IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sirine Martins
May 31st, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection using Web Scraping and SpaceX API
  - Data Wrangling using Pandas and NumPy
  - Exploratory Data Analysis (EDA) using SQL and Pandas & Matplotlib
  - Interactive Visual Analytics using Folium and Plotly Dash
  - Machine Learning Prediction with Classification Algorithms

- Summary of all results
  - Site with Most Successful Launches: KSC LC-39A (41.7%)
  - Site with Least Successful Launches: CCAFS SLC-40 (12.5%)
  - Best Prediction Algorithm: Decision Trees (88.9% accuracy)

Introduction

- Cost of Falcon 9 Rocket Launches
- Landing Success Rate of Falcon 9 First Stage
- Success/Failure Rates by Launch Site
- Landing Success Rate Prediction

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology
  - Data acquired through web scraping from Wikipedia and get request from SpaceX API
- Perform data wrangling
  - Data preparation performed using Pandas
- Perform exploratory data analysis (EDA) using visualization and SQL
  - EDA and Feature Engineering using Pandas and Matplotlib
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression
  - Support Vector Machine (SVM)
  - K Nearest Neighbors (KNN)
  - Decision Trees

Data Collection

- Data collected by web scraping Wikipedia page titled 'List of Falcon 9 and Falcon Heavy launches'
- Data acquired by performing a get request to the SpaceX API

Data Collection - Scraping

- Web scraping from Wikipedia page using requests and Beautiful Soup
- [Web Scraping Process]

1. Import required libraries

```
import sys
import requests
from bs4 import BeautifulSoup
import re
import unicodedata
import pandas as pd
```

2. Use requests.get() method and assign response to an object, then create a Beautiful Soup object

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
response = requests.get(static_url).text
soup = BeautifulSoup(response, 'html.parser')
```

3. Find all tables on Wiki page, select the one with the launch records, then extract column names one by one

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
column_names = []
headers = first_launch_table.find_all('th')
for i in range(len(headers)):
    name = extract_column_from_header(headers[i])
    if (name is not None) and (len(name) > 0):
        column_names.append(name)
```

4. Create dictionary with extracted column names, then create a dataframe by parsing the HTML tables

```
launch_dict= dict.fromkeys(column_names)

del launch_dict['Date and time ( )']
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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]

df=pd.DataFrame(launch_dict)
```

Data Collection – SpaceX API

- Data collection using SpaceX REST API
- Data Collection Process

1. Import required libraries

```
[ ]: import requests
import pandas as pd
import numpy as np
import datetime
```

2. Request and parse the SpaceX launch data using the GET request, then store the data in pre-defined lists and construct dataset (from dictionary to dataframe)

```
[ ]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
response = requests.get(static_json_url)
data = pd.json_normalize(response.json())

getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)

launch_dict = {'FlightNumber': list(data['flight_number']),'Date': list(data['date']),'BoosterVersion':BoosterVersion,'PayloadMass':PayloadMass,
'Orbit':Orbit,'LaunchSite':LaunchSite,'Outcome':Outcome,'Flights':Flights,'GridFins':GridFins,'Reused':Reused,'Legs':Legs,'LandingPad':LandingPad,
'Block':Block,'ReusedCount':ReusedCount,'Serial':Serial,'Longitude': Longitude,'Latitude': Latitude}

launch_data = pd.DataFrame(launch_dict)
```

3. Filter the dataframe so that it contains only Falcon 9 launches, then replace missing values with the mean for the Payload mass in kg and export it to a CSV file

```
[ ]: data_falcon9 = launch_data[launch_data['BoosterVersion']!='Falcon 1']

data_falcon9['PayloadMass'].replace(np.nan, data_falcon9['PayloadMass'].mean(), inplace=True)

data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

Data Wrangling

- Data was prepared and processed using Pandas

1. Import required libraries

```
[ ]: import pandas as pd
     import numpy as np
```

2. Load the data

```
[ ]: df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv"
```

3. Calculate the number of launches on each site, the number and occurrence of each orbit, and the number and occurrence of mission outcome per orbit type

```
[ ]: df['LaunchSite'].value_counts()

     df['Orbit'].value_counts()

     df['Outcome'].value_counts()
```

4. Create a list 'Class' that stores the outcome of each launch (1 for successful, 0 for unsuccessful), then determine the success rate by calculating the mean of the data

```
[ ]: landing_class = []
     for key, value in df['Outcome'].items():
         if value in bad_outcomes:
             landing_class.append(0)
         else:
             landing_class.append(1)

     df['Class']=landing_class
     df["Class"].mean()
```

- Data Wrangling Process

EDA with Data Visualization

- Scatter Point Plots
  - Flight Number vs. Launch Site
  - Payload Mass (kg) vs. Launch Site
  - Flight Number vs. Orbit Type
  - Payload Mass (kg) vs. Orbit Type
- Bar and Line Charts
  - Orbit Type vs. Launch Success Rate (Bar Chart)
  - Year vs. Launch Success Rate (Line Chart)
- Exploratory Data Analysis with Pandas and Matplotlib Process

EDA with SQL

- Selected unique launch sites in the space mission
- Displayed 5 records where launch sites being with 'CCA' using % operator
- Displayed the total payload mass carried by boosters launched by NASA (CRS)
- Displayed the average payload mass carried by booster version F9 v1.1
- Listed the date of the first successful landing outcome in ground pad
- Listed the names of the boosters which have success in drone ship and payload mass between 4,000 kg and 6,000 kg
- Listed total number of successful and failure missions
- Listed the names of the booster versions which have carried the maximum payload mass using a subquery
- Listed the failed landing outcomes in drone ship, their booster versions, and launch site names in 2015
- Ranked the count of landing outcomes between 04-06-2010 and 20-03-2017 in descending order
- Exploratory Data Analysis with SQL Process

Build an Interactive Map with Folium

- Marked all launch sites on a map using circles and markers using latitude and longitude coordinates
- Marked the success/failed launches for each site on the map using MarkerCluster object to avoid too many markers having the same coordinate
- Calculated the distance between a launch site and its proximities using MousePosition object
- Drew a line between a launch site to a selected coastline point using PolyLine object

- Interactive Visual Analytics with Folium Process

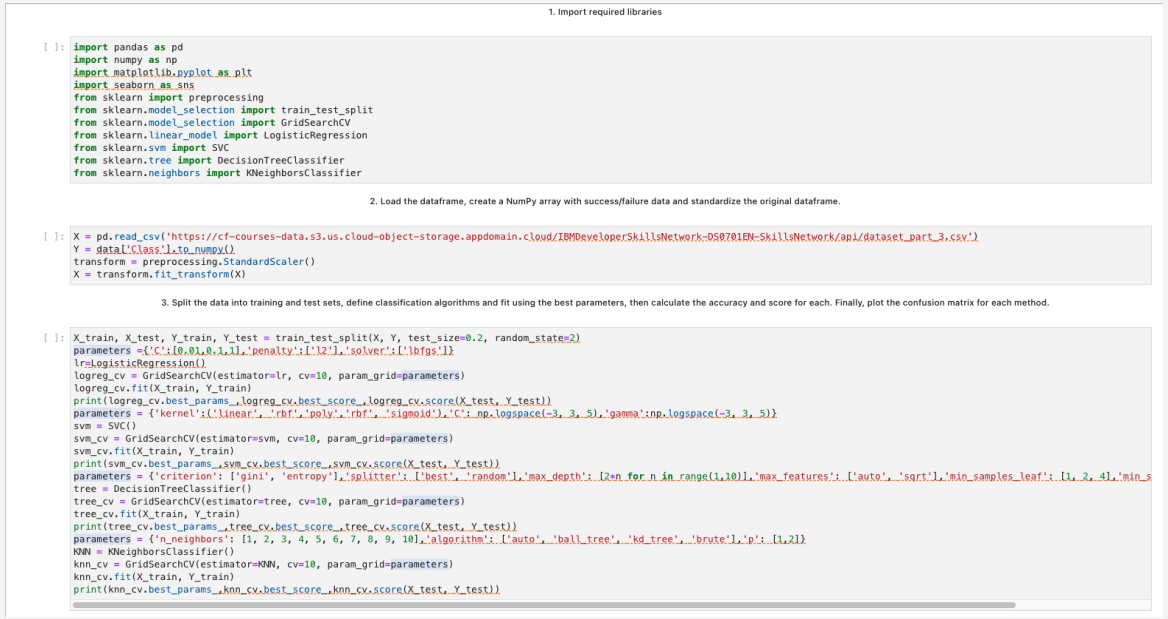
Build a Dashboard with Plotly Dash

- Created a pie chart with relative amount of successful launches by site for 4 sites
- Created 4 pie charts with the percentage of successful launches for each site
- Created a scatter point plot showing the correlation between Payload mass and launching success/failure for 4 sites
- Created 4 scatter point plots showing the correlation between Payload mass and launching success/failure for each site

- SpaceX Dash App Code

Predictive Analysis (Classification)

- Performed predictive analysis using Logistic Regression, SVM, Decision Trees and KNN classification algorithms



1. Import required libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

2. Load the dataframe, create a NumPy array with success/failure data and standardize the original dataframe.

