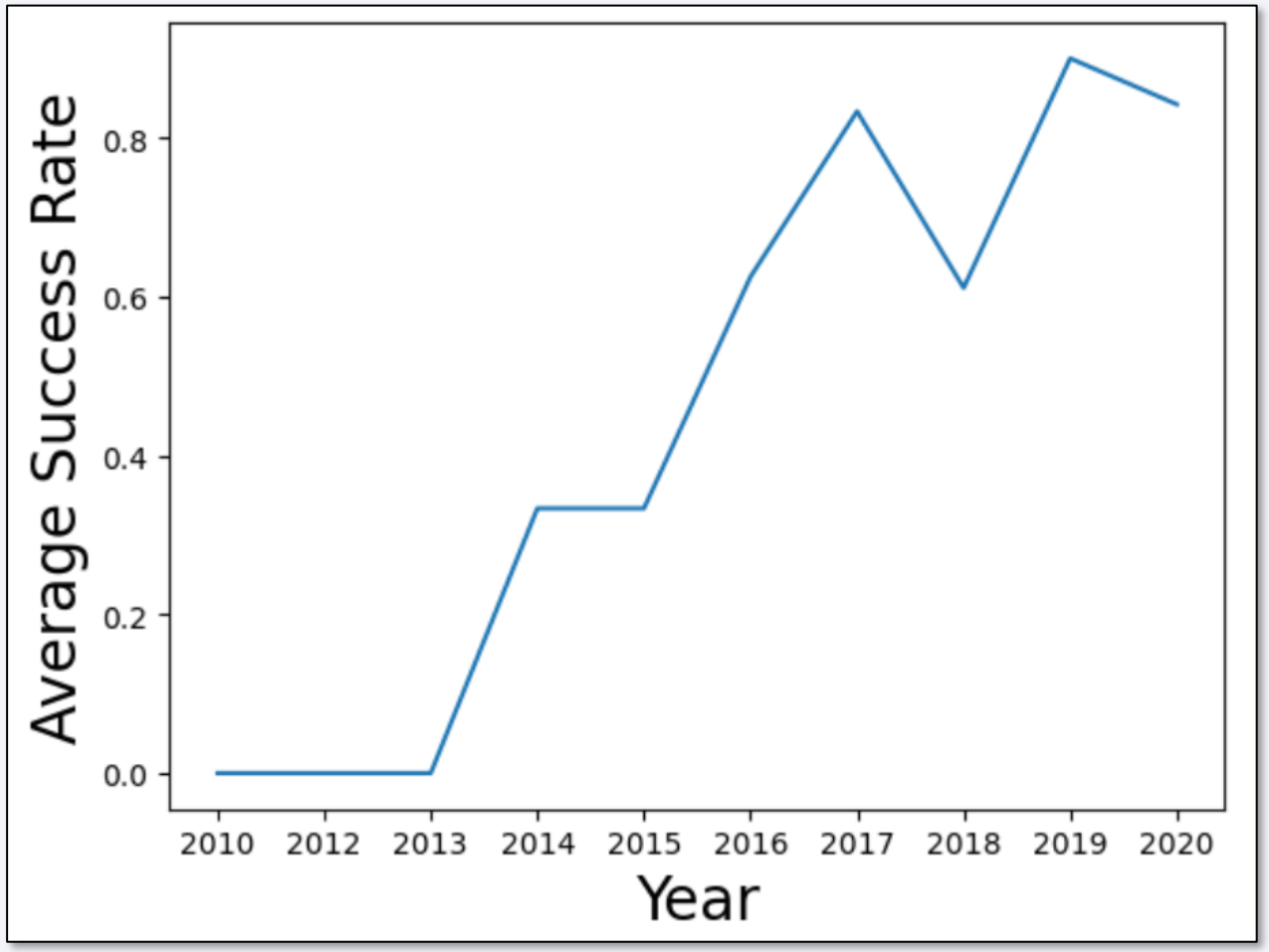
Payload vs. Orbit Type



- This scatter chart depicts Orbit Type vs. Payload Mass (in kilograms), with Landing Outcome depicted in orange (successful landing) and blue (unsuccessful landing).
- For the same group of Orbit types as the last chart (ES-L1 and below), there appears to be a higher rate of successful landings.

Launch Success Yearly Trend

- This line chart depicts a clear positive correlation between Launch Date and Landing Success Rate.
- This makes sense based on the analysis in the scatter charts – Orbit type and Flight Number have a high correlation with Launch Date, which is why they correlated with successful landing outcomes.



All Launch Site Names

Using the DISTINCT keyword in SQLite, we can easily create a list of all Launch Sites from the dataset.

```
In [32]:  %%sql
          SELECT DISTINCT Launch_Site FROM SPACEXTABLE;

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

Out[32]:

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

Launch Site Names Begin with 'CCA'

- Using the WHERE and LIKE keywords, we can filter the data to rows where the Launch Site starts with the letters ‘CCA.’
- Using the LIMIT keyword, the table displays only the first five rows of the query results.

In [41]:
```
%%sql
SELECT * FROM SPACEXTABLE
WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[41]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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 (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

```
In [25]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE;
 * sqlite:///my_data1.db
Done.
Out[25]: SUM(PAYLOAD_MASS__KG_)
         619967
```

- Using the SUM keyword in SQLite, we can calculate the total payload mass carried for all launches.

