IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Anjali Singh
February 6, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection with various means including API, Databases & Web-scraping.
  - Data Wrangling
  - Interactive maps using Folium to visualize the problem
  - Predictive modeling
- Summary of all results
  - Data analysis & interactive visualization
  - Identification of the best model

Introduction

- Background
  - SpaceX advertises launch services starting at $62 million for missions that allow some fuel to be reserved for landing the 1st stage rocket booster, so that it can be reused.
  - SpaceX public statements indicate a 1st stage Falcon 9 booster to cost upwards of $15 million to build without including R&D cost recoupment or profit margin.
  - Space X has the best pricing ($62 million vs. $165 million USD of competitors) largely due to its ability to recover part of the rocket (Stage 1)
  - SpaceY is a new commercial rocket launch provider who wants to go head-to-head with SpaceX.
- Problem
  - Space Y requires us to train a machine learning model to predict successful Stage 1 recovery.
  - As a result, SpaceY will be able to make more informed bids against SpaceX by using 1st stage landing predictions as a proxy for the cost of a launch.

Section 1

Methodology

Methodology

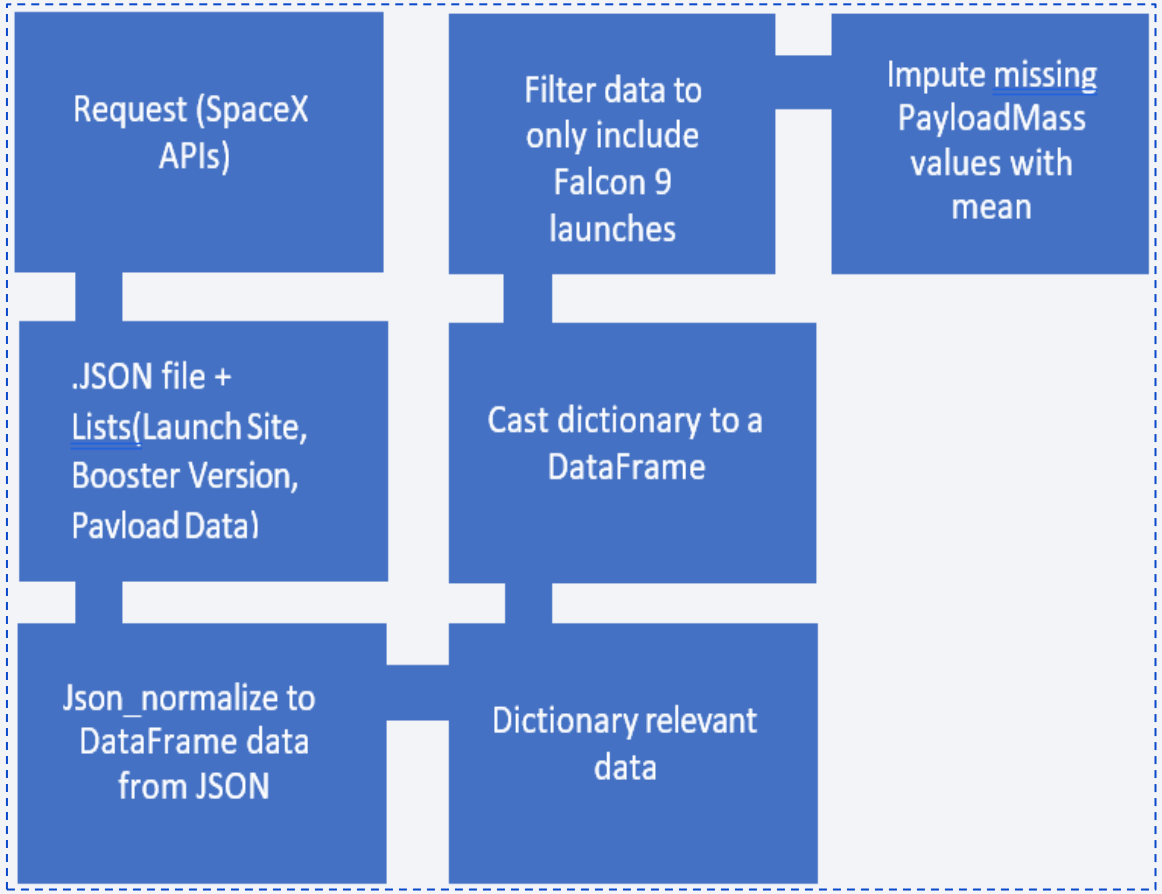
- Data collection methodology:
  - Data was collected from the SpaceX Public API and SpaceX Wikipedia page.
- Perform data wrangling
  - True landings were classified as successful, otherwise, unsuccessful landings.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree Classifier
  - K Nearest Neighbors Classifier
  - Tuned models using GridSearchCV.

Data Collection

- The data collection process involved a combination of API requests from Space X public API and web scraping data from a table in Space X's Wikipedia entry.
- The next slide will show the flowchart of data collection from API and the one after will show the flowchart of data collection from webscraping.
- Space X API Data Columns:
- FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins,
- Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude Wikipedia Webscrape Data Columns:
- Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time

Data Collection – SpaceX API

- Github URL:
  - https://github.com/anjaliprograms/IBM-Applied-Data-Science-Capstone/blob/main/Week%201%20(i)%20Data%20Collection%20using%20API%20Lab.ipynb



Data Collection – SpaceX API

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

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

```
# Use json_normalize meethod to convert the json result into a dataframe
jlist = requests.get(static_json_url).json()
df2 = pd.json_normalize(jlist)
df2.head()
```

```
# Hint data['BoosterVersion']!='Falcon 1'

data_falcon9.drop(data_falcon9[data_falcon9['BoosterVersion']!='Falcon 9'].index, inplace = True)
```

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

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

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	Re
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the data with the mean you calculated.

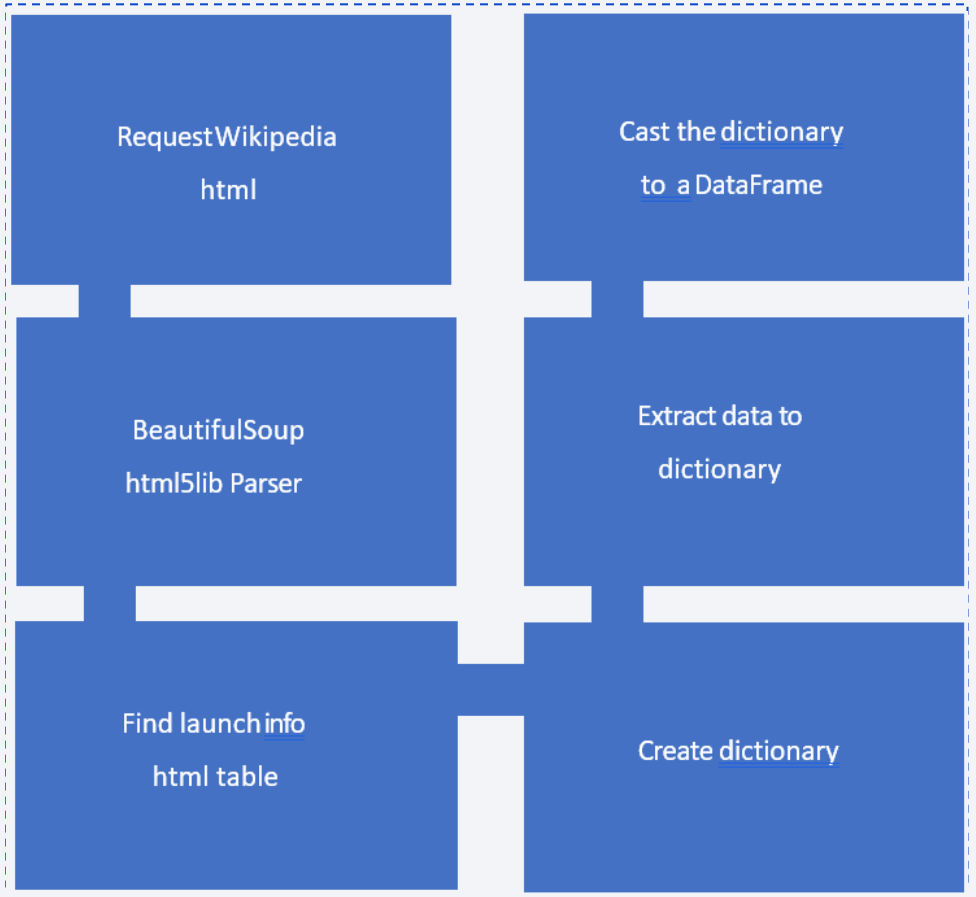
```
# Calculate the mean value of PayloadMass column
avg_payload_mass = data_falcon9["PayloadMass"].astype("float").mean(axis=0)
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, avg_payload_mass, inplace=True)
```

You should see the number of missing values of the `PayLoadMass` change to zero.

