Winning Space Race with Data Science

Jin “Max” Li

Oct 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
  - Prediction Analytics with Machine Learning
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  The Space X data revealed that the launch of its Falcon 9 rocket cost 62 million dollars, while other rocket providers had cost up to 165 million dollars. The primary cause of the gap in cost is that Space X can reuse the first-stage rocket. We can determine the price of a launch, given whether the first-stage rocket will land. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. This project predicts whether the first-stage rocket will land successfully with a machine learning pipeline.

- Problems you want to find answers

  - Factors that determine if the rocket will land successfully
  - The interaction of various features that determine the landing success rate
  - Conditions to ensure a successful landing

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
  - The logistic regression, SVM, decision tree classifier, and KNN models were used with the Python scikit-learn library.

Data Collection

- The data were collected with various methods
  - Firstly, we requested rocket launch data from SpaceX API
  - Then we checked the content response and decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - Finally, we then filtered the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We requested the SpaceX API to collect data, decoded the requested data and did some basic data wrangling and formatting.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w1_1.%20Collecting%20the%20data.ipynb

Get request for the rocket launch data using API

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

Use json_normalize( ) to convert response to data frame

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

```
In [12]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
In [16]: # Get the head of the dataframe
         data.head()
```

Filter the data using BoosterVersion then deal with missing values

```
In [38]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9 = data[data.BoosterVersion == 'Falcon 9']
         data_falcon9
```

```
In [42]: # Calculate the mean value of PayloadMass column
         PayloadMass_Mean = data_falcon9.PayloadMass.mean()
         # Replace the np.nan values with its mean value
         data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PayloadMass_Mean)
```

Data Collection - Scraping

- We applied web scrapping to get Falcon 9 launch records with BeautifulSoup, parsed the table, and converted it into a pandas dataframe.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w1_2.%20Web%20scraping%20launches%20records.ipynb

HTTP GET Method to request the Falcon 9 rocket launch page

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1(

In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        response = requests.get(static_url).text
```

Create BeautifulSoup from the HTML Response

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(response, 'html.parser')

In [7]: # Use soup.title attribute
        print(soup.title)
```

Extract column names from the HTML table headers

```
In [10]: column_names = []

         # Apply find_all() function with `th` element on first_launch_table
         # Iterate each th element and apply the provided extract_column_from_header() to get a column name
         # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

         temp = soup.find_all('th')
         for x in range(len(temp)):
             try:
                 name = extract_column_from_header(temp[x])
                 if (name is not None and len(name) > 0):
                     column_names.append(name)
             except:
                 pass
```

Data Collection - Scraping

- We applied web scrapping to get Falcon 9 launch records with BeautifulSoup, parsed the table, and converted it into a pandas dataframe.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w1_2.%20Web%20scraping%20launches%20records.ipynb

Create a data frame by parsing the launch HTML tables

```
In [24]:
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']=[]
```

Export CSV

Data Wrangling

- We calculated the number of launches on each site, the number and occurrence of each orbit, and occurrence of mission outcome per orbit type. Then we summarized the data and created a landing outcome.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w1_3.%20Data%20wrangling.ipynb

Calculate the number of launches on each site

In [5]:
```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

Calculate the number and occurrence of each orbit

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

Calculate the number and occurrence of mission outcome per orbit type

In [7]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

In [8]:
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

In [9]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Create a landing outcome label from Outcome column

In [10]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

In [11]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

EDA with Data Visualization

- We explored the relationship between flight number and launch site, payload and launch site with scatterplots. Since the causal relationship between variables can be better observed.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w2_2.%20EDA%20Data%20Visualization.ipynb

Relationship between Flight Number and Launch Site

In [25]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be
sns.set_theme(style="white", palette=None)
sns.catplot(x="FlightNumber",y="LaunchSite",hue='Class',data=df, aspect=3)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

Relationship between Payload and Launch Site

In [26]:
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis
sns.set_theme(style="white", palette=None)
sns.catplot(x="PayloadMass",y="LaunchSite",hue='Class',data=df, aspect=3)
plt.xlabel("Payload Mass KG",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

EDA with Data Visualization

- Then we evaluated the relationship between success rate of each orbit type with bar chart, relationship between flight number and orbit type, payload and orbit type with scatterplot as well
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w2_2.%20EDA%20Data%20Visualization.ipynb

Relationship between success rate of each orbit type

```
In [19]:
bar_index=bar.index
bar_values=bar.values

In [21]:
sns.set_theme(style="whitegrid", palette="pastel")
sns.catplot(x=bar_index,y=bar_values, data=df, kind='bar', aspect=3)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Success rate of each orbit", fontsize=20)
plt.show()
```

Relationship between FlightNumber and Orbit type

```
In [24]:
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be
sns.set_theme(style="white", palette=None)
sns.catplot(x="FlightNumber",y="Orbit",hue='Class',data=df, aspect=3)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

Relationship between Payload and Orbit type

```
In [28]:
# Plot a scatter point chart with x axis to be Payload and y axis to be the O
sns.set_theme(style="white", palette=None)
sns.catplot(x="PayloadMass",y="Orbit",hue='Class',data=df, aspect=3)
plt.xlabel("Payload Mass KG",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

EDA with Data Visualization

- For the launch success yearly trend, we extracted the yearly launch data and merged with the date. We used a line graph to better show the trend and change over years.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w2_2.%20EDA%20Data%20Visualization.ipynb

Launch success yearly trend

In [14]:

```
# Plot a line chart with x axis to be the extracted year and y axis to be the
df1=pd.DataFrame(Extract_year(df['Date']),columns =['year'])
df1['Class']=df['Class']
sns.lineplot(x=Unique_year, y=Success_rate)
plt.xlabel("Year", fontsize=20)
plt.ylabel("Success Rate", fontsize=20)
plt.show()
```

EDA with SQL

- In this section we loaded the SpaceX datasets with iPython SQL and queried in the Jupyter Notebook. We wrote SQL queries to get insights from the data, details are as follow:

- Display the names of the unique launch sites in the space mission
- Display records where launch sites begin with the string 'CCA‘ (string)
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass in a certain range.
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. (subquery)
- Rank counts according to date

The link to the notebook is (click and straight redirect):
https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w2_1.%20SQL%20Assignment.ipynb

Build an Interactive Map with Folium

- We marked all launch sites and added map objects such as markers, circles, and lines to mark the success or failure of launches for each site. We then assigned the feature launch outcomes failure to class 0 and success to class 1.
- We used the color-labeled marker clusters to identify which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w3_1.%20Launch%20Site%20Location.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash.
- We plotted pie charts showing the successful launches by sites.
- We plotted scatter graph showing the relationship with successful launches and Payload Mass (Kg) for the different booster versions.
- With the pie charts we can discern the proportion difference much easier, while the scatter plots enable us to find out the trend and correlation patterns.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w3_2_build_a_dashboard_application_with_plotly_dash_.py

Predictive Analysis (Classification)

- We loaded and transformed the data using numpy and pandas, then we split our data into training set and testing set with scikit-learn.
- We built machine learning models such as Logistic Regression, SVM, Decision Tree Classification, and KNN. The models were tested with different hyperparameters using the GridSearchCV function.
- The accuracy were then calculated and compared.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w4_1.%20SpaceX%20Machine%20Learning%20Prediction.ipynb

Data standardization and transformation

