IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Juan P. Sierra Fdez
June 2024

Outline

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

Executive Summary

Objective: Analyze SpaceX Falcon 9 data and use Machine Learning to predict first stage landing success, aiding decision-making for competing space agencies.

Methodologies:

- Data collection: API integration and web scraping.
- Data transformation: Rigorous data wrangling.
- Analysis: SQL queries and data visualization.
- Tools: Interactive maps (Folium) and dashboards (Plotly Dash).
- Prediction: Machine Learning model for first stage landing success.

Findings:

- Data analysis results.
- Visualizations and interactive dashboards.
- Predictive model performance.

Introduction

Project Background:

With the rise of private space travel, the space industry is becoming more accessible. Launch costs remain a significant barrier for new entrants.

SpaceX's ability to reuse Falcon 9 first stages significantly reduces launch costs ($62 million per launch compared to competitors' $165 million).

Objectives:

Predict the success of SpaceX Falcon 9 first stage landings.

Analyze the impact of variables like launch site, payload mass, and booster version on landing outcomes.

Investigate correlations between launch sites and landing success rates.

Section 1

Methodology

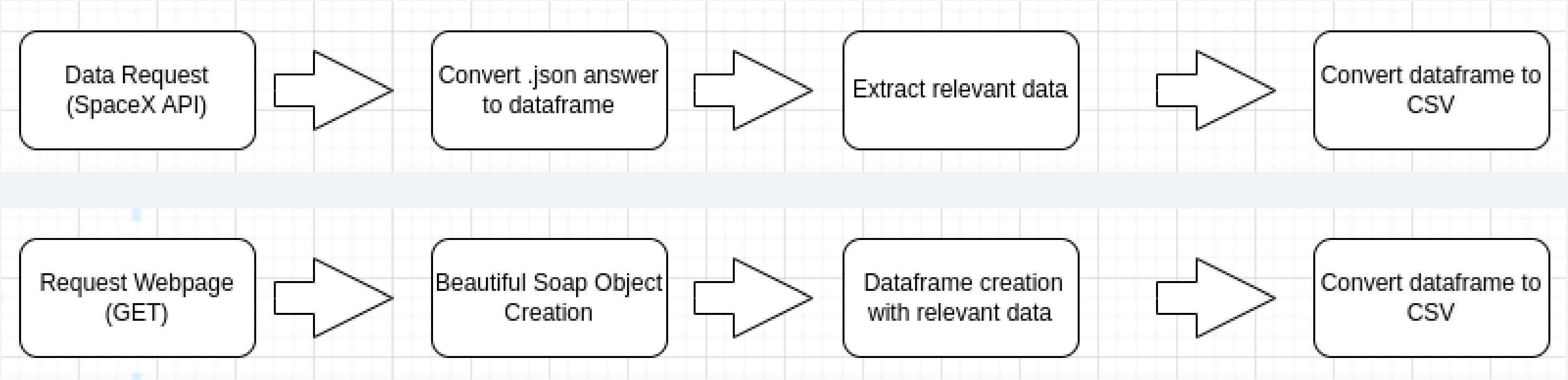
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX API
  - Falcon launch records scrapped from the wikipedia
- Perform data wrangling
  - Performed by assigning labels (1-successful, 0-unsuccessful) for training the supervised models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - We categorized mission outcomes into 'Class' labels (0 for Unsuccessful, 1 for Successful), standardized the data, and split it into training and test sets. We then evaluated Logistic Regression, SVM, Decision Trees, and KNN algorithms using the test data to determine the most effective classification model.

Data Collection

Firstly, we utilized SpaceX's API to directly access and retrieve structured data regarding their launch history. Secondly, we performed web scraping on Wiki pages to extract relevant launch data that was not available through the API. These combined efforts ensured comprehensive data coverage, enabling thorough analysis and insights generation for our project.



Data Collection – SpaceX API

1. Request, data normalization and dataframe

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

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

```
# Use json_normalize meethod to convert the json
data = pd.json_normalize(response.json())
```

2. Lists of global variables

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

3. Use of helper functions for relevant data

4. Dataset creation by means of 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}
```

5. Dataframe creation and exclusion of Falcon 1 launches

```
# Create the data
df_launch = pd.DataFrame(launc_dict)

data_falcon9 = df_launch[df_launch['BoosterVersion']!= 'Falcon 1']

data_falcon9.t0_csv('dataset_part\_1.csv',index=False)
```

https://github.com/psierrajs/capstone_project.git

Data Collection - Scraping

1. Request HTML page

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

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

2. Creation of Beautiful Soap object

```
soup = BeautifulSoup(html_data,"html.parser")
```

3. Extraction of relevant columns

```
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
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)
```

4. Creation of empty Dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each vc
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']=[]
```

5. Fill in the dictionary with extracted data

```
def date_time(table_cells):

def booster_version(table_cells):

def landing_status(table_cells):

def get_mass(table_cells):
```

6. Transform dicionary into dataframe

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

https://github.com/psierrajs/capstone_project.git

Data Wrangling

- Performed Exploratory Data Analysis (EDA) to identify patterns in the data and define labels for training supervised models.

- The dataset contained various mission outcomes, converted into training labels: 1 for successful booster landings and 0 for unsuccessful landings. The following landing scenarios were considered to create labels:

  - True Ocean: Successfully landed in a specific ocean region
  - False Ocean: Unsuccessfully landed in a specific ocean region
  - RTLS: Successfully landed on a ground pad
  - False RTLS: Unsuccessfully landed on a ground pad
  - True ASDS: Successfully landed on a drone ship
  - False ASDS: Unsuccessfully landed on a drone ship

https://github.com/psierrajs/capstone_project.git

Data Wrangling

1. load dataset

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

2. Explore patterns

i. Calculate the number of launches on each site

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

```
CCAFS SLC 40     55
KSC LC 39A       22
VAFB SLC 4E      13
```

