IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Marc Lodian M. Mier
December 31, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection through a combination of API utilization and web scraping techniques.
  - Data Processing by implementing rigorous data wringling to transform raw data into an analyzable format.
  - Data Analysis with SQL for querying and data visuals for pattern recognition.
  - Build an interactive map with Folium to analyze launch site proximity.
  - Build an interactive dashboard with Plotly Dash to analyze launch records.
  - Lastly, Build a predictive model to predict the success of the first stage landing of Falcon 9.
- Summary of all results
  - Data Analysis outcomes
  - Visual Representations
  - Predictive Model Evaluation

Introduction

In the evolving landscape of space exploration, recent successes in private space travel have propelled the industry into mainstream accessibility. However, the persistent barrier for new entrants remains the high cost of launches.

SpaceX, with its revolutionary first stage reuse capabilities, stands out by offering a distinct advantage over competitors. While the average launch cost for competitors hovers around 165 million dollars, each SpaceX launch costs approximately 62 million dollars. The reusable stage 1 technology significantly reduces expenditure, solidifying SpaceX's position in the space race.

Key Objectives:

1. **Predicting Successful Landings** – The primary objective is to determine the likelihood of a successful landing for the first stage of the SpaceX Falcon 9.
2. **Analyzing Impactful Variables** – Investigating the influence of key parameters, such as launch site, payload mass, and booster version, on landing outcomes.
3. **Correlating Launch Sites and Success Rates** – Exploring correlations between Launch sites and the success rates of Falcon 9

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX API
  - Web scrap Falcon 9 and Falcon Heavy launch records form Wikipedia
- Perform data wrangling
  - Determined labels for training the supervised models by converting mission outcomes in to training labels: 0 = unsuccessful , 1 = successful
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

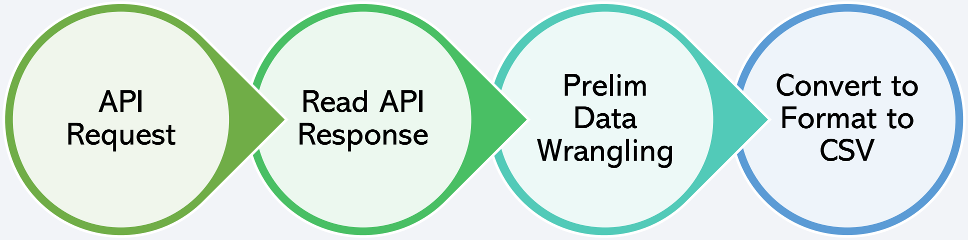
Created a column for ‘class’; standardized and transformed data; train/test split data; find best classification algorithm (Logistic regression, SVM, decision tree, & KNN) using test data.

Data Collection

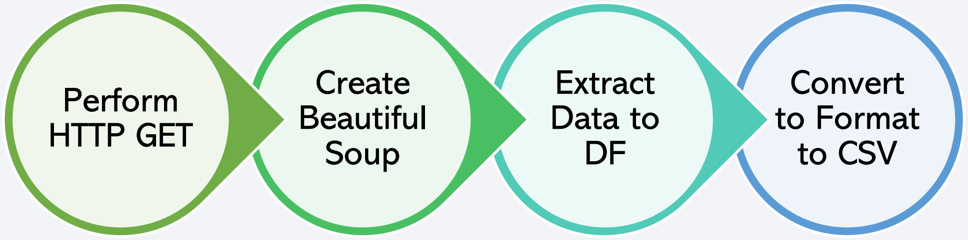
We utilized SpaceX API for structured data and employed web scraping techniques to extract relevant launch data from Wiki pages.

- **SpaceX API** – Leveraging the SpaceX API provided us with structured and oraganized data, offering a foundation for our comprehensive analysis.
- **Web Scraping Wiki Pages** – In addition to the API, we Utilized we scrapping to extract pertinent launch data from Wiki pages. This method allowed us to gather unstructured information, complementing the structured data obtained through the API.

SpaceX API



Web Scrapping



Data Collection – SpaceX API

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

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

```
response.status_code
```

200

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
data = pd.json_normalize(response.json())
```

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

```
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
BoosterVersion[0:5]
```

['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']

we can apply the rest of the functions here:

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
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}
```

```
# Create a data from launch_dict
df_launch = pd.DataFrame(launch_dict)
```

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

```
data_falcon9.to_csv('dataset_part_1.csv',
index=False)
```

API Request and read response into DF

Declare Global Variables

Call Helper Functions with API

Construct Data Using Directory

Convert Dict to DF

Filter for Falcon9 Launches

Convert to CSV

Github URL

Data Collection - Scraping

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

html_data = requests.get(static_url).text
```

```
# Use BeautifulSoup() to create a BeautifulSoup object fro
soup = BeautifulSoup(html_data,"html.parser")
```

```
column_names = []

# Apply find_all() function with `th` element on first_lau
# Iterate each th element and apply the provided extract_c
# Append the Non-empty column name (`if name is not None a
colnames = soup.find_all('th')
for x in range (len(colnames)):
    name2 = extract_column_from_header(colnames[x])
    if (name2 is not None and len(name2) > 3):
        column_names.append(name2)
```