```
In [5]:
Y = data['Class'].to_numpy()
Y

In [6]:
# students get this
transform = preprocessing.StandardScaler()

In [7]:
X = transform.fit_transform(X)
```

Split the data to training set and testing set

```
In [8]:
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

Logistic Regression

```
In [10]:
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}

In [11]:
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()

In [12]:
logreg_cv = GridSearchCV(lr,parameters,cv=10)
logreg_cv.fit(X_train, Y_train)
```

Predictive Analysis (Classification)

- We loaded and transformed the data using numpy and pandas, then we split our data into training set and testing set with scikit-learn.
- We built machine learning models such as Logistic Regression, SVM, Decision Tree Classification, and KNN. The models were tested with different hyperparameters using the GridSearchCV function.
- The accuracy were then calculated and compared.
- The link to the notebook is (click and straight redirect):

https://github.com/herecomesmax/newrepo/blob/Applied-Data-Science-Capstone/w4_1.%20SpaceX%20Machine%20Learning%20Prediction.ipynb

SVM

```
In [16]: parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
                      'C': np.logspace(-3, 3, 5),
                      'gamma':np.logspace(-3, 3, 5)}
         svm = SVC()

In [17]: svm_cv = GridSearchCV(svm,parameters,cv=10)
         svm_cv.fit(X_train, Y_train)
```

Decision Tree

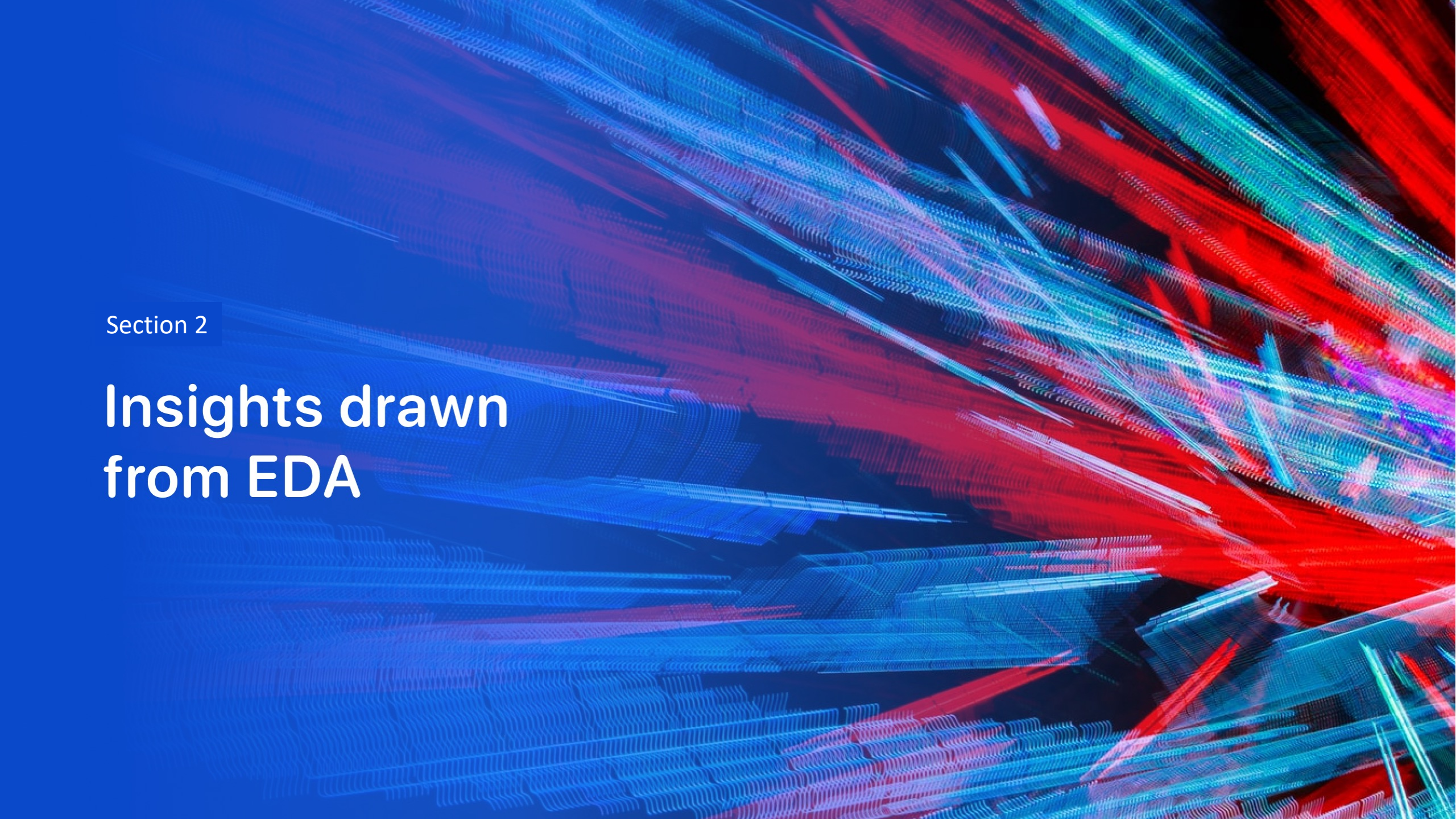
```
In [21]: parameters = {'criterion': ['gini', 'entropy'],
              'splitter': ['best', 'random'],
              'max_depth': [2*n for n in range(1,10)],
              'max_features': ['auto', 'sqrt'],
              'min_samples_leaf': [1, 2, 4],
              'min_samples_split': [2, 5, 10]}

         tree = DecisionTreeClassifier()

In [22]: tree_cv = GridSearchCV(tree,parameters,cv=10)
         tree_cv.fit(X_train, Y_train)
```