Average Payload Mass by F9 v1.1

```
In [27]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE;
 * sqlite:///my_data1.db
Done.
Out[27]: AVG(PAYLOAD_MASS__KG_)
         6138.287128712871
```

- Using the AVG keyword in SQLite, we can the average payload mass from all launches.

First Successful Ground Landing Date

```
In [40]: %%sql
         SELECT Date FROM SPACEXTABLE
         WHERE Landing_Outcome = 'Success (ground pad)'
         ORDER BY Date LIMIT 1;

          * sqlite:///my_data1.db
         Done.
Out[40]:      Date
         ----------
         2015-12-22
```

- Using the WHERE keyword, we can filter the data to show only successful landings on a ground pad.
- Using the ORDER BY and LIMIT keywords, we can find the date of the first successful landing on a ground pad.

Successful Drone Ship Landing with Payload between 4000 and 6000

In [48]:
```
%%sql
SELECT DISTINCT Booster_Version, PAYLOAD_MASS__KG_, Landing_Outcome FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
    AND Landing_Outcome = 'Success (drone ship)';
```

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

Out[48]:

Booster_Version	PAYLOAD_MASS__KG_	Landing_Outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

- Using the BETWEEN keyword, we can create a list of the names of boosters which had a payload mass greater than 4000 but less than 6000.
- Using the AND logic operator, we can add an additional filter, returning only results for rockets that have successfully landed on a drone ship.

Total Number of Successful and Failure Mission Outcomes

- Using the COUNT keyword and WHERE / LIKE filter, we can calculate the total number of successful and failed mission outcomes.
- Mission outcomes are different than landing outcomes. A mission can succeed in launching its payload into orbit but then fail to successfully recover the first-stage rocket.

In [58]:
```
%%sql
SELECT COUNT(Mission_Outcome) FROM SPACEXTABLE
WHERE Mission_Outcome LIKE '%Success%';
```

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

Out[58]:

COUNT(Mission_Outcome)
100

In [60]:
```
%%sql
SELECT COUNT(Mission_Outcome) FROM SPACEXTABLE
WHERE Mission_Outcome LIKE '%Failure%';
```

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

Out[60]:

COUNT(Mission_Outcome)
1

Boosters Carried Maximum Payload

- Using a subquery, we can find the maximum payload mass, and then list the names of the boosters which have carried a payload mass in that amount.

In [62]:
```
%%sql
SELECT Booster_Version FROM SPACEXTABLE
    WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

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

Out[62]:

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

- This query filters the data, returning Landing Outcomes, Booster Versions, and Launch Sites only from launches in 2015.

In [11]:
```
%%sql
SELECT SUBSTR(Date,6,2), SUBSTR(DATE,0,5), Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTABLE
WHERE Date LIKE '%2015%';
```

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

Out[11]:

SUBSTR(Date,6,2)	SUBSTR(DATE,0,5)	Landing_Outcome	Booster_Version	Launch_Site
01	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
02	2015	Controlled (ocean)	F9 v1.1 B1013	CCAFS LC-40
03	2015	No attempt	F9 v1.1 B1014	CCAFS LC-40
04	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
04	2015	No attempt	F9 v1.1 B1016	CCAFS LC-40
06	2015	Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40
12	2015	Success (ground pad)	F9 FT B1019	CCAFS LC-40

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

- This query ranks the number of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the dates 2010-06-04 and 2017-03-20, in descending order.

In [22]:

```
%%sql
SELECT Landing_Outcome, COUNT(*) FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY COUNT(*) DESC;
```

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

Out[22]:

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

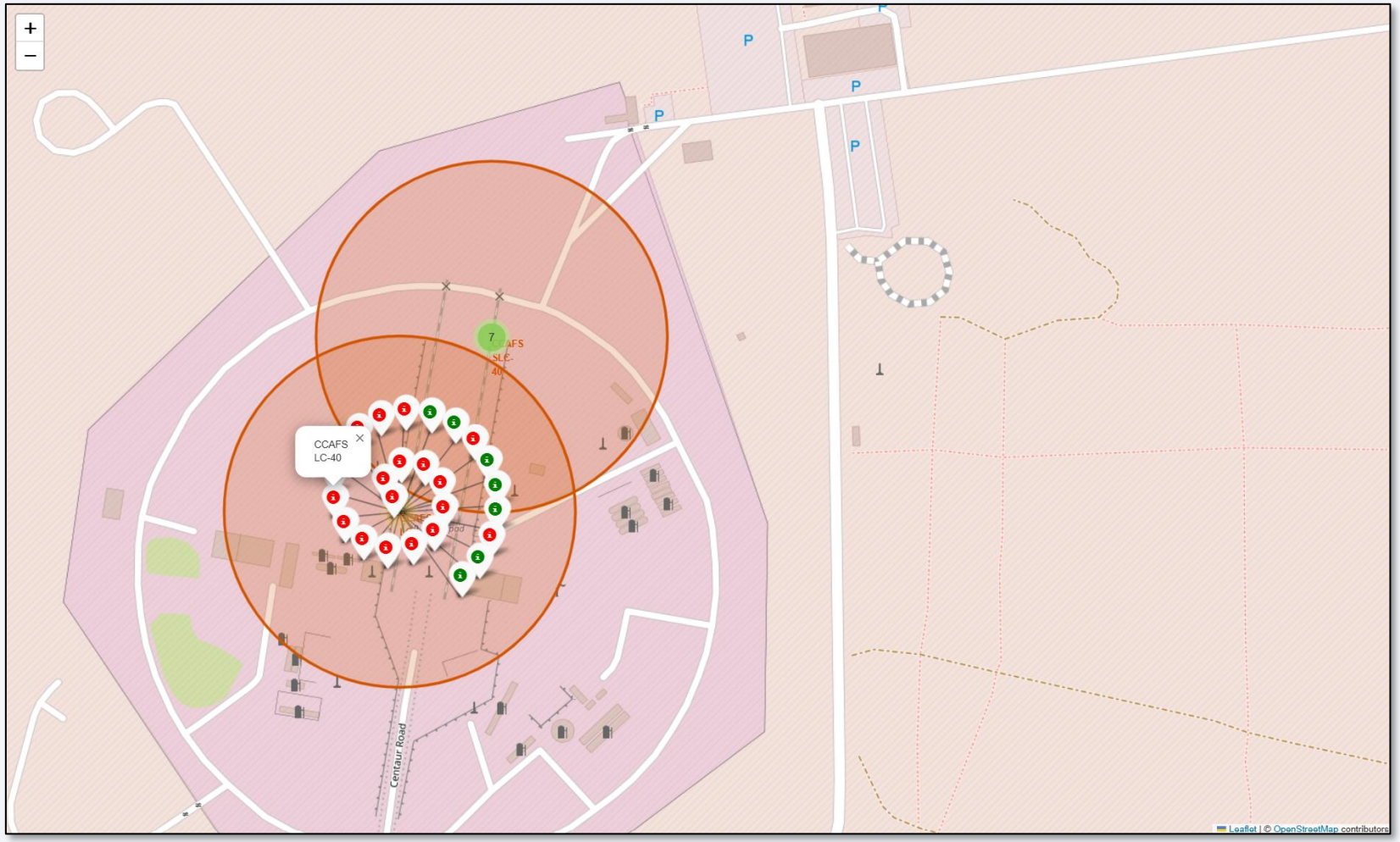
Launch Sites Proximities Analysis

Launch Sites Map

- This map depicts all launch sites with red circles and labels.
- You can see that one launch site exists in Southern California, and several appear to be in close proximity in Florida.
- It is interesting to see that all launch sites are in coastal locations.

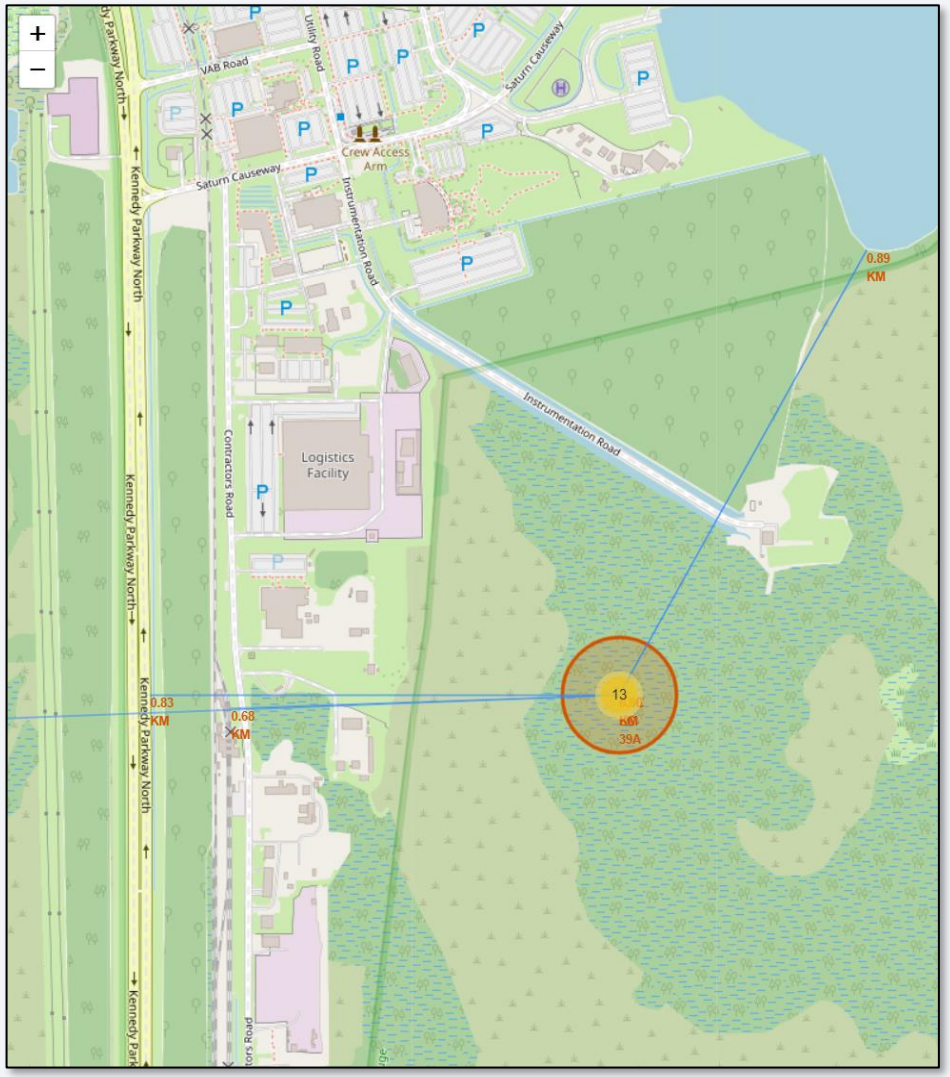
Successful Landing Outcomes by Site

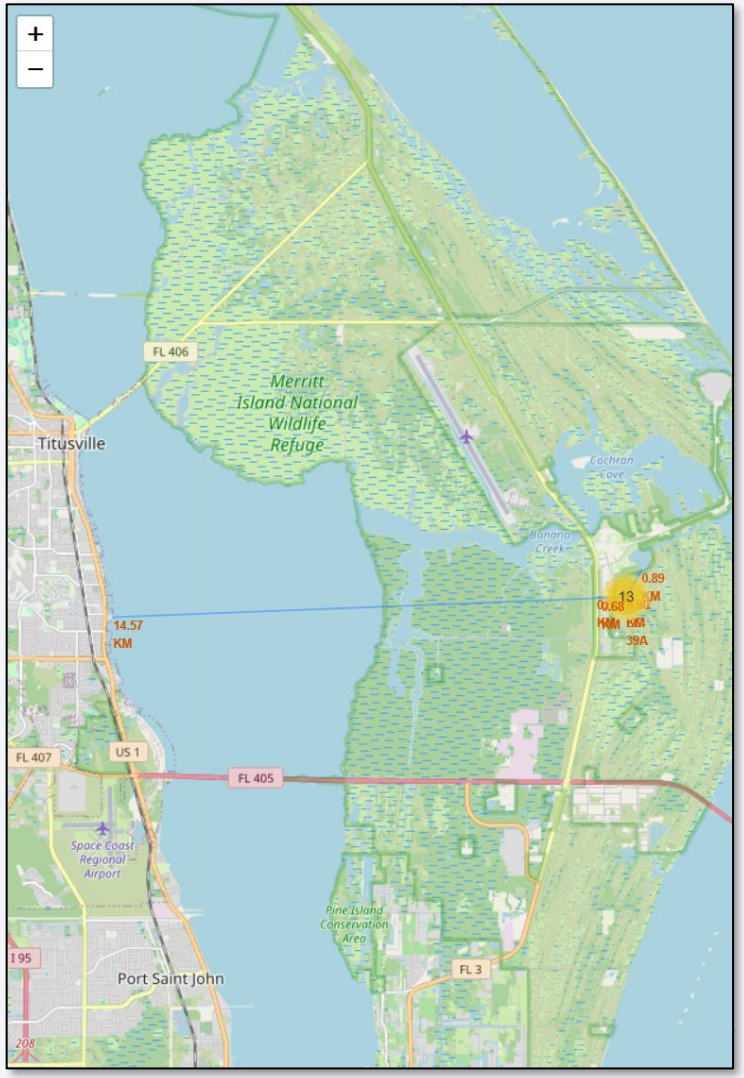
- This map visualization shows a cluster of icons with red markers denoting unsuccessful Landing Outcomes and green markers denoting successful Landing Outcomes for each site.