```
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 e
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(launch_dict)
```

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Perform HTTP GET to request HTML page

Create BeautifulSoup Object

Extract Column Names

Create Dictionary

Call Helper Functions to Fill up Dict

Convert Dictionary to Data Frame

Github URL

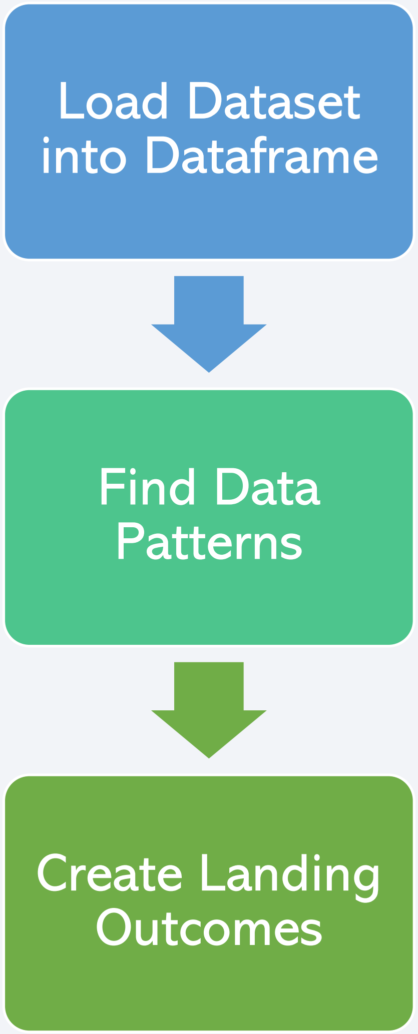
Data Wrangling

To prepare our data for analysis, we conducted Exploratory Data Analysis (EDA) to identify patterns and define labels for training supervised models.

Landing Scenarios Considered:

- True Ocean: Successful landing in a specific region of the ocean.
- False Ocean: Unsuccessful landing in a specific region of the ocean.
- RTLS (Return to Launch Site): Successful landing on a ground pad.
- False RTLS: Unsuccessful landing on a ground pad.
- True ASDS (Autonomous Spaceport Drone Ship): Successful landing on a drone ship.
- False ASDS: Unsuccessful landing on a drone ship.

By categorizing Mission Outcomes into these scenarios, we create a well-defined set of labels for training our supervised models



Github URL

EDA with Data Visualization

As part of our Exploratory Data Analysis (EDA), we employed various charts to extract insights from the dataset:

1. Scatter Plots:
   - Utilized for visualizing relationships or correlations between two variables, making patterns easily observable.
   - Plotted scatter charts to examine relationships between Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit Type, as well as Payload and Orbit Type.

2. Bar Chart:
   - Commonly used for comparing the values of a variable at a specific point in time.
   - Plotted a bar chart to visualize the relationship between the success rates of each orbit type. The length of each bar is proportional to the success rate, facilitating clear comparisons.

3. Line Chart:
   - Applied to track changes over a period, helping depict trends.
   - Plotted a line chart to observe the yearly trend in the average launch success rate. This chart provides a concise overview of the data's temporal patterns.

Github URL

EDA with SQL

To gain deeper insights into the SpaceX dataset, several SQL queries and operations were performed on an IBM DB2 cloud instance:

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA’
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was achieved.
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes
8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
9. List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
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

Github URL

Build an Interactive Map with Folium

Utilizing Folium for geospatial data analysis enhances interactive visual analytics, providing insights into factors such as launch site location and proximity, which significantly impact launch success rates.

Map Objects Created:

1. **Launch Site Markers** - Marked all launch sites on the map for visual reference.
2. **Circle and Marker Highlights** - Implemented 'folium.circle' and 'folium.marker' to highlight circular areas with text labels over each launch site. This enhances the visual representation of launch site features.
3. **Launch Success/Failure Clusters** - Integrated 'MarkerCluster()' to display launch success (green) and failure (red) markers for each launch site. This clustering provides a clear overview of success rates.
4. **Distance Calculation** - Calculated distances between launch sites and proximities (e.g., coastline, railroad, highway, city) for a comprehensive spatial analysis.
5. **Mouse Position Tracking** - Added 'MousePosition()' to retrieve coordinates for a mouse position over a point on the map. This facilitates precise location identification.
6. **Distance Display Markers** - Utilized 'folium.Marker()' to display distances (in KM) on the map at specific points (e.g., coastline, railroad, highway, city), aiding in spatial understanding.
7. **Line Visualization** - Employed 'folium.Polyline()' to draw lines between launch sites and proximities (coastline, railroad, highway, city). This visual representation enhances the connectivity visualization.

The combination of these map objects offers a dynamic and informative visualization, enabling a more in-depth understanding of the spatial relationships influencing launch success

Github URL

Build a Dashboard with Plotly Dash

Crafted using Plotly Dash, this dynamic web application offers real-time interactive visual analytics on SpaceX launch data, providing insights through various components:

1. Launch Site Dropdown:

   Introduced a Launch Site Dropdown Input to enable users to filter visualizations by selecting all launch sites or a specific launch site. Notably, KSC LC 39A emerges with 10 successful launches, showcasing its significance.

2. Pie Chart:

   Implemented a Pie Chart that displays total successful launches when 'All Sites' is selected. When a specific site is chosen, it reveals success/failure counts. For instance, KSC LC 39A stands out with a remarkable 76.9% success rate.

3. Payload Range Slider:

   Integrated a Payload Range Slider, allowing users to easily select different payload ranges. This feature aids in identifying visual patterns. Notably, payload ranges of 2000-5000 kg demonstrate the highest launch success rate.

4. Scatter Chart:

   Included a Scatter Chart to explore potential correlations between payload and mission outcomes for selected site(s). Each point is color-labeled by Booster version. Notably, the FT Booster version exhibits the highest launch success rate among various versions (v1.0, v1.1, FT, B4, B5, etc.).

Github URL

Predictive Analysis (Classification)

```
data = pd.read_csv("https://cf-courses-data.s3.us.cl
```

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

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

```
# Split data for training and testing data sets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

```
Train set: (72, 83) (72,)
Test set: (18, 83) (18,)
```

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

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

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'sol
LR = LogisticRegression()
logreg_cv = GridSearchCV(LR, parameters,cv=10)
logreg_cv.fit(X_train, Y_train)
```

```
# Create a DF for algorithm type and respective best

Model_Performance_df = pd.DataFrame({'Algo Type': ['
'Accuracy Score': [logreg_cv.best_score_, svm_cv.bes
'Test Data Accuracy Score': [logreg_cv.score(X_test,
tree_cv.score(X_test, Y_test), knn_cv.score(X_test,
```