KNN

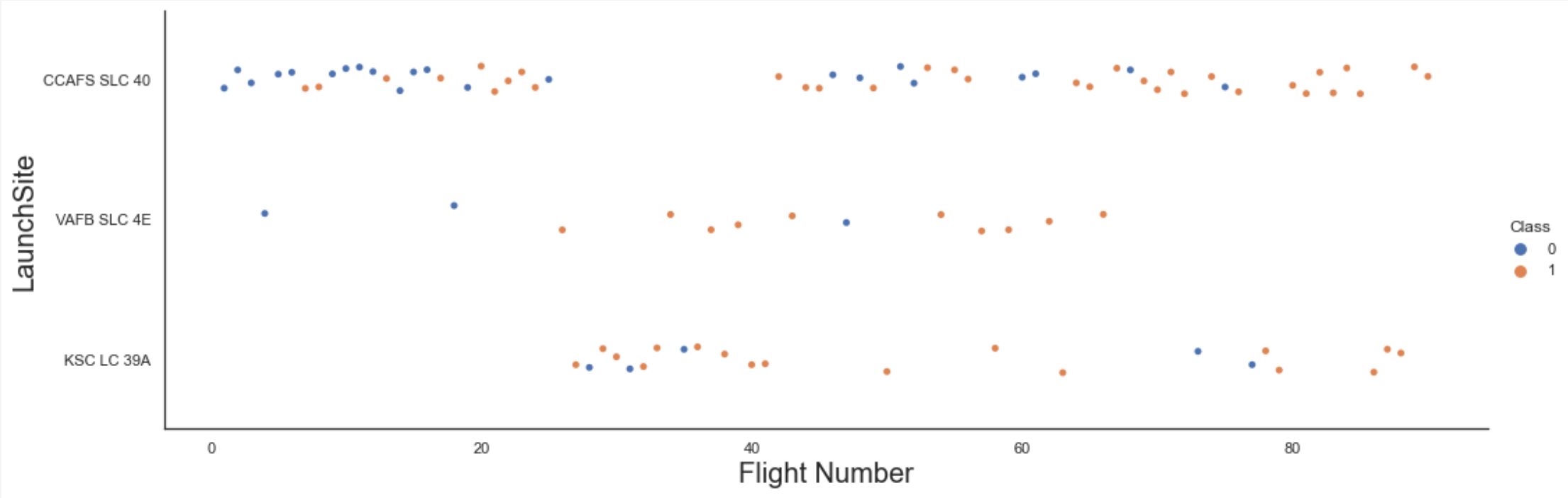
```
In [26]: parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                      'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                      'p': [1,2]}

         KNN = KNeighborsClassifier()

In [27]: knn_cv = GridSearchCV(KNN,parameters,cv=10)
         knn_cv.fit(X_train, Y_train)
```

