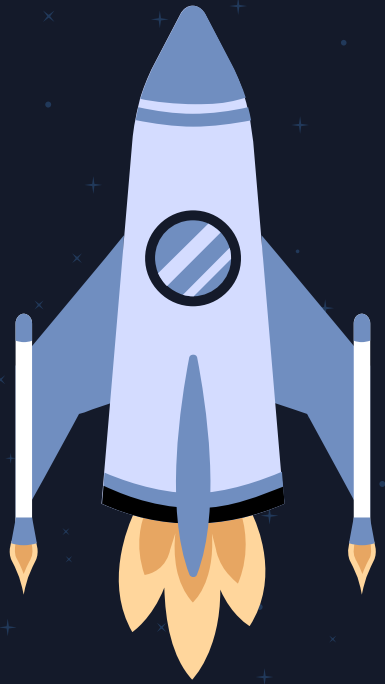
Winning the Space Race with Data Science

Savio | April 24th 2025

Outline



Executive Summary

This project uses open-source data and web scraping to analyze and model SpaceX Falcon 9 mission outcomes, focusing on predicting first-stage landing success. The aim is to assess SpaceX's reliability for future mission bidding.

Methodology:

- **Data Collection & Wrangling:** Launch data were gathered from public APIs and Wikipedia, then cleaned and structured for analysis.
- **Exploratory & Geospatial Analysis:** SQL and visualization tools revealed patterns; Folium maps highlighted geographic influences on outcomes.
- **Dashboard:** A Plotly Dash dashboard enabled interactive exploration of missions and key metrics.
- **Machine Learning:** A predictive model estimated landing success based on mission features (e.g., payload, launch site, orbit).

Key Insights:

- **Launch success correlated with factors like booster version, site, and orbit.**
- **The dashboard supported data-driven decision-making.**
- **The model showed high accuracy in predicting landings, affirming SpaceX's reliability.**

Introduction

Project background and context

- The space industry is becoming increasingly mainstream and accessible, driven by recent successes in private space travel.
- Despite innovation, high launch costs remain a major barrier for new competitors. Reusability of the first stage is key to this cost advantage, making SpaceX a dominant force in commercial launches.
- SpaceX revolutionized space travel with reusable Falcon 9 rockets, bringing launch costs down to ~$62 million versus $165M+ for competitors using expendable rockets.

Key Questions This Project Aims to Answer

- Can we predict the success of Falcon 9 first-stage landings?
- What is the impact of variables like launch site, payload mass, and booster version on landing outcomes?
- Are there significant correlations between launch sites and landing success rates?

01

Methodology

Data Collection

The data collection involved combining information from:

- **SpaceX Public API**
- **Web scraping a structured table from SpaceX's Wikipedia page**

SpaceX API – Key Data Columns

FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSiteOutcome, Flights, GridFins, Reused, Legs, LandingPadBlock, ReusedCount, Serial, Longitude, Latitude

Wikipedia Web Scrape – Key Data Columns

Flight No., Launch site, Payload, PayloadMass, Orbit, CustomerLaunch outcome, Version Booster, Booster landing, Date, Time

Data Collection – SpaceX API (Github Link)

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Ski
```

We should see that the request was successfull with the 200 status response code

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

```
# Use json_normalize meethod to convert the json result into a dataframe
resj = response.json()
data = pd.json_normalize(resj)
```

Request the data via an API request and store it in normalized JSON format.

The data from these requests will be stored in lists and will be used to create a new dataframe.

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

Create column bins for each data in the JSON value to then be used to construct a Dataframe

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
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}
```

Use the created columns to construct the Dataframe using the dictionary data structure

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data_falcon9[data_falcon9['BoosterVersion'] != 'Falcon 1']
```

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
```

Clear out all data that is not pertaining to the Falcon 9 data we are focusing on

Data Collection – Scraping (Github Link)

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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(page.content, "html.parser")
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Request the wiki URL via an and pass it through a BeautifulSoup html parser

```
# 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')
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<
br/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[note 8]</a></sup>
```

Extract all column names from the parsed html

```
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
tt = first_launch_table.find_all('th')
for i in tt:
    ii = extract_column_from_header(i)
    if ii is not None and len(ii) > 0:
        column_names.append(ii)
```

Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

Create column bins to store the column data

```
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']=[]
```

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

Use the column bins to create a dataframe that stores all the scraped information

Data Wrangling (Github Link)

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

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

Calculate the number of launches on each site

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

```
Orbit
GTO     27
ISS     21
VLEO    14
PO       9
LEO      7
SSO      5
MEO      3
HEO      1
ES-L1    1
SO       1
GEO      1
Name: count, dtype: int64
```

Calculate the number and occurrence of each orbit

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

Calculate the number and occurrence of mission outcome of the orbits

```
c = 0
landing_class = []
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
for i in df['Outcome']:
    if i in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
        c += 1
print(c)
```

```
60
```

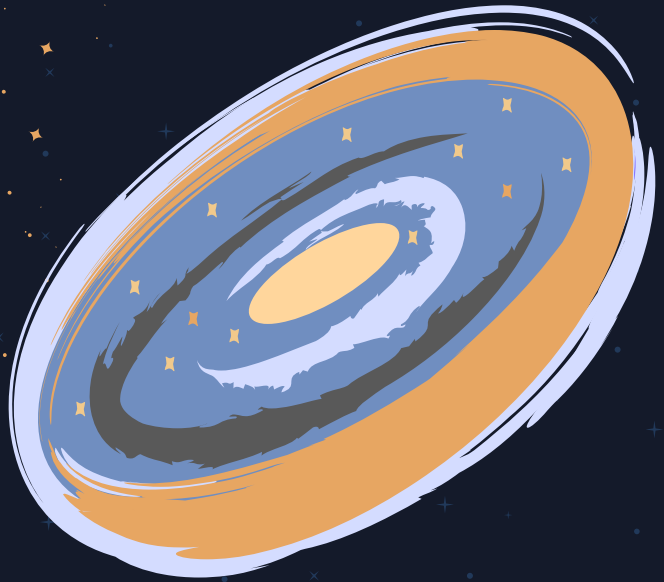
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

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

Create a landing outcome label from Outcome column

EDA with Data Visualization (Github Link)

To explore trends and relationships in the SpaceX launch data, we used various visualizations:



Scatter Plots (Seaborn catplot)

- ***Flight Number vs. PayloadMass (coloured by mission success)***
- ***Flight Number vs. Launch Site***
- ***Launch Site vs. Payload Mass***
- ***Flight Number vs. Orbit***
- ***Payload Mass vs. Orbit***

These charts helped us analyze how different mission parameters like flight sequence, payload, and launch site correlate with success rates.