```
Model_Performance_df.sort_values(['Accuracy Score'],
```

```
Model_Performance_df
```

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

Read Dataset into Data Frame

Create 'Class' array

Standardize the Data

Train, Test, Spill Data

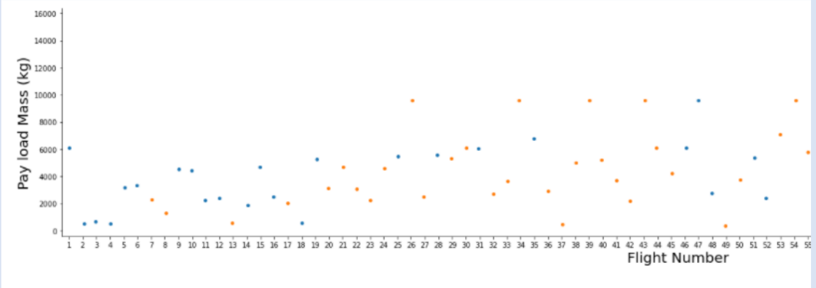
Create and Refine

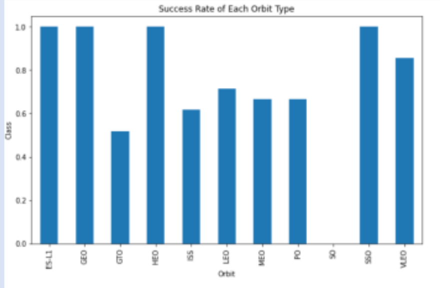
Find the Best Performing Model

Github URL

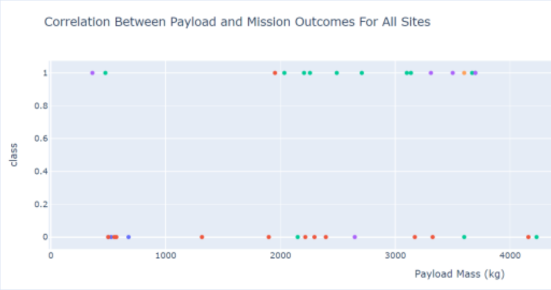
Results

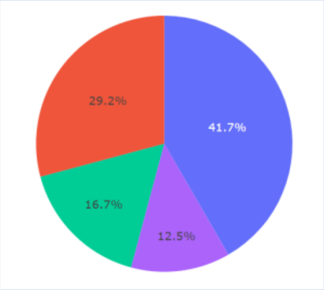
Exploratory Data Analysis Results





Interactive Analytics Demo





Predictive Analysis

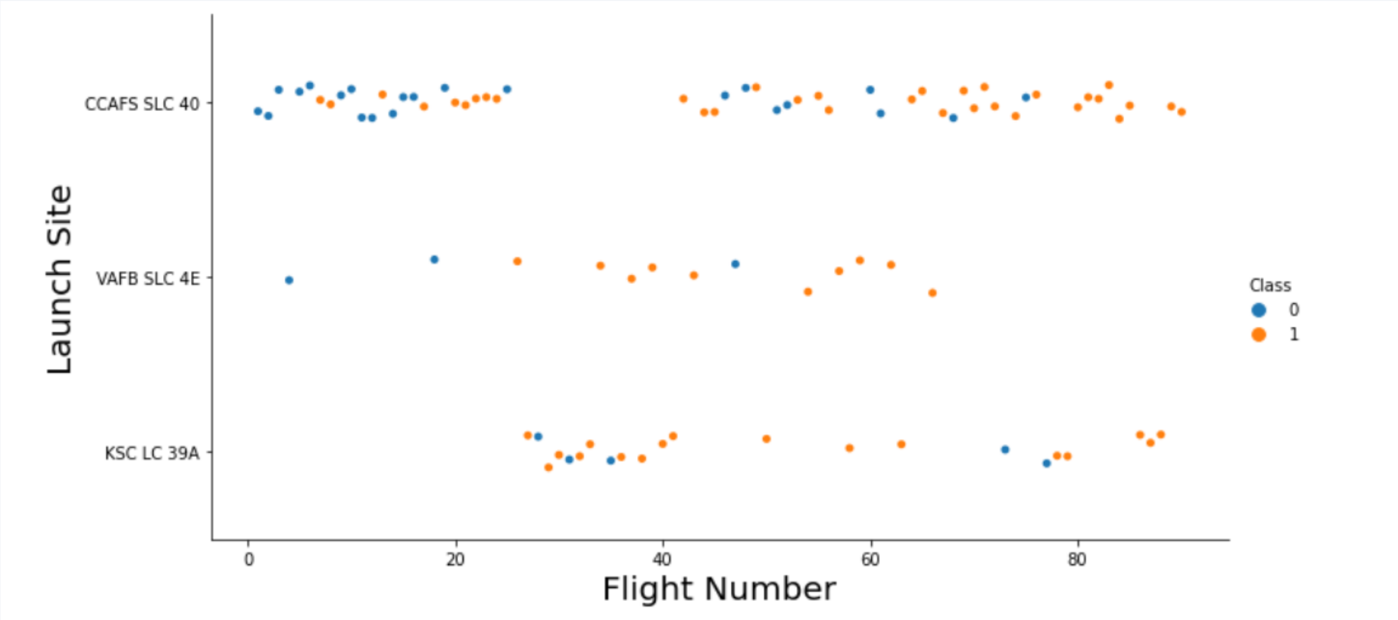
	Algo Type	Accuracy Score
2	Decision Tree	0.903571
3	KNN	0.848214
1	SVM	0.848214
0	Logistic Regression	0.846429

Section 2

Insights drawn from EDA

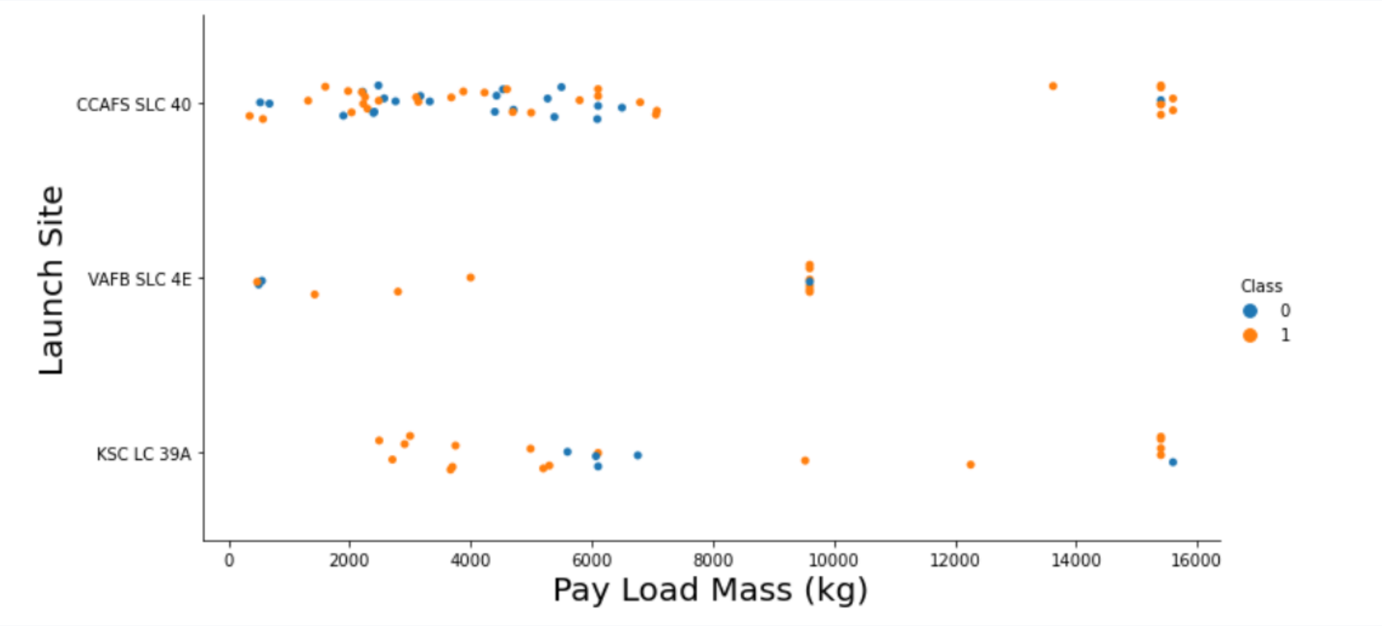
Flight Number vs. Launch Site

- Success rates (Class=1) increases as the number of flights increase
- For launch site 'KSC LC 39A', it takes at least around 25 launches before a first successful launch



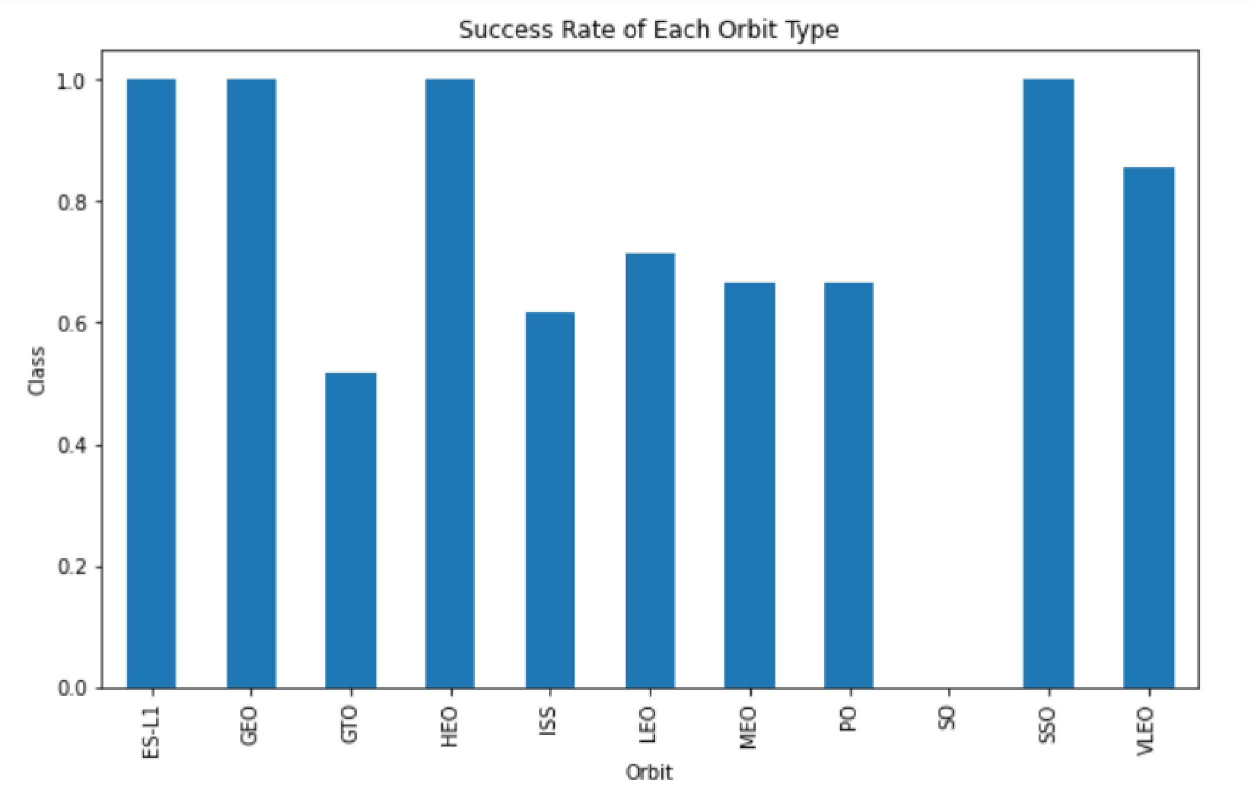
Payload vs. Launch Site

- For launch site ‘VAFB SLC 4E’, there are no rockets launched for payload greater than 10,000 kg
- Percentage of successful launch (Class=1) increases for launch site ‘VAFB SLC 4E’ as the payload mass increases
- There is no clear correlation or pattern between launch site and payloadmass



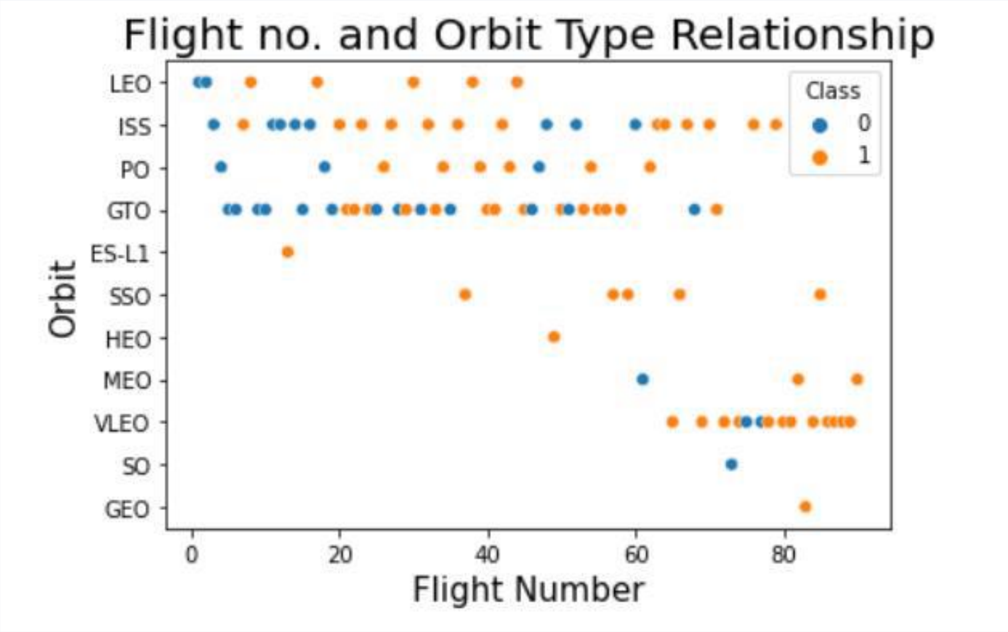
Success Rate vs. Orbit Type

- Orbits ES-LI, GEO, HEO, and SSO have the highest success rates
- GTO orbit has the lowest success rate



Flight Number vs. Orbit Type