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

Data Collection - Scraping

- Github URL:
  - https://github.com/anjaliprograms/IBM-Applied-Data-Science-Capstone/blob/main/Week%201%20(iii)%20Data%20Collection%20with%20Web%20Scraping%20lab.ipynb



Data Collection - Scraping

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
data  = requests.get(static_url).text
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text con
soup = BeautifulSoup(data, 'html5lib')
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
df=pd.DataFrame(launch_dict)
df.head()
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, ple
the end of this lab

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables=soup.find_all("table")
html_tables
```

```
[<table class="multicol" role="presentation" style="border-collapse: collapse; padding: 0; border: 0; background:transparent;
width:100%;">
```

```
for th in ths:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Removing \n from all columns ¶

```
df = df.replace('\n','', regex=True)
df.head()
```

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43

Data Wrangling

- Create a training label with landing outcomes where successful = 1 and failure = 0. Outcome column has two components: ‘Mission Outcome’ and ‘Landing Location’
- New training label column ‘class’ with a value of 1 if ‘Mission Outcome’ is True and 0 otherwise.
- Value Mapping:
  - True ASDS, True RTLS, & True Ocean -> 1
  - None None, False ASDS, None ASDS, False Ocean, False RTLS -> 0
- Github URL:
  - https://github.com/anjaliprograms/IBM-Applied-Data-Science-Capstone/blob/main/Week%201%20(ii)%20Data%20Wrangling%20-%20EDA%20lab.ipynb

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

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

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

```
True ASDS       41
None None       19
True RTLS       14
False ASDS       6
```

```
#df['Class']=landing_class

# for outcome in df['Outcome']:
#     if outcome in bad_outcomes:
#         landing_class.append(0)
#     else:
#         landing_class.append(1)

df['Class'] = df['Outcome'].apply(lambda landing_class: 0 if landing_class in bad_outcomes else 1)

df[['Class']].head(8)
```

EDA with Data Visualization

- Exploratory Data Analysis performed on variables Flight Number, Payload Mass, Launch Site, Orbit, Class and Year.
- Plots Used:
- Flight Number vs. Payload Mass, Flight Number vs. Launch Site, Payload Mass vs. Launch Site, Orbit vs. Success Rate, Flight Number vs. Orbit, Payload vs Orbit, and Success Yearly Trend.
- Scatter plots, line charts, and bar plots were used to compare relationships between variables to decide if a relationship exists so that they could be used in training the machine learning model
- GitHub URL:
  - https://github.com/anjaliprograms/IBM-Applied-Data-Science-Capstone/blob/main/Week%202%20(ii)%20EDA%20with%20Visualization%20lab.ipynb

EDA with SQL

- Loaded data set into IBM DB2 Database. Queried using SQL Alchemy.
- Queries were used to get a better understanding of the dataset.
- Queried information about the following:
  - Launch sites
  - Payload masses
  - Booster versions
  - Mission outcomes
  - Booster landings
- Github URL:
  - https://github.com/anjaliprograms/IBM-Applied-Data-Science-Capstone/blob/main/Week%202%20(i)%20Exploratory%20Data%20Analysis%20Using%20SQL.ipynb

Build an Interactive Map with Folium

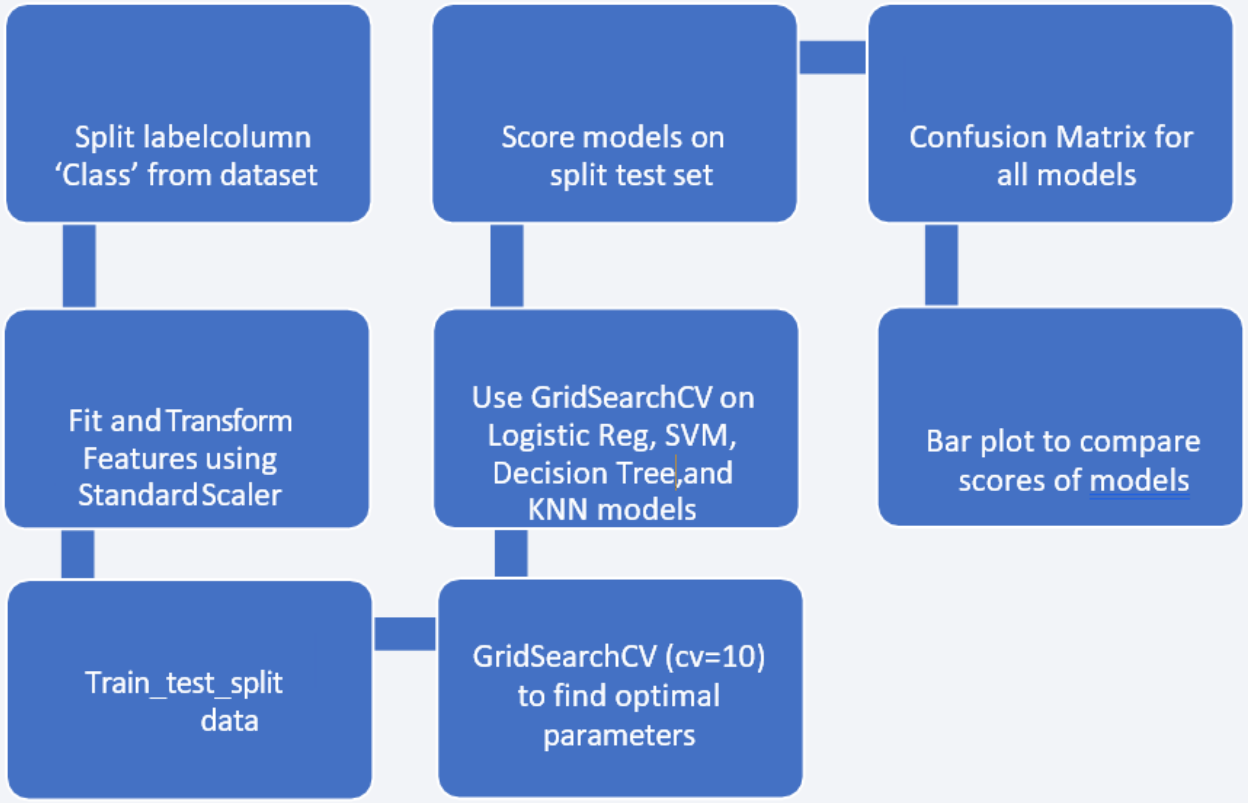
- Launch Sites Location Analysis
- Used Python interactive mapping library called Folium
- Marked all launch sites on a map
- Marked the successful/failed launches for each site on map
- Calculated the distances between a launch site to its proximities
  - Railways
  - Highways
  - Coastlines
  - Cities
- Github URL:
  - https://github.com/anjaliprograms/IBM-Applied-Data-Science-Capstone/blob/main/Week%203%20(i)%20Interactive%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- The dashboard includes a pie chart and a scatter plot.
- Pie chart can be selected to show distribution of successful landings across all launch sites and can be selected to show individual launch site success rates.
- Scatter plot receives two inputs: All sites or individual site and payload mass between 0 and 10000 kg. The pie chart is used to visualize launch site success rate.
- The scatter plot can help us see how success varies across launch sites, payload mass, and booster version category.
- Github URL:
  - https://github.com/anjaliprograms/IBM-Applied-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Github URL:
  - https://github.com/anjalipro grams/IBM-Applied-Data-Science-Capstone/blob/main/Week%204%20(i)%20Machine%20Learning%20Prediction%20Lab.ipynb