Bar Plot (sns.barplot)

Orbit vs. Success Rate (Class)

- *Provided a clear comparison of mission success across different orbit types.*

Line Plot (sns.lineplot)

Date vs. Success

- *RateShowed trends over time in launch success, indicating improvements in mission reliability.*

EDA with SQL (Github Link)

These SQL queries helped explore mission metadata, identify trends, and prepare insights for deeper analysis:

Filtered Data for Analysis

- *Created a working table (SPACEXTABLE) excluding rows with missing dates.*

Explored Launch Sites

- *Queried all unique launch sites to understand location diversity.*

Drilled into Specific Launch Locations

- *Fetched missions launched from Cape Canaveral (CCA%) to analyse localized performance.*

Analysed Payload Trends

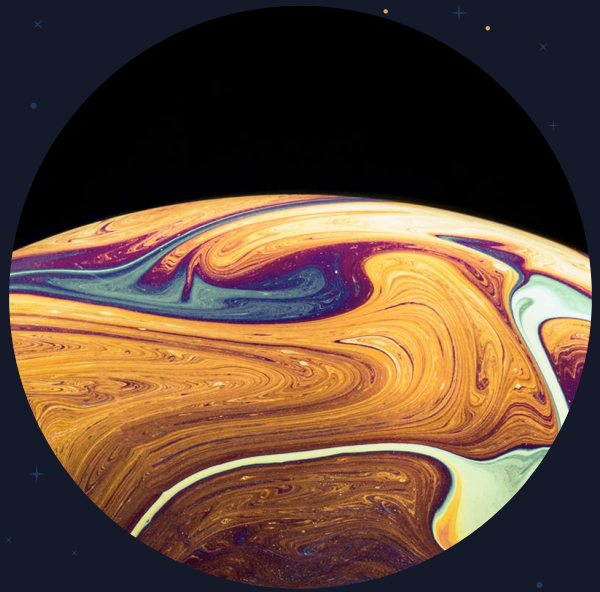
- *Calculated total payload mass for NASA (CRS) missions.*
- *Computed average payload mass for missions using the 'F9 v1.1' booster version.*

Evaluated Mission Outcomes by Date

- *Queried mission data to observe success rates over time (likely in other cells).*

Build an Interactive Map with Folium (Github Link)

Folium was used to create an interactive map to analyze geospatial data and understand factors influencing launch success rate.



Map Objects Added and Why:

- ***Markers***: *All launch sites were marked to visualize their locations.*
- ***Circles & Markers***: *folium.circle and folium.marker were used to highlight launch site areas with text labels.*
- ***Marker Cluster***: *A MarkerCluster() was added to differentiate launch success (green) and failure (red) at each site.*
- ***Proximity Analysis***: *Distances were calculated between each launch site and nearby geographical features (coastline, railroad, highway, city).*
- ***Distance Display***: *folium.Marker() was used to display these distances (in KM) on the map.*
- ***Proximity Lines***: *folium.Polyline() was used to draw lines connecting launch sites to their nearby coastline, railroad, highway, and city.*

Build a Dashboard with Plotly Dash (Github Link)

A Dashboard web application was built using Plotly to visualize and analyze SpaceX launch data in real-time.

Dashboard Components included:

- ***Launch Site Dropdown:*** *Filters dashboard visuals by launch site.*
- ***Pie Chart:*** *Displays total successful launches (all sites) or success/failure counts (selected site).*
- ***Payload Range Slider:*** *Filters data by payload mass range.*
- ***Scatter Chart:*** *Shows the correlation between payload mass and mission outcomes, colored by booster version.*

Predictive Analysis (Github Link)

```
URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv"
data = pd.read_csv(URL1)
```

```
URL2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv'
X = pd.read_csv(URL2)
```

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

Import the data, normalize it and define X and Y data

```
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()
logreg_cv=GridSearchCV(lr,parameters,scoring='accuracy',cv=10)
logreg_cv.fit(X_train,Y_train)
```

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

```
svm_cv = GridSearchCV(svm, parameters, scoring='accuracy', cv=10)
svm_cv.fit(X_train, Y_train)
```

```
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()
```

```
tree_cv = GridSearchCV(tree, parameters, scoring='accuracy', cv=10)
tree_cv.fit(X_train, Y_train)
```

```
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(KNN, parameters, scoring='accuracy', cv=10)
knn_cv.fit(X_train, Y_train)
```

Train the data on different models with hyperparameter tuning and note the accuracy of each.

```
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,)
```

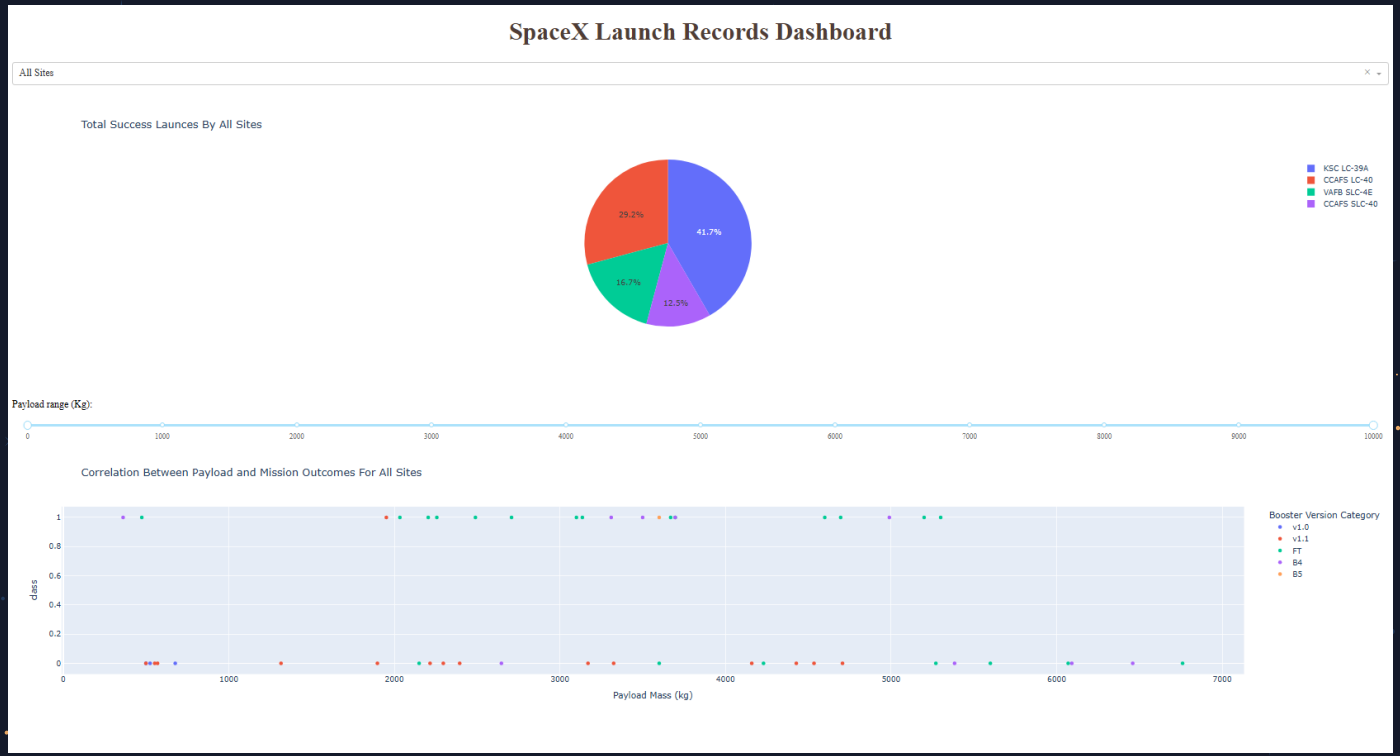
Split the data into training and testing

```
Model_Performance_df = pd.DataFrame({'Algo Type': ['Logistic Regression', 'SVM','Decision Tree','KNN'],
'Accuracy Score': [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_],
'Test Data Accuracy Score': [logreg_cv.score(X_test, Y_test), svm_cv.score(X_test, Y_test),
tree_cv.score(X_test, Y_test), knn_cv.score(X_test, Y_test)]})
Model_Performance_df.sort_values(['Accuracy Score'], ascending = False, inplace=True)
i = Model_Performance_df['Accuracy Score'].idxmax()
print(Model_Performance_df['Algo Type'][i]
+ ' -> ' + str(Model_Performance_df['Accuracy Score'][i]))
```

```
Decision Tree -> 0.8875
```

