CAN THE FIRST STAGE BE REUSED?

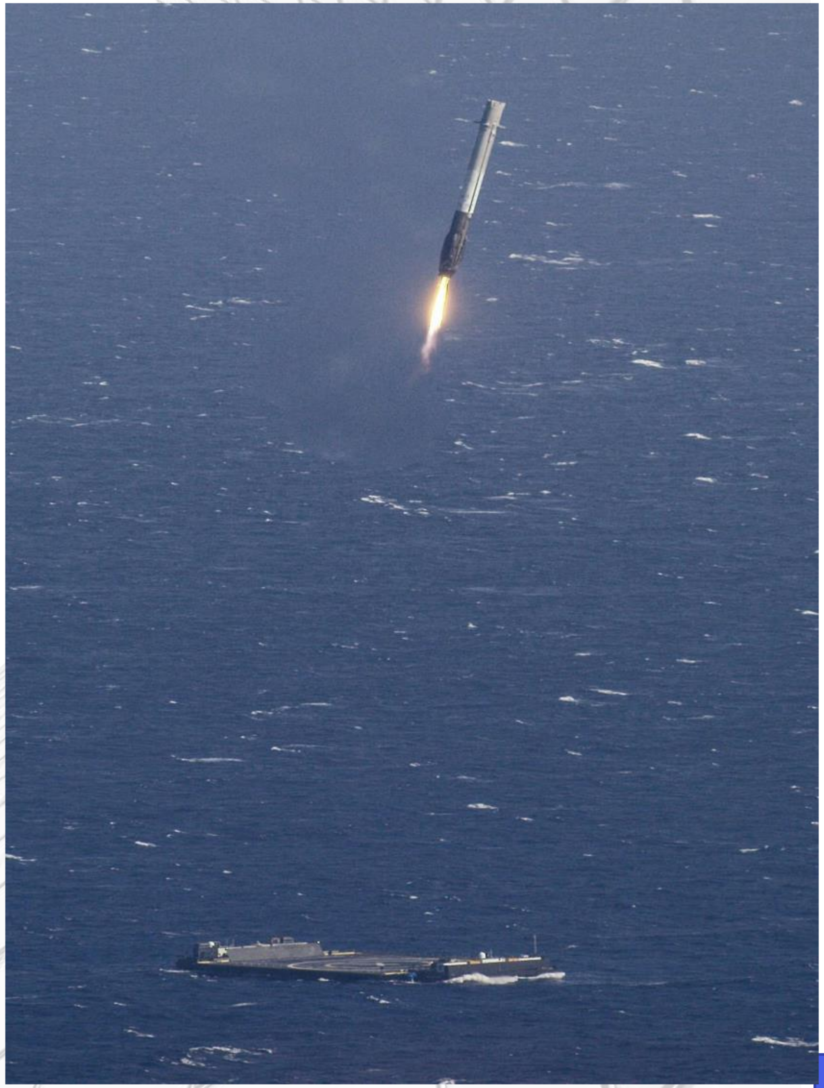
SAURAV JOHN SATHEESH

05/12/2024

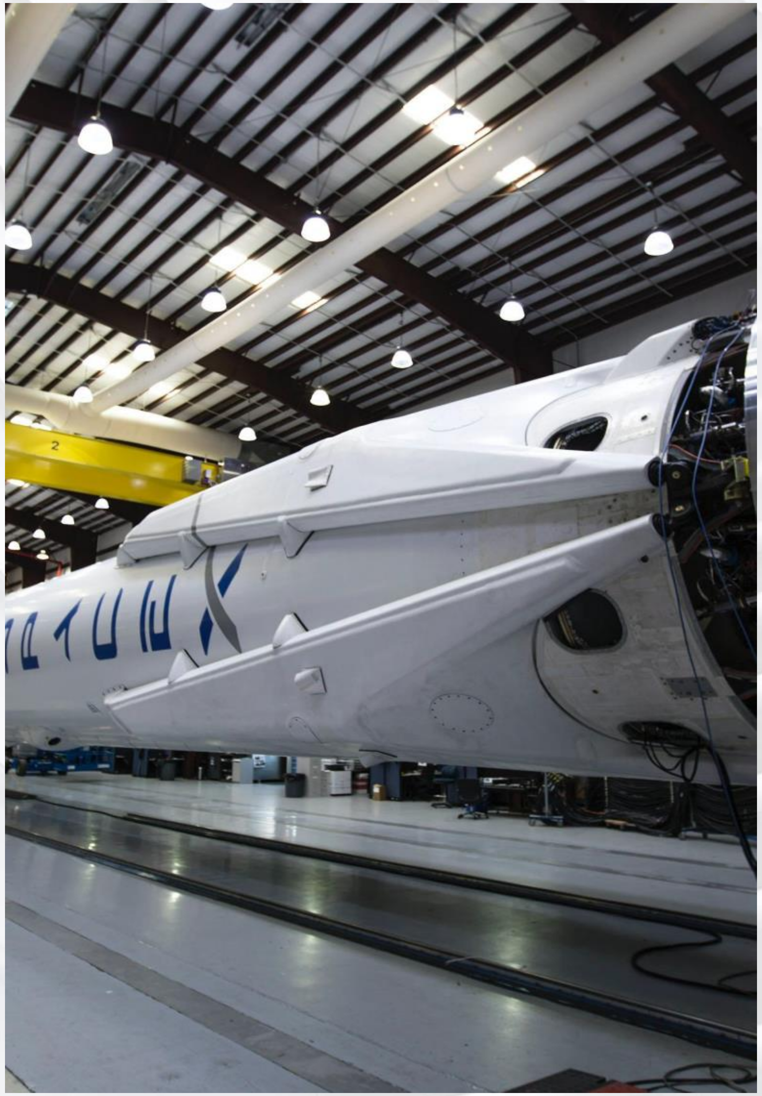
OUTLINE

EXECUTIVE SUMMARY

- Methodology
  - Data collection and Wrangling
  - Exploratory Data Analysis (EDA)
  - Interactive Visual Analytics
  - Predictive Modeling
- Results
  - Insights drawn from EDA
  - Proximities Analysis with Folium
  - Dashboard with Plotly Dash
  - Predictive Analytics



INTRODUCTION



Goal- This project aims to predict if the Falcon 9 first stage will land successfully.