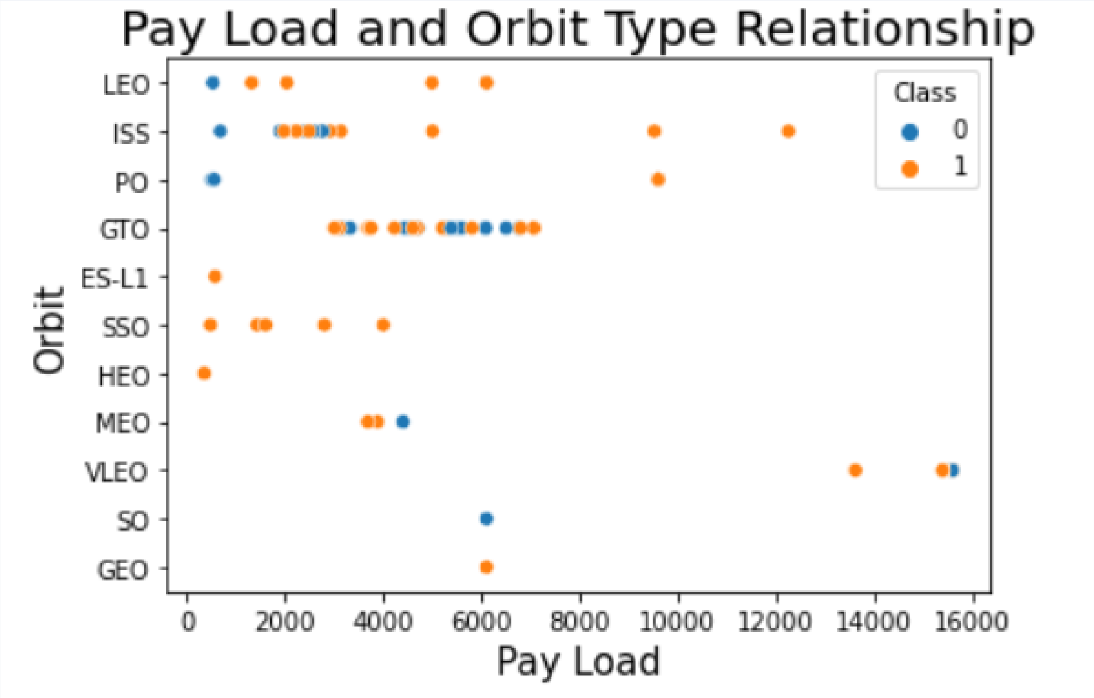
- For orbit VLEO, first successful landing (class=1) doesn't occur until 60+number of flights
- For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers
- There is no relationship between flight number and orbit for GTO



Payload vs. Orbit Type

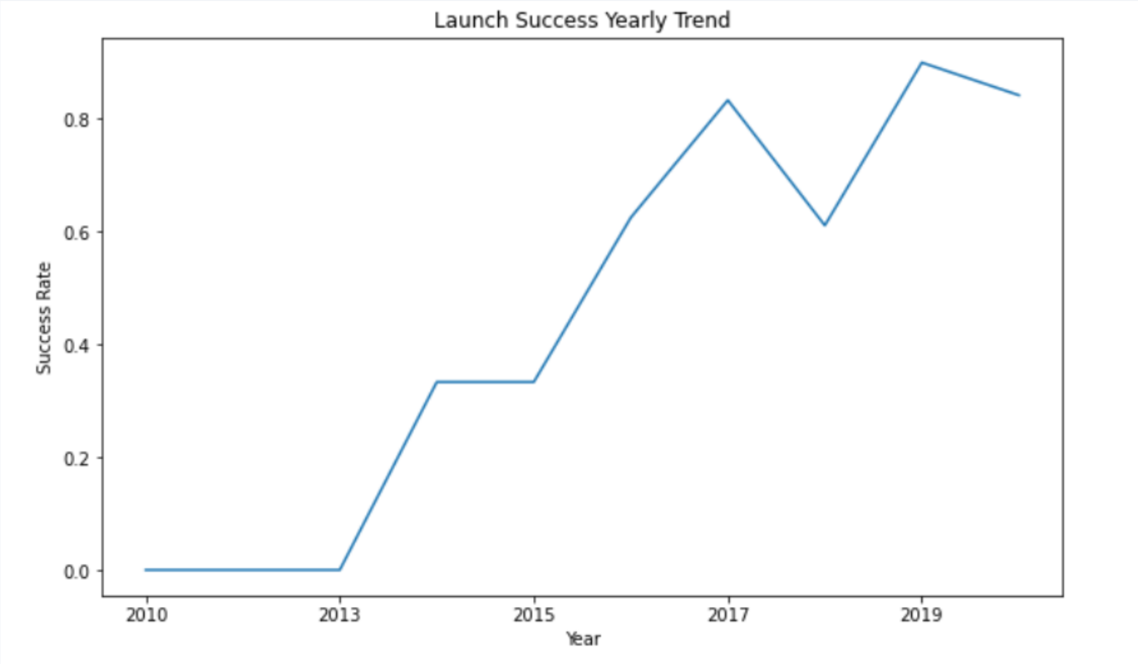
- Successful landing rates (Class=1) appear to increase with pay load for orbits LEO, ISS, PO, and SSO

- For GEO orbit, there is not clear pattern between payload and orbit for successful or unsuccessful landing



Launch Success Yearly Trend

- Success rate (Class=1) increased by about 80% between 2013 and 2020
- Success rates remained the same between 2010 and 2013 and between 2014 and 2015
- Success rates decreased between 2017 and 2018 and between 2019 and 2020



All Launch Site Names

The query SELECT DISTINCT Launch_Site FROM spacextbl utilizes the 'DISTINCT' keyword to retrieve unique values from the 'Launch_Site' column. In this case, it results in 4 unique launch sites.

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

Launch Site Names Begin with 'CCA'

Utilizing the 'LIKE' keyword with the format 'CCA%' retrieves records where the 'Launch_Site' column starts with "CCA". The query then limits the number of returned records to 5.

select * from spacextbl where Launch_Site LIKE %’CCA%’ limit 5;

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

The 'sum' operation adds up the values in the 'PAYLOAD_MASS_KG' column, providing the total payload mass for customers named 'NASA (CRS)'.

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