Predictive Analysis (Classification)

- The training data was standardized & split into training and testing sets.
- Data was fit using the following models:
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree Classifier
  - K Nearest Neighbors Classifier
- A cross-validated grid-search over a variety of hyperparameters was performed to select the best ones for each model.
- Evaluated accuracy of each model using test data to select the best model.

```
y = data['Class'].to_numpy()
```

```
# students get this
transform = preprocessing.StandardScaler()
```

```
X = transform.fit(X).transform(X)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

```
Y_test.shape
```

```
(18,)
```

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

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

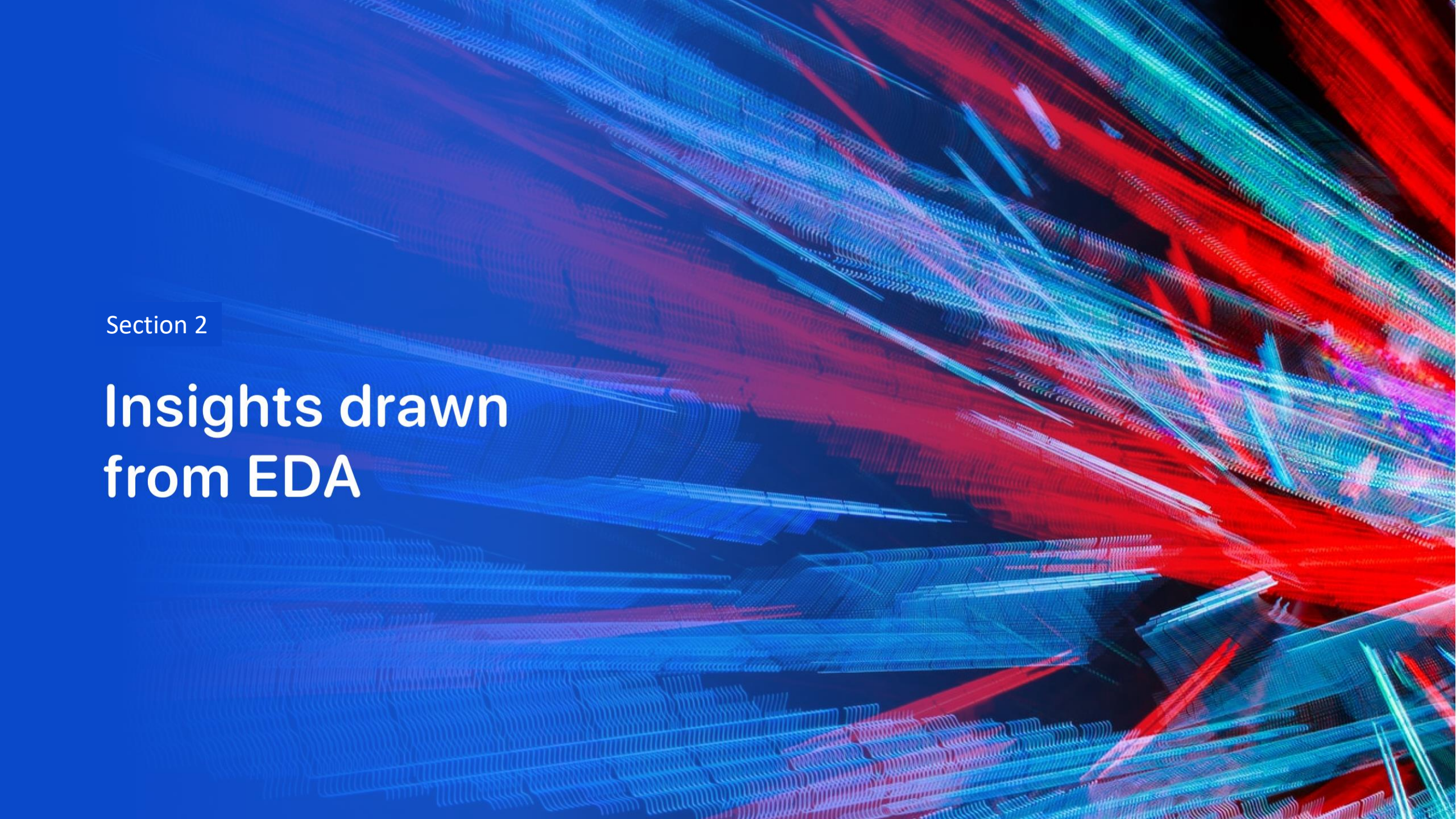
```
algo_df.sort_values(['Accuracy'], inplace=True)
```