Compare each score and determine the best fitting model

Result



Resulting Dashboard:

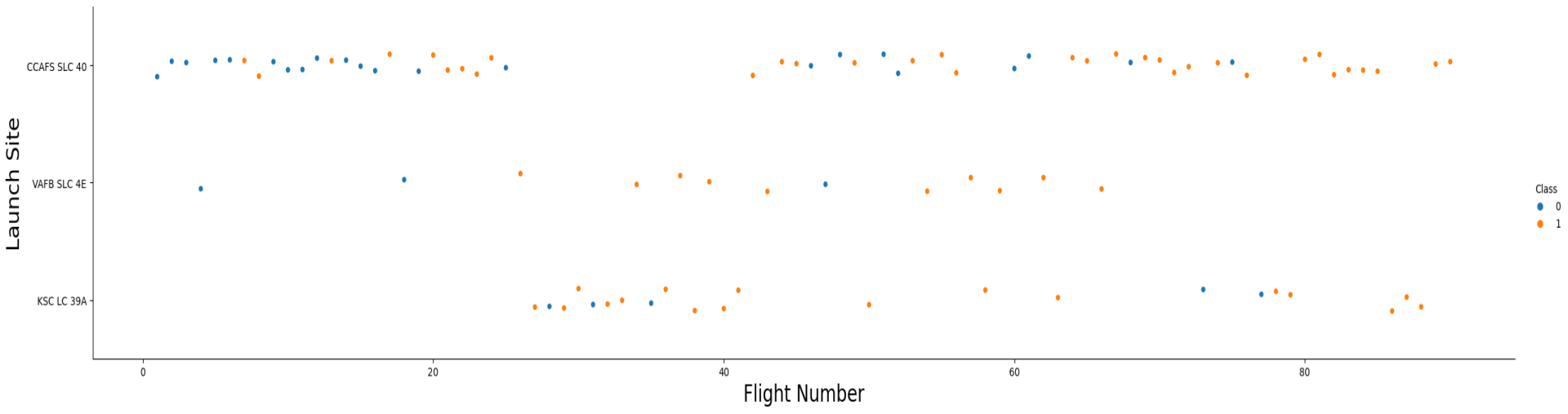
- ***Pie Chart*** *displayed for each launch status in dropdown*
- ***Scatter Chart:*** *Scatter plot shown with adjustment for each payload range*

The best model chosen was the ***Decision tree*** *with the highest accuracy of* ***88.75%***

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.887500	0.888889
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

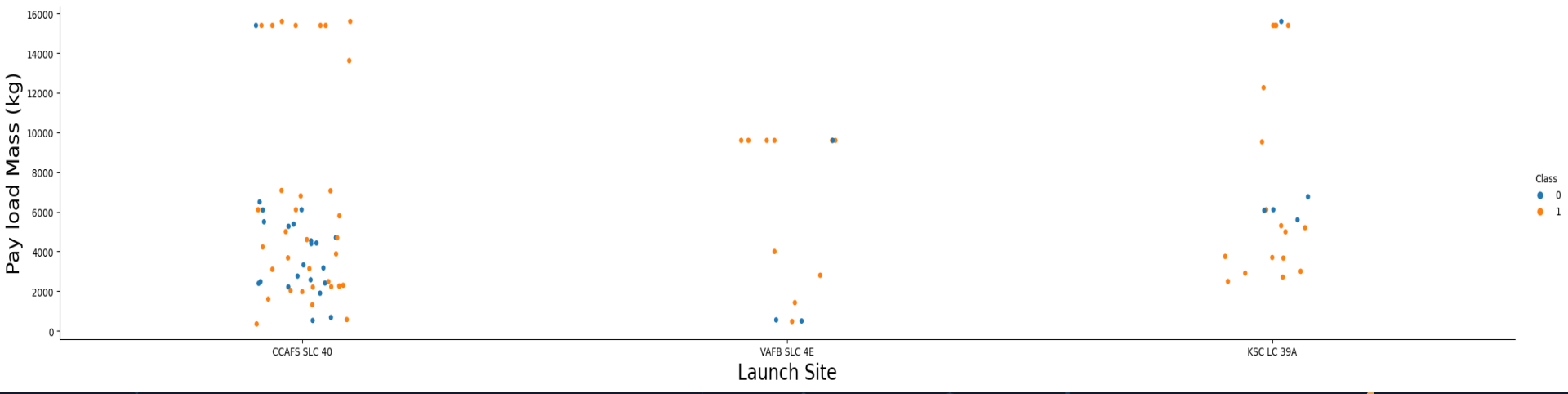
02

Insights drawn from EDA



The data indicates that launch success rates tend to improve with increasing flight number, showing a significant jump in success probability around the 20th flight. The CCAFS launch site shows the highest volume of launches with the lowest being the VADB SLC 4E.