```
%%sql

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

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

1
45596

Average Payload Mass by F9 v1.1

The 'avg' keyword calculates the average payload mass in the 'PAYLOAD_MASS_KG' column specifically for cases where the booster version is 'F9 v1.1'.

```
%%sql

select avg(PAYLOAD_MASS__KG_) from spacextbl where Booster_Version LIKE 'F9 v1.1';
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

1
2928

First Successful Ground Landing Date

min(Date) selects the earliest date from the 'Date' column where the first successful landing on a ground pad was achieved. The WHERE clause specifies the condition to retrieve the date when the 'Landing_Outcome' value is 'Success (ground pad)'.

```
%%sql

select min(Date) as min_date from spacextbl where Landing__Outcome = 'Success (ground pad)';
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

min_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

The query identifies the booster version with a payload mass greater than 4000 but less than 6000, and a successful landing outcome on a drone ship. The 'and' operator in the WHERE clause ensures that both conditions are met for the returned results.

```
%%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://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

The 'group by' keyword organizes identical data in a column into groups. In this context, the number of mission outcomes is grouped by outcome types in a column.

```
%%sql

select Mission_Outcome, count(Mission_Outcome) as counts from spacextbl group by Mission_Outcome;
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

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

Boosters Carried Maximum Payload

The subquery identifies the maximum payload mass using the 'max' keyword on the payload mass column. The main query retrieves booster versions and their corresponding payload mass when the payload mass is at its maximum value of 15600.

```
%%sql

select Booster_Version, PAYLOAD_MASS__KG_ from spacextbl where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from space
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
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

2015 Launch Records

The query retrieves landing outcomes, booster versions, and launch sites where the landing outcome is a failure on a drone ship, and the year is 2015. The 'AND' operator in the WHERE clause ensures both conditions are met. The 'year' keyword extracts the year from the 'Date' column. The results pinpoint 'CCAFS LC 40' as the launch site and booster versions 'F9 v1.1 B1012' and 'B1015' with failed landing outcomes on a drone ship in the year 2015.

```
%%sql

select Landing__Outcome, Booster_Version, Launch_Site from spacextbl where Landing__Outcome = 'Failure (drone ship)' and yea
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