Context- SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch.

My Tasks:

- As a data scientist working for rocket company, SpaceY, gathering essential information about SpaceX and creating dashboards for my team help in ultimately figuring out price of every launch for a possibility to bid against SpaceX.
- Also, determining if the first stage will land successfully, involves training machine learning models, and using public information, that help predict if SpaceX will eventually reuse the first stage.

POINTS TO VALIDATE

- What factors of a mission influence Falcon 9 launch success?
- How payload mass, launch site, orbit type, and other features affect first stage landing success?
- What conditions does SpaceX have to meet to get the best results and ensure the best successful first stage landing rates?
- Best predictive model for successful first stage landing.

METHODOLOGY

1. DATA COLLECTION

- SpaceX REST API
- Web scraping from Wikipedia

2. DATA WRANGLING

- Handle missing values, filter data to include only Falcon 9 launches
- Transform variables by one-hot encoding, for performing analysis and modeling

3. EXPLORATORY DATA ANALYSIS (EDA) USING DATA VISUALIZATION AND SQL

- Visualize relationship of factors with one another using scatter points chart and bar chart to understand patterns
- Execute SQL queries to list and display relevant information like regarding past landing outcomes

4. INTERACTIVE VISUAL ANALYTICS USING FOLIUM AND PLOTLY DASH

- Explore and analyze the proximities of launch sites using Folium
- Build a dashboard using Plotly Dash comparing launch sites and its success rates over the years

5. PREDICTIVE ANALYSIS USING CLASSIFICATION MODELS

- Build classification models – Logistic Regression, SVM, Decision Tree, KNN
- Evaluate and tune to find best model, its accuracy and its best hyperparameters

DATA COLLECTION

Data was collected primarily through SpaceX API and through web scraping from Wikipedia source in the following ways:

- Requesting for rocket launch data from SpaceX API
  https://api.spacexdata.com/v4/launches/past
- Web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled **List of Falcon 9 and Falcon Heavy launches** using BeautifulSoup
  https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

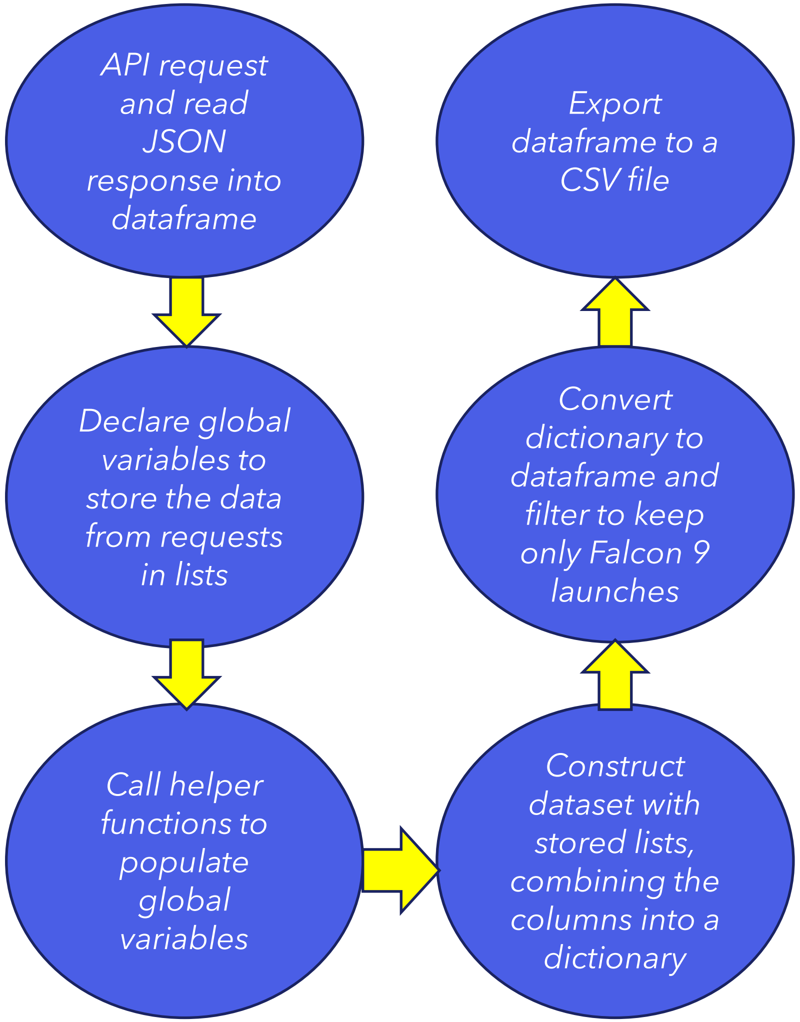
SpaceX API

API request for launch data → Read response in JSON format, normalize to create dataframe → Clean data and filter dataframe for Falcon 9 → Export data to csv file

Web scraping from Wikipedia

HTTP GET to request Falcon 9 Launch HTML page → Create BeautifulSoup object → Extract column names, collect data from parsed tables to create dataframe → Export data to csv file