```
X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_part_3.csv')
Y = data['Class'].to_numpy()
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

3. Split the data into training and test sets, define classification algorithms and fit using the best parameters, then calculate the accuracy and score for each. Finally, plot the confusion matrix for each method.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
parameters ={'C':[0.01,0.1,1],'penalty':['l2'],'solver':['lbfgs']}
lr=LogisticRegression()
logreg_cv = GridSearchCV(estimator=lr, cv=10, param_grid=parameters)
logreg_cv.fit(X_train, Y_train)
print(logreg_cv.best_params_,logreg_cv.best_score_,logreg_cv.score(X_test, Y_test))
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),'C': np.logspace(-3, 3, 5),'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
svm_cv = GridSearchCV(estimator=svm, cv=10, param_grid=parameters)
svm_cv.fit(X_train, Y_train)
print(svm_cv.best_params_,svm_cv.best_score_,svm_cv.score(X_test, Y_test))
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_s
tree = DecisionTreeClassifier()
tree_cv = GridSearchCV(estimator=tree, cv=10, param_grid=parameters)
tree_cv.fit(X_train, Y_train)
print(tree_cv.best_params_,tree_cv.best_score_,tree_cv.score(X_test, Y_test))
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],'p': [1,2]}
KNN = KNeighborsClassifier()
knn_cv = GridSearchCV(estimator=KNN, cv=10, param_grid=parameters)
knn_cv.fit(X_train, Y_train)
print(knn_cv.best_params_,knn_cv.best_score_,knn_cv.score(X_test, Y_test))
```

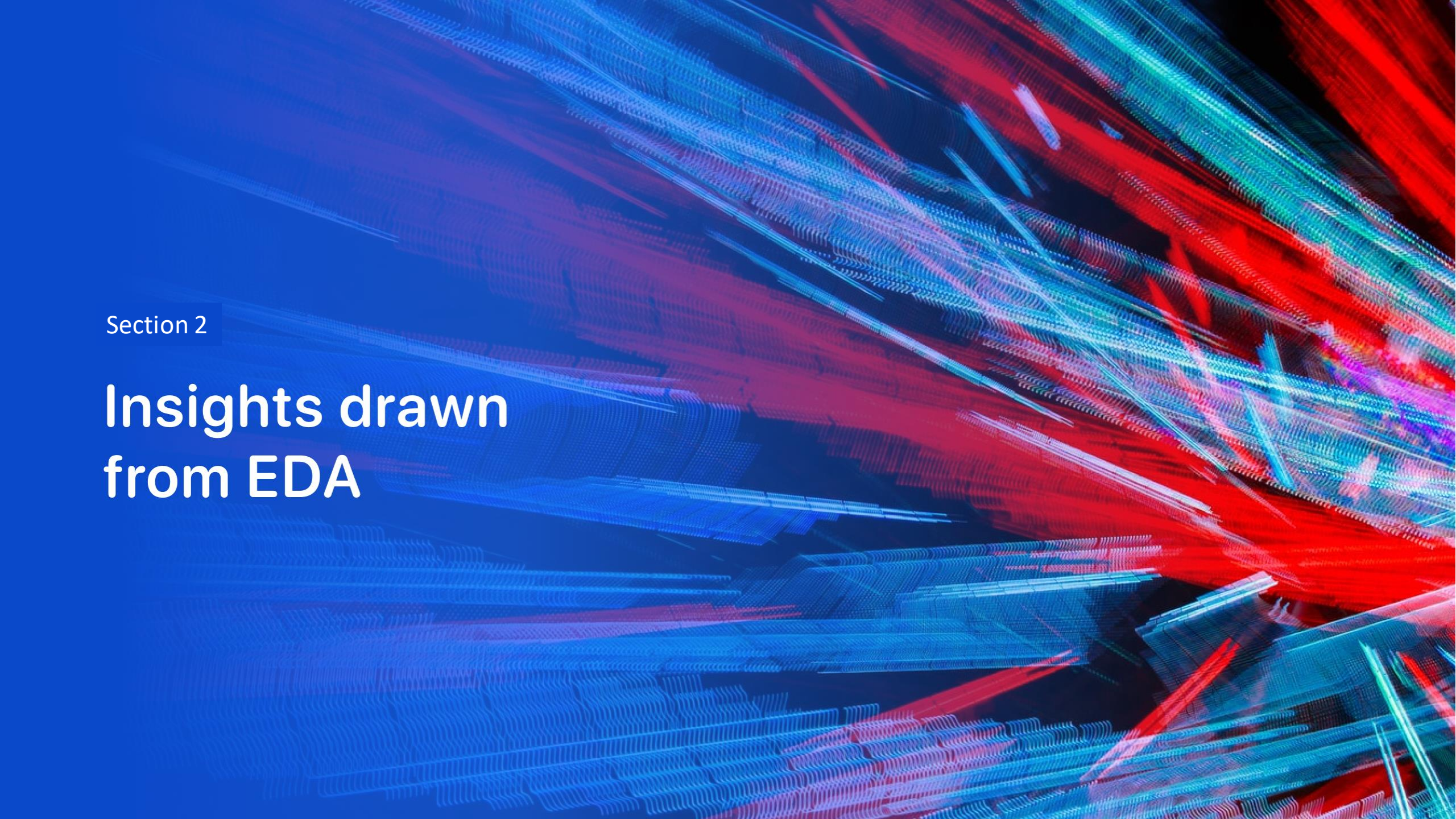
- Predictive Analysis (Classification) Process

Results

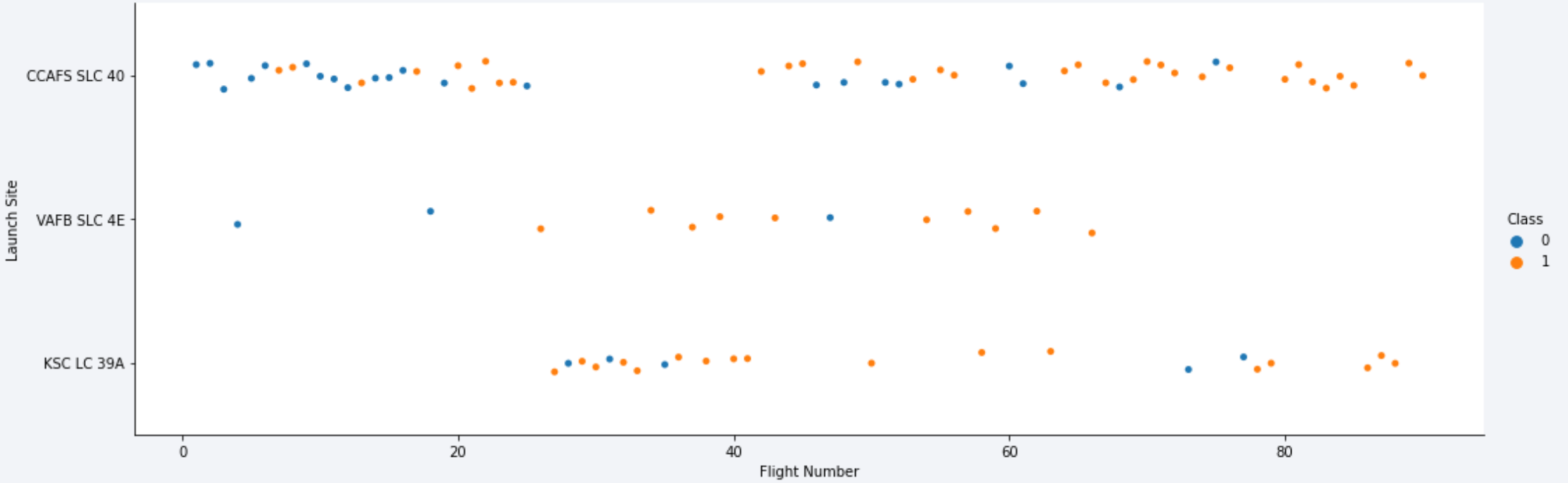
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

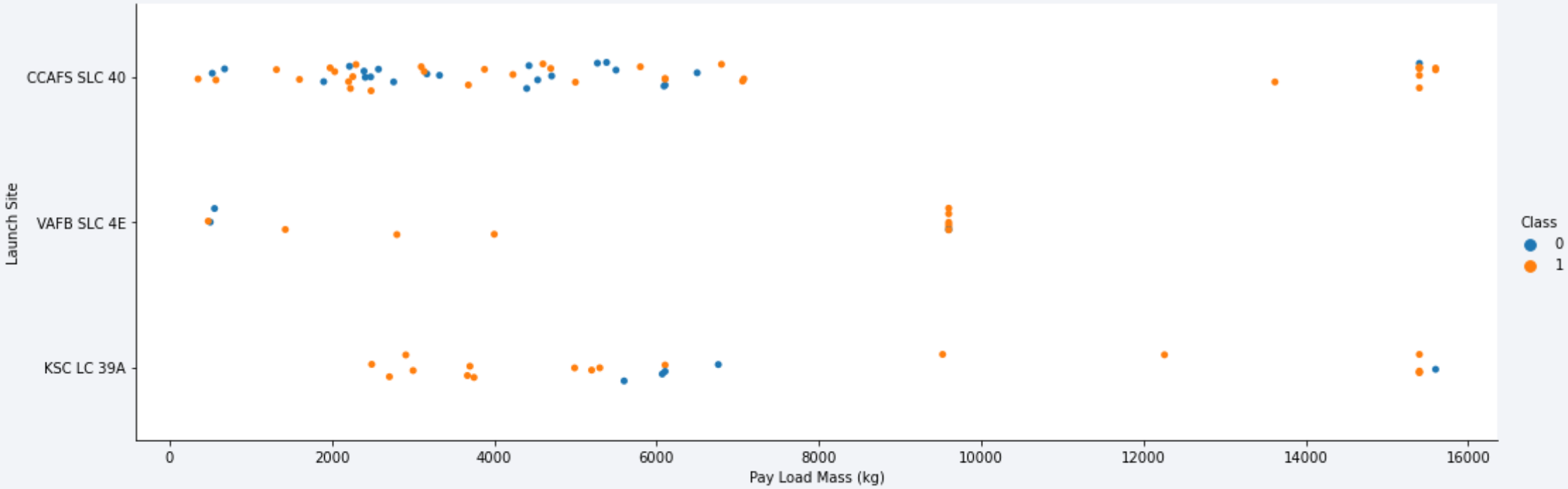


Flight Number vs. Launch Site



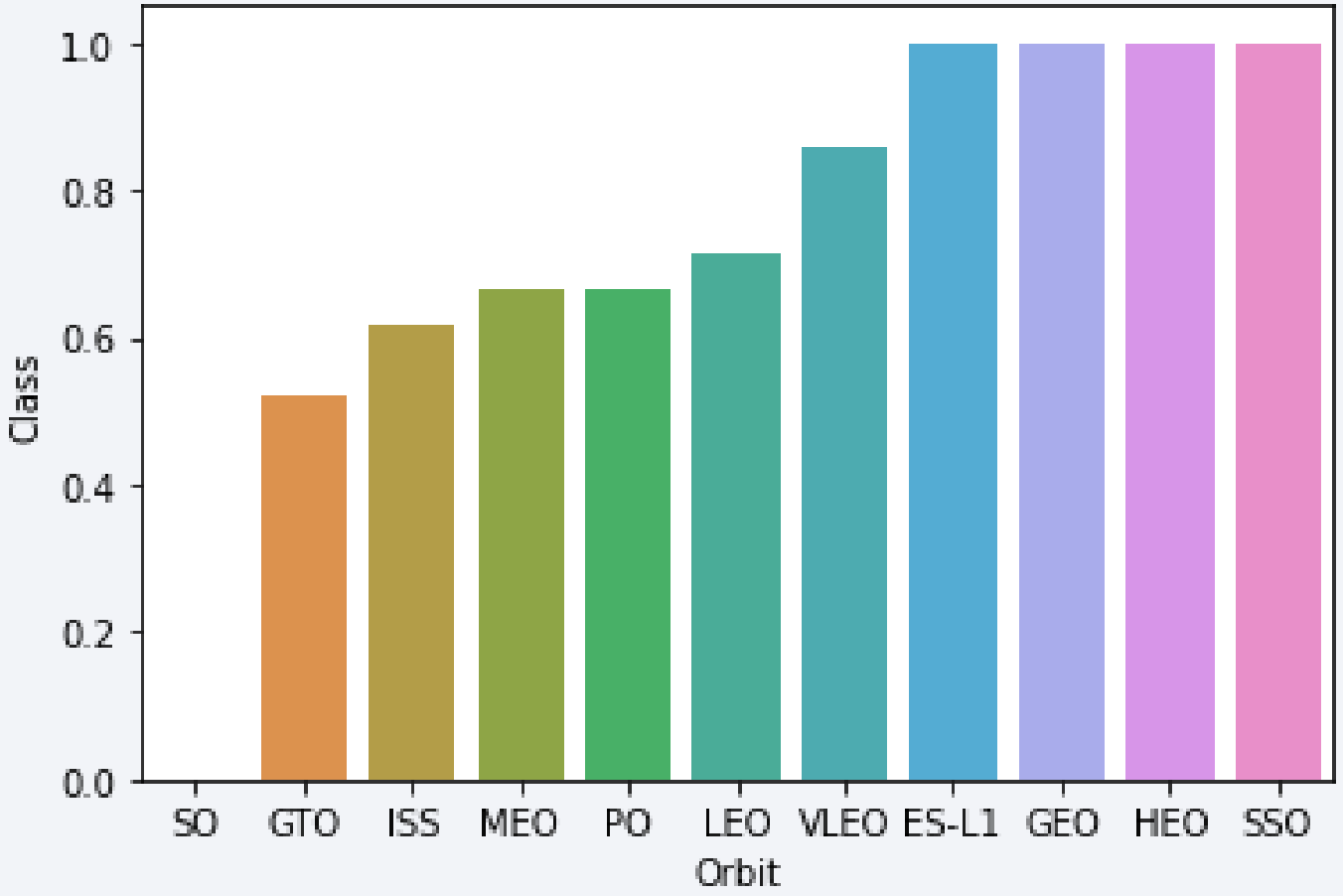
- As the flight number increased for each site, so did the success rate

Payload vs. Launch Site



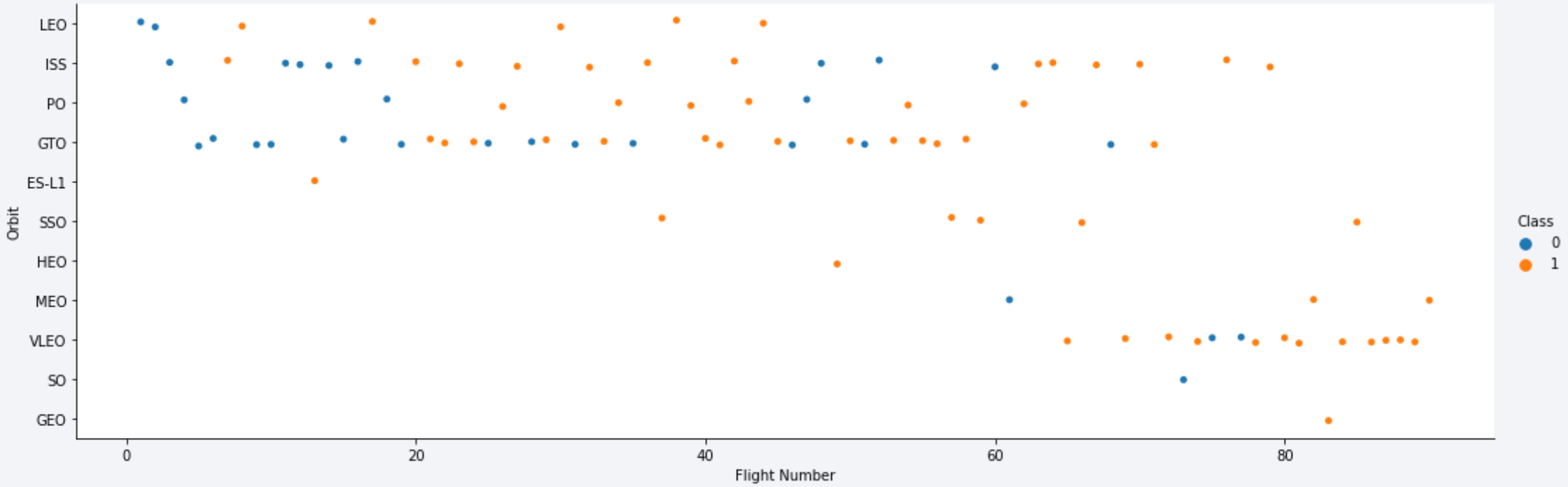
- As the payload mass (kg) increased for each site, so did the success with the exception of a few outliers in the maximum payload range