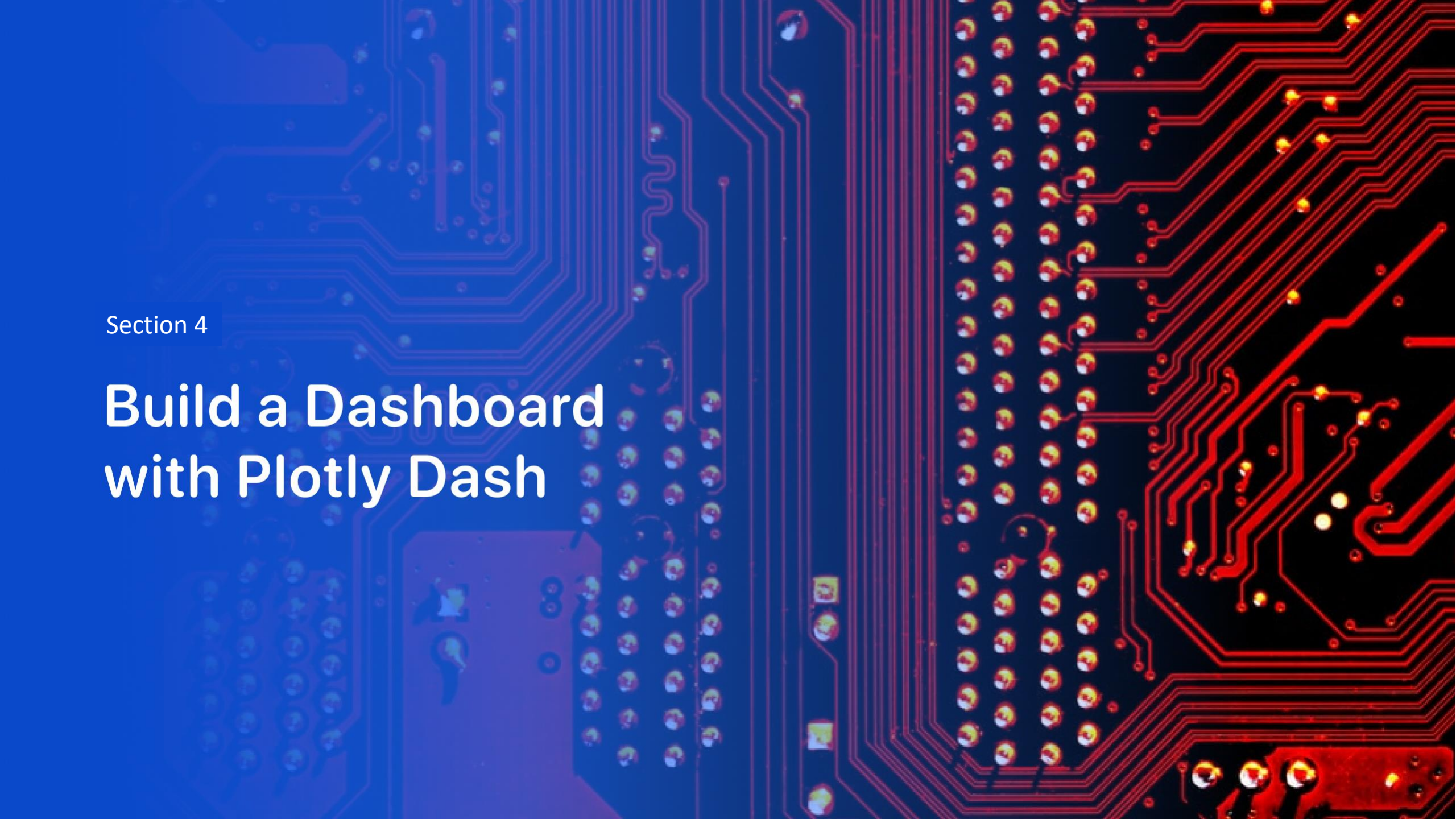


Launch Site Proximity to Geographic Features

- This map visualization depicts lines showing the distance to geographic features:
  - Nearest coastline
  - Nearest railroad
  - Nearest highway
  - Nearest city
- It appears that launch sites must be in close proximity (less than 1km) to bodies of water, railroads, and highways. This is likely to facilitate transportation of heavy spacecraft components.
- It appears that launch sites must be some distance away from populated areas. This is likely to mitigate risk to people in the event of a rocket explosion / crash during launch.

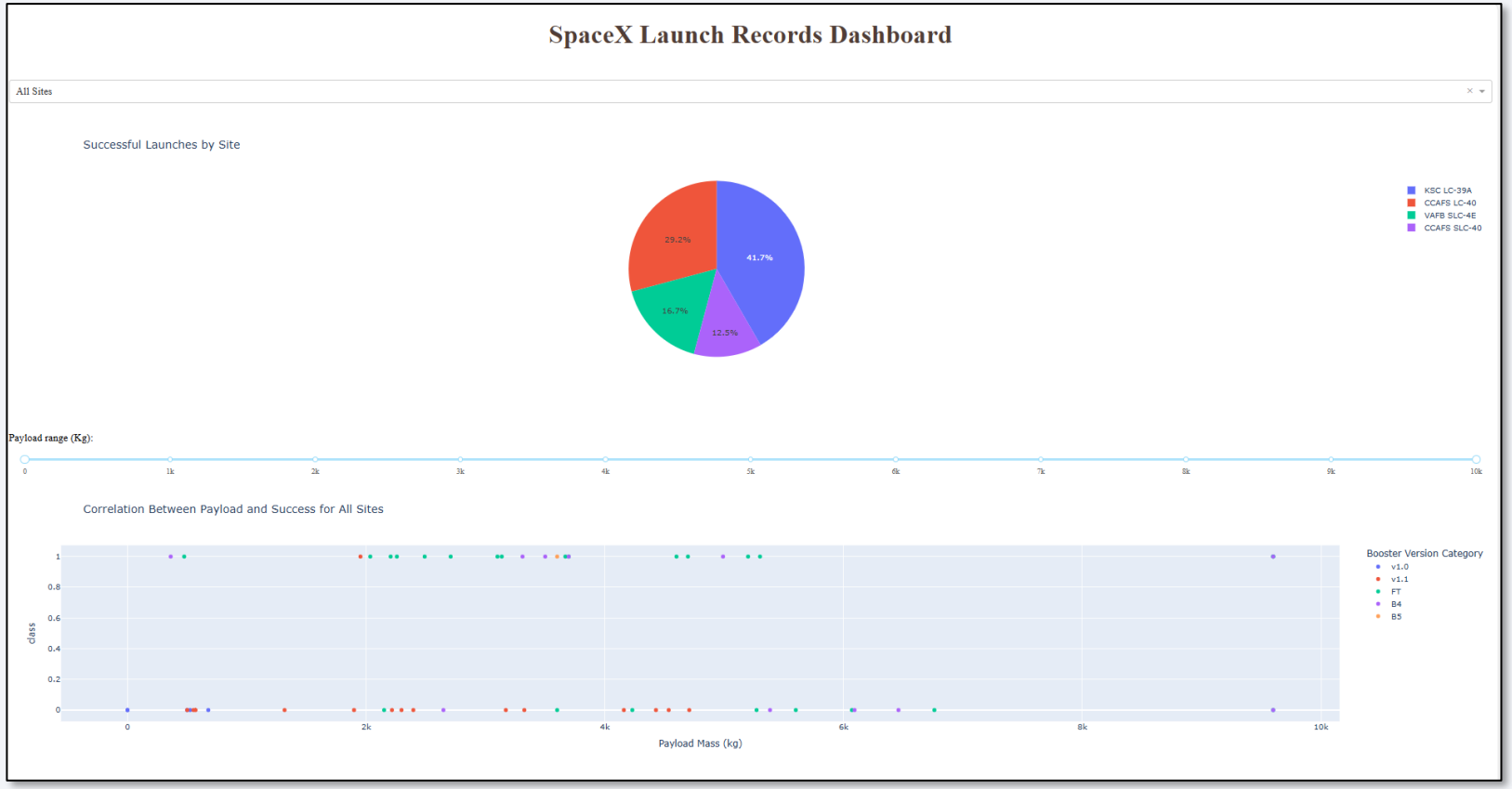





Section 4
Build a Dashboard with Plotly Dash

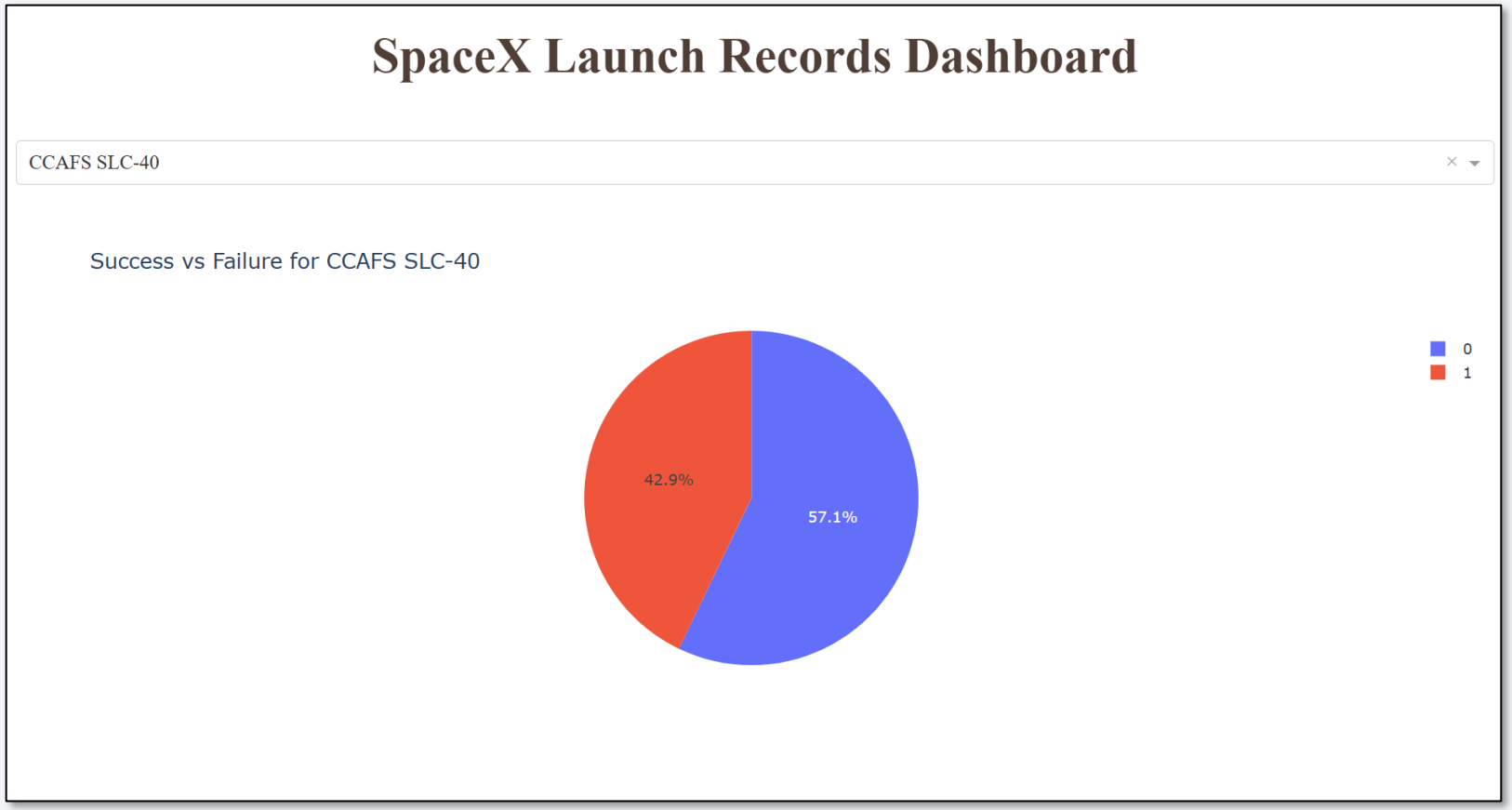