Results

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

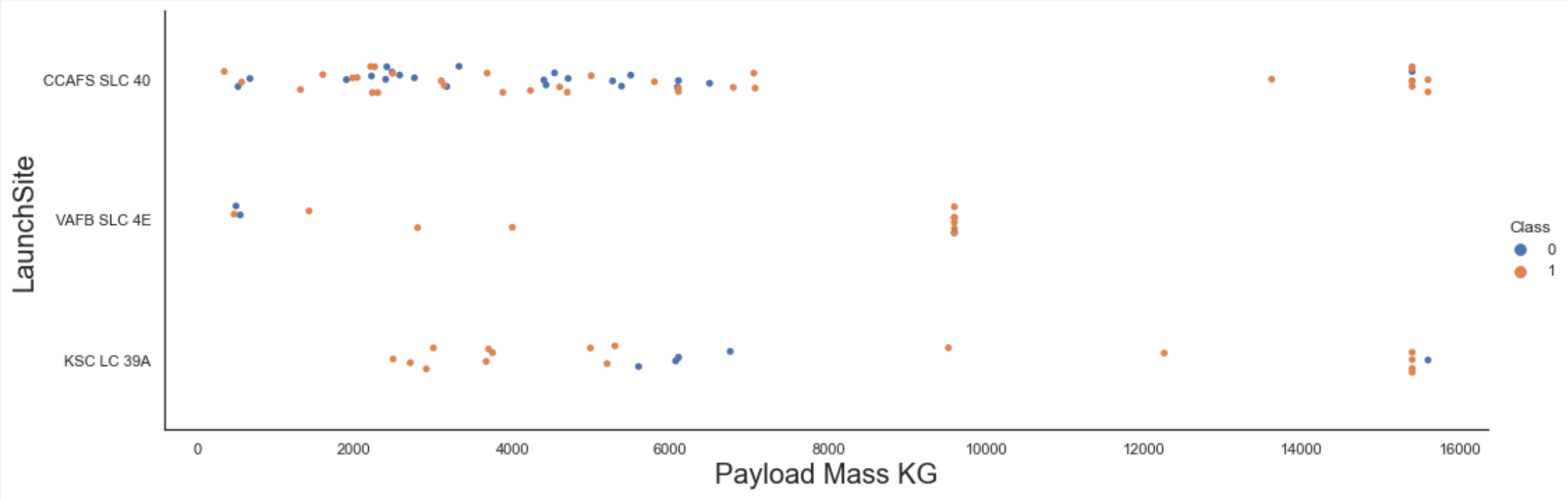

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



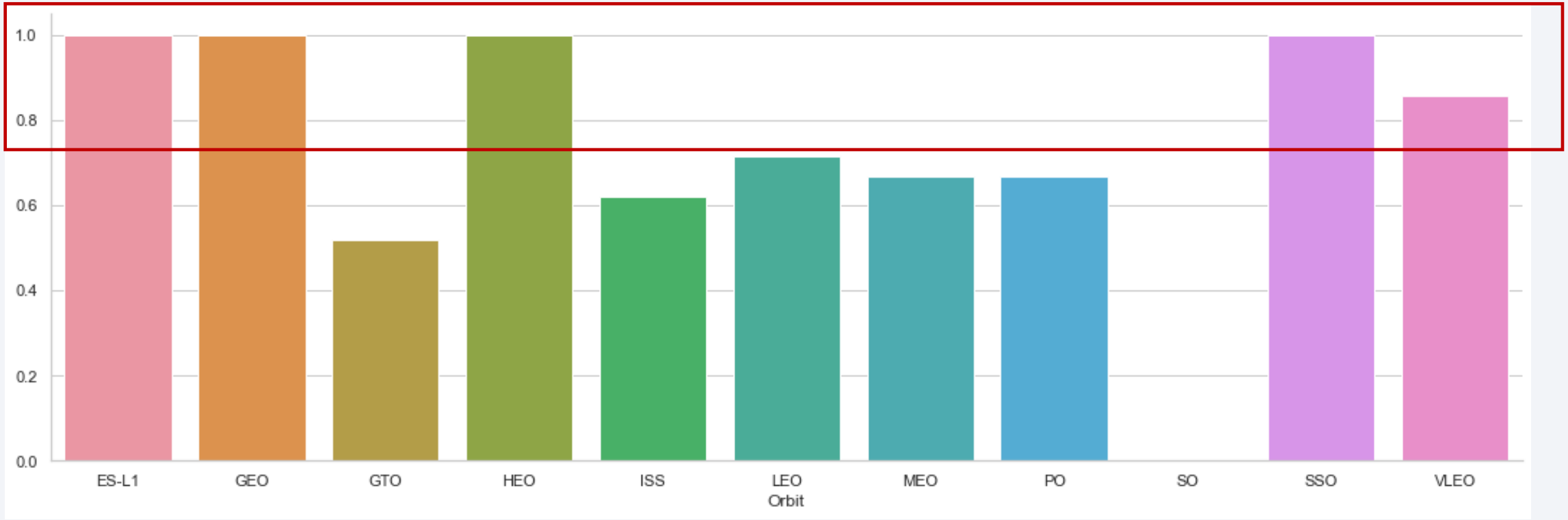
- The scatterplot showed that as the flight number increases at a launch site, the success rate tend to increase as well. Since the variability in total flights is decreasing.

Payload vs. Launch Site



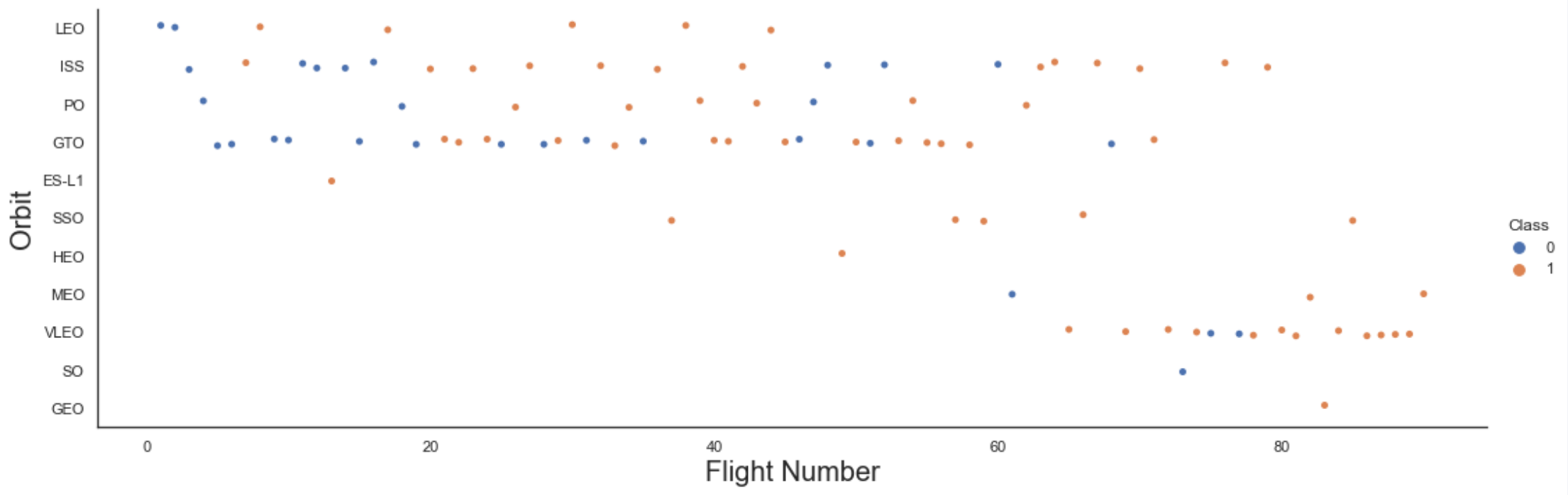
- At launch site CCAFS SLC 40, the higher the payload mass tend to result in a higher success rate. While at VAFB SLC 4E and KSC LC 39A, the relationship of payload mass and success rate are relatively hard to discern.

Success Rate vs. Orbit Type



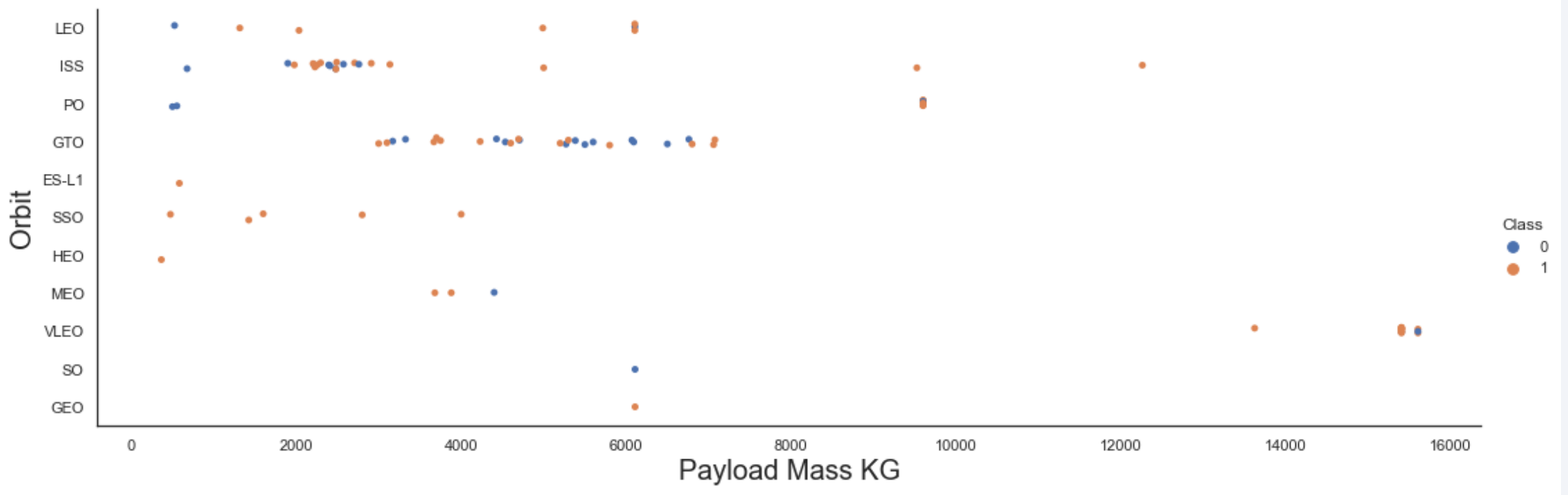
- The bar chart shows the relationship of success rate and orbit type. And we can see that some specific orbits such as ES-L1, GEO, HEO, SSO, and VLEO have higher success rates.