Success Rate vs. Orbit Type



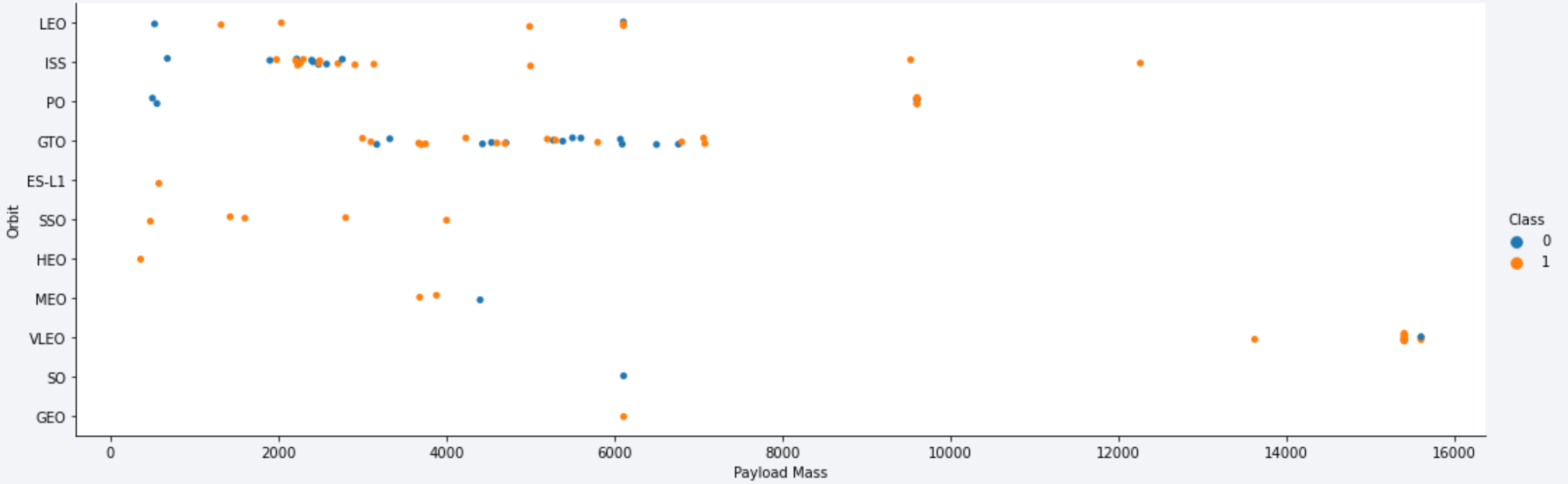
- Orbit types with the highest success rate (1.0):
  - SSO
  - HEO
  - GEO
  - ES-L1

Flight Number vs. Orbit Type



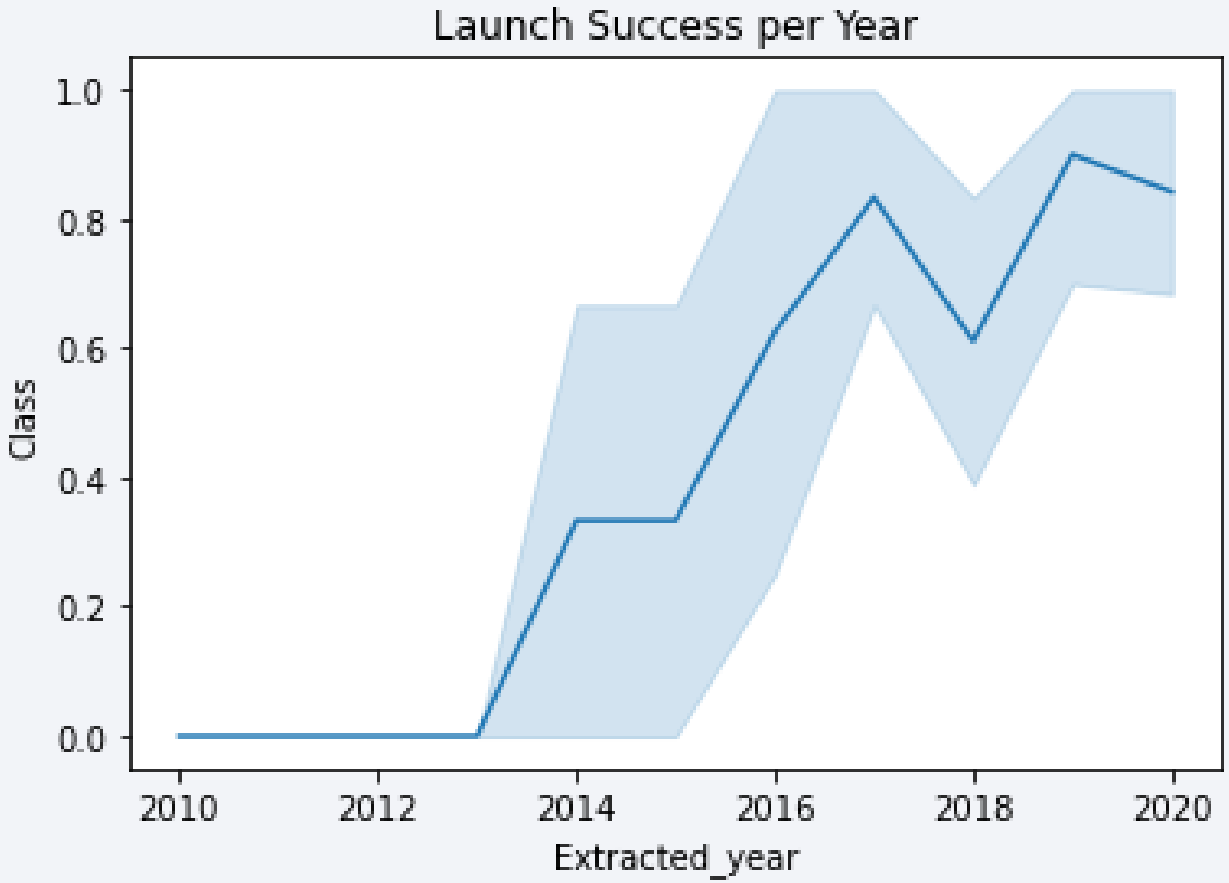
- For the most part, it seems that as the flight number increases, so does the success rate for every orbit type except for GTO where no correlations can be extracted from this plot

Payload vs. Orbit Type



- For the most part, it seems that as the payload mass increases, so does the success rate for every orbit type except for GTO where no correlations can be extracted from this plot

Launch Success Yearly Trend



- It can be observed that the success rate has been increasing rapidly since 2013, had slight decrease in 2018 and then resumed its upwards trajectory since

All Launch Site Names

Task 1

Display the names of the unique launch sites in the space mission

In [8]:
```
%sql SELECT DISTINCT(launch_site) FROM SPACEXTBL;
```