```
algo_df.head()
```

	Accuracy
Logistic Regression	0.846429
SVM	0.848214
KNN	0.848214
Decision Tree	0.889286

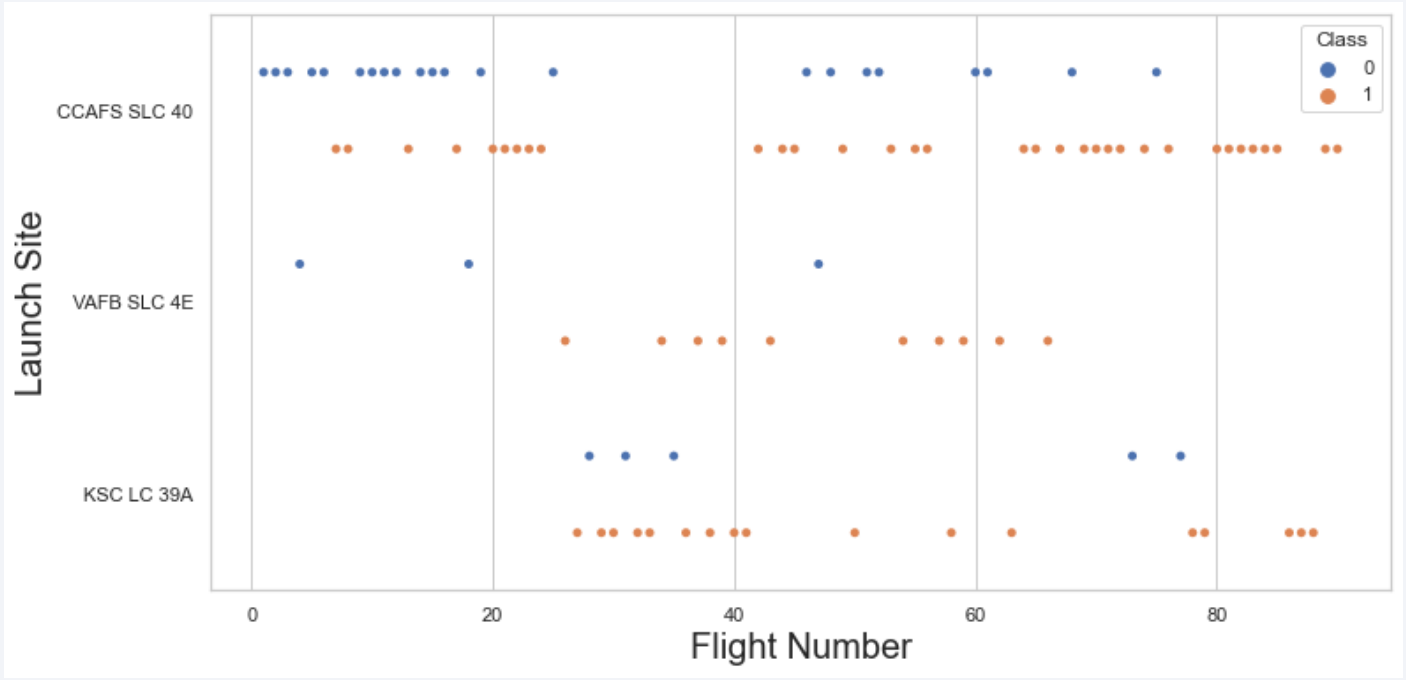
Results

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


Section 2
Insights drawn from EDA

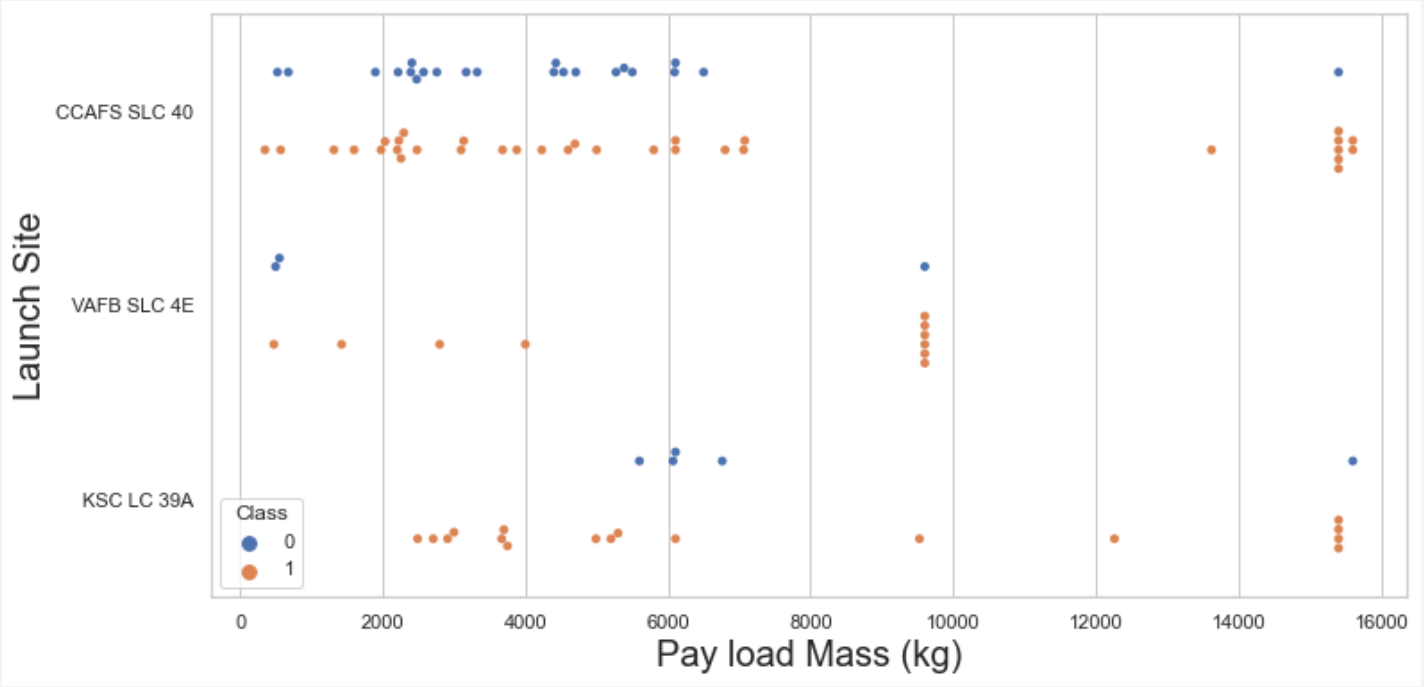
Flight Number vs. Launch Site

- Orange indicates successful launch; Blue indicates unsuccessful launch.
- With more flight numbers (after 40) higher the success rate for the launch.
- CCAFS SLC 40 appears to have been where most of the early 1st stage landing failures took place
- There is no definitive pattern to identify the success of a launch.



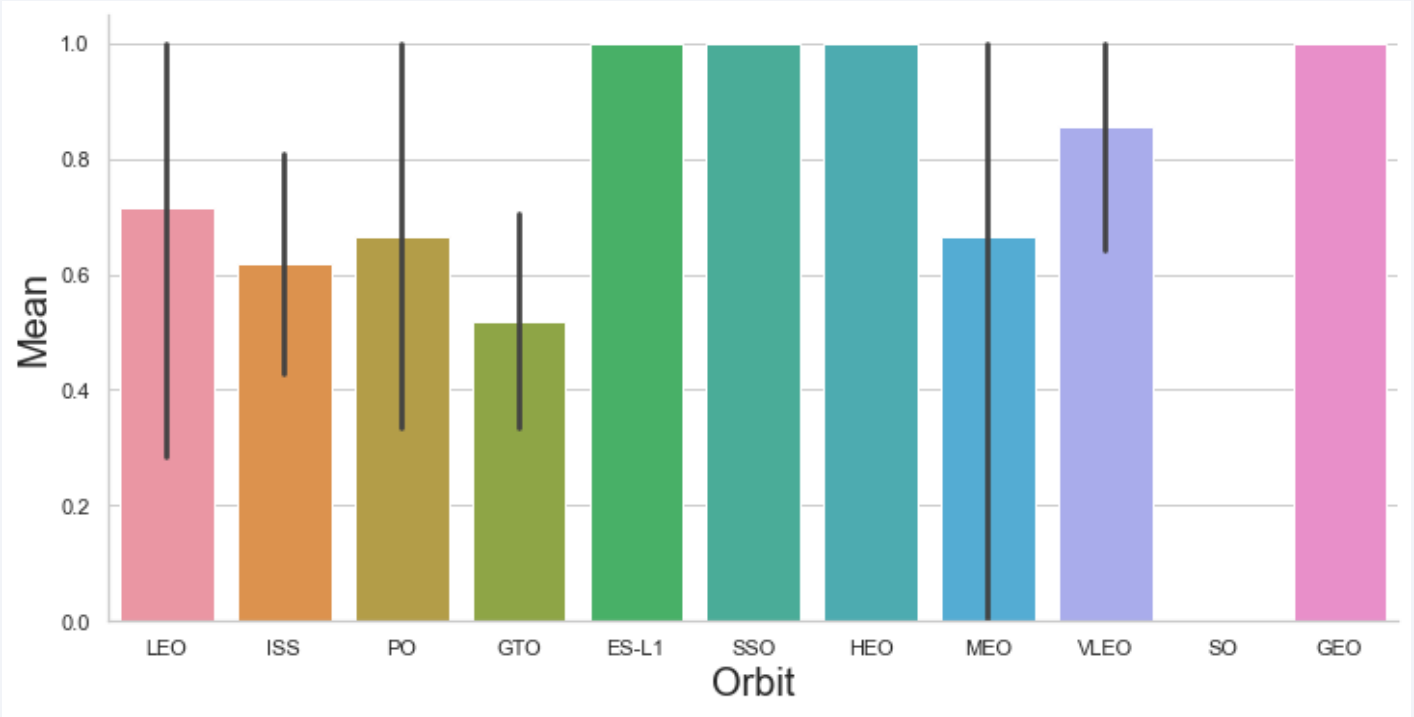
Payload vs. Launch Site

- Orange indicates successful launch; Blue indicates unsuccessful launch.
- CCAFS SLC 40 and KSC LC 39A appear to be favored for heavier payloads
- The greater the payload mass (greater than 8000) higher the success rate for the Rocket.



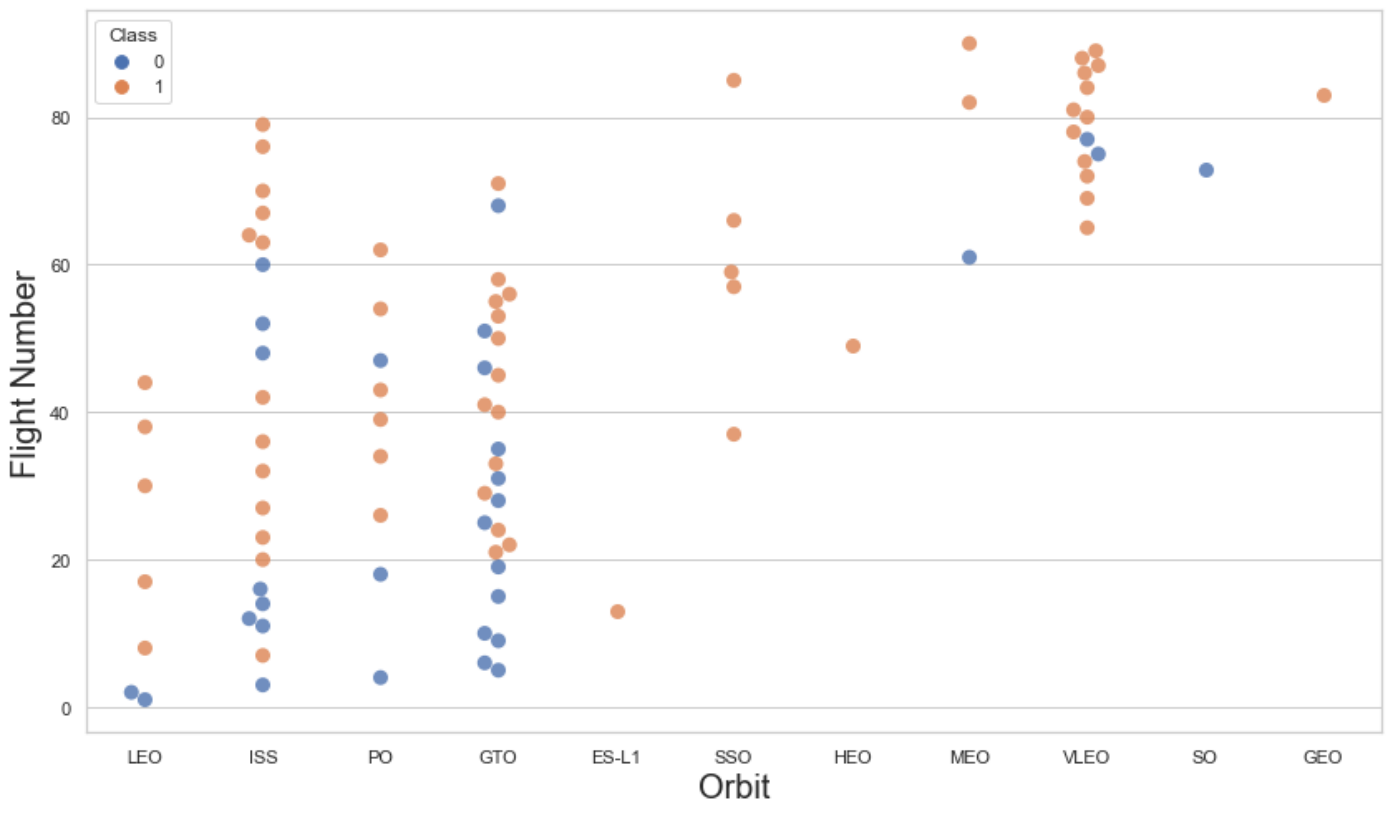
Success Rate vs. Orbit Type

- ES-L1, GEO, SSO and HEO have 100% success rate;
- VLEO has decent success rate (around 80%) and attempts;
- SO has 0% success rate;
- GTO has the around 50% success rate, but it also has the highest amount of samples.