SpaceX Launch Records Interactive Dashboard

- This pie chart depicts the number of successful landings, grouped by launch sites.
- This chart shows that the KSC LC-39A launch site had **the highest number** of rockets with successful first stage landings.
- This data could be misleading – it appears like the likelihood of a successful landing may be highest for rockets launched from the KSC LC-39A launch site…



SpaceX Launch Records Dashboard – Site Filter

However, when we filter the pie chart to show the proportion of successful vs. unsuccessful launches at each site, we see that rockets launched from the CCAFS SLC-40 launch site actually had the highest proportion of successful first stage recoveries.



SpaceX Launch Records Dashboard – Payload Mass Filter

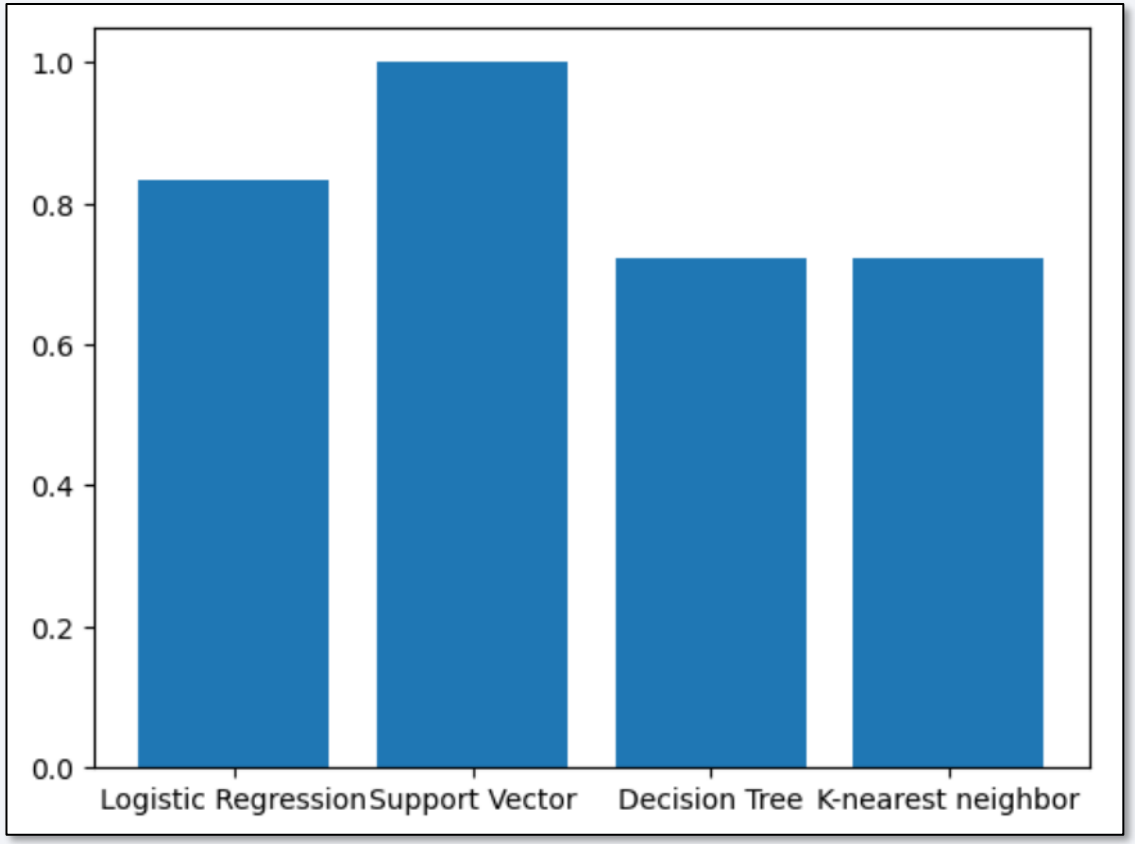