DATA COLLECTION - SpaceX API




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

response = requests.get(spacex_url)

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

API request and json_normalize into dataframe

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

Global variables declared

```python
# Call getBoosterVersion
getBoosterVersion(data)

# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

Helper functions invoked uses API to extract information using IDs of columns, 'rocket', 'launchpad', 'payload', and 'cores'

```python
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}
```

Dataset created, combining columns into a dictionary

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

# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = pd.loc[pd['BoosterVersion']!="Falcon 1"]
```

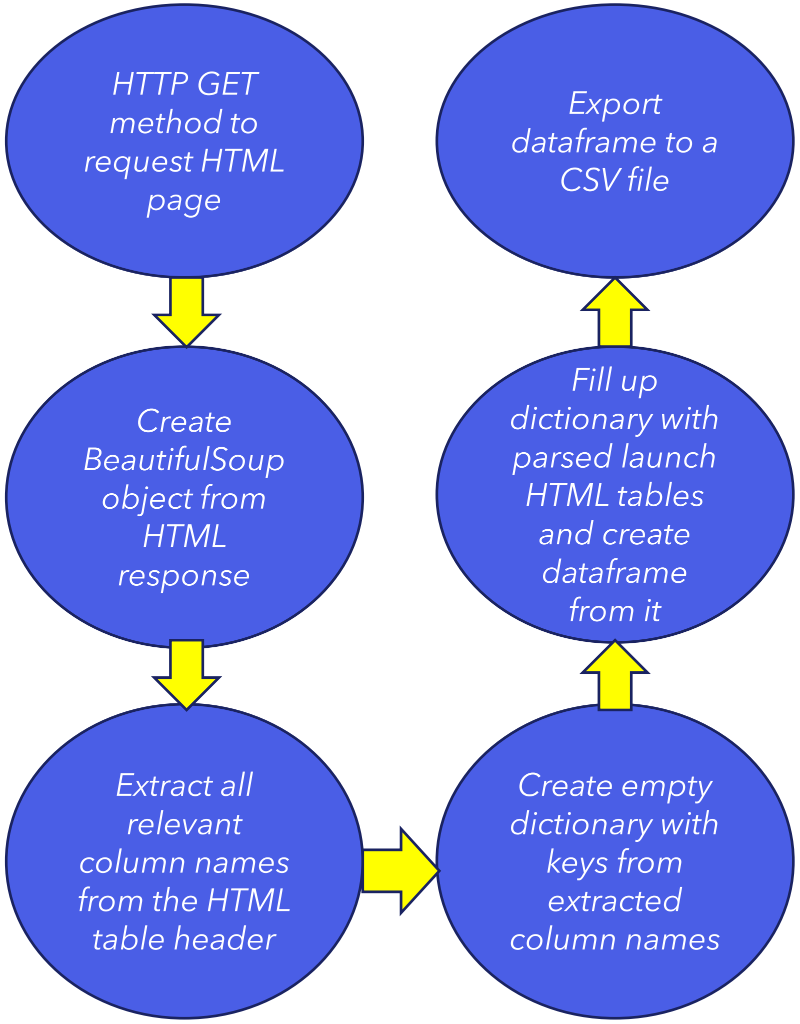
Dictionary converted to required dataframe

```python
# Save the DataFrame to a CSV file
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Saved to a csv file

[GitHub URL - Data Collection (SpaceX API)]

DATA COLLECTION - Web Scraping




```python
# Make the GET request
response = requests.get(static_url)
```

GET request with URL of HTML page

```python
# Parse HTML response
soup = BeautifulSoup(response.content, 'html.parser')
```

BeautifulSoup object created by parsing HTML response

```python
# 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
html_tables = soup.find_all('table')

# Assign the third table to first_launch_table
first_launch_table = html_tables[2]

# Find all th elements in the first table
th_elements = first_launch_table.find_all('th')

# Initialize the list for column names
column_names = []

# Iterate through each th element
for th in th_elements:
    # Apply the extract_column_from_header function
    column_name = extract_column_from_header(th)

    # Append non-empty column names to the list
    if column_name is not None and len(column_name) > 0:
        column_names.append(column_name)
```

Applying find_all() function to find all the HTML tables and then iterating through each table header element to extract column names

```python
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']=[]
```

Empty dictionary created with extracted columns

```python
def date_time(table_cells):
def booster_version(table_cells):
def landing_status(table_cells):
def get_mass(table_cells):
```

Appending data to dictionary keys by calling the above helper functions and then creating a dataframe with the code

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

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

Saved to a csv file

[GitHub URL - Data Collection (Web Scraping)]()

DATA WRANGLING

Dealing with missing values of ‘PayloadMass’

2 columns were found to have missing values- ‘PayloadMass’, and ‘LandingPad’. Out of which ‘LandingPad’ missing values retained as it is to represent when landing pads were not used. So, left with only ‘PayloadMass’. To replace its missing values the mean was calculated using .mean(). Then using the mean and .replace() function, the missing values were replaced with mean.

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

FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        5
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9["PayloadMass"].mean()
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, mean)
```

5 missing values of ‘PayloadMass’ replaced with mean

Converting Landing outcomes to Training Labels with 1 meaning booster landed successfully and 2 meaning unsuccessful landing

Outcomes – Different cases where the booster did not land successfully, sometimes a landing was attempted but failed due to an accident; is represented as follows:

- *True/False Ocean – Successfully/Unsuccessfully landed to a specific region of the ocean*
- *True/False RTLS – Successfully/Unsuccessfully landed to a ground pad*
- *True/False ASDS – Successfully/Unsuccessfully landed on a drone ship*