Flight Number vs. Launch Site

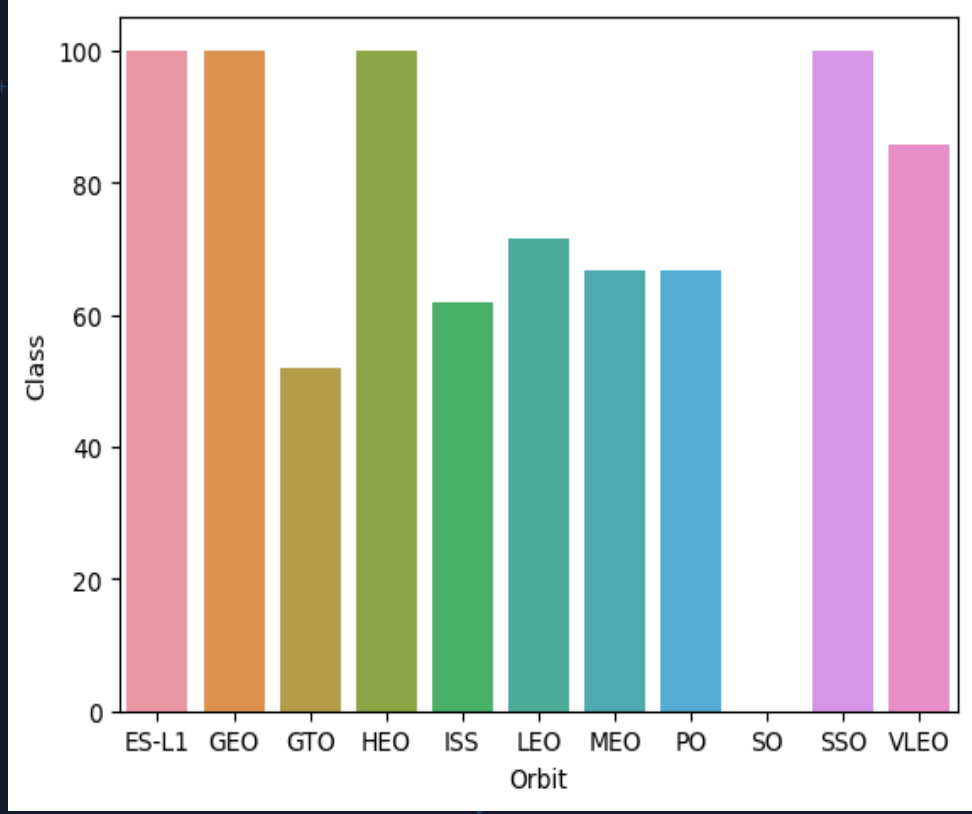


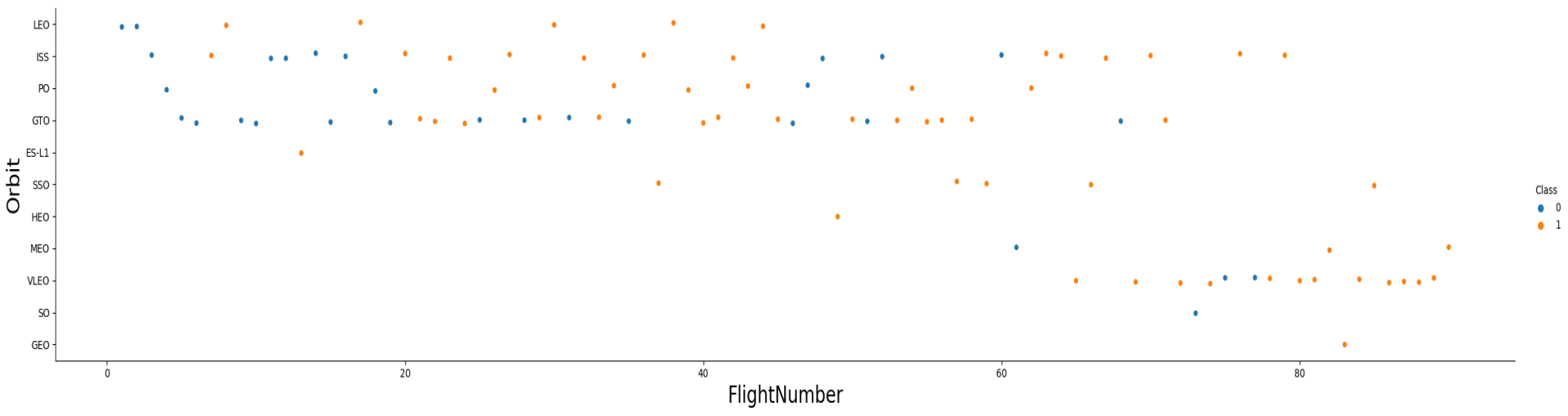
Most Payload mass resides below the 8000 Kg limit but it can also be noticeable that different launch sites handle different distributions of payload as well as different frequencies with the most popular site being the CCAFS SLC 40 and the least popular being the VAFB SLC 4E.

Payload vs. Launch Site

Success rate vs. Orbit type

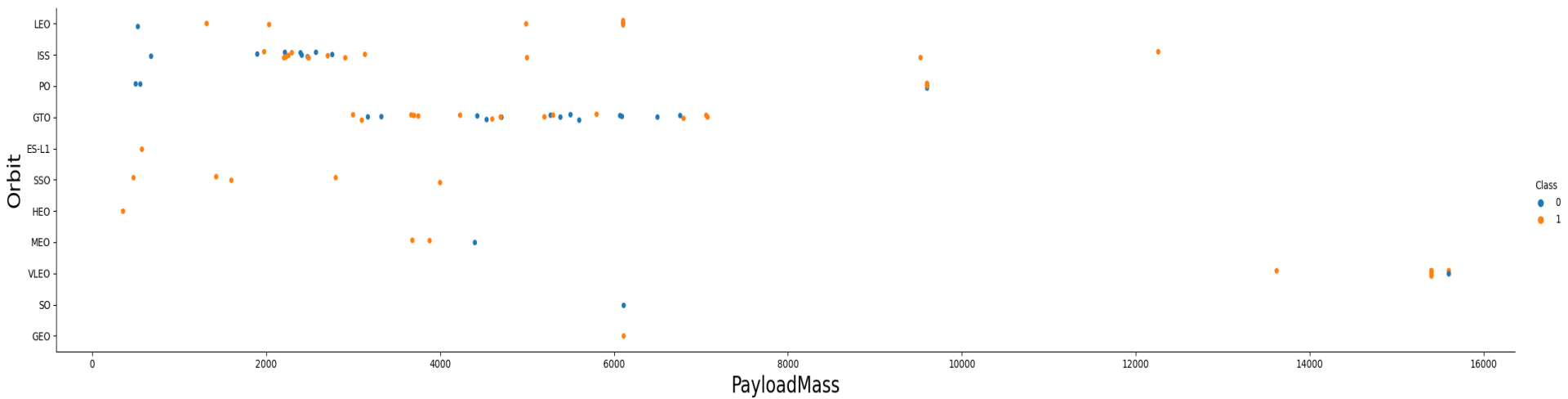
The missions of Orbit type ES-L1, GEO, HEO and SSO tend to have the highest success rates of 100% whereas the lowest success rate falls on missions with the GTO orbit. It is also notable that no successful missions have been undertaken for the SO orbit.