```
 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqblod8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[8]:

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

- SELECT DISTINCT statement used to avoid repeated launch sites, yielding a result set with unique names only

Launch Site Names Begin with 'CCA'

Task 2

Display 5 records where launch sites begin with the string 'CCA'

In [9]:
```
%sql SELECT * FROM SPACEXTBL WHERE launch_site LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[9]:

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

- % operator used to limit result set to launch sites beginning with string 'CCA'

Total Payload Mass

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

In [12]:
```
%sql SELECT SUM(payload_mass__kg_) FROM SPACEXTBL WHERE customer = 'NASA (CRS)';
```

```
 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqblod8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[12]:

1
45596

- SUM() function used to calculate the total payload mass
- WHERE clause used to limit result set to boosters launched by NASA (CRS)

Average Payload Mass by F9 v1.1

Task 4

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

```
In [13]: %sql SELECT AVG(payload_mass__kg_) FROM SPACEXTBL WHERE booster_version = 'F9 v1.1';

 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
Out[13]:    1
           2928
```

- AVG() function used to calculate the average payload mass
- WHERE clause used to limit result set to booster version F9 v1.1

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

In [17]:
```
%sql SELECT MIN(date) FROM SPACEXTBL WHERE landing__outcome = 'Success (ground pad)';
```

```
 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90108kqblod8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[17]:

1
2015-12-22

- MIN() function used to find the oldest date recorded
- WHERE clause used to limit result set to successful launches on ground pad

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

```
[5]: %sql SELECT booster_version FROM SPACEXTBL WHERE (payload_mass__kg_ > 4000 and payload_mass__kg_ < 6000) and landing__outcome = 'Success (drone ship)';
 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

[5]:

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

- WHERE clause used to limit the result set to 2 conditions:
  - Payload mass greater than 4,000 kg and smaller than 6,000 kg
  - Successful landing in drone ship

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
[19]: %sql SELECT COUNT(*) AS successful_outcome, (SELECT COUNT(*) FROM SPACEXTBL WHERE mission_outcome LIKE 'Failur%') AS failure_outcome FROM SPACEXTBL WHERE mission_outcome LIKE 'Succes%';
 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

[19]:

successful_outcome	failure_outcome
100	1

- COUNT() function used to return the number of records
- Subquery used to avoid the need of writing multiple queries
- % operator and WHERE clause used to limit result set to failure outcomes for the 2nd column, and limit result set to successful outcomes for 1st column

Boosters Carried Maximum Payload

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
[20]: %sql SELECT booster_version FROM SPACEXTBL WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL);
 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

[20]:

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

- WHERE clause used to limit result set to booster versions carrying maximum payload mass
- Subquery used to allow use of MAX() function in the WHERE clause

2015 Launch Records

Task 9

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

[21]:
```
%sql SELECT landing__outcome, booster_version, launch_site FROM SPACEXTBL WHERE landing__outcome = 'Failure (drone ship)' AND year(date) = 2015;
```

```
 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

[21]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- SELECT statement used to list landing outcome, booster version, and launch sites
- WHERE clause used to limit result set to 2 conditions:
  - Failed landings in drone ship
  - Year of 2015

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

Task 10

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

[6]:
```
%sql SELECT landing__outcome, COUNT(*) AS count FROM SPACEXTBL WHERE date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing__outcome ORDER BY 2 DESC;
```

```
 * ibm_db_sa://rhx09276:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

[6]:

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

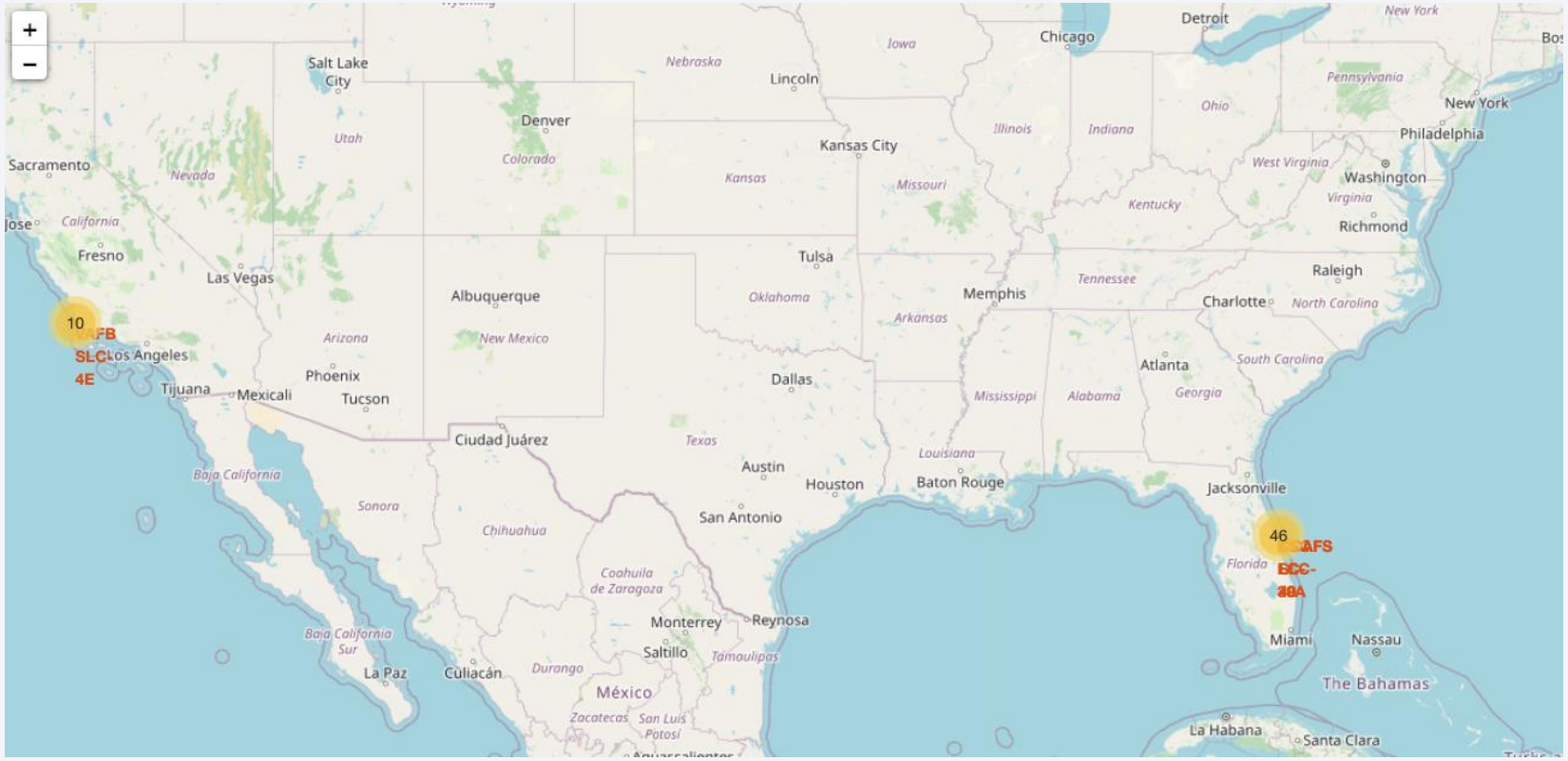
- COUNT() function used to calculate the number of records
- WHERE clause used to limit the result set to landings between 04-06-2010 and 20-03-2017
- GROUP BY clause used to organize the result set by landing outcome
- ORDER BY DESC clause used to display set in descending order of the number of landing outcomes