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)

0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

Landing outcome values

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes

{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

Failure outcome values

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = np.array(~df['Outcome'].isin(bad_outcomes), dtype=int).tolist()
```

Classifying successful and unsuccessful outcomes

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

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

Wrangling – ***Helps perform EDA by finding patterns and label for training supervised models***

Identify missing values → Identify numerical and categorical columns → Calculate no. of launches on each site → Calculate no. and occurrence of each orbit → Calculate no. and occurrence of mission outcome per orbit type → Create landing outcome label

GitHub URL - Data Wrangling

EDA WITH DATA VISUALIZATION

1) SCATTER POINT CHARTS (Catplot)

- Shows relationship or correlation between two variables helping visualize patterns through easy observation.
- ***Flight Number Vs Payload, Flight Number Vs Launch Site, Payload Vs Launch Site, Flight Number Vs Orbit type, Payload Vs Orbit type***

2) BAR CHARTS

- Used to compare values of a variable at a given point in time. Bar charts makes it easy to see which is the highest or common group among various other groups. The length of the bar is proportional to the value of the items the group represents.
- ***Success rate of each orbit type***

3) LINE CHARTS

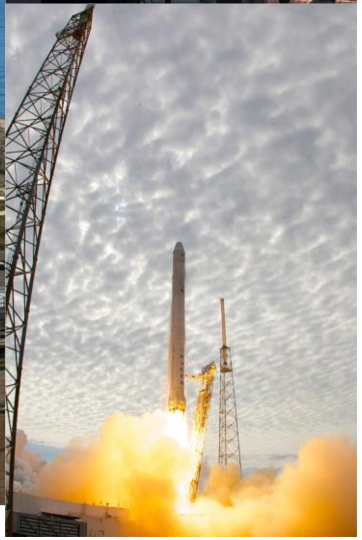
- Tracks changes over a period of time helping depict trends over time.
- ***Average launch success yearly trend***

[GitHub URL - EDA with Data Visualization]

EDA WITH SQL

Performed queries for the following tasks given below:

- Display names of unique launch sites in the space mission
- Display 5 records where launch sites begin with the string ‘CCA’
- Display total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000kg but less than 6000kg
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass
- List the records which will display the month names, failure landing outcomes in drone ship ,booster versions, launch site for the months in year 2015
- 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

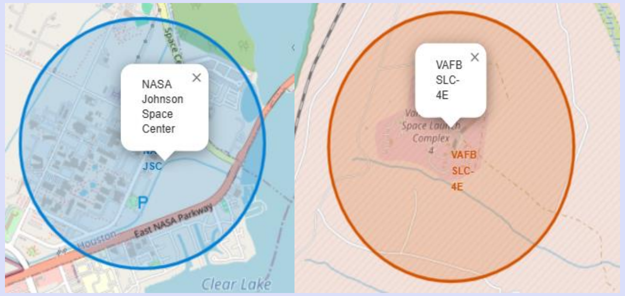



[GitHub URL - EDA with SQL]

BUILDING INTERACTIVE MAPS WITH FOLIUM

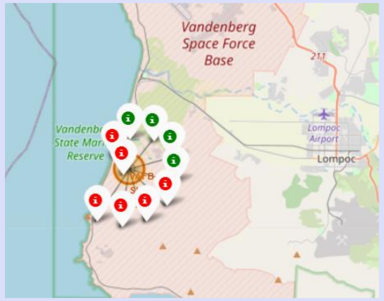
Map Objects added:

1) Circles indicating launch sites



- ***Blue*** circle at ***NASA Johnson Space Centre's coordinate*** with a popup label showing its name using its latitude and longitude coordinates
- ***Red*** circle at ***all launch sites coordinates*** with a popup label showing its name using its latitude and longitude coordinates

2) Markers indicating launch outcomes



- ***Green*** marker used for a ***successful launch outcome (class=1)*** from a launch site
- ***Red*** marker used for ***unsuccessful launch outcome (class=0)*** from a launch site
- These markers form clusters for a particular site, helping easily identify which launch sites have relatively high success rates

3) Lines indicating launch outcomes

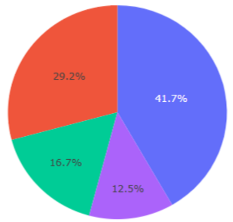
- ***Blue*** lines to show the distance from a launch site to ***closest city, railway, highway, and coastline***
- This helped in unraveling questions about distances between launch sites to these proximities and whether the sites kept a certain distance away from cities

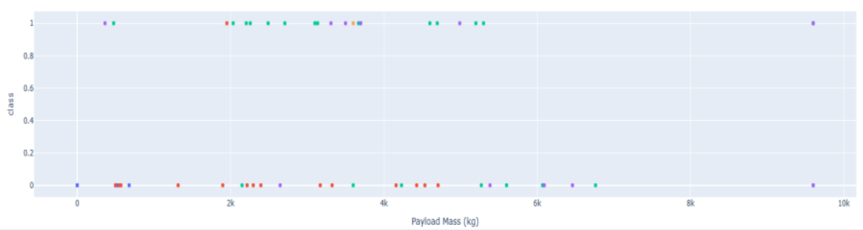
[GitHub URL - Interactive Maps with Folium]

BUILDING DASHBOARD WITH PLOTLY DASH

Charts and Interactions added:

1) Charts





- ***Pie Chart***- Represents the ***total success launches*** for selection of ***all launch sites*** and ***specific launch site***. In case of selection of specific launch site, the success launch rate and failure launch rate are shown. Whereas, if selected all sites only the success launch rate for each of the 4 launch sites is shown.
- ***Scatter Chart***- ***Payload Mass (kg) Vs Class***. Class (0/1), the launch outcome variable, is represented in y-axis, and the Payload Mass (kg) is represented in x-axis. This scatter chart helps understand the ***correlation*** between payload mass and success launch for ***all launch sites*** and ***specific launch site***. With a point in the chart representing the success launch outcome for a particular booster version.

3) Interactions

- ***Launch Site(s) Selection Dropdown List***- This dropdown list enables in selecting ***all launch sites*** and ***specific launch site***. It also plays a key role in visualizing pie chart and scatter chart.
- ***Interactive Slider for Payload Mass Range Selection***- This interactive slider helps in selecting a range of payload mass of min value, 0 kg, and max value, 10000 kg. On selecting a particular range, all the points, representing launch outcomes of relevant boosters from the booster version category, are shown only for that range and for the initial selection of ***all launch sites*** or a ***specific launch site***.

[GitHub URL - Dashboard with Plotly Dash]()

PREDICTIVE ANALYSIS (CLASSIFICATION)

Read dataset and create target array → Standardize the data of independent features → Splitting data into training and testing sets → Create classifier models and evaluate them based on tuned hyperparameters → Finding the best performing model using accuracy and score metrics

Build — **Load**, **transform**, and **split** data to **train** and **fit** model

Evaluate — **Finding** best parameters, **checking** accuracy for both sets of data, and **plotting** confusion matrix

Improve — **Engineering** feature variables, and **tuning** hyperparameters

Find best model — **Comparing** model accuracies, and **selecting** best performing model

[GitHub URL - Predictive Modeling]