ii. Calculate the number and occurrence of each orbit

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

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
GEO       1
HEO       1
SO        1
ES-L1     1
```

iii. Calculate number/occurrence of mission outcomes per orbit type

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

2. Create landing label

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

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

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

https://github.com/psierrajs/capstone_project.git

EDA with Data Visualization

The following charts were created to gain deeper insights into the dataset as part of the Exploratory Data Analysis (EDA):

1) Scatter Plot:
    1) Allows to study the correlation between two variables.
    2) Follows a list of charts used:
        1) Flight number vs launch site
        2) Payload vs Launch site
        3) Flight Number vs Orbit Type
        4) Payload vs orbit type
    3) Bar Chart:
        1) Used to compare values of a variable
        2) The only bar chart used was success rate vs orbit type
    4) Line Chart:
        1) Shows progression of values through time
        2) The only line chart used was average launch success vs year

https://github.com/psierrajs/capstone_project.git

EDA with SQL

The following queries where performed:

- Names of the launch sites
- Five record of launch sited staring with ‘CCA’
- Payload carried by boosters launches by NASA
- Average payload carried by booster F9 v1.1
- Date when the first landing outcome in ground pad was successfull
- Names of the boosters successfull in drone ship and with payload mass between 4000 and 6000
- Total number of successful and failed missions
- Names of booster versions which carried the highest payload mass
- Failed landing outcomes in drone ship, their booster versions, and launch site names in 2015
- Classify landing outcomes according to their number in descending order

Build an Interactive Map with Folium

Folium is a Python library designed to create interactive maps by leveraging the capabilities of the Leaflet.js library. It enables users to visualize geospatial data effectively by allowing the addition of markers, popups, and layers to the maps. With Folium, users can interact with the map elements, such as zooming and panning, to explore data in a more intuitive and engaging way. This tool is particularly useful for mapping locations, visualizing geographical distributions, and overlaying various data sets on maps to enhance data analysis and presentation.

In order to fullfill the requirements of the task, the following was added to the map:
- Maks to signal launching sites

- MarkerCluster to mark successful and failed launches using green and red colored markers respectively

- Calculation of distances between a lauch site and geographical highlights

https://github.com/psierrajs/capstone_project.git

Build a Dashboard with Plotly Dash

I built a Plotly Dash web application to perform interactive visual analytics on SpaceX launch data in real-time. The dashboard includes a Launch Site drop-down, a Pie Chart, a Payload range slider, and a Scatter chart. The Launch Site drop-down input component allows users to filter the dashboard visuals by all launch sites or a specific launch site. The Pie Chart displays total successful launches when 'All Sites' is selected, and shows success and failure counts for a particular site when selected. The Payload range slider facilitates the selection of different payload ranges to help identify visual patterns. Additionally, the Scatter chart helps observe how payload correlates with mission outcomes for selected sites, with each scatter point color-labeled by booster version to indicate mission outcomes with different boosters.

The purpose of this procedure was to obtain information necessary to answer questions regarding the largest successfull launches, highest launch success rate, range of payloads with the highest range of success rate and booster version with the highest success rate.

https://github.com/psierrajs/capstone_project.git

Build a Dashboard with Plotly Dash

I built a Plotly Dash web application to perform interactive visual analytics on SpaceX launch data in real-time, featuring a Launch Site drop-down, Pie Chart, Payload range slider, and Scatter chart. The Launch Site drop-down allows filtering by all or specific launch sites, while the Pie Chart shows total successful launches for all sites or success and failure counts for a selected site. The Payload range slider enables the selection of different payload ranges to identify visual patterns, and the Scatter chart reveals how payload correlates with mission outcomes, with color-labeled scatter points indicating booster versions. The dashboard helped answer key questions: KSC LC-39A had the largest number of successful launches (10) and the highest success rate (76.9%). The payload range with the highest success rate was 2000-5000 kg, while 0-2000 kg and 5500-7000 kg had the lowest. The F9 Booster version FT had the highest success rate.

Predictive Analysis (Classification)

1. Load SpaceX dataset (csv) in to a Dataframe and create NumPy array from the column class in data

```
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object
et_part_2.csv")
```

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

2. Data Standardization

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

3. Train/test/split X and Y in to training and test data sets.

```
# 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
( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

4. Create and refine Models based on following classification Algorithms: (below is LR example)
   i. Create Logistic Regression object and then create a GridSearchCV object
   ii. Fit train data set in to the GridSearchCV object and train the Model

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

   iii. Find and display best hyperparameters and accuracy score

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_
print("accuracy :",logreg_cv.best_score_)
```

   iv. Check the accuracy on the test data by creating a confusion matrix

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

   v. Repeat above steps for Decision Tree, KNN, and SVM algorithms

5. Optimize for the best perfoming model