Section 3
Launch Sites
Proximities Analysis

Location of Launch Sites

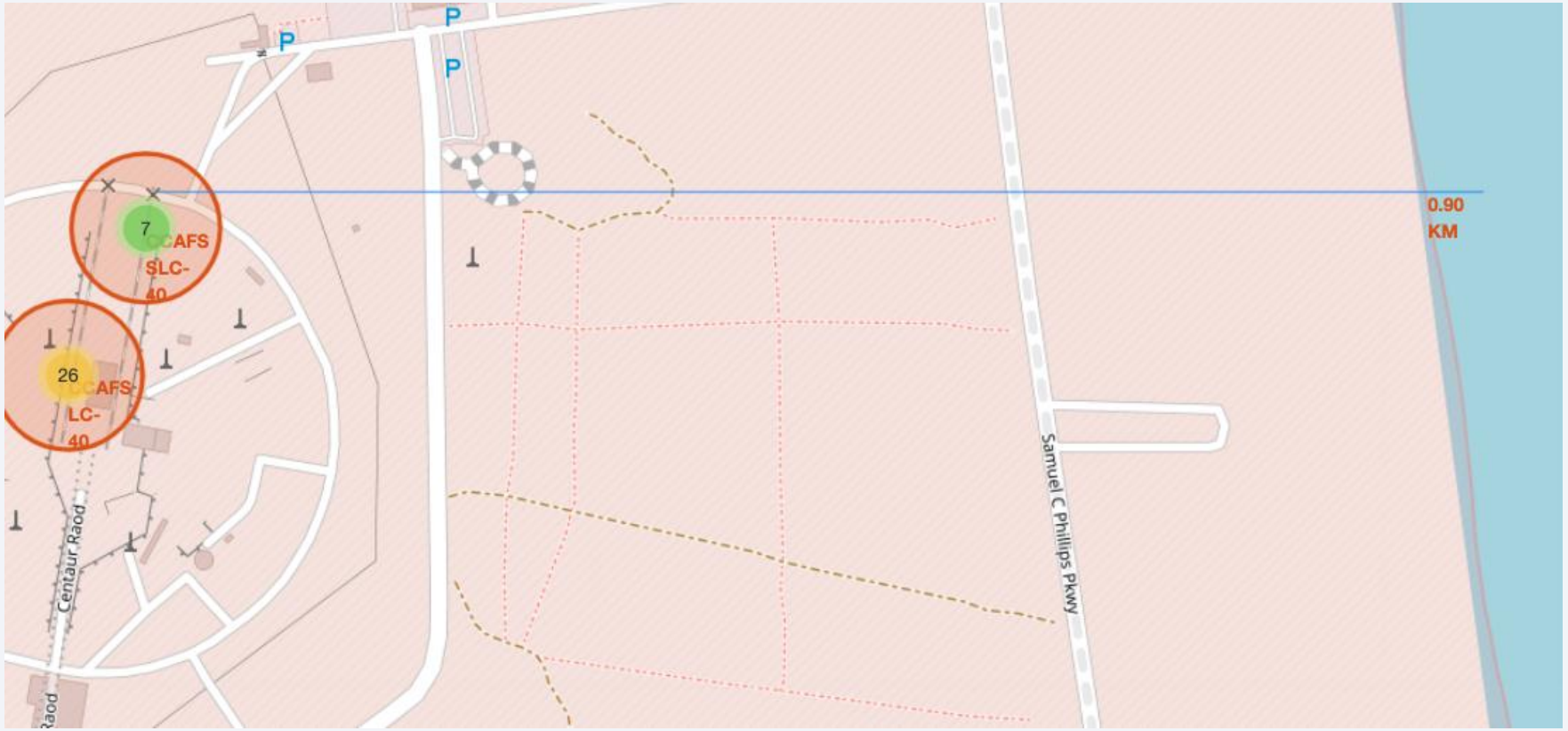
- It can be observed that all the launch sites are located near the eastern and western coasts of the United States, namely in the states of Florida and Los Angeles