Flight Number vs. Orbit Type



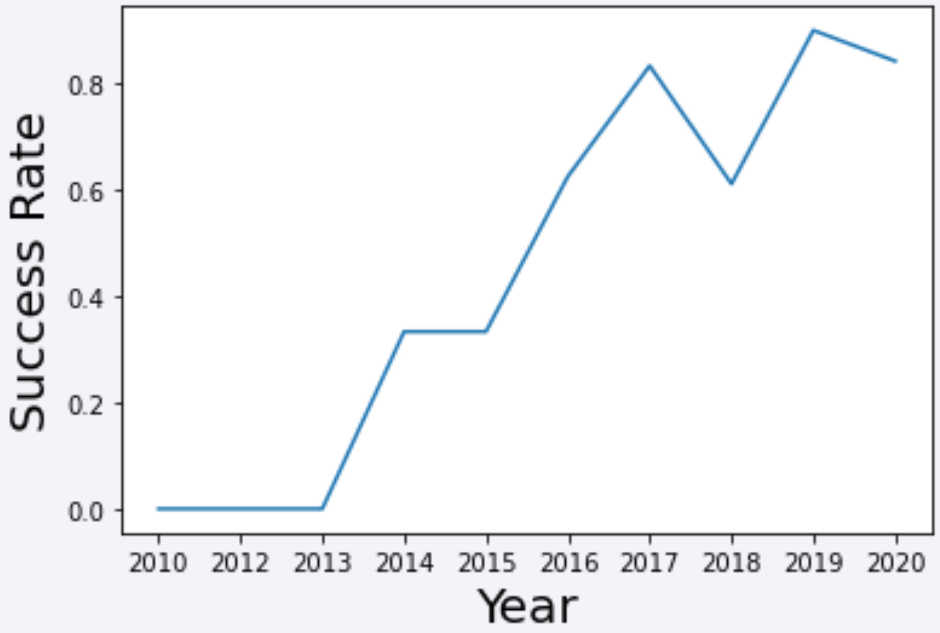
- The scatterplot shows that with orbit LEO, the success rate is positively correlated with the flight number whereas other orbits such as GTO and PO, no relationship between flight number and success rate is observed.

Payload vs. Orbit Type



- The scatterplot shows that with orbit LEO, PO and ISS, the success rate is positively correlated with the payload mass.

Launch Success Yearly Trend



- The line graph shows that the success rate of the launches has an increasing trend over years from 2013 to 2020.

All Launch Site Names

- We used the DISTINCT query function to display the names of the unique launch sites in the space mission.

In [7]:

```
%%sql
SELECT distinct(LAUNCH_SITE)
FROM SPACEXTBL
```

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

Out[7]:

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

Launch Site Names Begin with 'CCA'

- We used the LIKE and LIMIT query function to display 5 records where launch sites begin with the string 'CCA'.

In [8]:

```
%%sql
SELECT *
FROM SPACEXTBL where LAUNCH_SITE like 'CCA%' limit 5
```

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

Out[8]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payloa
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Drago Spacecra Qualificatio Un
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Drago demo fligh C1, tw CubeSats barrel o Brouer chees
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Drago demo fligh C
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	Space CRS-
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	Space CRS-

Total Payload Mass

- We used the SUM query function to calculate the total payload carried by boosters from NASA.

In [10]:

```
%%sql
SELECT sum(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE CUSTOMER = 'NASA (CRS)'
```

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

Out[10]:

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- We used the AVERAGE query function to calculate the average payload mass carried by booster version F9 v1.1.

In [11]:

```
%%sql
SELECT avg(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE BOOSTER_VERSION = 'F9 v1.1'
```

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

Out[11]:

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- We used to MIN query function to find the dates of the first successful landing outcome on ground pad. Before that, we used Pandas TO_DATE_TIME to convert the string from object to datetime format.

In [26]:

```
%%sql
SELECT min(date)
FROM SPACEXTBL
WHERE landing__outcome = 'Success (ground pad)'
```

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

Out[26]:

min(date)
2015-12-22 00:00:00

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the WHERE, AND, and constraints to list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000.

In [15]:

```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE Landing__Outcome = 'Success (drone ship)'
    and PAYLOAD_MASS__KG_ > 4000
    and PAYLOAD_MASS__KG_ < 6000
```

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

Out[15]:

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

Total Number of Successful and Failure Mission Outcomes

- We used the COUNT and LIKE query functions to calculate the total number of successful and failure mission outcomes.

```
In [16]: %%sql
         SELECT count(MISSION_OUTCOME)
         FROM SPACEXTBL
         WHERE MISSION_OUTCOME LIKE 'Success%' or MISSION_OUTCOME LIKE 'Failure%'

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

Out[16]:

count(MISSION_OUTCOME)
101

Boosters Carried Maximum Payload

- We used a subquery and MAX query function to list the names of the booster which have carried the maximum payload mass.

In [17]:

```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT max(PAYLOAD_MASS__KG_)
                           FROM SPACEXTBL)
```

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

Out[17]:

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

- We used query functions WHERE and date features to list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015. We used substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [18]:
```
%%sql
SELECT substr(Date, 4, 2),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE
FROM SPACEXTBL
WHERE substr(Date,7,4)='2015'
```

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

Out[18]:

substr(Date, 4, 2)	Mission_Outcome	Booster_Version	Launch_Site
01	Success	F9 v1.1 B1012	CCAFS LC-40
02	Success	F9 v1.1 B1013	CCAFS LC-40
03	Success	F9 v1.1 B1014	CCAFS LC-40
04	Success	F9 v1.1 B1015	CCAFS LC-40
04	Success	F9 v1.1 B1016	CCAFS LC-40
06	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

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

- We used the BETWEEN, GROUP BY, and ORDER BY to rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

In [29]:

```
%%sql
SELECT LANDING__OUTCOME, COUNT(*) AS COUNT_LAUNCHES
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY COUNT_LAUNCHES DESC
```

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

Out[29]:

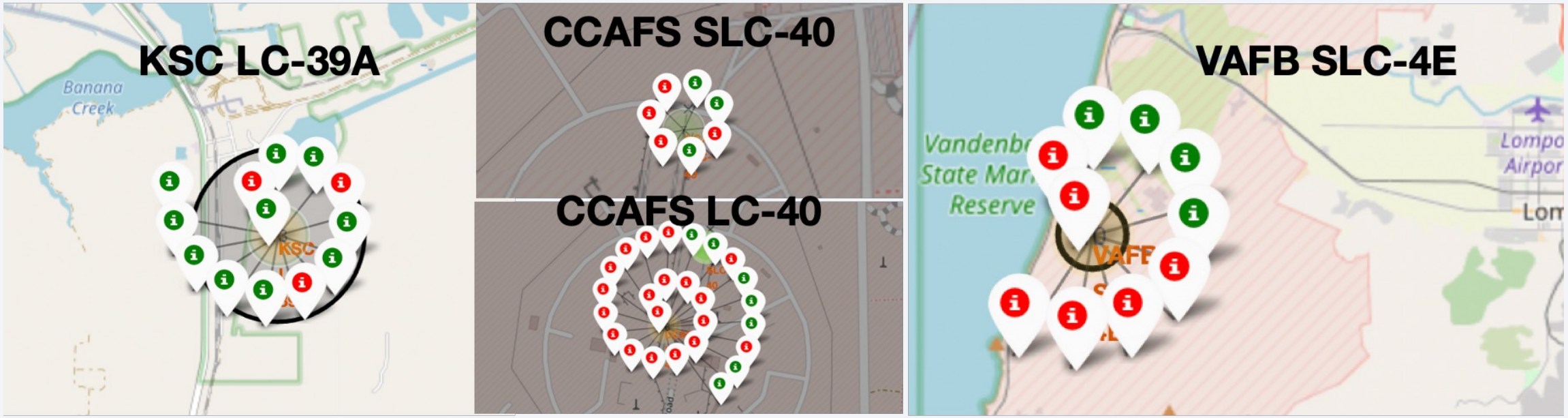
Landing__Outcome	COUNT_LAUNCHES
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1


Section 3
Launch Sites
Proximities Analysis

All Launch Sites: Global Markers

- From the global perspective, we see that the SpaceX launch sites are along the coastline of the United States of America. In Florida and California.

Launch Record and Color Markers

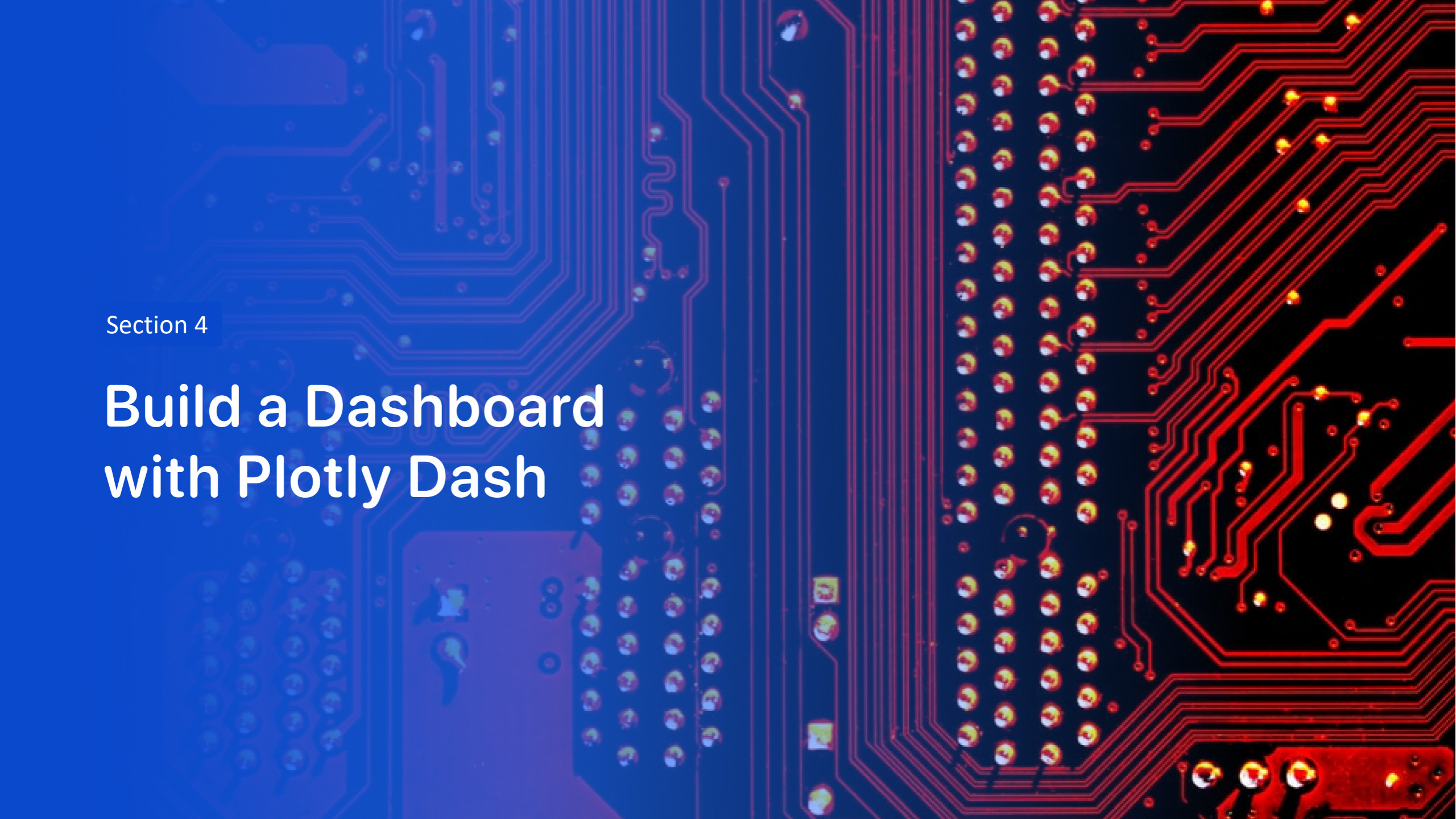


Florida Launch Sites **California Launch Sites**

Color markers were used to label launches. Green markers represent successful launches and red markers represent failures

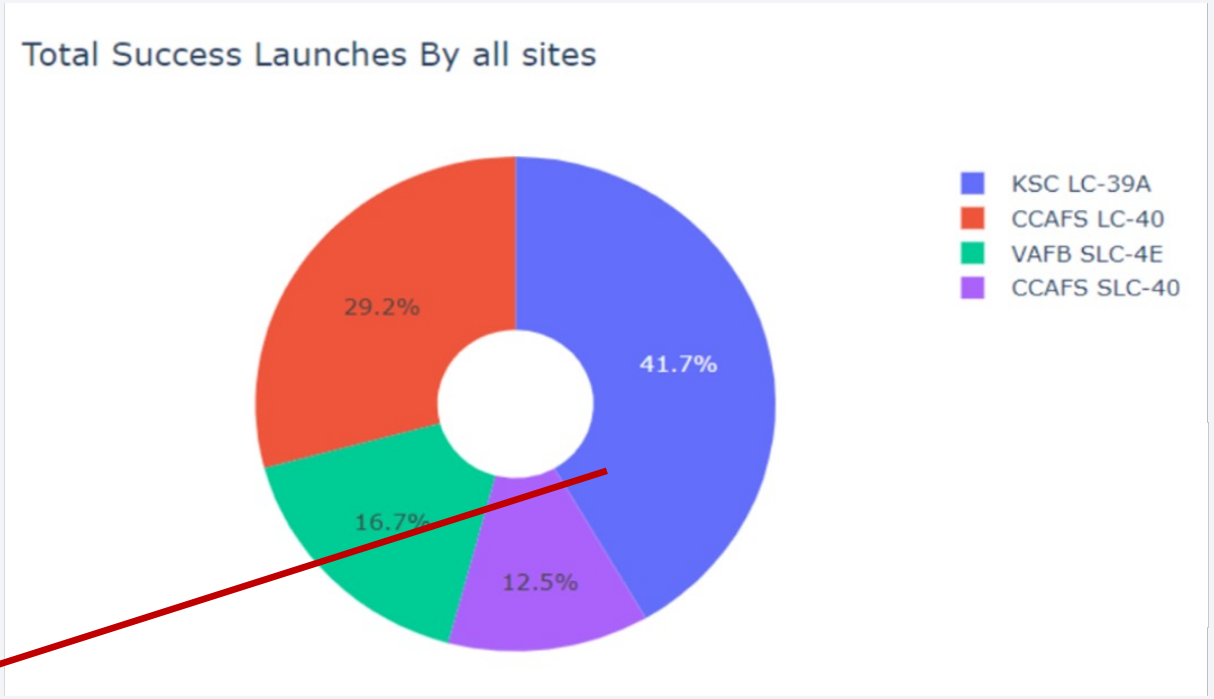
Distance From Launch Sites to Landmarks