```
URL2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv'
resp2 = await fetch(URL2)
text2 = io.BytesIO((await resp2.arrayBuffer()).to_py())
X = pd.read_csv(text2)
```

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

Load datasets for feature variables dataframe and NumPy array creation from target variable column 'Class'

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

Transform data by standardizing it to perform model fitting

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

Data split into training and testing sets with parameter test_size=0.2, setting 80% of data for training and rest 20% for testing

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

```
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, param_grid=parameters, cv=10)
logreg_cv.fit(X_train,Y_train)
```

Create, Train and Evaluate models (For Example- Logistic Regression)

- *Logistic regression model creation, applying GridSearchCV to tune grid of hyperparameters for several combinations through cross-validation*
- *Fit model on training data of X and Y*

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

Finding the best parameters from tuned hyperparameters using best_params_ attribute and the accuracy on the validation data using best_score_ attribute

```
print("Accuracy of logistic regression classifier: ", logreg_cv.score(X_test, Y_test))
```

Calculating accuracy on the test data using score method

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

Plot Confusion Matrix.
Further, repeat the above steps for the Decision Tree, KNN, and SVM classification algorithms

```
all_models = [logreg_cv, svm_cv, tree_cv, knn_cv]

all_scores = [model.score(X_test,Y_test) for model in all_models]
all_scores
```

```
from sklearn.metrics import jaccard_score


[jaccard_score(Y_test, model.predict(X_test)) for model in all_models]
```

```
best_model = all_models[all_scores.index(max(all_scores))]
best_model
```

```
best_model.score(X_test,Y_test)
```

Finding the model that performs the best, by examining score metrics including Jaccard score. Comparing the metrics for each model, the best and top scoring model was predicted.

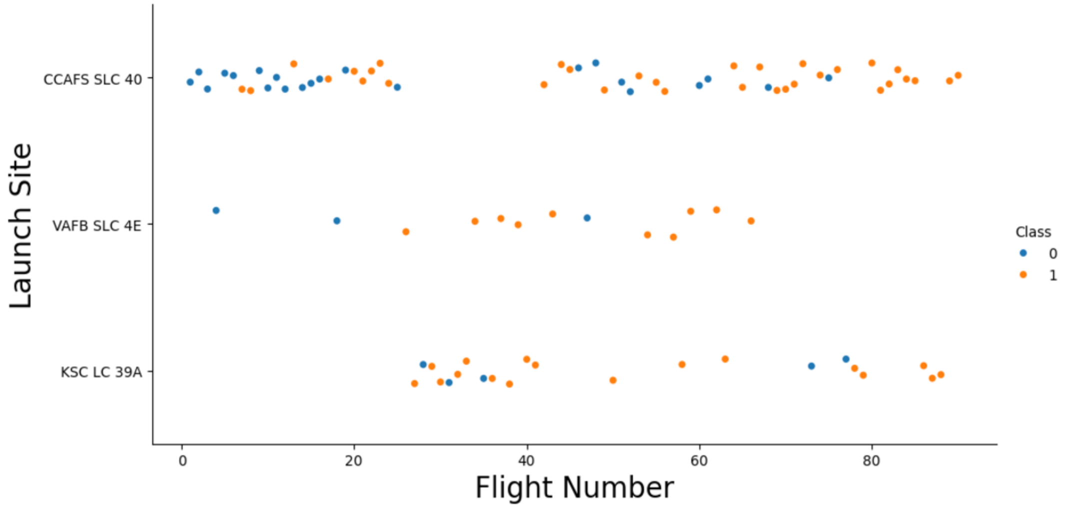
RESULTS

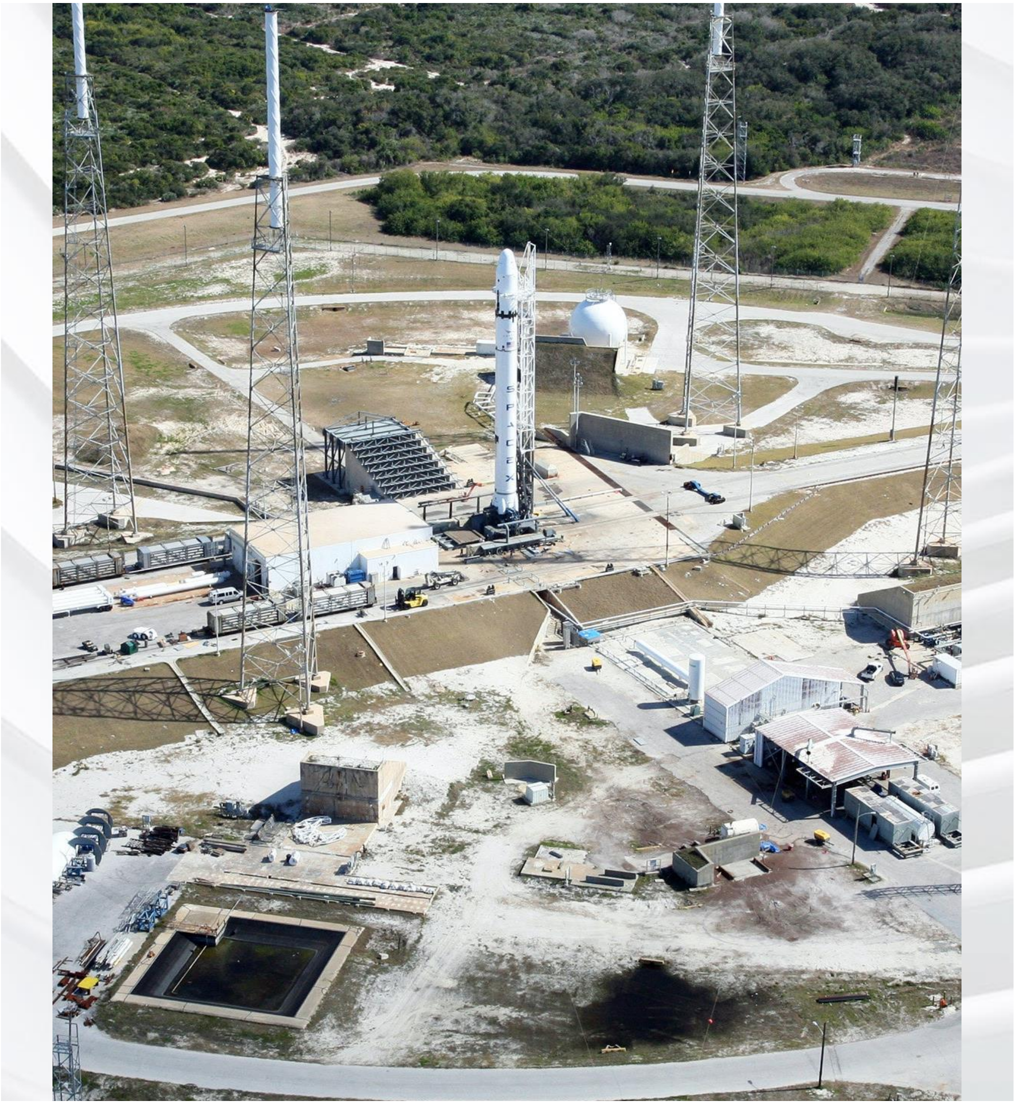