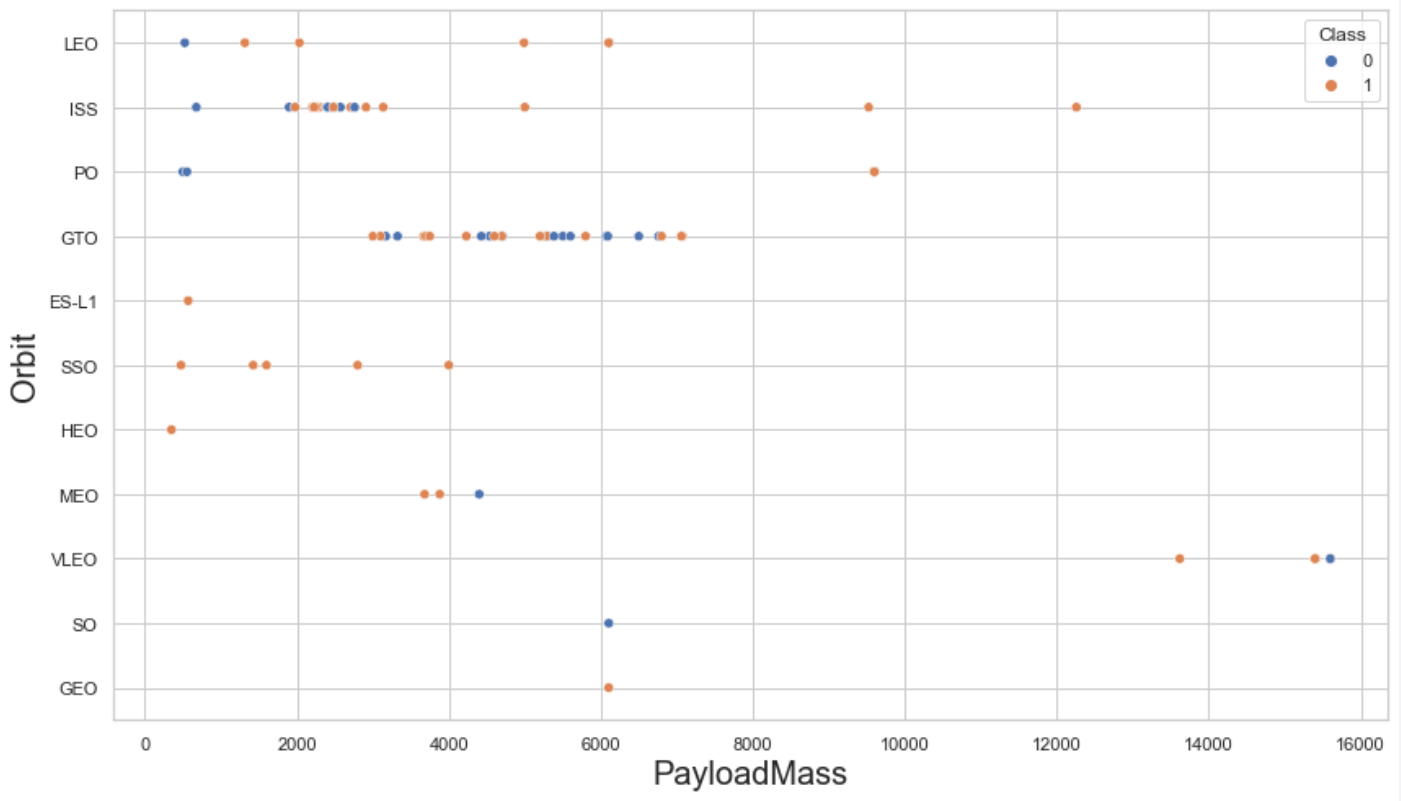
Flight Number vs. Orbit Type

- Orange indicates successful launch; Blue indicates unsuccessful launch.
- We see that in the LEO orbit the success rate appears related to the flight number;
- On the other extreme, there seems to be no relationship between flight number with the GTO orbit;
- Launch Orbit preferences changed over Flight Number;
- SpaceX appears to perform better in lower orbits or Sun-Synchronous-Orbits.



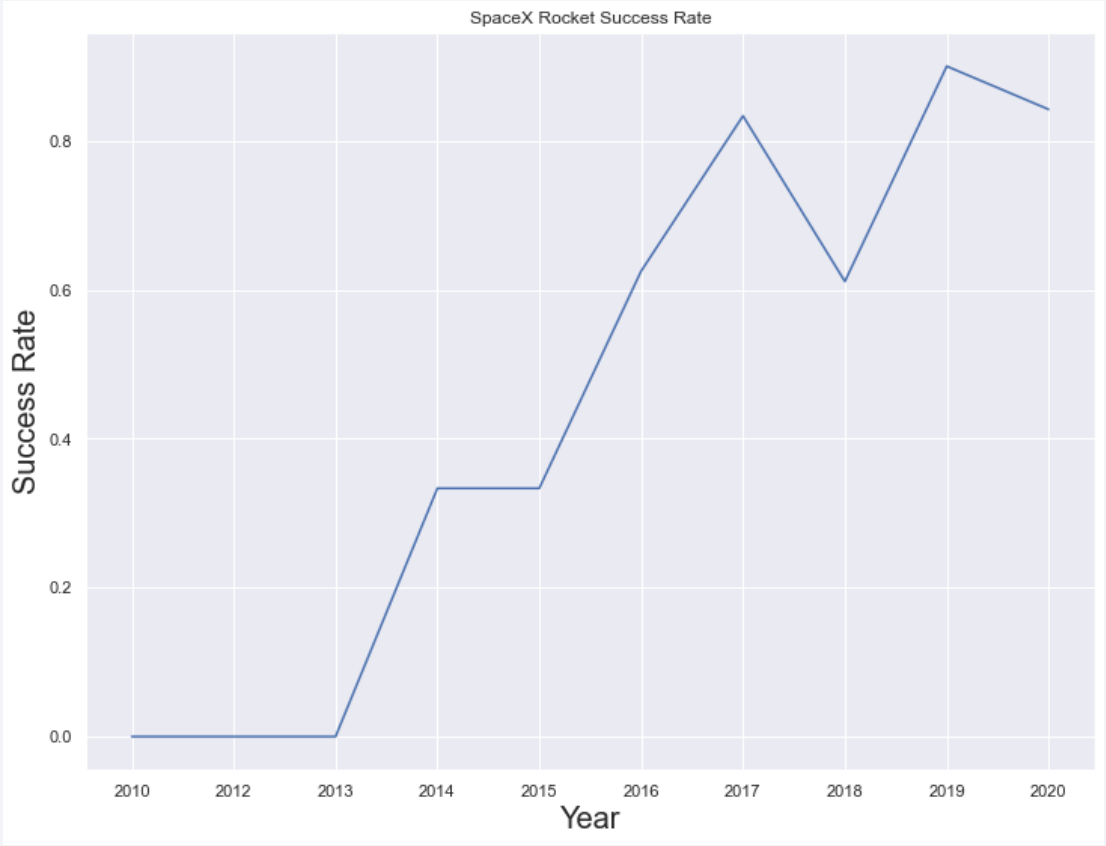
Payload vs. Orbit Type

- Orange indicates successful launch; Blue indicates unsuccessful launch;
- Payload mass seems to correlate with orbit;
- LEO and SSO seem to have relatively low payload mass;
- The other most successful orbit VLEO only has payload mass values in the higher end of the range.



Launch Success Yearly Trend

- The launch success rate has vastly increased over the years.
- Since 2013 there has been a steady increase except for dips around 2017 and 2019.



All Launch Site Names

- CCAFS SLC-40 and CCAFSSLC-40 represent the same launch site, but the data provided had errors;
- CCAFS LC-40 was the previous name.
- The unique names are:
  - CCAFS SLC-40, KSC LC-39A, VAFB SLC-4E.