From the above graph it seems that the initial 10 flights primarily focused on the LEO, ISS, PO and GTO orbits with more orbits being focused on with more flights to having done almost all orbits after the 80th flight.

Flight Number vs. Orbit type

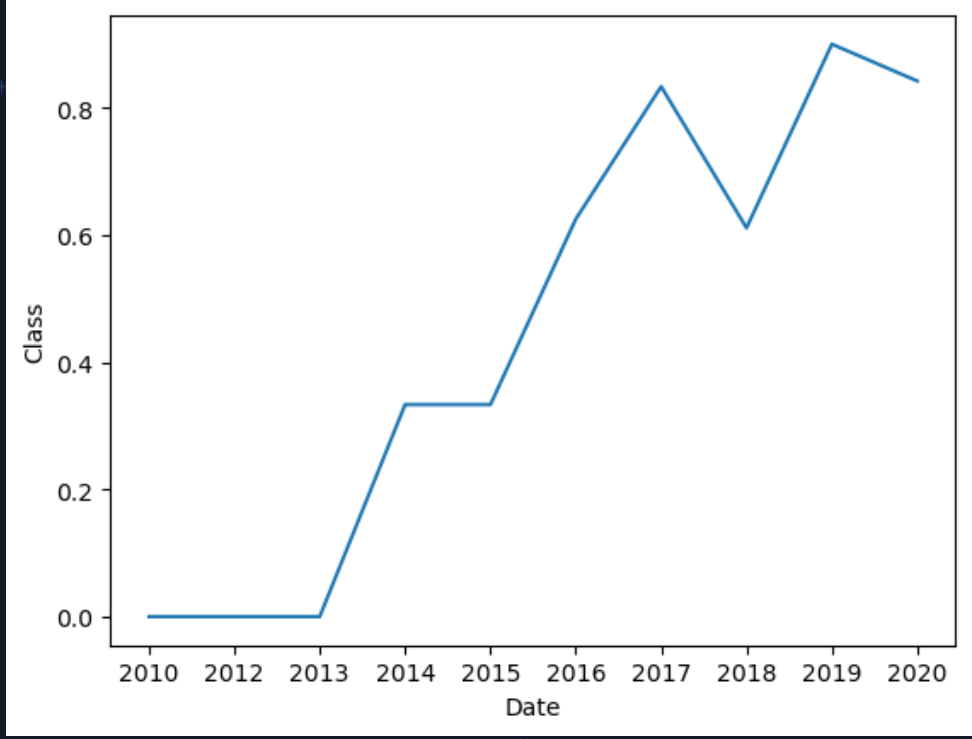


Payload mass appears to be related to orbit type. LEO and SSO tend to have lower payload masses, whereas VLEO, another successful orbit, is characterized by higher payload masses.

Payload vs. Orbit type

Launch Success Yearly Trend

Missions have 0% success rate up until 2013 with a stead increase in success rate per year with a slight stagnation from 2017 and a dip in success rate in 2018, followed by the highest success rate being recorded in 2019.



03

EDA with SQL

All Launch Site Names

There are a total of 4 distinct launch sites given in the query result

```
%sql select distinct(Launch_Site) from SPACEXTABLE
```

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

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

Launch Site Names Beginning with ‘CCA’

The query gives only 5 launch entries, all of which share the launch site having the same starting letters ‘CCA’

```
%sql select * from SPACEXTABLE where Launch_Site like 'CCA%' limit 5
```

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

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 sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = 'NASA (CRS)'
```

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

sum(PAYLOAD_MASS__KG_)
45596

The query returns the total payload mass accounted for in launch instances affiliated with the customer NASA (CRS)

Total Payload Mass

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = 'F9 v1.1'
```

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

avg(PAYLOAD_MASS__KG_)
2928.4

The Average payload is taken with launches involving boosters of version F9 v1.1 is displayed above.

Average Payload Mass by F9 v1.1

```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome like "Success (ground pad)"
```

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

min(Date)
2015-12-22

The first ever successful landing on the ground pad was recorded on the 22nd of December, 2015.

First Successful Ground Landing Date

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

```
%sql select distinct(Booster_Version) from SPACEXTABLE where Landing_Outcome like "Success (drone ship)" and PAYLOAD_MASS__
```

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

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

The above is a list of 4 different booster versions that have been able to successfully landed on the ground pad with a drone ship and a payload mass between 4000 and 6000 Kgs.

Successful Drone Ship Landing with Payload Between 4000 and 6000

```
%sql select Mission_Outcome, count(Mission_Outcome) from SPACEXTABLE group by Mission_Outcome
```

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

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

There have been a total of 100 success outcomes, out of which only 1 of them has an unclear payload status. Meanwhile there have only been one in-flight failure recorded.

Total Number of Each Mission Outcome

Boosters that Carried Maximum Payload

These listed booster versions have all been recorded carrying the heaviest payload i.e. 15600 kg. All these versions tend to be of F9 B5 B10xx.x version.

```
%sql select distinct(Booster_Version), PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_ in (select max(PAYLOAD_MA
```

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

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

```
%sql select *, substr(Date, 6,2) as Month from SPACEXTABLE where Landing_Outcome like "Failure%" and substr(Date,0,5)='2015
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome	Mo
2015-01-10	9:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)	
2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)	

This query retrieves data about 2015 launches where the first stage failed to land on a drone ship, including the month, landing outcome, booster version, payload mass (in kg), and launch site. There were two such instances.

2015 Failed Drone Ship Landing Records