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

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

The 'group by' keyword organizes data in the 'Landing_Outcome' column into groups. The 'between' and 'and' keywords filter data between the dates 2010-06-04 and 2017-03-20. The 'order by' keyword arranges the counts column in descending order. The result of the query is a ranked list of landing outcome counts within the specified date range.

```
%%sql

select Landing__Outcome, count(*) as LandingCounts from spacextbl where Date between '2010-06-04' and '2017-03-20'
group by Landing__Outcome
order by count(*) desc;
```

```
 * ibm_db_sa://jjk92789:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

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

Section 3

Launch Sites Proximities Analysis

SpaceX Falcon9: Launch Sites

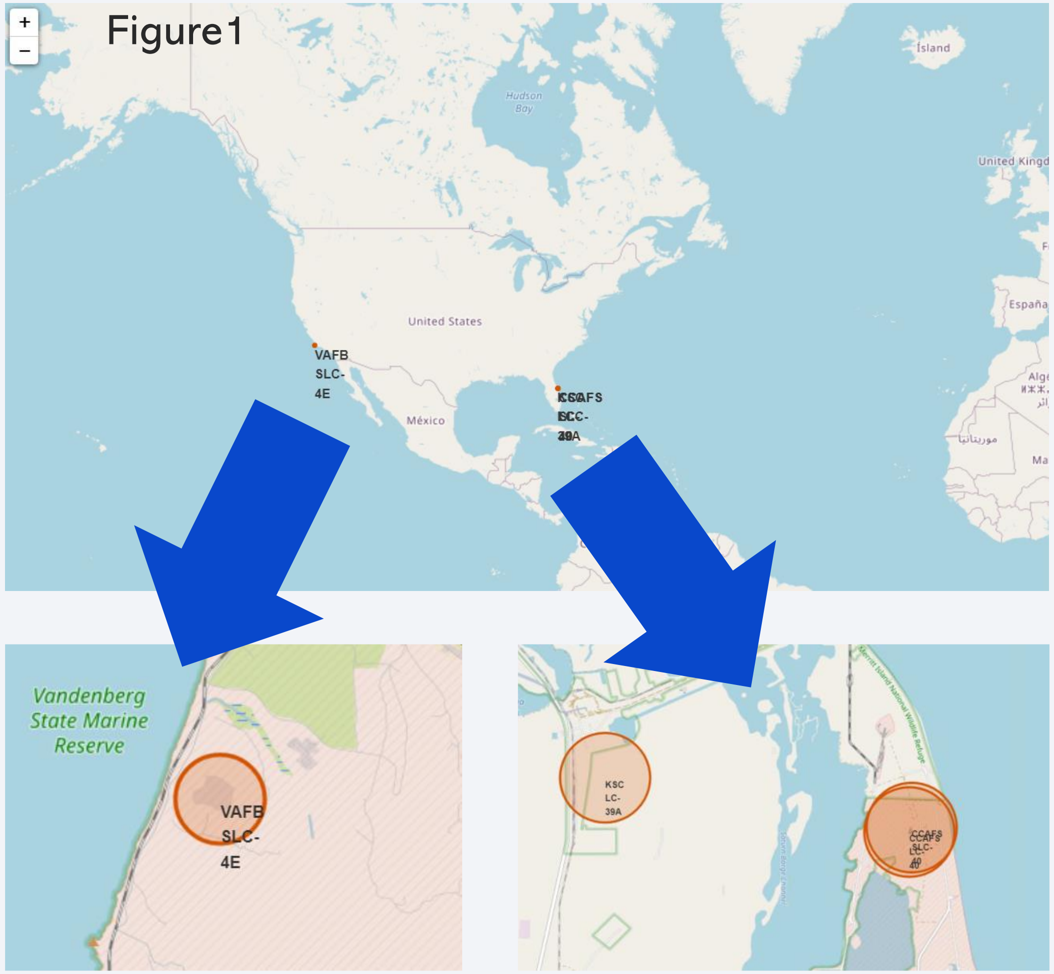


Figure 2

Figure 3

On the left, Figure 1 presents a global map featuring Falcon 9 launch sites situated in the United States, specifically in California and Florida. Each launch site is represented by a circle, a label, and a popup, providing details on the location and the name of the launch site. Notably, all launch sites are positioned near the coast. Figures 2 and 3 offer a closer view of four specific launch sites:

- VAFB SLC 4E (CA)
- CCAFS LC 40 (FL)
- KSC LC 39A (FL)
- CCAFS SLC 40 (FL)
- These detailed figures provide an in-depth visualization of the specified launch sites.

SpaceX Falcon9: Success/Fail Launch Maps

In Figure 1, the U.S. map showcases all the Launch Sites, with each site displaying the total number of successful and failed launches through annotated numbers.

Figures 2, 3, 4, and 5 provide detailed zoom-ins for each site, illustrating success (green) and failure (red) markers. Upon closer inspection, it is evident that the KSC LC 39A Launch Site stands out with the highest number of successful launches among the displayed sites.

Figure 1

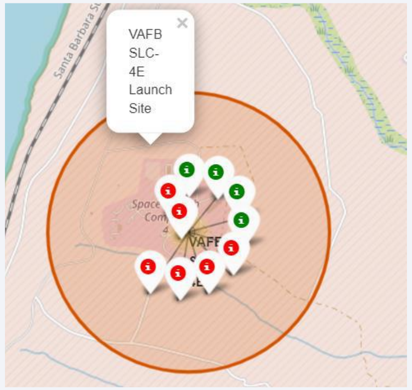


Fig. 2 – VAFB

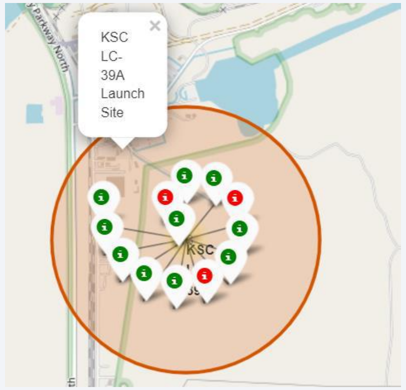


Fig. 3 – KSC LC-39A

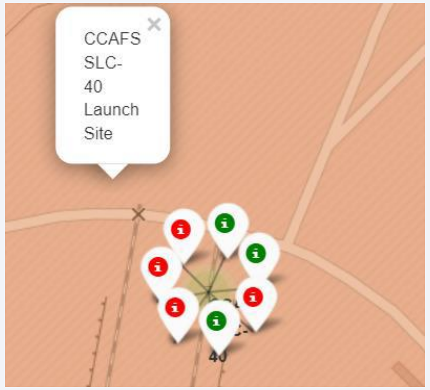


Fig. 4 – CCAFS SLC-40

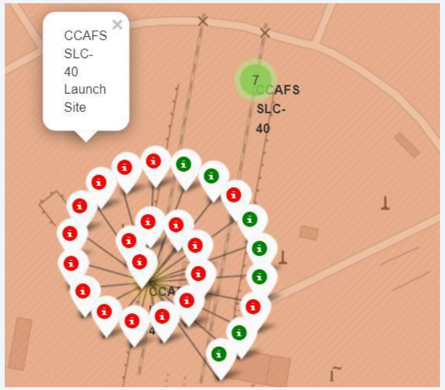


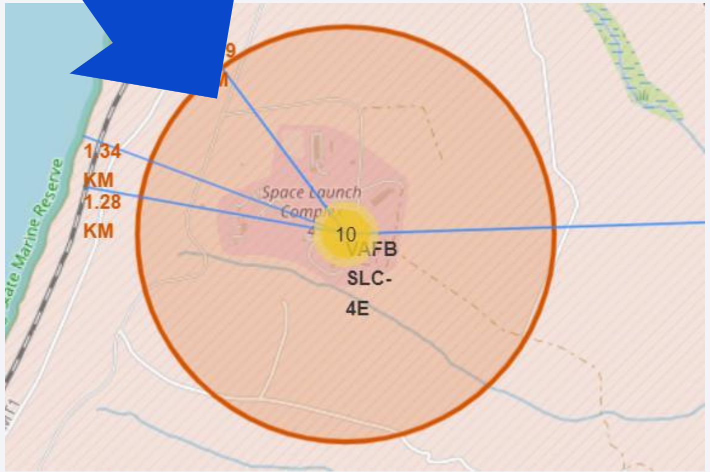