```
In [4]: %%sql
        SELECT UNIQUE LAUNCH_SITE
        FROM SPACEXDATASET;
```

Out[4]:

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

Launch Site Names Begin with 'CCA'

- First five rows in the database with Launch Site name beginning with CCA;
- This is a standard SQL query with WHERE clause and LIKE.

In [5]:
```
%%sql
SELECT *
FROM SPACEXDATASET
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

Out[5]:

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

Total Payload Mass

- This query sums the total payload mass in kg where NASA was the customer.
- We use SUM as VAR for all data where the customer is NASA (CRS).

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS SUM_PAYLOAD_MASS_KG
FROM SPACEXDATASET
WHERE CUSTOMER = 'NASA (CRS)';
```

sum_payload_mass_kg
45596

Average Payload Mass by F9 v1.1

- This query calculates the average payload mass for the flights which used booster version F9 v1.1
- We use AVG as VAR for all data where the booster_version is F9 v1.1.

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_MASS_KG
FROM SPACEXDATASET
WHERE booster_version = 'F9 v1.1'
```

avg_payload_mass_kg
2928

First Successful Ground Landing Date

- This query returns the first successful ground pad landing date.

```
%%sql
SELECT MIN(DATE) AS FIRST_SUCCESS
FROM SPACEXDATASET
WHERE landing__outcome = 'Success (ground pad)';
```

first_success
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- This query returns the four booster versions that had successful drone ship landings and a payload mass between 4000 and 6000.

```
%%sql
SELECT booster_version
FROM SPACEXDATASET
WHERE landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ BETWEEN 4001 AND 5999;
```

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

Total Number of Successful and Failure Mission Outcomes

- This query returns a count of each mission outcome.
- SpaceX appears to achieve a successful outcome 99% of the times.
- This means that most of the landing failures were intended.

```
%%sql
SELECT mission_outcome, COUNT(*) AS no_outcome
FROM SPACEXDATASET
GROUP BY mission_outcome;
```

mission_outcome	no_outcome
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- This query returns the booster versions that had the highest payload mass of 15600 kg.
- The query determines the maximum payload and lists the booster versions which carried it.

```
%%sql
SELECT booster_version, PAYLOAD_MASS__KG_
FROM SPACEXDATASET
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXDATASET);
```

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- This query returns the Month, Landing Outcome, Booster Version, Payload Mass (kg), and Launch site of 2015 launches where stage 1 failed to land on a drone ship.

```
%%sql
SELECT MONTHNAME(DATE) AS MONTH, landing__outcome, booster_version, PAYLOAD_MASS__KG_, launch_site
FROM SPACEXDATASET
WHERE landing__outcome = 'Failure (drone ship)' AND YEAR(DATE) = 2015;
```

MONTH	landing__outcome	booster_version	payload_mass__kg_	launch_site
January	Failure (drone ship)	F9 v1.1 B1012	2395	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	1898	CCAFS LC-40

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

- This query returns a list of successful landings and between 2010-06-04 and 2017-03-20.
- This is one of the more complex SQL queries we have used here. It selects based on a criteria on value and date, then groups the results and orders them as shown here.

```
%%sql
SELECT landing__outcome, COUNT(*) AS no_outcome
FROM SPACEXDATASET
WHERE landing__outcome LIKE 'Succes%' AND DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY landing__outcome
ORDER BY no_outcome DESC;
```

```
 * ibm_db_sa://
Done.
```

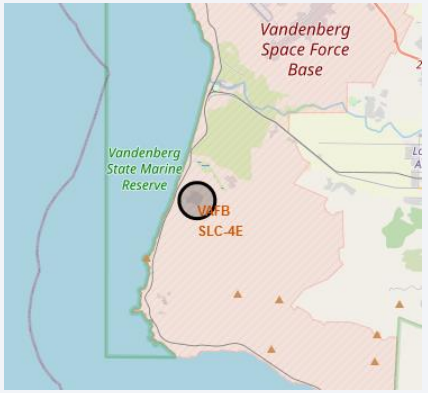
landing__outcome	no_outcome
Success (drone ship)	5
Success (ground pad)	3