Exploratory Data Analysis	• Launch success rate since 2013 kept increasing • For VAFB-SLC launch site there are no rockets launched for heavy payload mass (greater than 10000kg) • Even with more massive payloads, the first stage often returns successfully
Interactive Visual Analytics	• Most launch sites are near the equator, and all are close to coastline • Launch sites are not at close proximities to cities, highways and railways. But in case of launch mission resource management, the launch sites are located well within reach making it suitable for aiding launch activities, also implying that a good logistic infrastructure is around
Predictive Analysis (Classification)	• Decision tree classifier is the best performing model • Validation data accuracy = ~ 89% • Test data accuracy = ~ 94%

INSIGHTS DRAWN FROM EDA

Flight Number Vs Launch Site

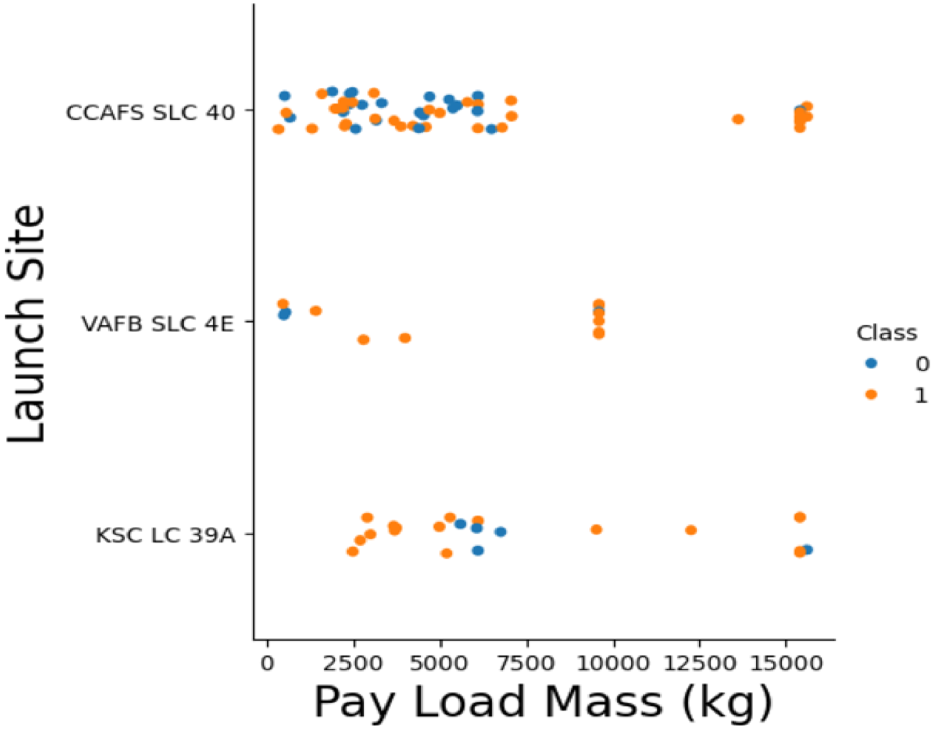
- Initial flights had less success outcomes compared to much latter flights (Success outcome is represented by ***orange*** (Class = 1) and unsuccessful outcome is represented by ***blue*** (Class = 0))
- CCAFS SLC 40 launch site has more than half of all launches