```
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)]})
```

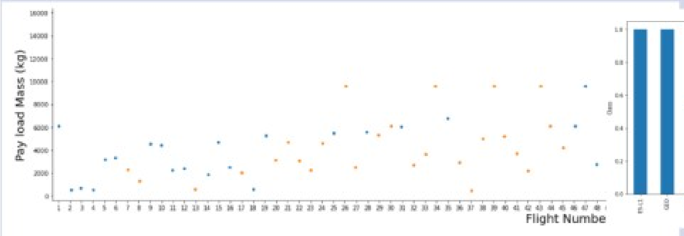
```
i = Model_Performance_df['Accuracy Score'].idxmax()
print('The best performing alogrithm is '+ Model_Performance_df['Algo Type'][i]
+ ' with score ' + str(Model_Performance_df['Accuracy Score'][i]))
```

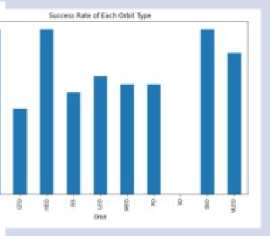
```
The best performing alogrithm is Decision Tree with score 0.875
```

	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

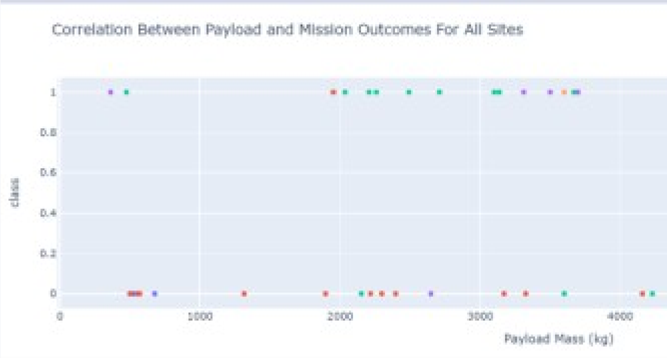
Results

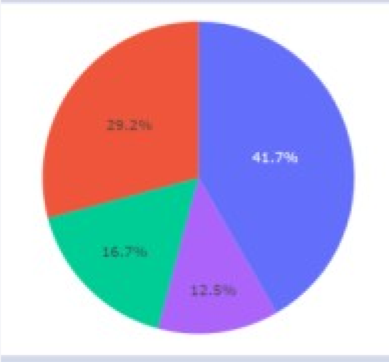
- Exploratory data analysis results





Interactive analytics demo in screenshots





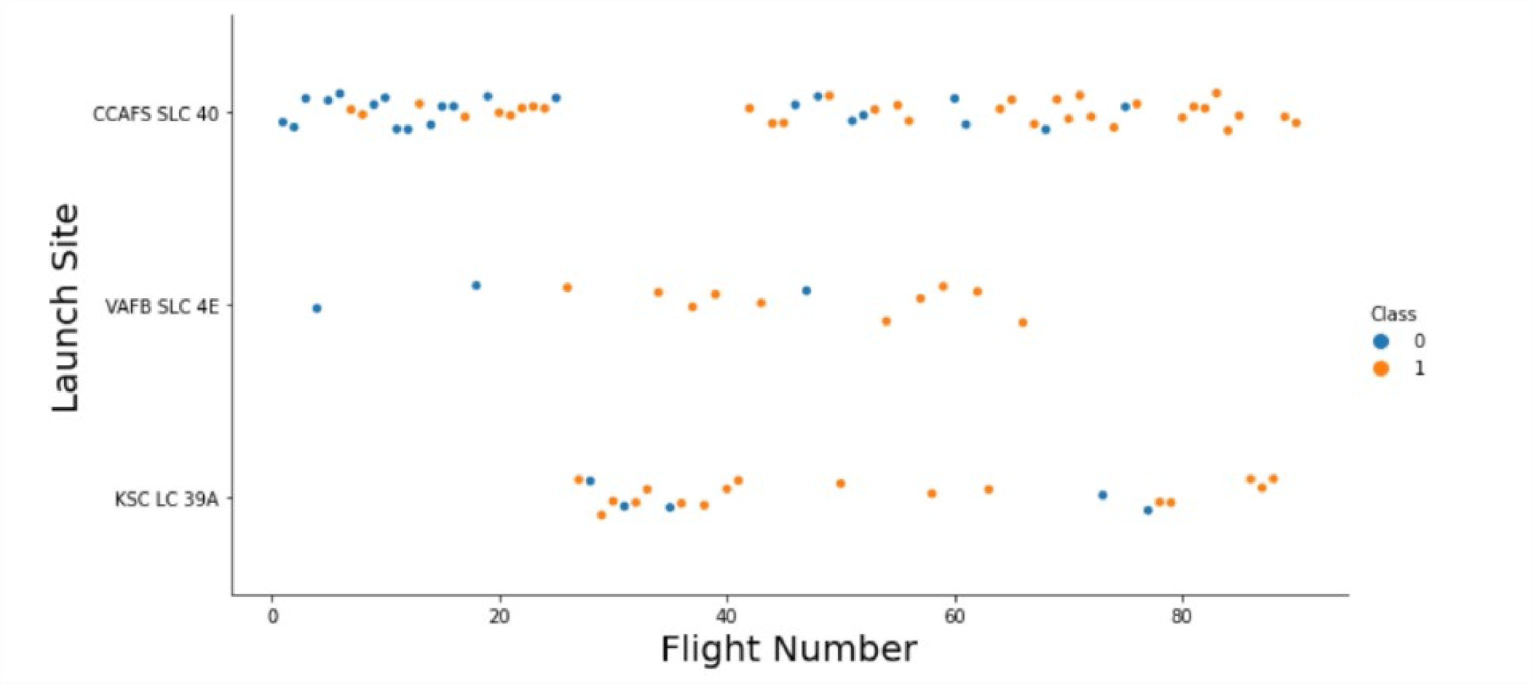
Predictive analysis results

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

Section 2

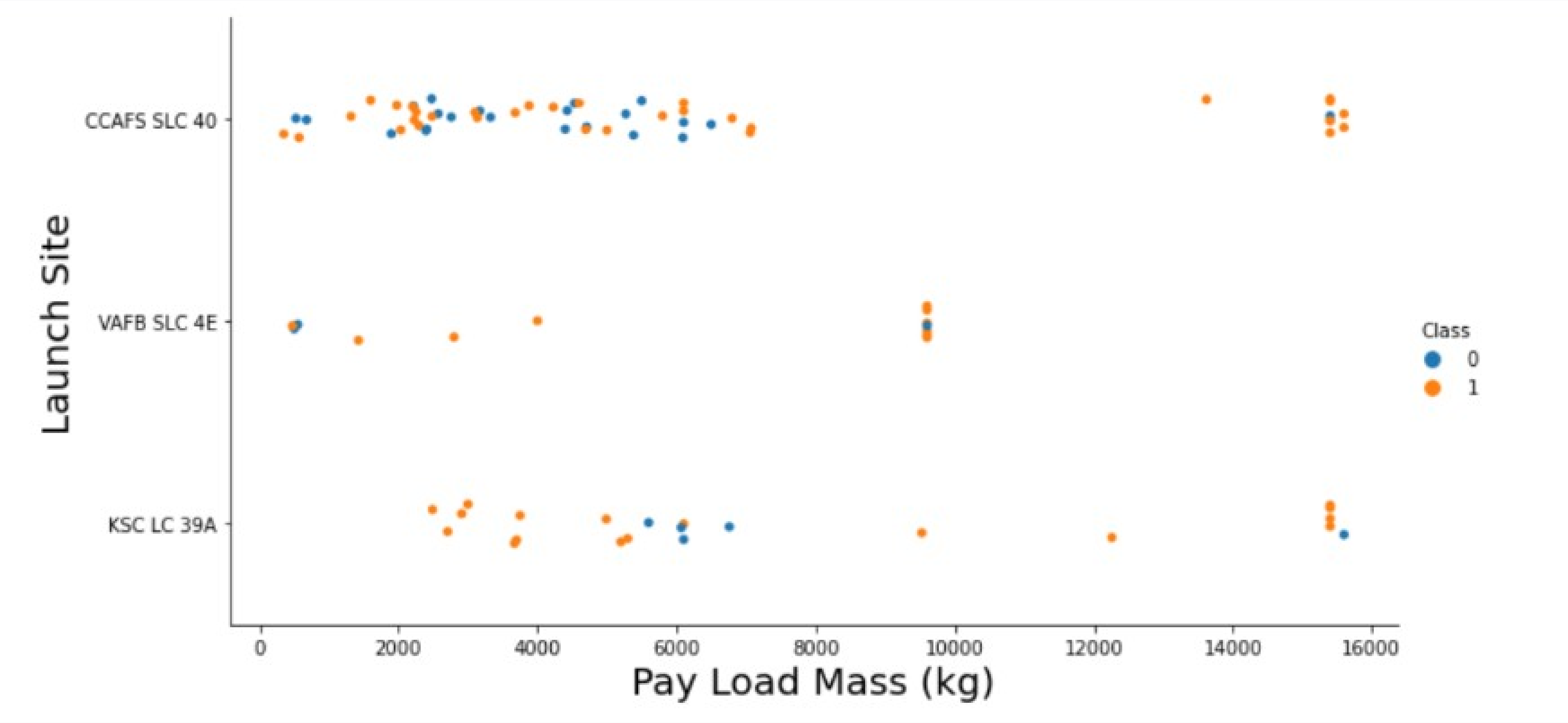
Insights drawn from EDA

Flight Number vs. Launch Site



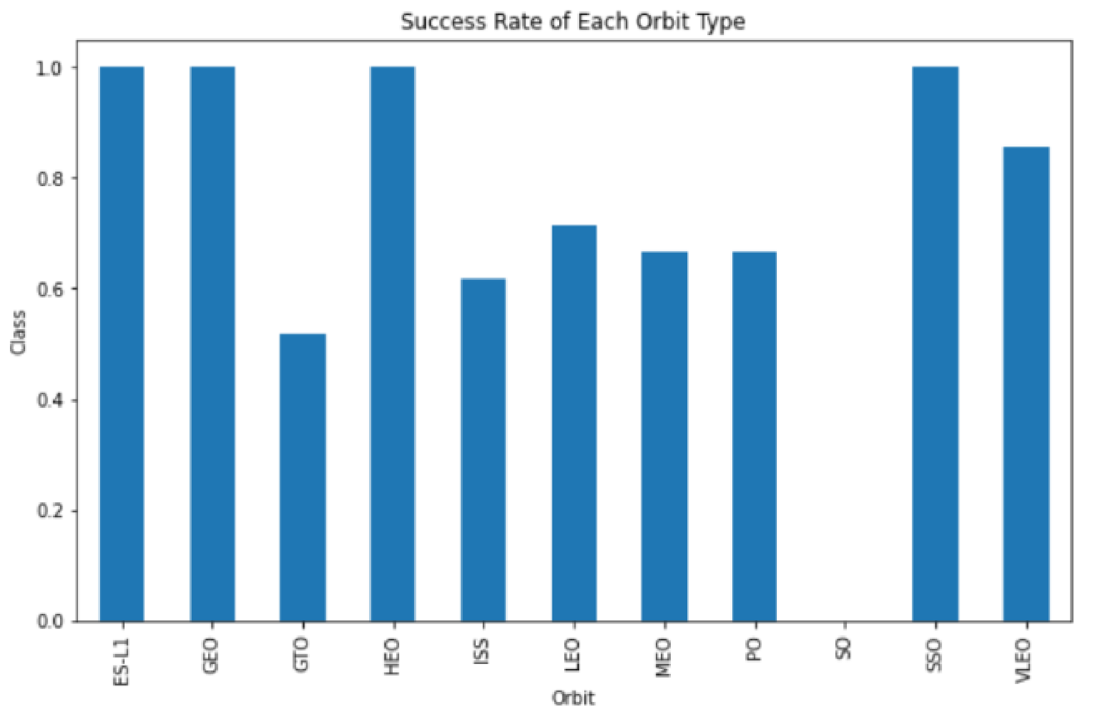
The results presented in the figure above suggest that the success rate if class 1 increases as the number of flights increase.
The launch site with code KSC LC 39A requires a higher number of flights to reach a successful launch (25) compared to the rest.

Payload vs. Launch Site



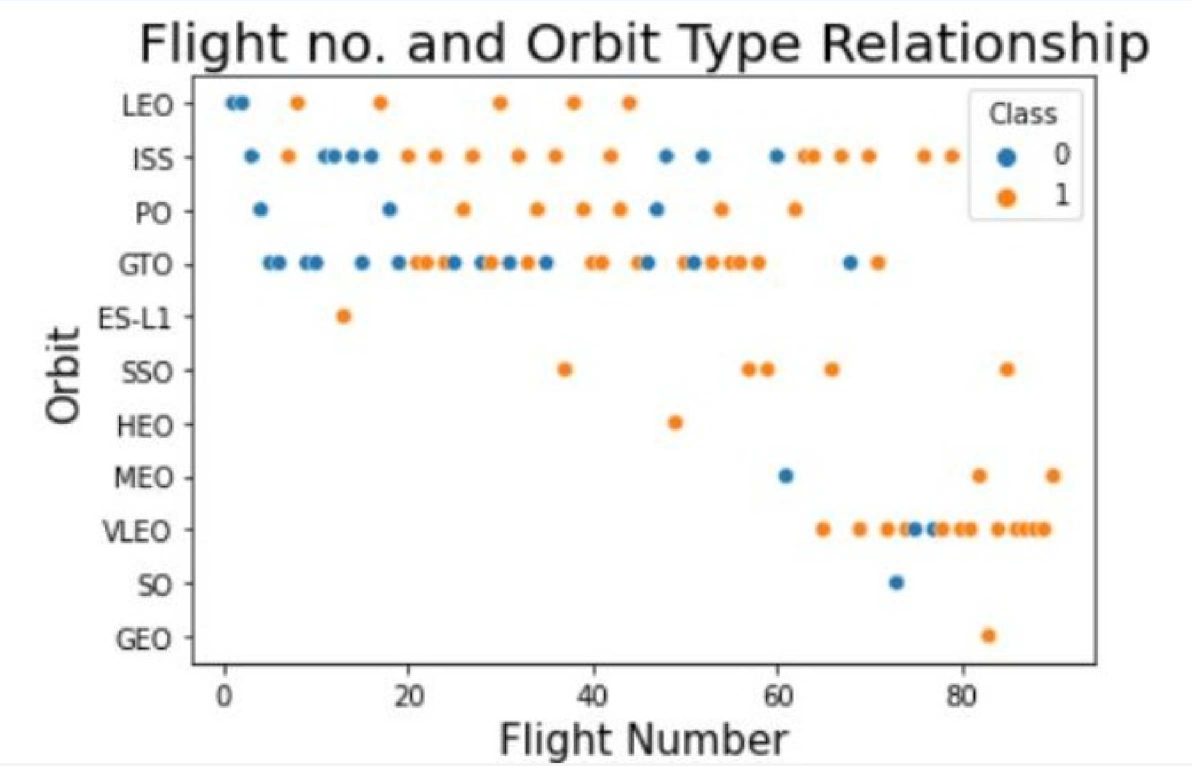