Section 3

Launch Sites Proximities Analysis

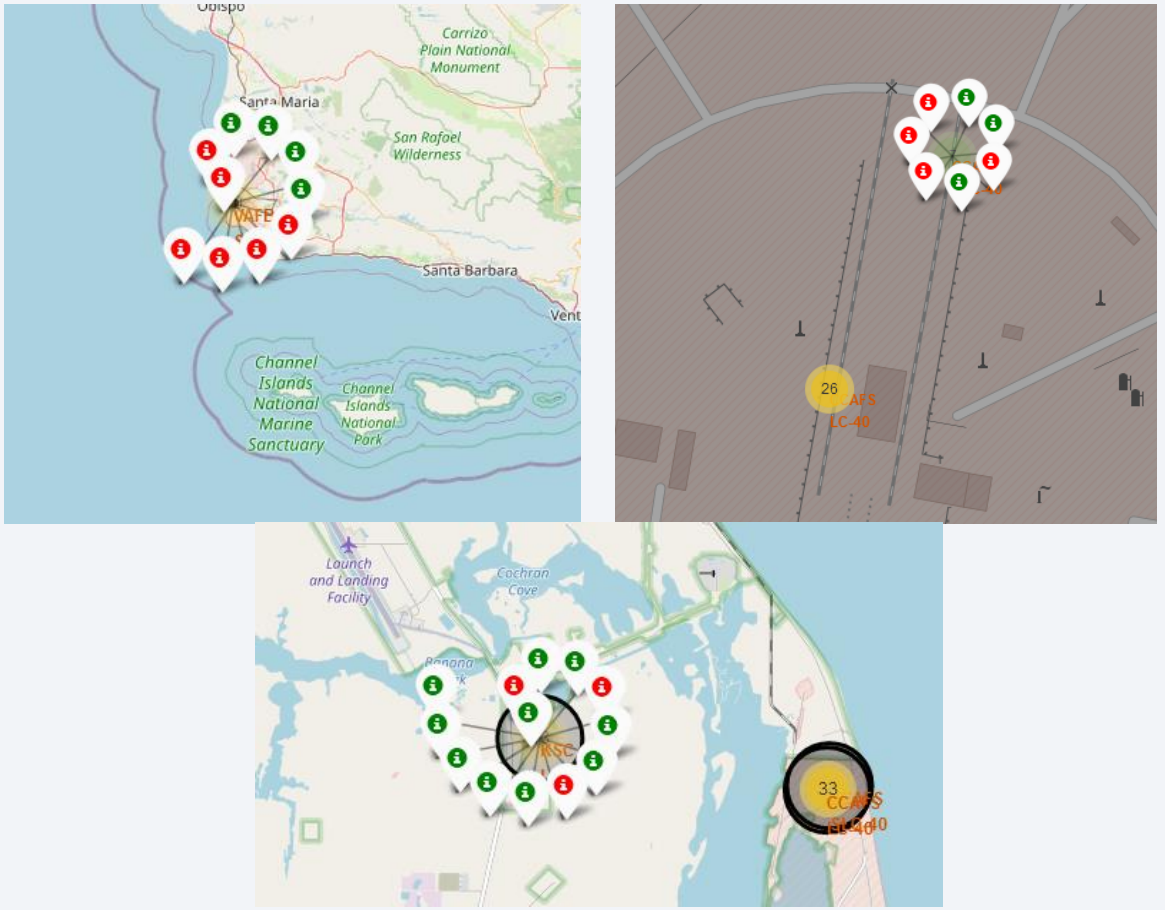
Folium: Launch Sites Locations

- This map shows all the launch sites in the USA map. All the launches appear to have happened next to the ocean.



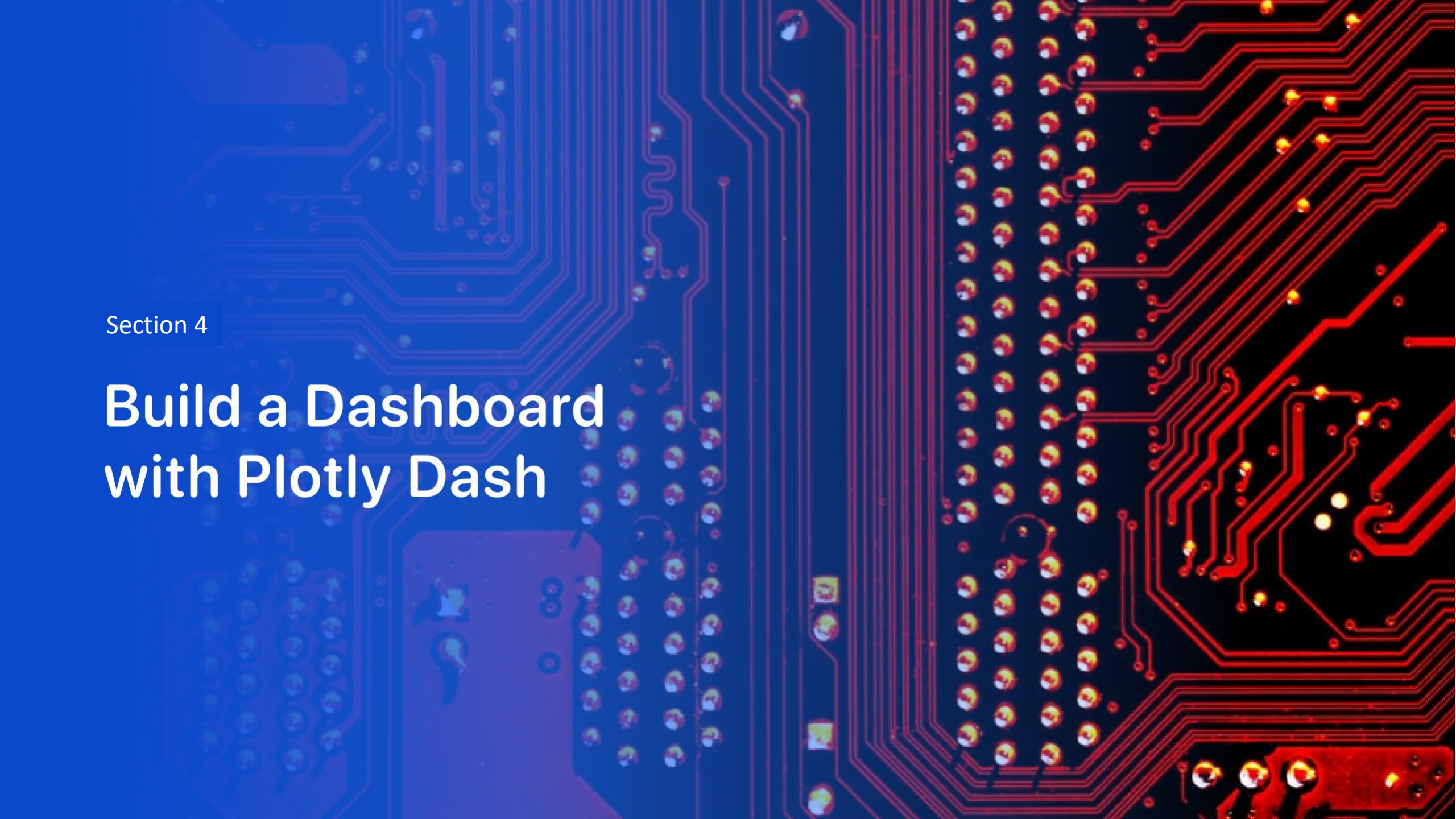
Folium: Clusters (Launch Markers)

- Clusters on Folium map can be clicked on to display each successful landing (green icon) and failed landing (red icon).