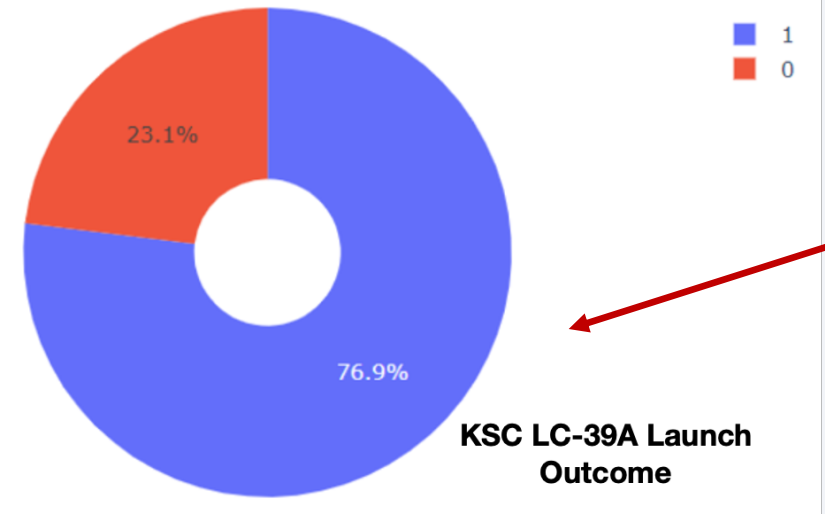
- Launch sites are close to the equator for minimizing the fuel to get the earth's orbit
- Launch sites are close to highways and railways, making the transportation of people and heavy cargo relatively easy.
- Launch sites close to the coastline. It enables the debris to land at sea and makes the option of sea landing an option when there is an emergency abortion.


Section 4
Build a Dashboard with Plotly Dash

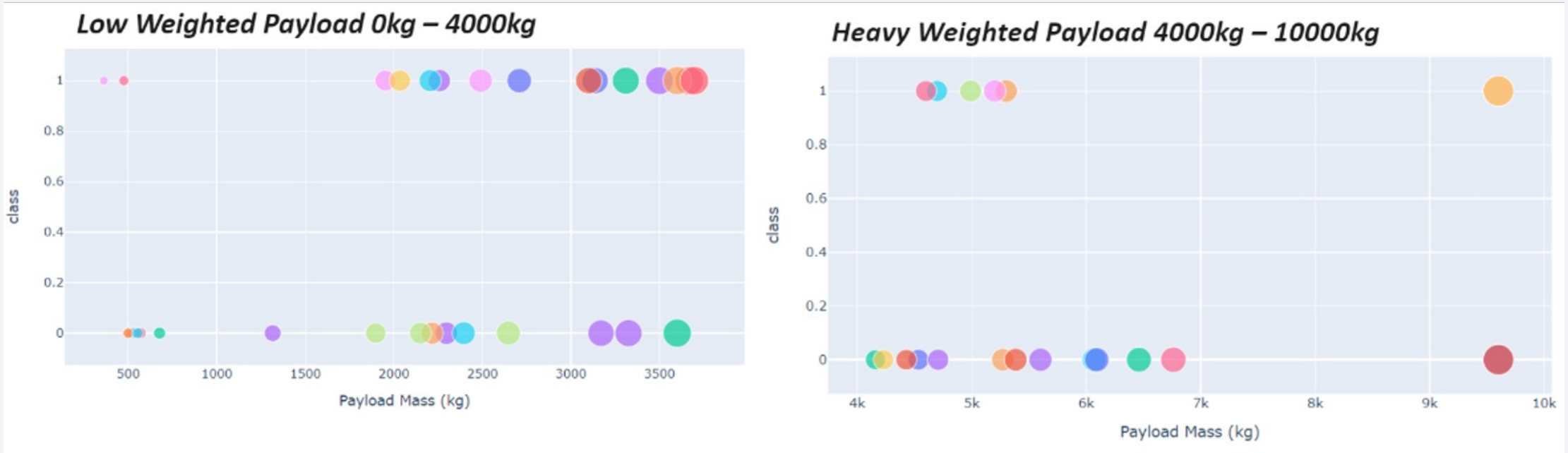
Pie Chart Showing Success Rates by Launch Sites

- The total successful launches were listed by launch sites. We can observe that the KSC LC-39A has the highest rate of successful launches (76.9%).





Scatterplot of Payload Mass and Outcomes (With Slider feature)



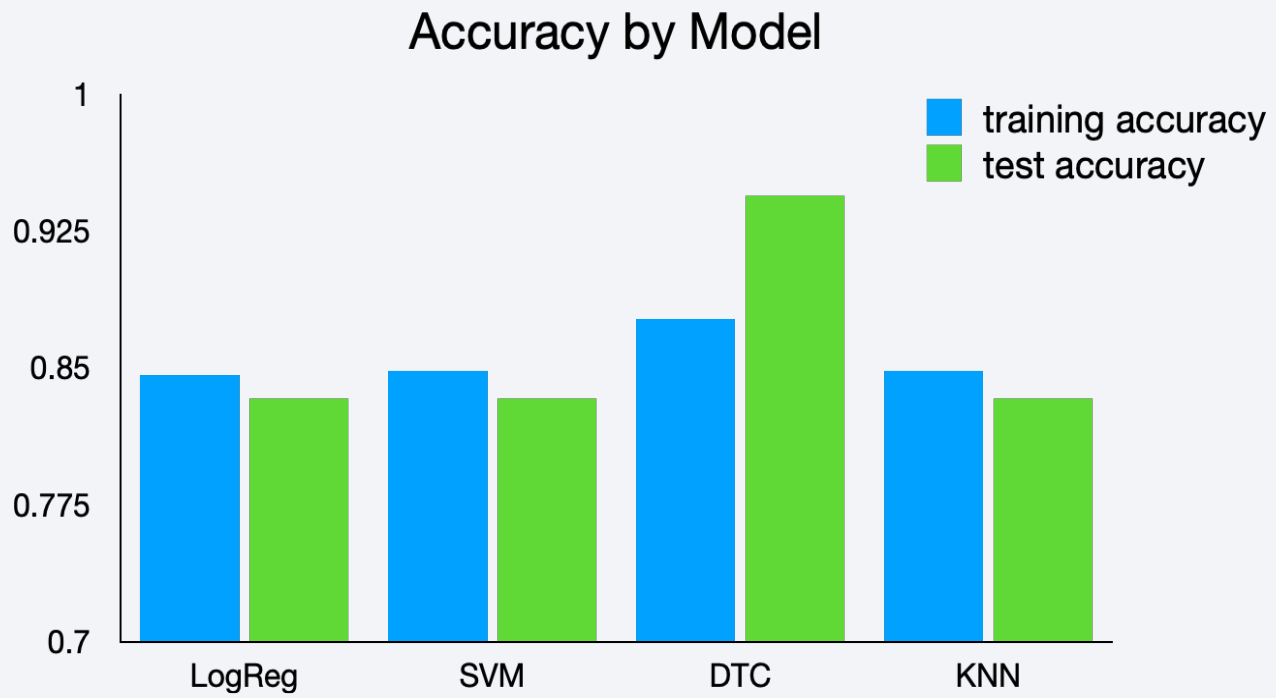
- The success rates of the launches for low weighted payload mass is comparatively higher than heavy weighted payload mass, as we can see from the graphs.