For the launch site 'VAFB SLC 4E', no rockets have been launched with a payload exceeding 10,000 kg. Additionally the success rate (Class=1) for the launch site 'VAFB SLC 4E' tends to rise as the payload mass increases. Finally, no distinct correlation or pattern exists between the launch site and payload mass.

Success Rate vs. Orbit Type



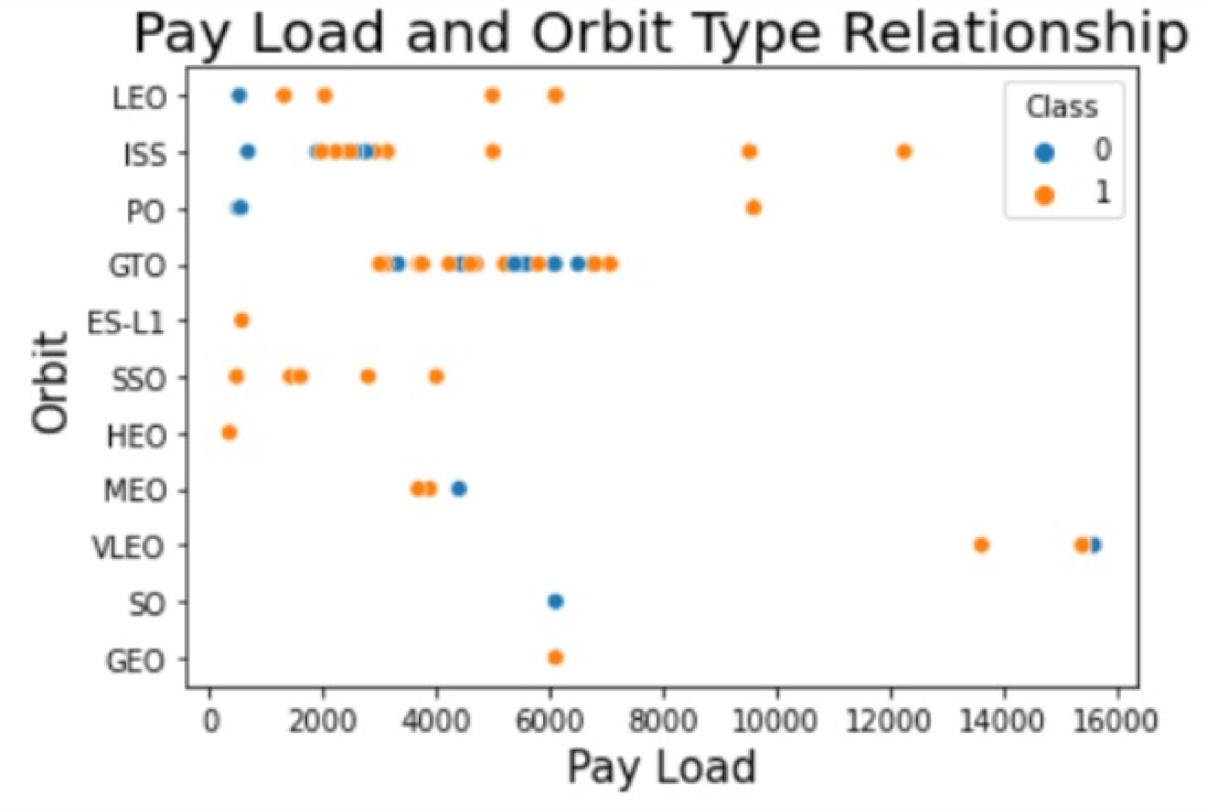
Orbits ES-LI, GEO, HEO, and SSO exhibit the highest success rates whereas the GTO orbit has the lowest success rate.

Flight Number vs. Orbit Type



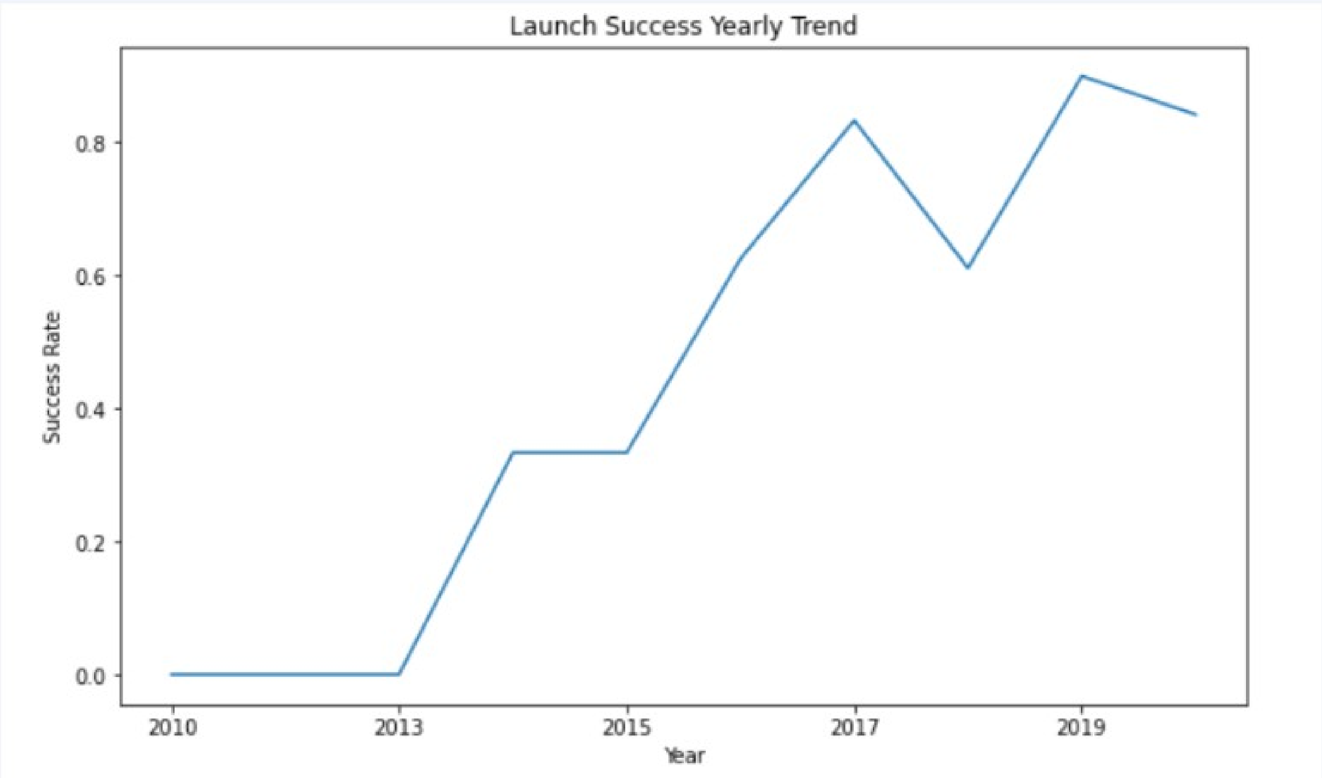
- For the VLEO orbit, the first successful landing (Class=1) occurs only after more than 60 flights.
- For most orbits, including LEO, ISS, PO, SSO, MEO, and VLEO, the rate of successful landings tends to increase with the number of flights.
- There is no discernible relationship between flight number and orbit for the GTO orbit.

Payload vs. Orbit Type



- Successful landing rates (Class=1) seem to rise with payload for LEO, ISS, PO, and SSO orbits.
- There is no clear pattern between payload and successful or unsuccessful landing for the GEO orbit.

Launch Success Yearly Trend



- The success rate (Class=1) saw an approximately 80% increase from 2013 to 2020.
- Success rates remained unchanged between 2010 and 2013, as well as between 2014 and 2015.
- Success rates declined between 2017 and 2018, and again between 2019 and 2020.

All Launch Site Names

To find the names of the unique launch sites we used the following query:

```
select distinct Launch_Site from spacextbl
```

We included ‘disctinct’ to avoid douplicated results and extracted the data from the table spacextbl.

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

Launch Site Names Begin with 'CCA'

To find 5 records where launch sites begin with `CCA` we used the following query:

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

The results are as follows:

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

To calculate the total payload carried by boosters from NASA:

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

The query uses sum to add the contents of the column PAYLOAD_MASS_KG_ in the table spacextbl provided that the customer is NASA. Its execution returns:

45596

Average Payload Mass by F9 v1.1

The query used to calculate the average payload mass carried by booster version F9 v1.1:

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

‘avg’ keyword is used to return the average of the results obtaiuned in the query. Notice how we are including the restriction that ensures the booster version is F9 v1.1.

The query returns 2928.

First Successful Ground Landing Date

To find the dates of the first successful landing outcome on ground pad we used the following query:

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

In this query, min(Date) retrieves the earliest or oldest date from the 'Date' column where the first successful landing on a ground pad occurred. The WHERE clause specifies conditions to retrieve dates where the 'Landing_Outcome' value equals 'Success (ground pad)'.

The query returns 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

To list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 we used the following query:

```
select Booster_Version from spacextbl where (PAYLOAD_MASS__KG_> 4000 and PAYLOAD_MASS__KG_ < 6000)
and (Landing__Outcome = 'Success (drone ship)');
```

The 'AND' operator in the WHERE clause filters and returns booster versions where both conditions specified in the WHERE clause are true simultaneously.

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

Total Number of Successful and Failure Mission Outcomes

To calculate the total number of successful and failure mission outcomes we used:

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

The 'GROUP BY' keyword organizes identical data in a column into groups. In this instance, the number of mission outcomes by type are grouped in the 'counts' column.

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

Boosters Carried Maximum Payload

To kist the names of the booster which have carried the maximum payload mass the following query was used:

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

The subquery uses the 'MAX' keyword to retrieve the maximum payload mass from the payload_mass column. In the main query, booster versions and their respective payload masses are returned, specifically where the payload mass equals the maximum value of 15600 kilograms.

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

To list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 the following query was used:

```
select Landing__Outcome, Booster_Version, Launch_Site from spacextbl where Landing__Outcome = 'Failure (drone ship)' and year(Date) = '2015'
```

The query retrieves landing outcomes, booster versions, and launch sites where the landing outcome was 'failed' on a drone ship in the year 2015. The 'AND' operator in the WHERE clause filters booster versions where both specified conditions are true. The 'YEAR' keyword extracts the year from the 'Date' column. The results identify the launch site as 'CCAFS LC-40' and the booster versions as F9 v1.1 B1012 and B1015, which had unsuccessful landing outcomes on a drone ship in 2015.

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

To rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order we used:

```
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;
```

The 'GROUP BY' keyword organizes data in the 'Landing__Outcome' column into distinct groups based on similar values. Using the 'BETWEEN' and 'AND' keywords, data is filtered to include entries from June 4, 2010, to March 20, 2017. The 'ORDER BY' keyword sorts the 'counts' column in descending order. The query results in a ranked list of landing outcome counts within the specified date range.

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

Space X Falcon9 – Launch Sites Map

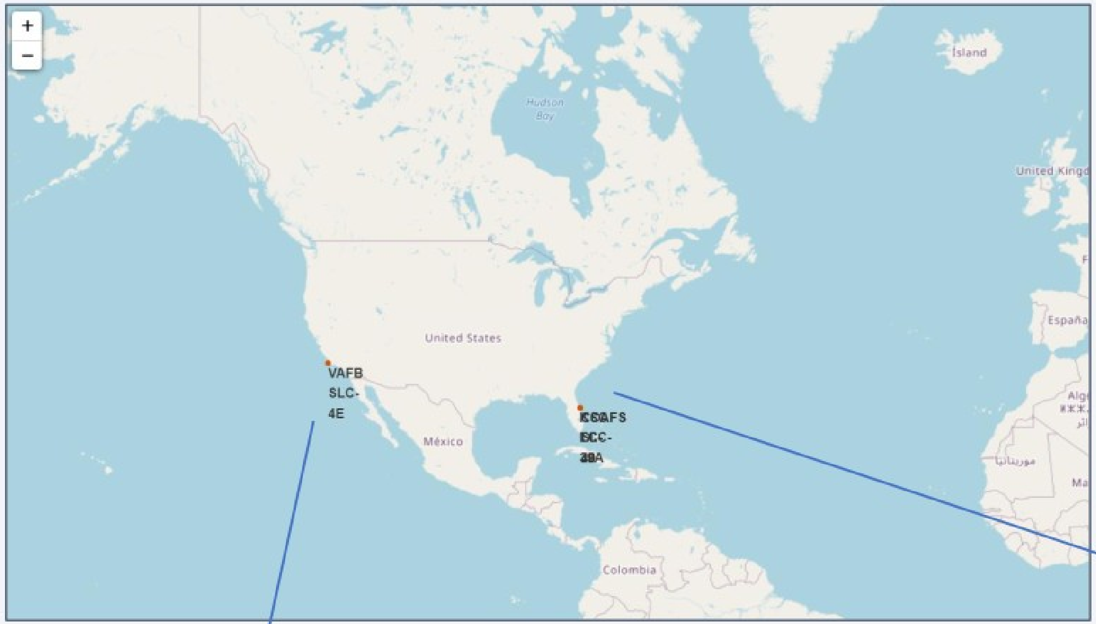


Fig 1 – Global Map

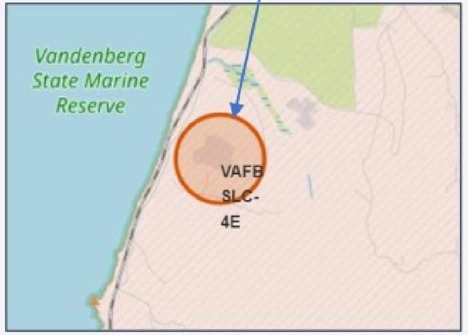


Fig 2 – Zoom 1

Figure 1 on left displays the Global map with Falcon 9 launch sites that are located in the United States (in California and Florida). Each launch site contains a circle, label, and a popup to highlight the location and the name of the launch site. It is also evident that all launch sites are near the coast.

Figure 2 and Figure 3 zoom in to the launch sites to display 4 launch sites:

- VAFB SLC-4E (CA)
- CCAFS LC-40 (FL)
- KSC LC-39A (FL)
- CCAFS SLC-40 (FL)

Fig 3 – Zoom 2

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

Fig 1 – US map with all Launch Sites

- Figure 1 is the US map with all the Launch Sites. The numbers on each site depict the total number of successful and failed launches
- Figure 2, 3, 4, and 5 zoom in to each site and displays the success/fail markers with green as success and red as failed
- By looking at each site map, KSC LC-39A Launch Site has the greatest number of successful launches

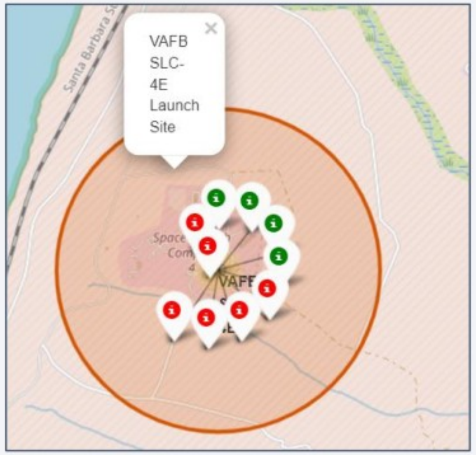


Fig 2 – VAFB Launch Site with success/failed markers

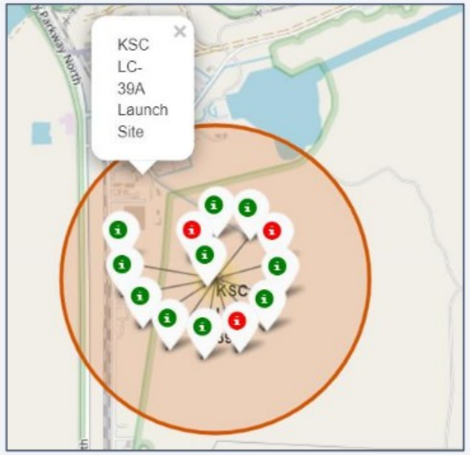


Fig 3 – KSC LC-39A success/failed markers

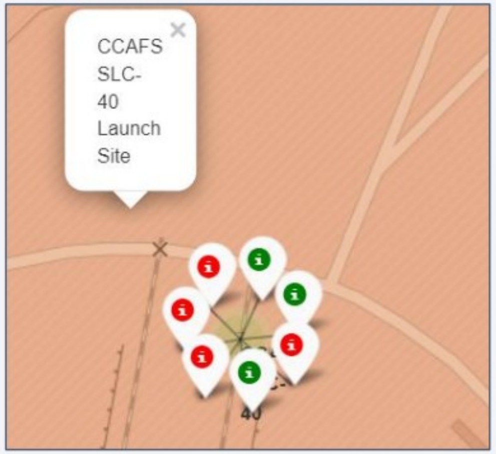


Fig 4 – CCAFS SLC-40 success/failed markers

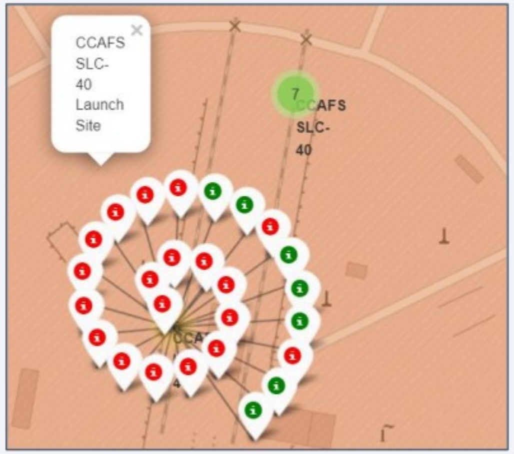


Fig 5 – CCAFS SLC-40 success/failed markers