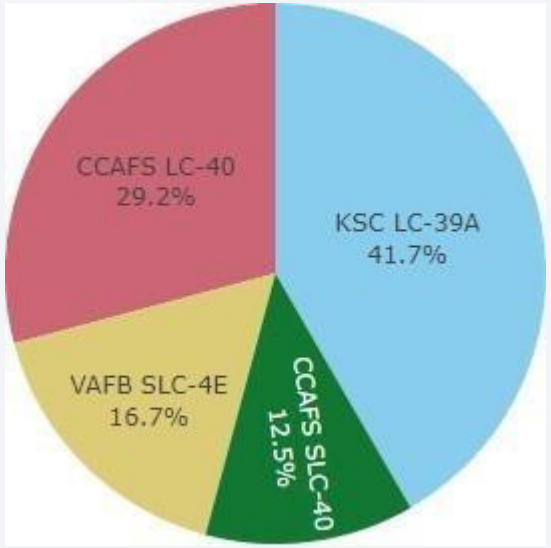
Folium: Launch site location analysis

- Visualizing the railway, highway, coastline, and city proximities for each launch site allows us to see how close each is, for example:
  - Proximities for CCAFS SLC-40:
  - railway: 1.28 km
  - transporting heavy cargo
  - highway: 0.58 km
  - transporting personel and equipment
  - coastline: 0.86 km
  - optionality to abort launch and attempt water landing
  - minimizing risk from falling debris
  - city: 51.43 km
  - minimizing danger to population dense areas.


Section 4
Build a Dashboard with Plotly Dash

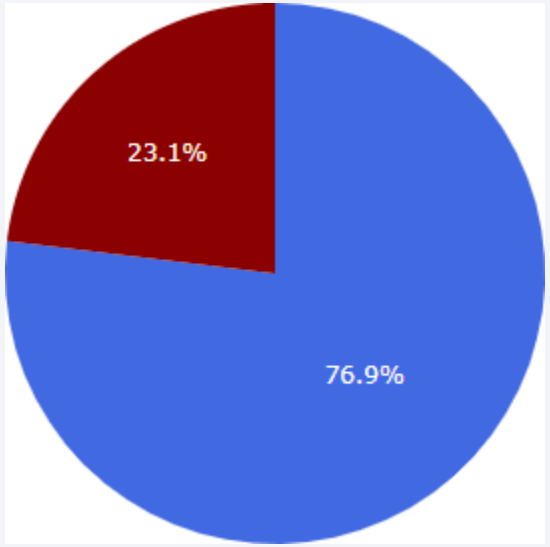
Dashboard: Successful Launches by Launch Sites

- This is the distribution of successful landings across all launch sites. CCAFS LC-40 is the old name of CCAFS SLC-40 so CCAFS and KSC have the same amount of successful landings, but the majority of the successful landings performed before the name change. VAFB has the smallest share of successful landings. This may be due to smaller sample and increase in difficulty of launching in the west coast.



Dashboard: Highest Success Rate

- KSC LC-39A has the highest success rate with 10 successful landings and 3 failures.
- Blue -> Success, Red -> Failure



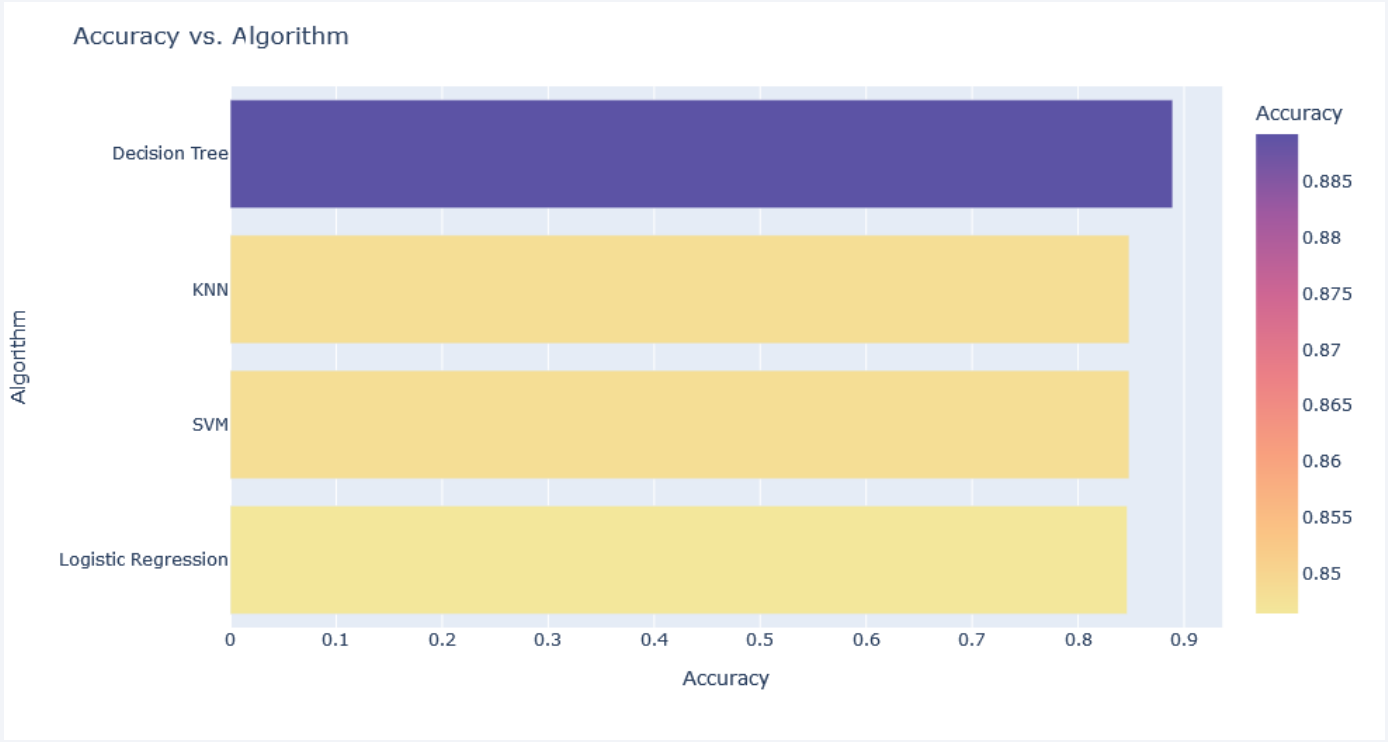
Section 5

Predictive Analysis (Classification)

Classification Accuracy

	Algorithm	Accuracy
0	Logistic Regression	0.846429
1	SVM	0.848214
2	KNN	0.848214
3	Decision Tree	0.889286

- The decision tree classifier achieves the maximum accuracy of nearly 90%
- The rest of the classifiers also perform well at around 0.84%



Confusion Matrix

- The model predicted 12 successful landings when the true label was successful landing.
- The models predicted 3 unsuccessful landings when the true label was unsuccessful landing.
- The models predicted 3 successful landings when the true label was unsuccessful landings (false positives). This model over predicts successful landings.

Conclusions

- Project deliverable: To develop a machine learning model for Space Y which would like to bid against SpaceX;
- The goal of model is to predict when Stage 1 will successfully land to save ~$100 million USD;
- Used data from a public SpaceX API and web scraping SpaceX Wikipedia page
- Created data labels and stored data into a DB2 SQL database;
- Created a dashboard for visualization;
- We created a machine learning model with an accuracy of 90%;
- Allon Mask of SpaceY can use this model to predict with relatively high accuracy whether a launch will have a successful Stage 1 landing before launch to determine whether the launch should be made or not;
- If possible, more data should be collected to better determine the best machine learning model and improve accuracy.

Appendix

- Github Repository for the project:
  - https://github.com/anjaliprograms/IBM-Applied-Data-Science-Capstone

Thank you!