Success/Failed Launches for Each Site

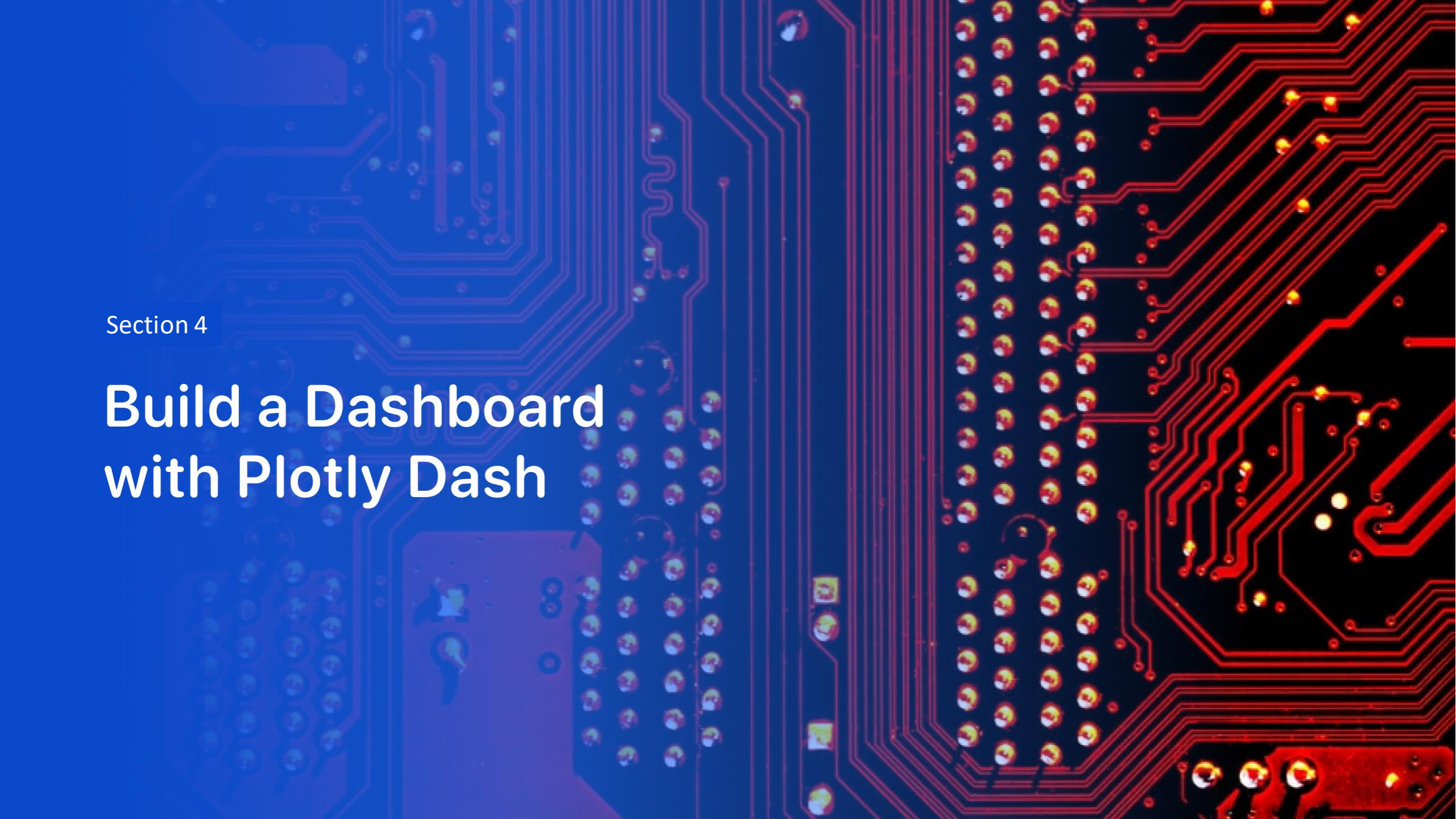


- This map shows the number of (clustered) success/failed launches for each site

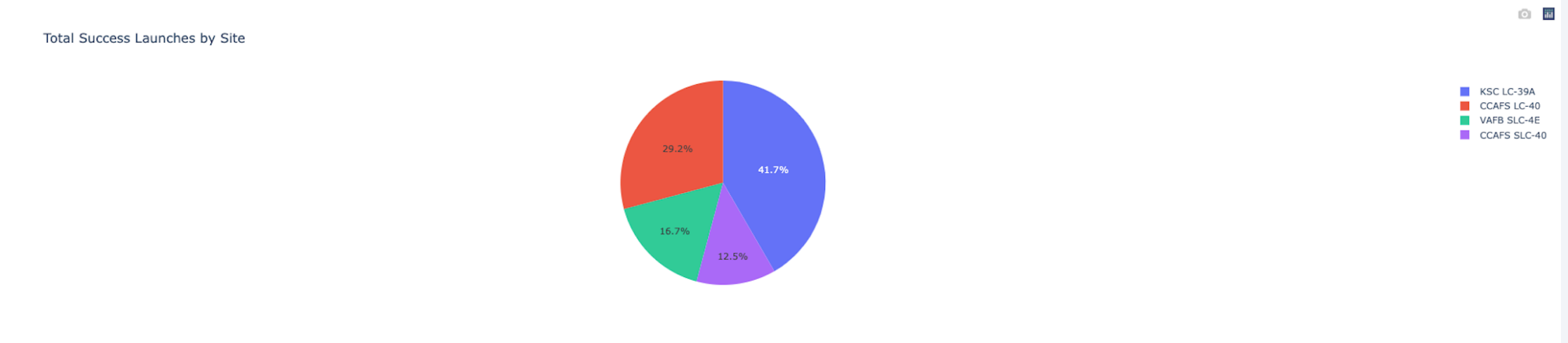
Distance from CCAFS SLC-40 to Coast



- This map shows that the distance between the site CCAFS SLC-40 and the nearest coastline point is 900 meters

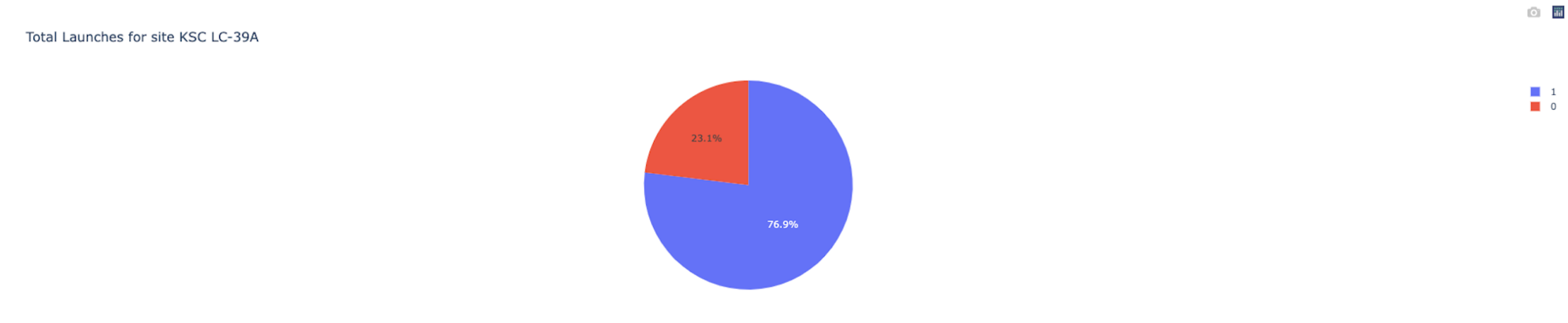

Section 4
Build a Dashboard with Plotly Dash

Launch Success Count by Site



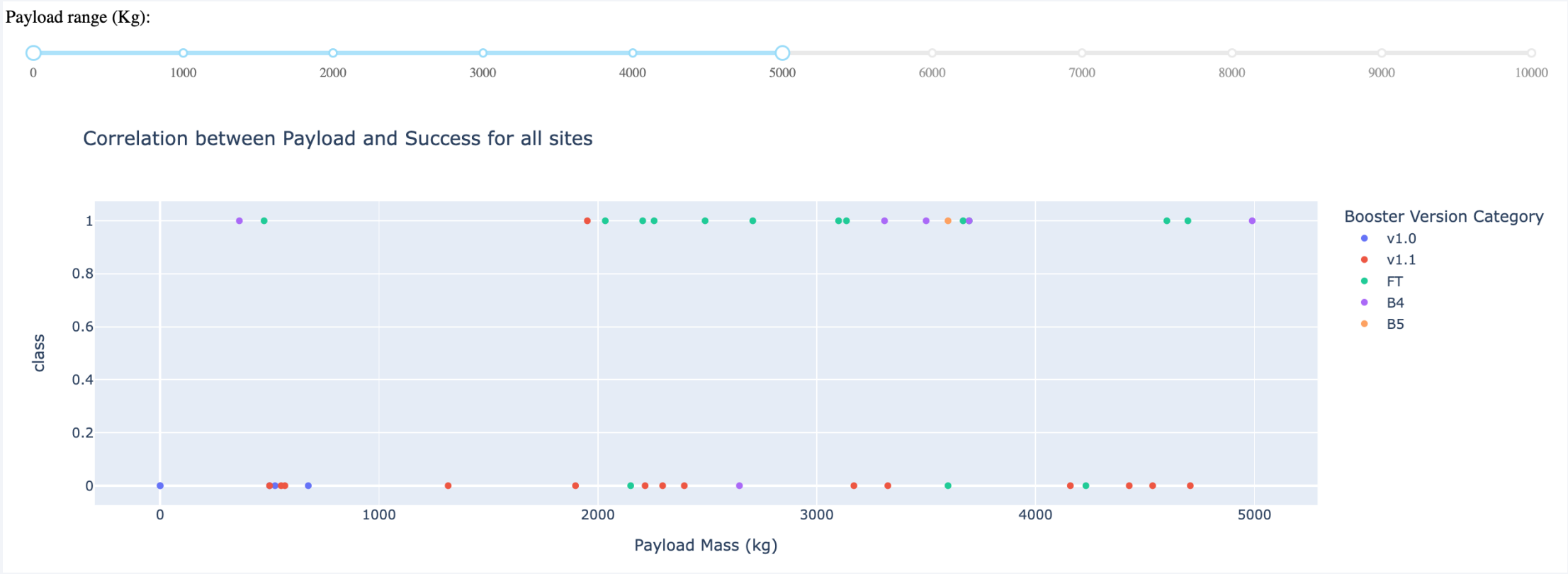
- It can be observed that most of the successful launches take place at KSC LC-39A (41.7%), followed by:
  - CCAFS LC-40 (29.2%)
  - VAFB SLC-4E (16.2%)
  - CCAFS SLC-40 (12.5%)

Launch Site with Highest Launch Success Ratio



- KSC LC-39A is the launch site with the highest launch success ratio (76.9%)

Payload vs. Launch Outcome (0 kg – 5,000 kg)