Fig. 5 – CCAFS SLC-40

SpaceX Falcon9: Launch Site Proximity Distance

In Figure 1, Launch Site VAFB SLC 4E displays proximity sites on the map, with the city of Lompoc positioned farther away compared to other proximities like the coastline, railroad, and highway. A marker indicates the city's distance from the Launch Site (14.09 km).

Figure 2 zooms in on proximities such as the coastline, railroad, and highway, providing their respective distances from the Launch Site.

Cities are strategically distanced from Launch Sites to minimize potential impacts on the public and infrastructure. Launch Sites are strategically located near the coastline, railroad, and highways for easy resource access.

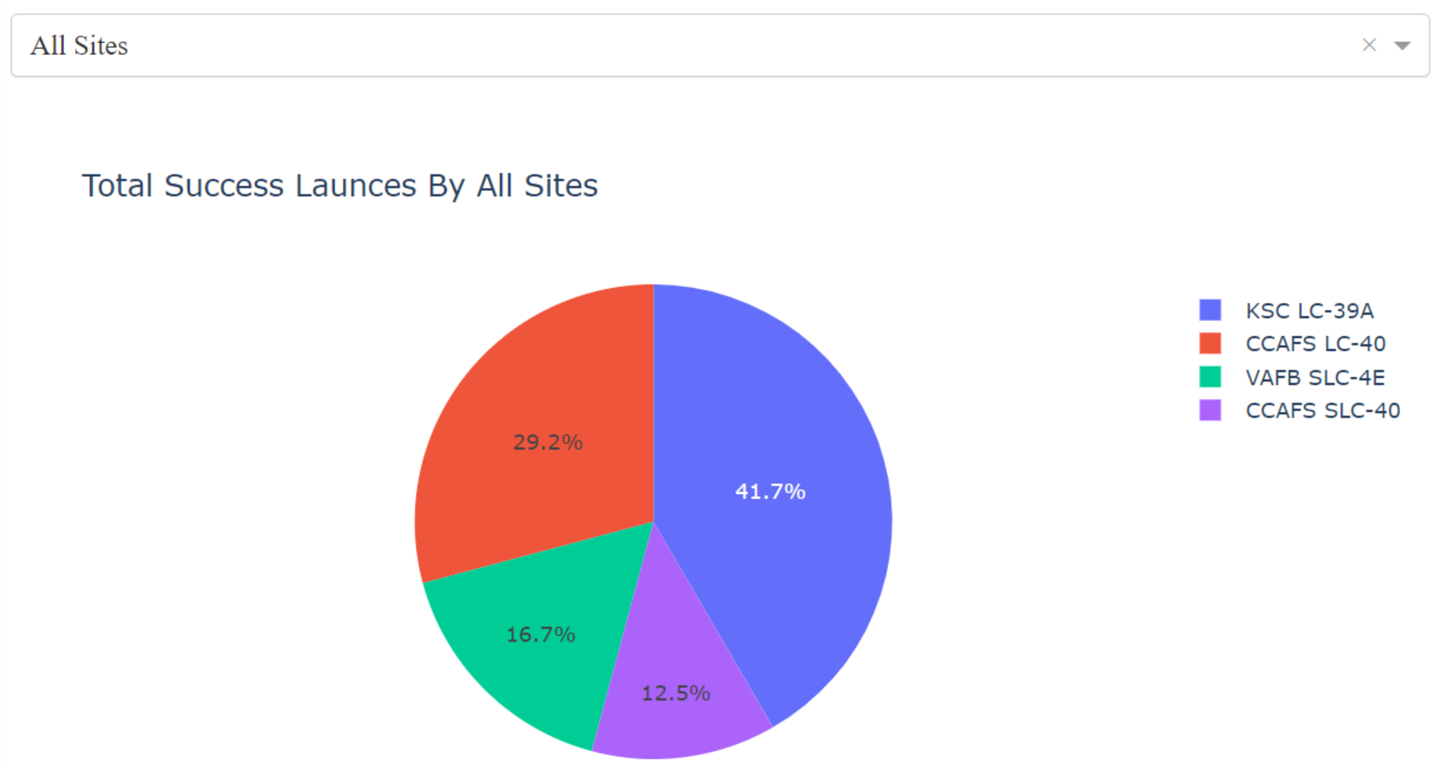


Section 4

Build a Dashboard with Plotly Dash

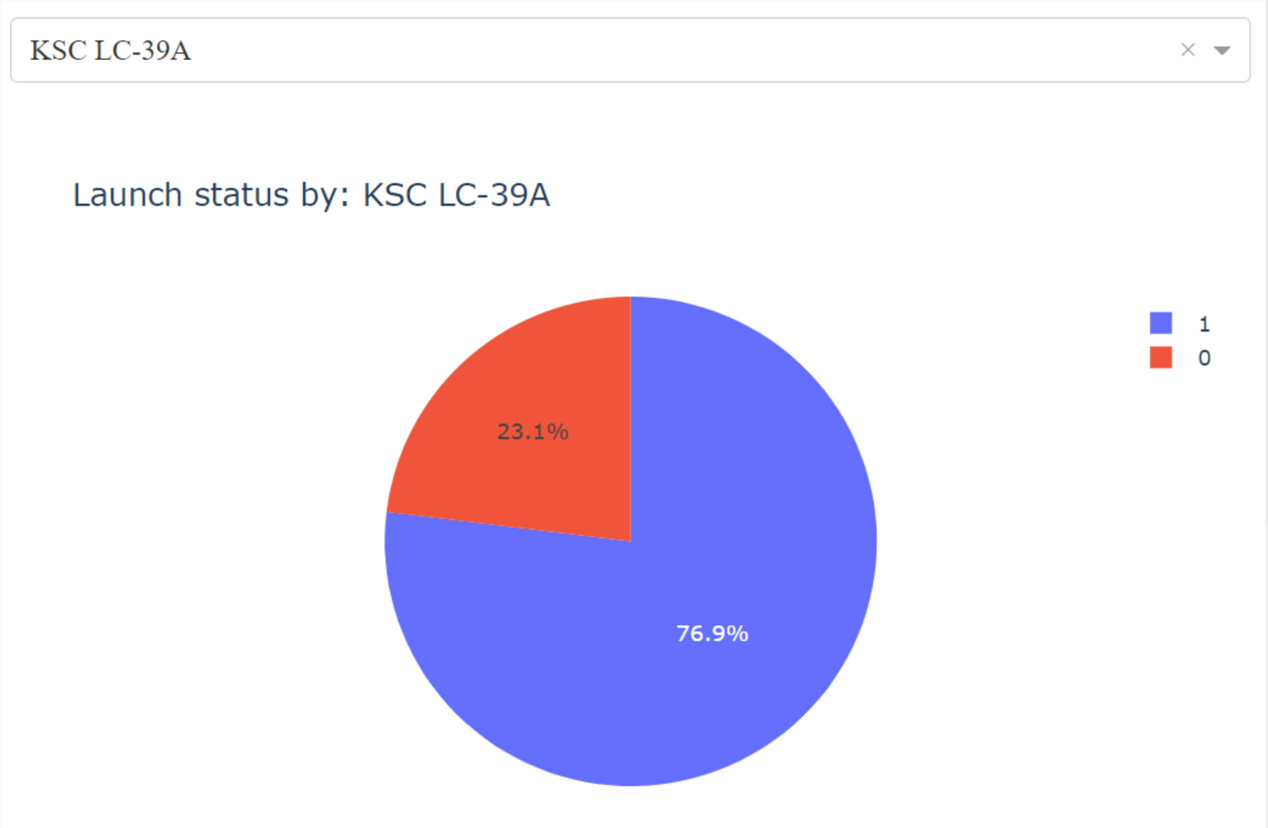
Launch Success for Each Site

'KSC LC 39A' boasts the highest launch success rate among the Launch Sites, while 'CCAFS SLC 40' has the lowest launch success rate.



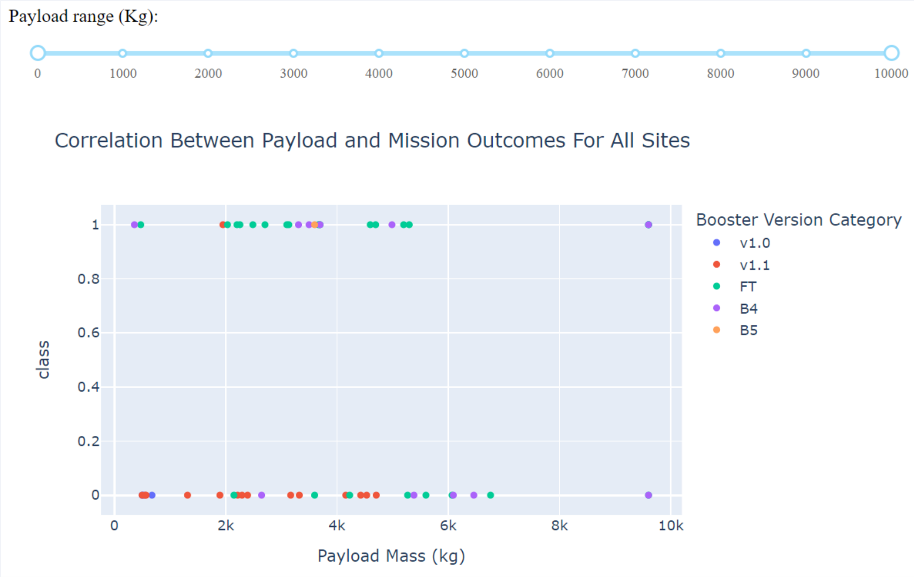
Launch Site with the Highest Success Ratio

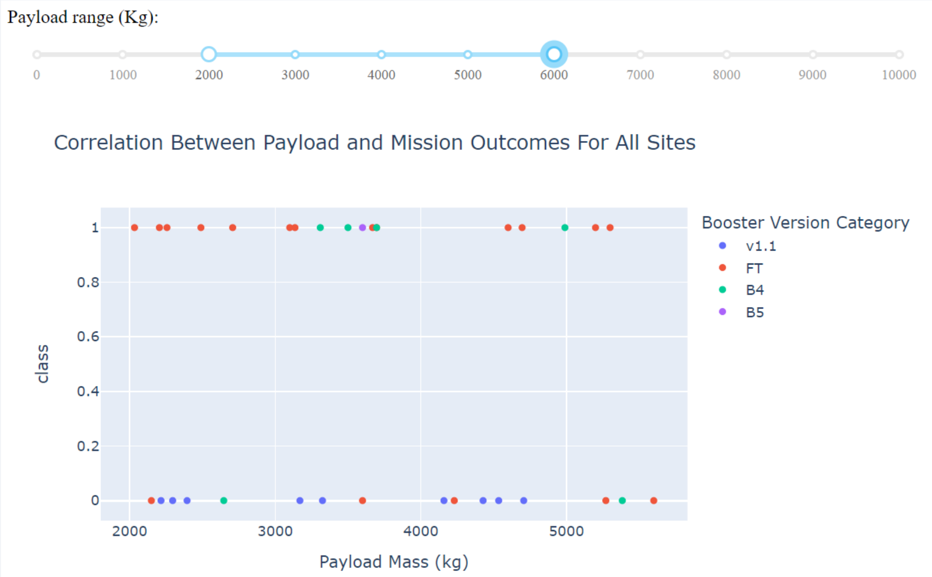
The KSC LC 39A Launch Site stands out with both the highest launch success rate and count. The launch success rate for this site is an impressive 76.9%, while the failure rate is 23.1%.



Payload vs. Launch Outcome

The majority of successful launches fall within the payload range of 2000 to approximately 5500. Among the booster version categories, 'FT' has the highest number of successful launches. Notably, the booster version 'B4' is the only one with a successful launch when the payload is greater than 6000 kg.



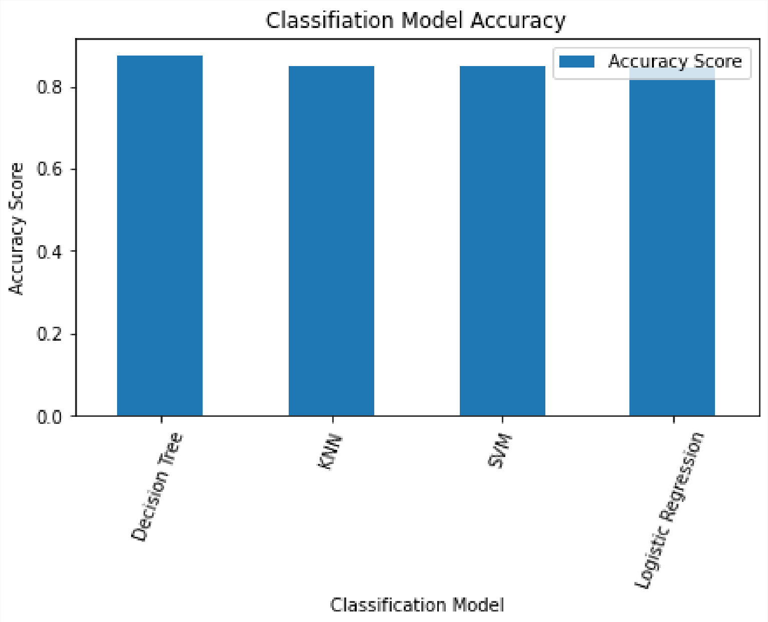


Section 5

Predictive Analysis (Classification)

Classification Accuracy

The Decision Tree algorithm stands out with the highest accuracy score of 0.8750, as indicated by both the scores and the bar chart. Interestingly, the accuracy score on the test data is consistent across all classification algorithms, registering at 0.8333. Given the close proximity of accuracy scores and identical test scores, further model tuning may be explored with a broader dataset to enhance performance.

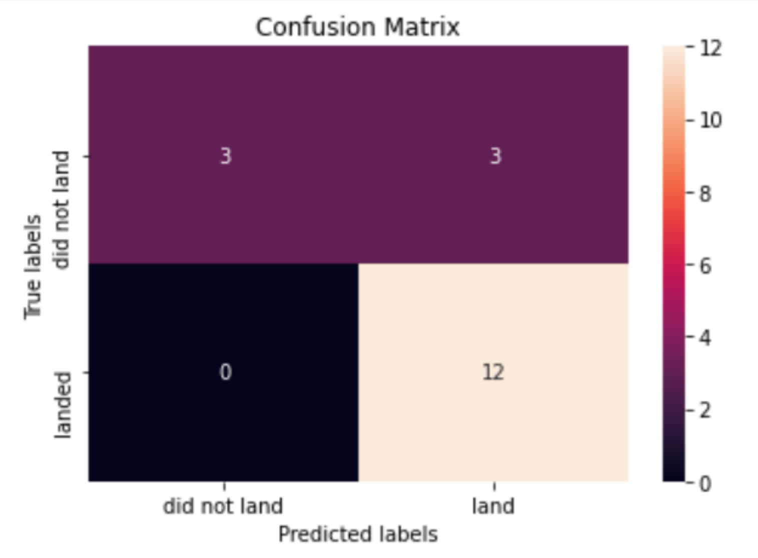