```
%sql select Landing_Outcome, count(Landing_Outcome) as count from SPACEXTABLE where Date between '2010-06-04' and '2017-03-
```

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

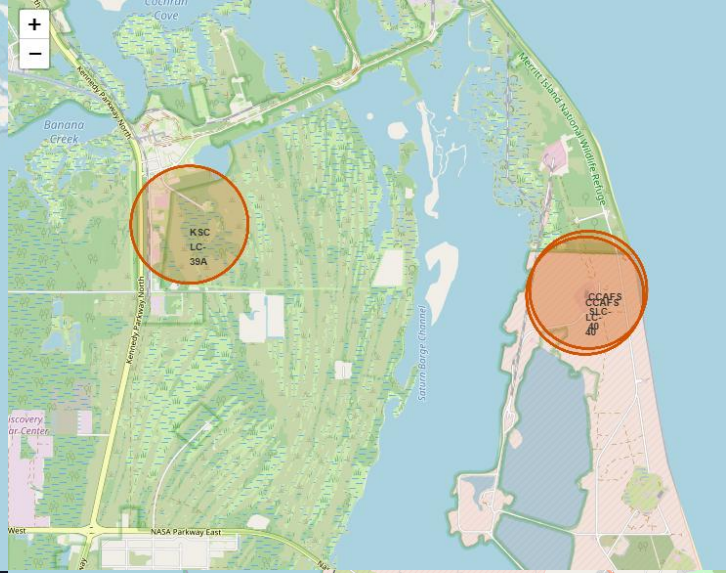
Landing_Outcome	count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

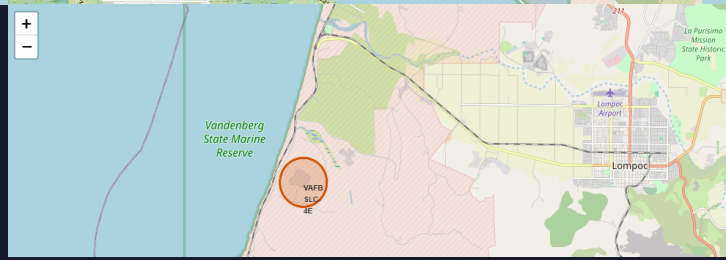
The query provides a list of successful landings from June 4, 2010, to March 20, 2017, inclusive. During this period, there were eight successful landings, which are categorized as either drone ship or ground pad landings.

Ranking Counts of Successful Landings Between 2010-06-04 and 2017-03-20

04

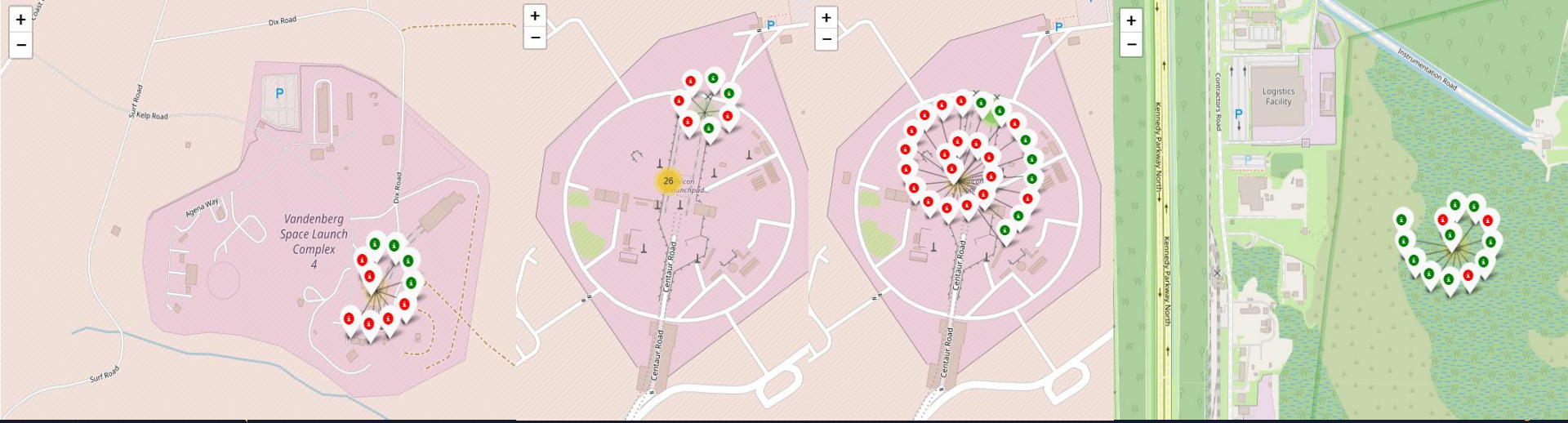
Launch Sites Proximities Analysis





The above maps show the general locations of the launch sites on the USA map as well as close-ups on each of the 4 different sites.

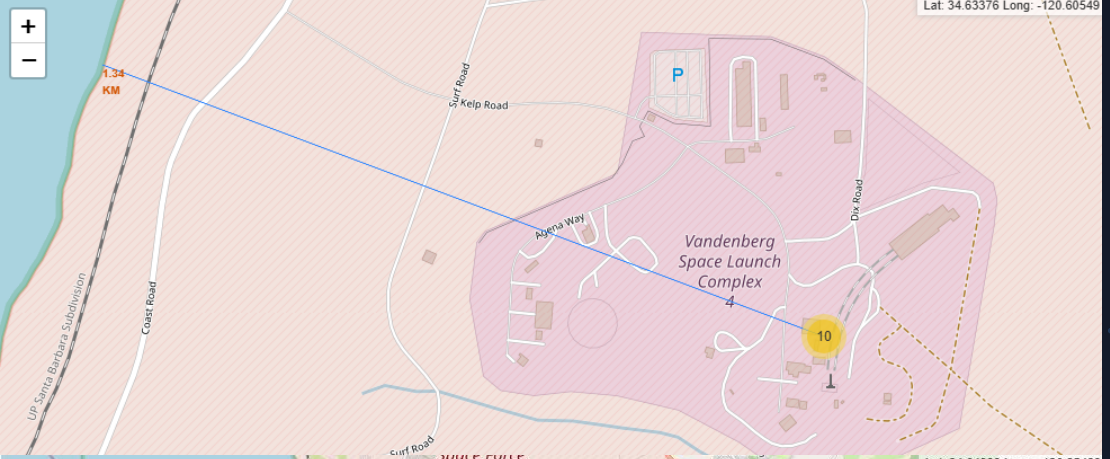
SpaceX Falcon9 - Launch Sites Map

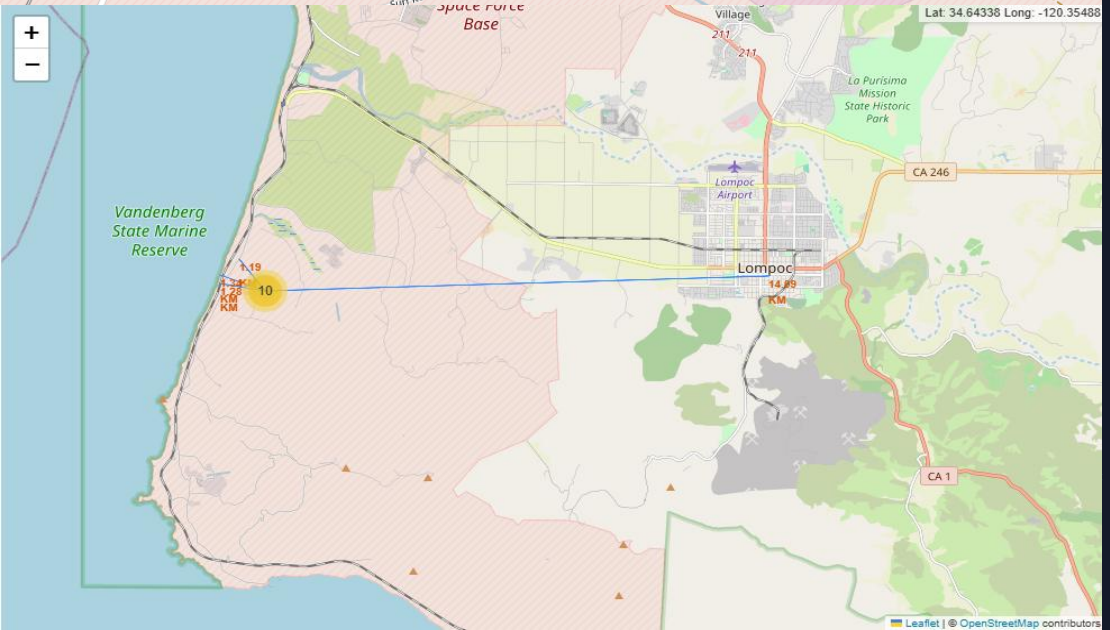


The above are the close-up shots of all 4 launch sites with each pointer on each location indicating the successful and failed launches on each site as indicated by the green and red points respectively.

SpaceX Falon9 - Success/Failed Launch Map for all Launch Sites

SpaceX Falcon9 - Launch Site to proximity Distance Map





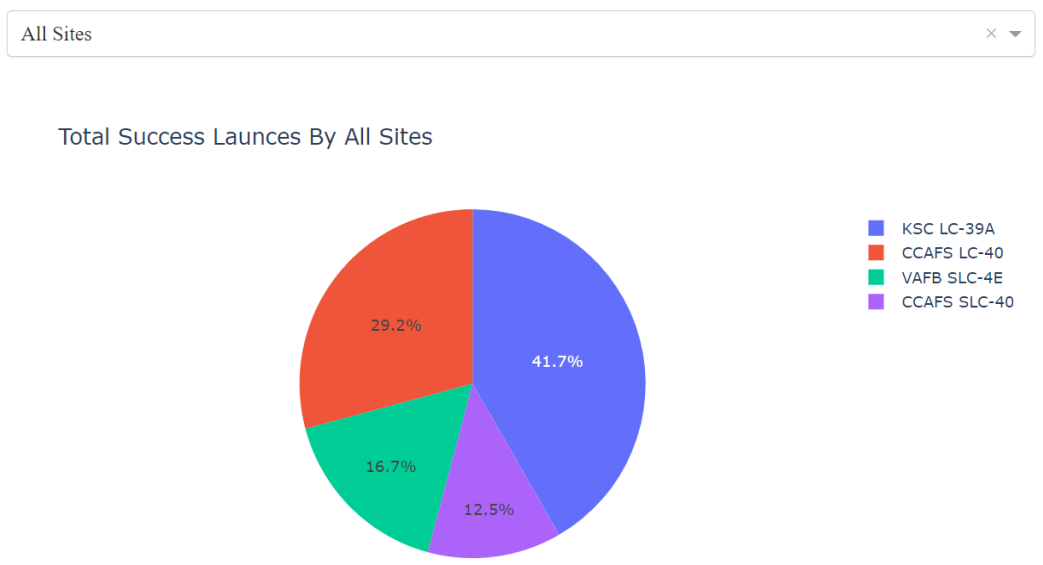
For Launch Site VAFB SLC 4E, the map shows that the city of Lompoc is farther away compared to other nearby locations like the coastline, railroad, and highway.

The distance from the launch site to the city is indicated on the map as 14.09 km. A closer view highlights the proximity of the coastline, railroad, and highway, including their respective distances from the launch site.