SpaceX Falcon9 – Launch Site to proximity Distance Map

Fig 1 – Proximity site map for VAFB SLC-4E

Fig 2 – Zoom in for sites – coastline, railroad, and highway

Figure 1 displays all the proximity sites marked on the map for Launch Site VAFB SLC-4E. City Lompoc is located further away from Launch Site compared to other proximities such as coastline, railroad, highway, etc. The map also displays a marker with city distance from the Launch Site (14.09 km)

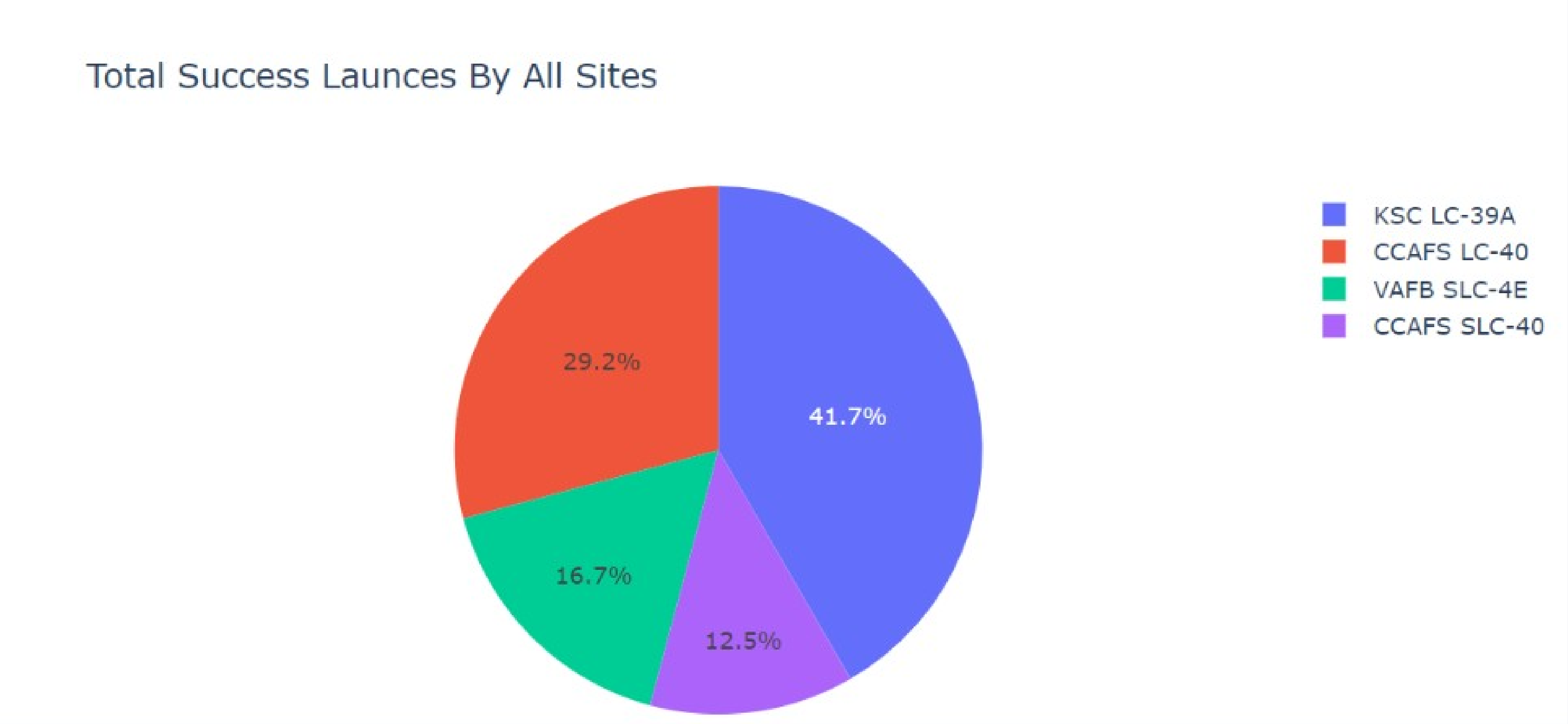
Figure 2 provides a zoom in view into other proximities such as coastline, railroad, and highway with respective distances from the Launch Site

In general, cities are located away from the Launch Sites to minimize impacts of any accidental impacts to the general public and infrastructure. Launch Sites are strategically located near the coastline, railroad, and highways to provide easy access to resources.

Section 4

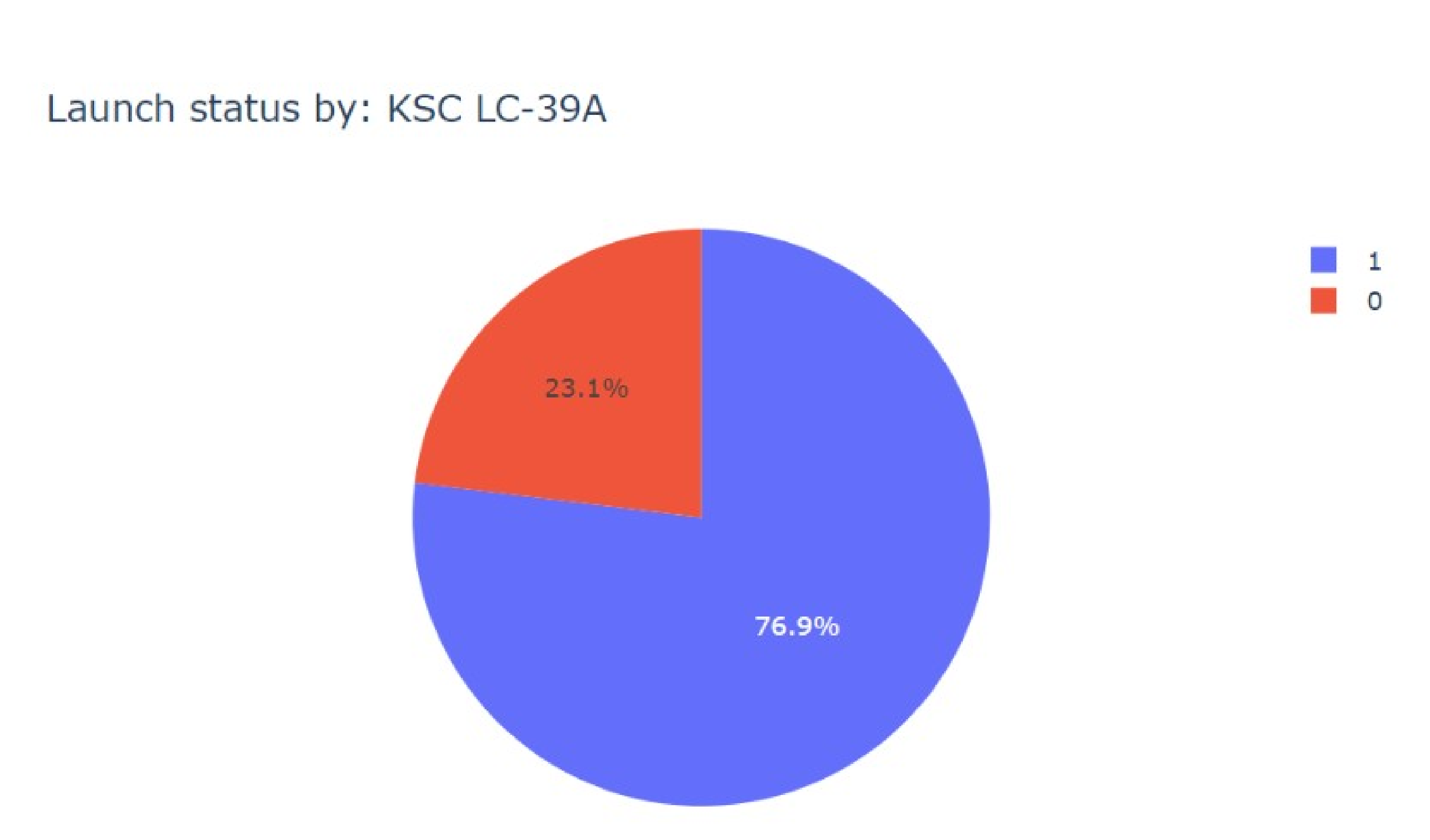
Build a Dashboard with Plotly Dash

Launch Success Counts For All Sites



KSC LC-39A boasts the highest launch success rate among all launch sites, while CCAFS SLC-40 exhibits the lowest.

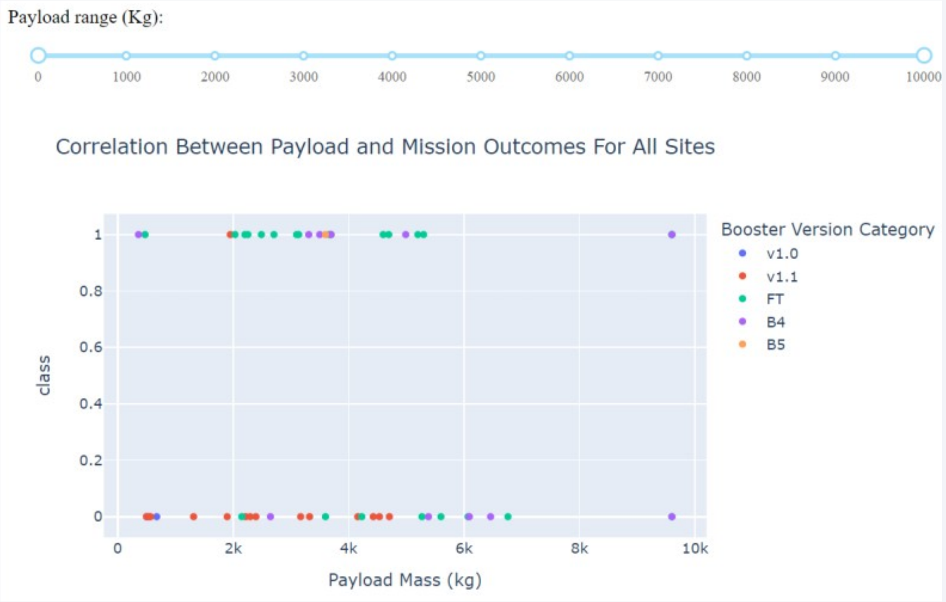
Launch Site with Highest Launch Success Ratio



The KSC LC-39A launch site achieves the highest launch success rate and count:

- Launch success rate: 76.9%
- Launch failure rate: 23.1%

Payload vs. Launch Outcome Scatter Plot for All Sites

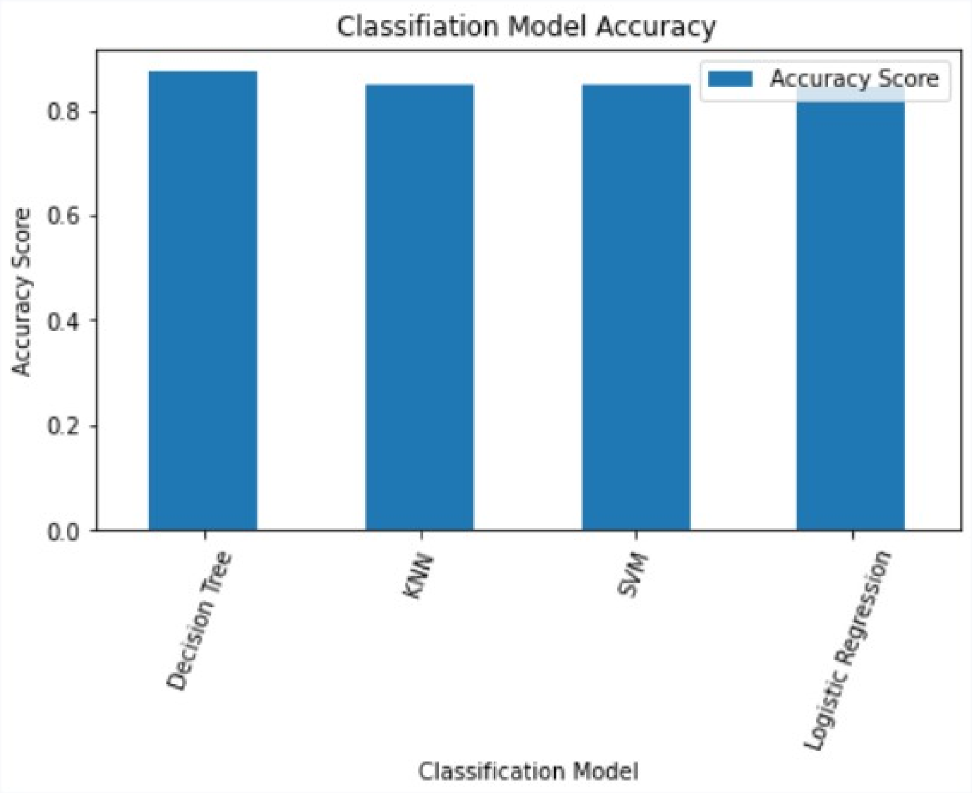


- The most successful launches occur in the payload range from 2000 to about 5500.
- The 'FT' booster version category records the highest number of successful launches.
- The only booster with a successful launch when the payload is greater than 6000 kilograms is 'B4'.

Section 5

Predictive Analysis (Classification)

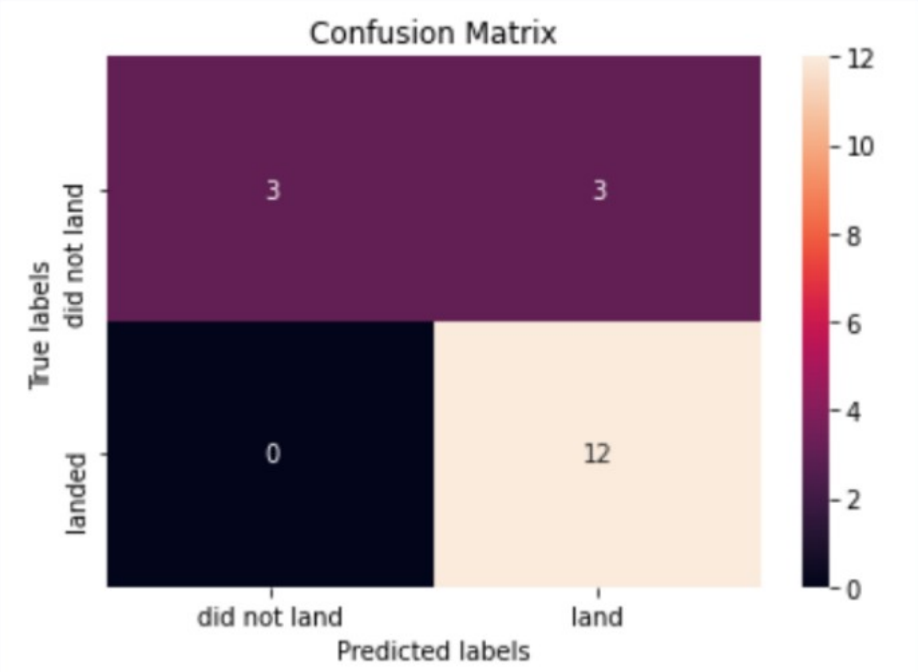
Classification Accuracy



	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