Section 5

Predictive Analysis (Classification)

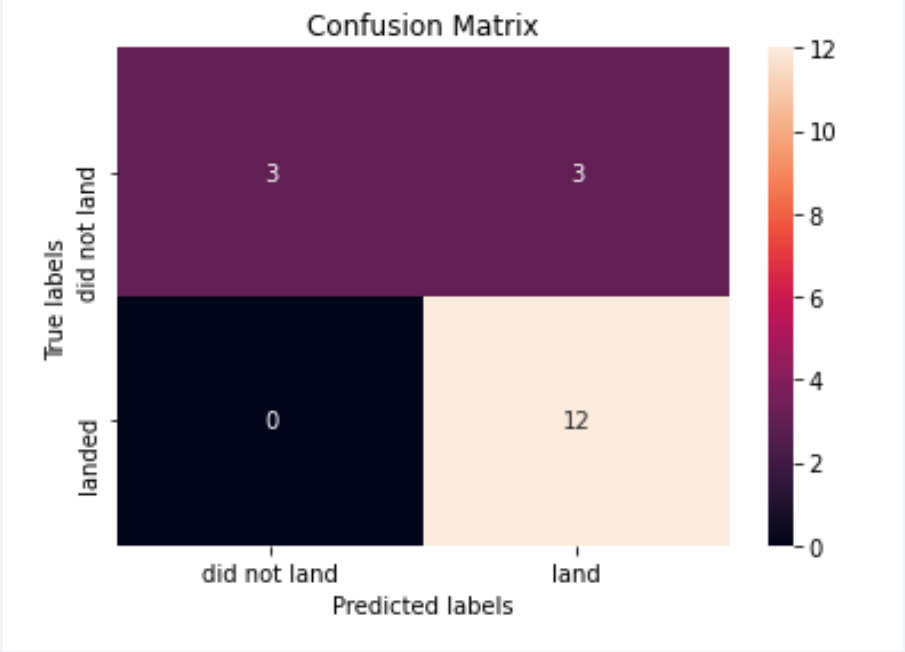
Classification Accuracy

- The model accuracy for all built classification models were shown in a bar chart. The decision tree classifier model is the one with the highest classification accuracy



Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives. In this case, it means unsuccessful landing marked as successful landing by the classifier.



Conclusions

From the case study, we can conclude that

- Sites with more launches tend to have a higher rate of success (Low variabilities)
- Correlations can be preserved in some Launch Sites with certain variables. But we can't generalize all cases with different variables since the launch outcomes are different by sites.
- The success rate of launch is increasing over years from 2013 to 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!