In general, cities are situated at a distance from launch sites, likely to mitigate potential risks to the public and infrastructure. Conversely, launch sites are strategically positioned near the coastline, railroad, and highways for convenient access to resources.

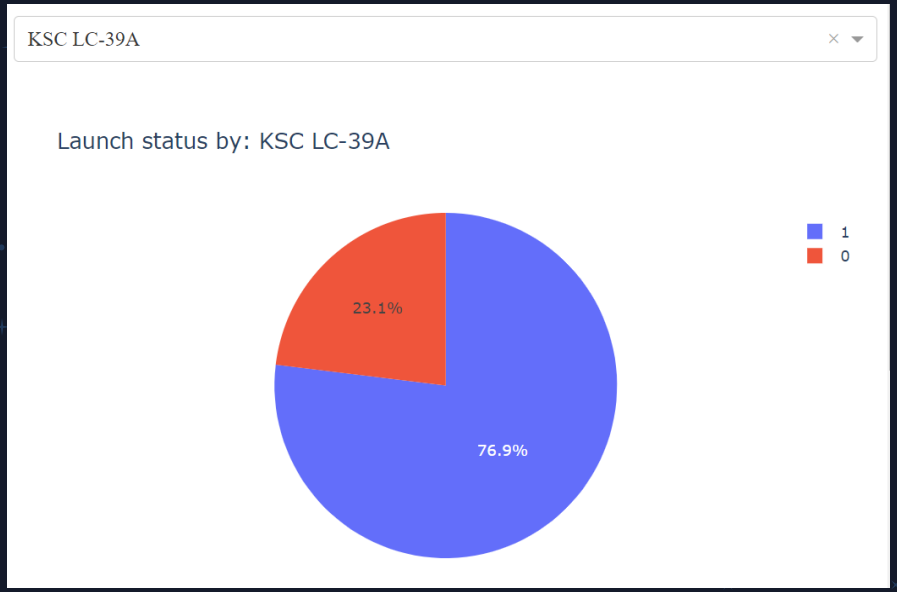
05

Build a Dashboard with Plotly Dash



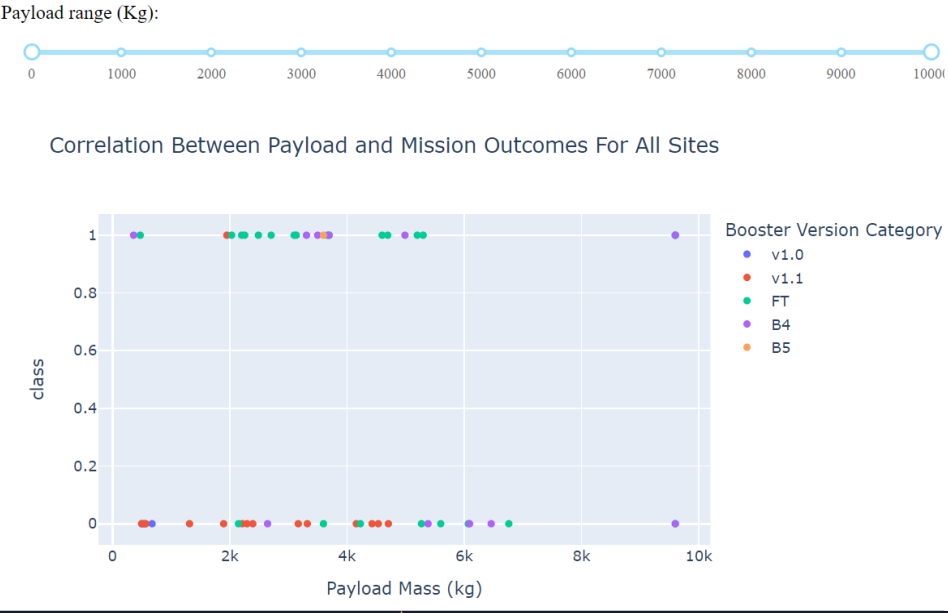
According to the above pie chart, the KSC LC 39A has the highest launch success rate, while CCAFS SLC 40 has the lowest.

Launch Success Counts For All Sites



The KSC LC 39A launch site demonstrates the highest launch success rate and number of successful launches, characterized by a 76.9% success rate and a 23.1% failure rate.

Launch Site with Highest Launch Success Ratio



For all sites, the payload vs. launch outcome scatter plot indicates that most successful launches occur with payloads between 2000 and 5500 kg. The 'FT' booster version exhibits the highest number of successful launches, while 'B4' is the only booster version that has successfully launched payloads exceeding 6000 kg.

Payload vs. Launch Outcome Scatter Plot for All Sites

06

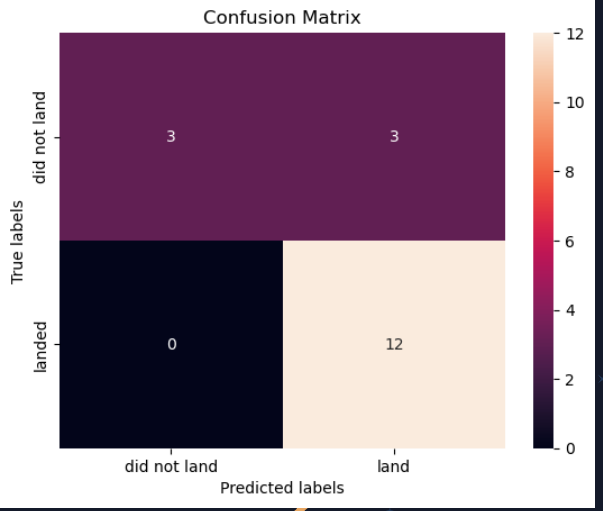
Predictive Analysis (Classification)

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.887500	0.888889
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

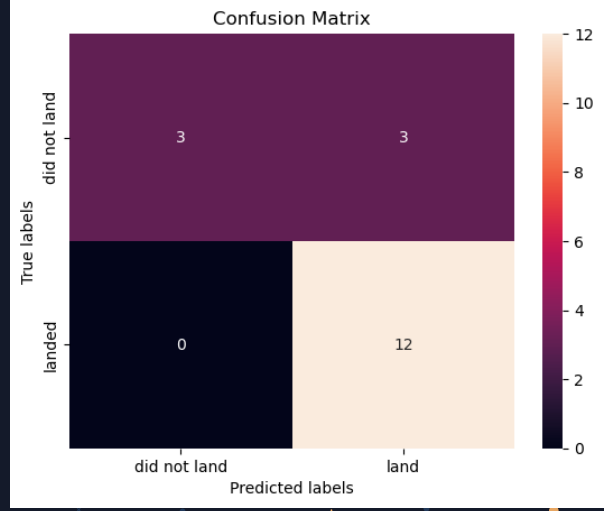
The Decision Tree algorithm demonstrates the highest classification score at 0.8875, as indicated by the accuracy scores and the bar chart. However, the accuracy score on the test data is consistent across all classification algorithms in this dataset, with a value of 0.8333 except for decision tree which is 0.888889. Given the close accuracy scores among these algorithms and the identical test scores, a broader dataset may be necessary to effectively tune the models further.