Based on the accuracy scores and as depicted in the bar chart, the Decision Tree algorithm achieves the highest classification score, reaching 0.8750. Interestingly, all classification algorithms show identical accuracy scores of 0.8333 on the test data. Given the minimal variance in accuracy scores among the classification algorithms and the consistent test scores, further improvement might require a larger and more diverse dataset to refine the models.

Confusion Matrix



The confusion matrix is consistent across all models (LR, SVM, Decision Tree, KNN). According to the matrix, the classifier made predictions for a total of 18 scenarios. Among these, 12 predictions of 'Yes' for landing were correct (True positives), while 3 predictions of 'No' for landing were also correct (True negatives). However, 3 scenarios predicted 'Yes' for landing did not succeed (False positives). Overall, the classifier's accuracy rate is approximately 83% (correct predictions divided by total predictions), with a misclassification rate of about 16.5% (incorrect predictions divided by total).

Conclusions

- As flight numbers increase, the likelihood of successful first stage landings also increases.
- Success rates tend to rise with payload increases, but there is no clear correlation between payload mass and success rates.
- Launch success rates saw an 80% increase from 2013 to 2020.
- KSC LC-39A has the highest launch success rate, while CCAFS SLC-40 has the lowest.
- Orbits ES-L1, GEO, HEO, and SSO boast the highest launch success rates, while GTO has the lowest.
- Launch sites are strategically located away from cities and nearer to coastlines, railroads, and highways.
- The top-performing Machine Learning classification model is the Decision Tree, achieving approximately 87.5% accuracy. All models scored about 83% accuracy on test data, suggesting the need for additional data to further refine and potentially improve model performance.

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!