	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

Confusion Matrix

The confusion matrix remains consistent across all models (LR, SVM, Decision Tree):

- The classifier made 18 predictions.
- Out of these, 12 scenarios were correctly predicted as successful landings (True positive).
- In 3 scenarios (top left), the classifier correctly predicted no landing, and indeed, there was no landing (True negative).
- In 3 scenarios (top right), the classifier incorrectly predicted a successful landing, but it did not happen (False positive).
- Overall, the classifier is correct about 83% of the time ((True Positives + True Negatives) / Total), with a misclassification or error rate of approximately 16.5% ((False Positives + False Negatives) / Total).



Conclusions

- **First Stage Landing Success Correlation** - As the number of flights increases, there is a higher likelihood of a successful first stage landing.
- **Payload Mass and Success Rates** - While success rates appear to increase with higher payload masses, there is no clear correlation between payload mass and success rates.
- **Temporal Success Trend** - Over the period from 2013 to 2020, the launch success rate increased by approximately 80%.
- **Launch Site Performance** - 'KSC LC 39A' emerges as the Launch Site with the highest success rate, while 'CCAFS SLC 40' has the lowest success rate.
- **Orbit Analysis** - Orbits ES L1, GEO, HEO, and SSO exhibit the highest launch success rates, with GTO having the lowest success rate.
- **Strategic Launch Site Locations** - Launch sites are strategically positioned away from cities and closer to coastline, railroads, and highways.
- **Machine Learning Model Performance** - The Decision Tree classification model outperforms others with an accuracy of approximately 87.5%. However, when scored on the test data, all models achieve an accuracy of about 83%. Further data may be necessary to fine-tune the models for potential improvements.

These insights provide a comprehensive understanding of SpaceX's launch outcomes, emphasizing factors influencing success rates and the performance of machine learning models in predicting these outcomes.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!