- This scatter chart depicting Payload Mass vs. Landing Outcomes is also filterable by Launch Site.
- This chart depicts data from ALL Launch Sites.
- The chart is also filterable by Payload Range, using a range slider. This enables analysts to change the minimum and maximum Payload Mass displayed in the chart, in 1,000kg increments.
- From this chart, you can see that:
  - The FT and B4 Booster Versions have relatively high landing success rates for this Payload Mass range..
  - The v1.1 has a very low success rate for this Payload Mass range.
  - There is only one B5 data point for this Payload Mass range, and its landing was successful.

Section 5

Predictive Analysis (Classification)

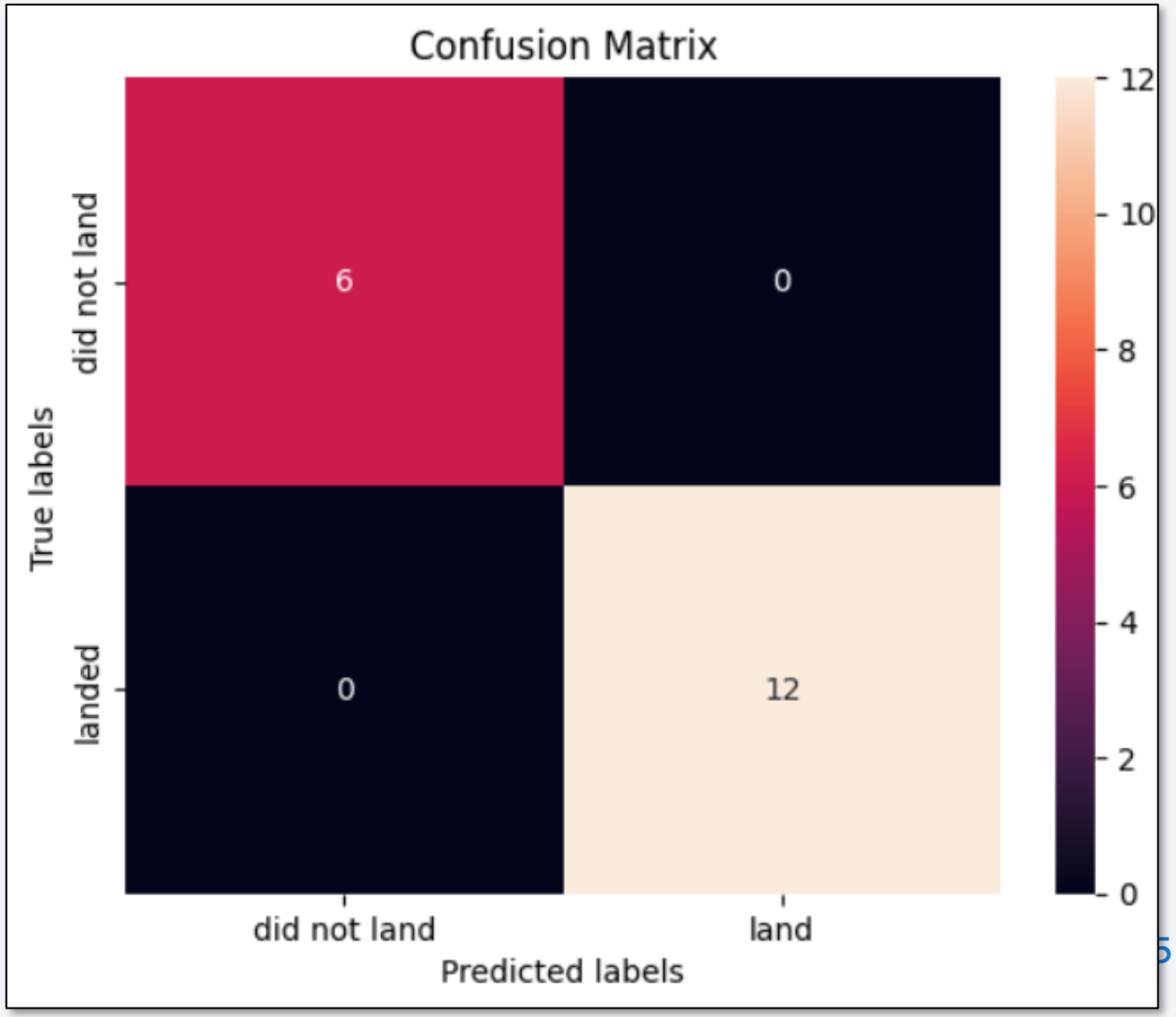
Classification Accuracy

- The Support Vector Machine (SVM) model scored the highest accuracy, correctly predicting 100% of the landing outcomes in the test data. However, the score using the best parameters was only 94%. This may be due to overfitting.