- Launch Success Rate by Booster Version for the Payload Range: 0 kg – 5,000 kg

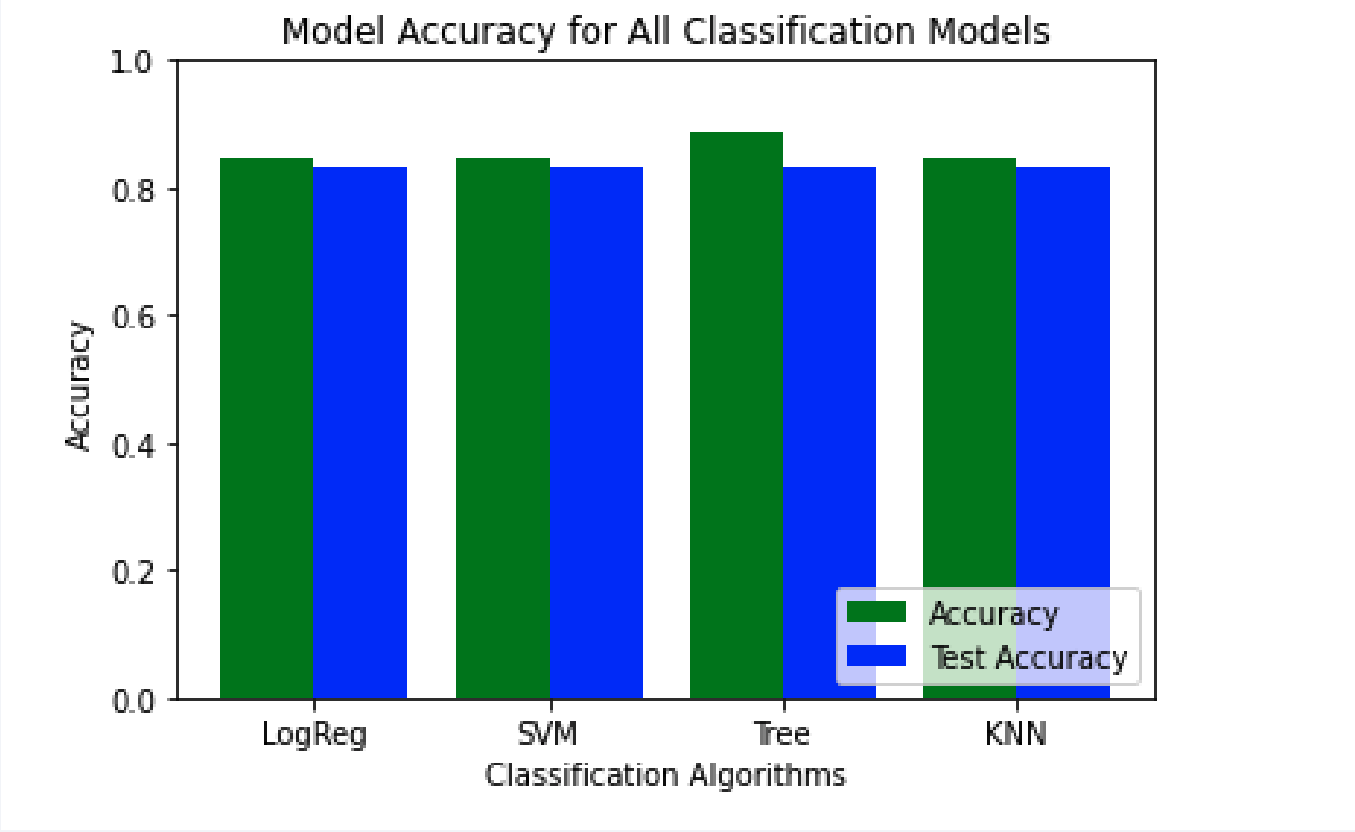
Payload vs. Launch Outcome (5,000 kg – 10,000 kg)

- Launch Success Rate by Booster Version for the Payload Range: 5,000 kg – 10,000 kg

Section 5

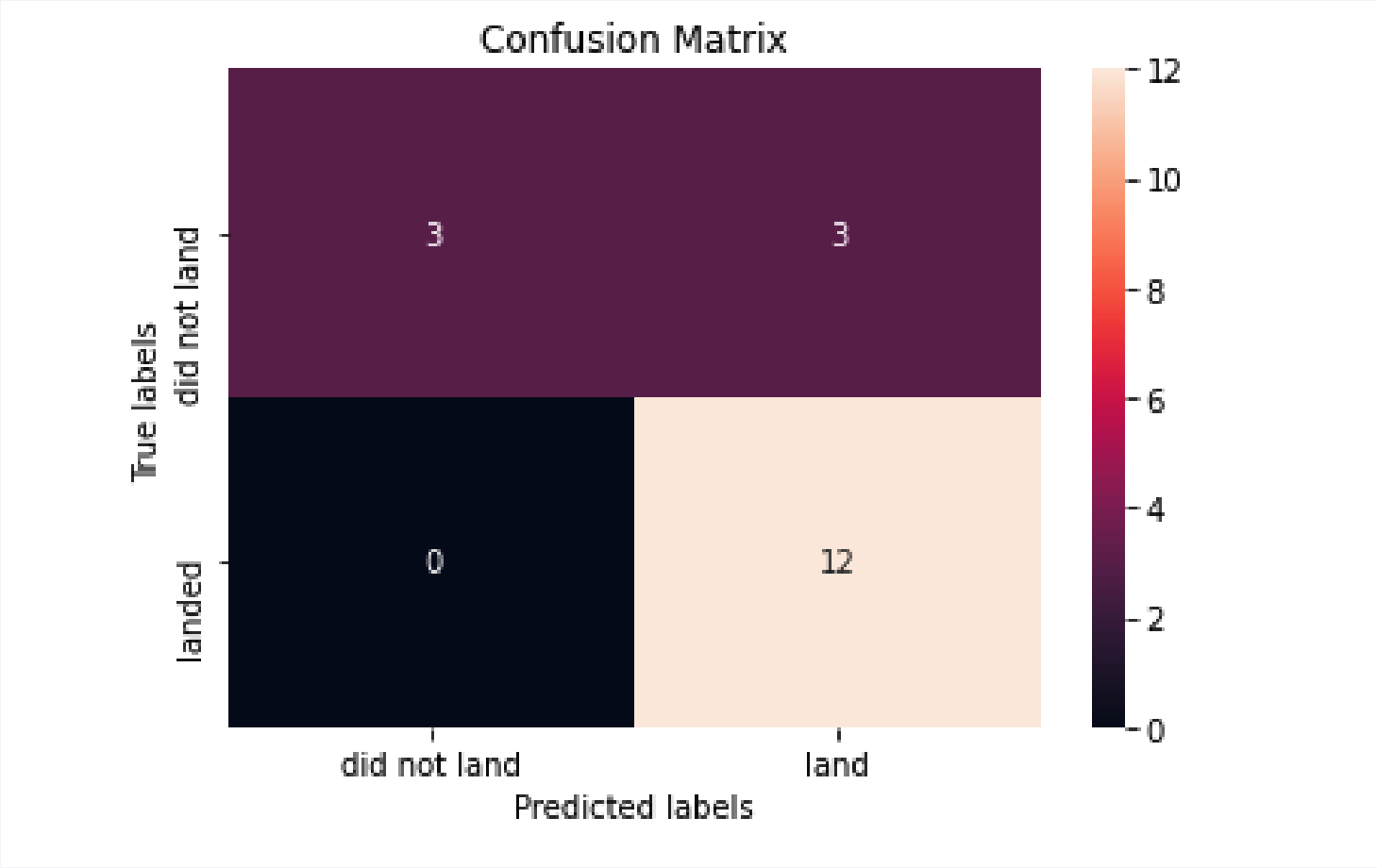
Predictive Analysis (Classification)

Classification Accuracy



- Decision Trees are the best, most accurate, classification model for this problem, having the highest accuracy

Confusion Matrix



- 15/18 (83.3%) landing outcomes were predicted correctly with 12 true positives and 3 true negatives
- Only 3 landing outcomes were not predicted correctly, and all were false positives

Conclusions

- Launching success rate is positively affected by increasing both flight Number and payload mass
- Orbit types with the highest success rate (1.0): SSO, HEO, GEO, ES-L1
- Yearly launch success rates have been trending upwards since 2013
- All launch sites are located near the eastern and western coasts of the United States, namely in the states of Florida and Los Angeles
- Most of successful launches take place at KSC LC-39A (41.7%) and the site has a 76.9% launch success ratio
- Decision Tree is the most accurate classification algorithm for this problem

Appendix

- Every detail, image, and code in this presentation can be found in the following link:
  - https://github.com/smart7294/IBM-Data-Science

Thank you!