Classification Accuracy

Confusion Matrix



Logistic Regression



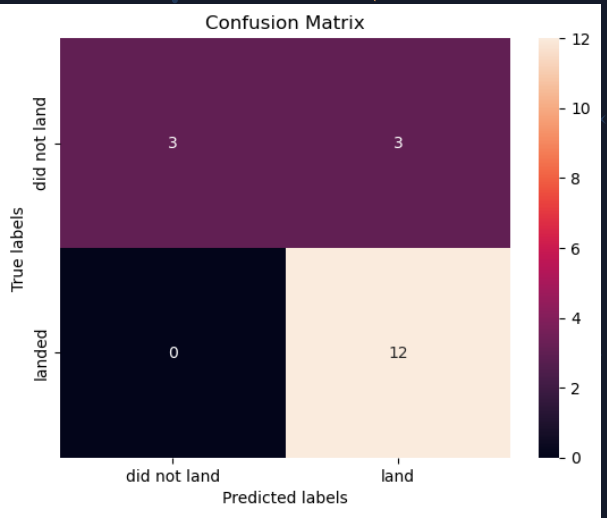
Support Vector Machine

The confusion matrix, consistent across all models (LR, SVM, Decision Tree), shows 18 total predictions.

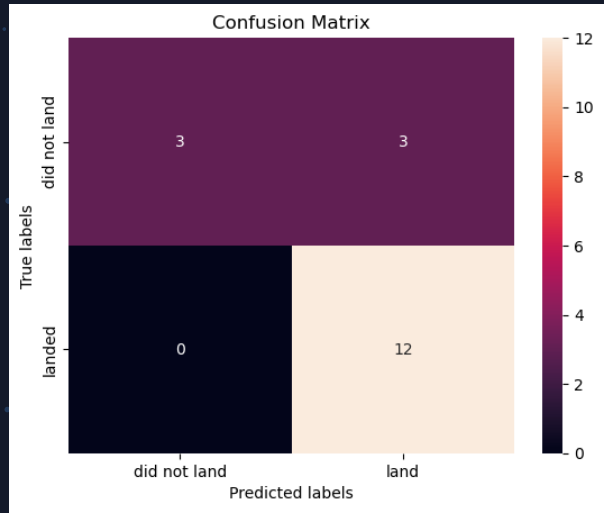
The classifier accurately predicted 12 successful landings (true positives) and 3 unsuccessful landings (true negatives).

However, it incorrectly predicted 3 unsuccessful landings as successful (false positives).

Overall, the classifier demonstrates an 83% accuracy rate, with a misclassification or error rate of approximately 16.5%.



Decision Tree



K-Nearest Neighbors

Conclusion

- ✓ First-stage landings are more likely to be successful as the number of flights increases.
- ✓ While success rates appear to improve with higher payload masses, there's no clear correlation between payload mass and success rate.
- ✓ The launch success rate increased by approximately 80% between 2013 and 2020. The KSC LC 39A launch site has the highest launch success rate, whereas the CCAFS SLC 40 launch site has the lowest.
- ✓ The ES L1, GEO, HEO, and SSO orbits have the highest launch success rates, while the GTO orbit has the lowest.
- ✓ Launch sites are strategically positioned away from cities and closer to coastlines, railroads, and highways.
- ✓ The Decision Tree is the best-performing machine learning classification model, with an accuracy of about 88.7%. When models were tested, the accuracy was about 83% for all except for decision tree which is 0.888889. Further model tuning with more data could potentially improve results.

Thanks!