- The Logistic Regression model scored the next best, with an accuracy score of 83%. Tuning the model with the best parameters boosted the LR score to 89%.
- Decision Tree and K-Nearest Neighbor (KNN) models scored 72%. However, tuning them to their best parameters boosted their scores to 95% and 91%, respectively.



Confusion Matrix

- The SVM model correctly predicted ALL landing outcomes when evaluated with the test data.
- The model accurately predicted all successful landings in the test data.
- The model accurately predicted all unsuccessful landings in the test data.



Conclusions

- ML models predicted landing outcomes with remarkable accuracy.
- Recommended ML Model: SVM
- The Decision Tree model could also be nearly as accurate as SVM if tuned to its best parameters. Recommend evaluating the importance of model accuracy vs. compute costs for each model.
- Further exploration into the discrepancy between the SVM model's overall score (100%) and 'best' score (94%) is needed.

Appendix

- GitHub folder for all project code: https://github.com/chrisdeitz/CourseraAssignments/tree/main
- Coding Resources:
  - W3 Schools Python Tutorial: https://www.w3schools.com/python/default.asp
  - Python Developer's Guide: https://devguide.python.org/
  - W3 Schools SQL Tutorial: https://www.w3schools.com/sql/default.asp
  - Pandas User Guide: https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html
  - SciKitLearn User Guide: https://scikit-learn.org/stable/user_guide.html
  - Dash Python User Guide: https://dash.plotly.com/

Thank you!