- VAFB SLC 4E and KSC LC 39A have fewer launches and higher success rates compared to CCAFS SLC 40
- Recent launches from every site were mostly successful indicating high success rate
- KSC LC 39A started launch much later compared to CCAFS SLC 40 and VAFB SLC 4E, launching their first flight as around 25th overall





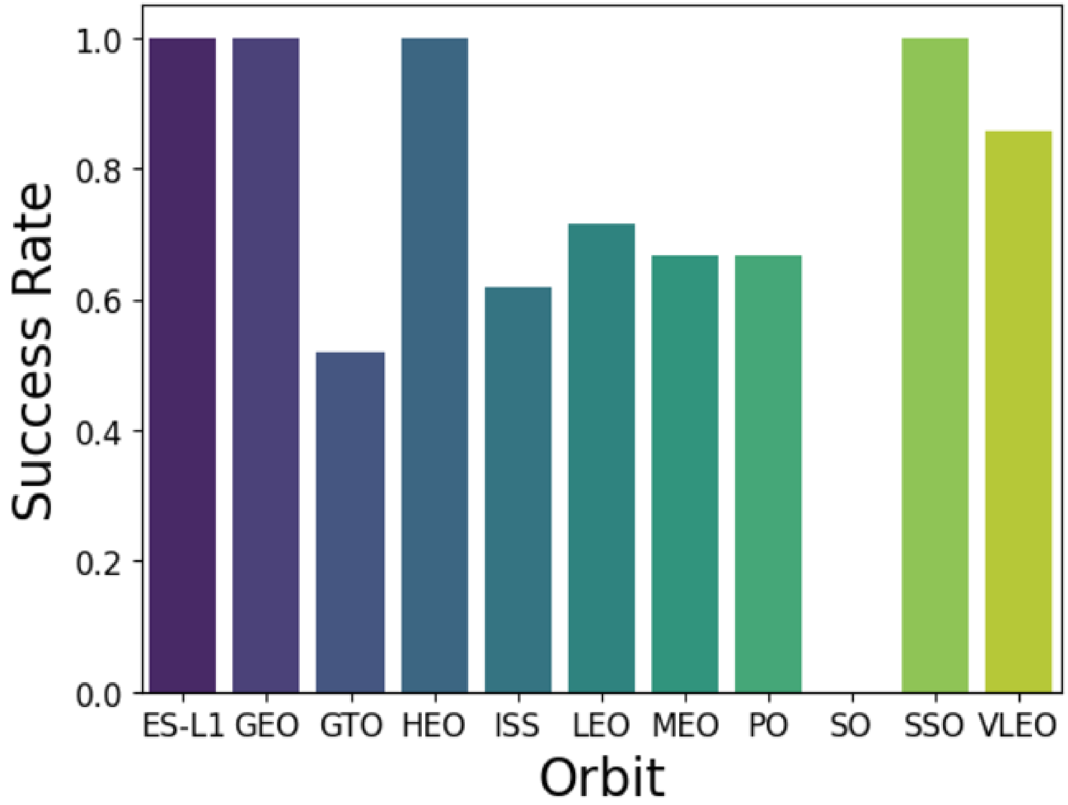
Payload Vs Launch Site

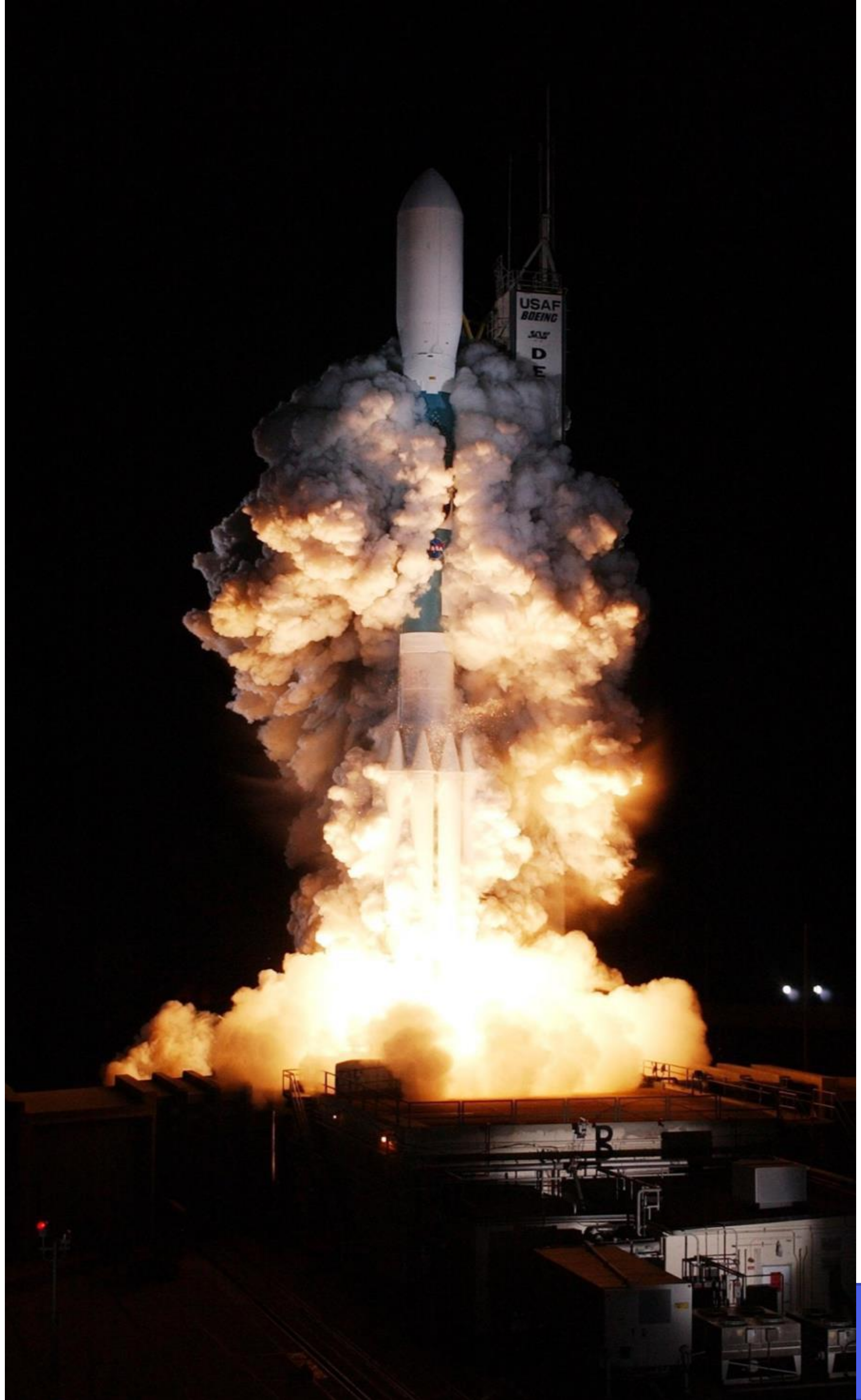
- ***Higher the payload mass in kg, higher the launch success rate***
- KSC LC 39A has 100% launch success rate for payload mass up to 5500kg
- Mostly all launches with payload mass greater than 7500kg were successful
- VAFB SLC 4E hasn't launched flights having payload mass greater than 10000kg
- CCAFS SLC 40 has launched most number of flights with heaviest payload masses (greater than 15000kg)



Success rate Vs Orbit type

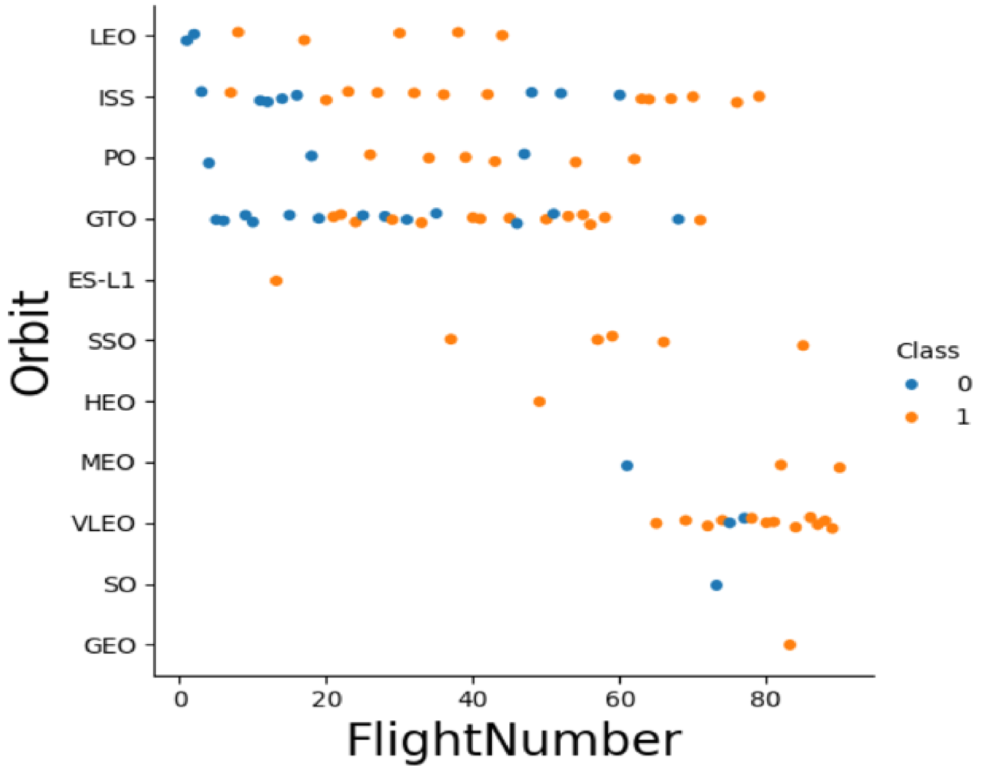
- SO orbit type has 0% launch success rate
- GTO, ISS, LEO, MEO, PO, and VLEO orbit types have launch success rates in the range of 50 – 80%
- ES-L1, GEO, HEO, and SSO orbit types have a 100% launch success rate





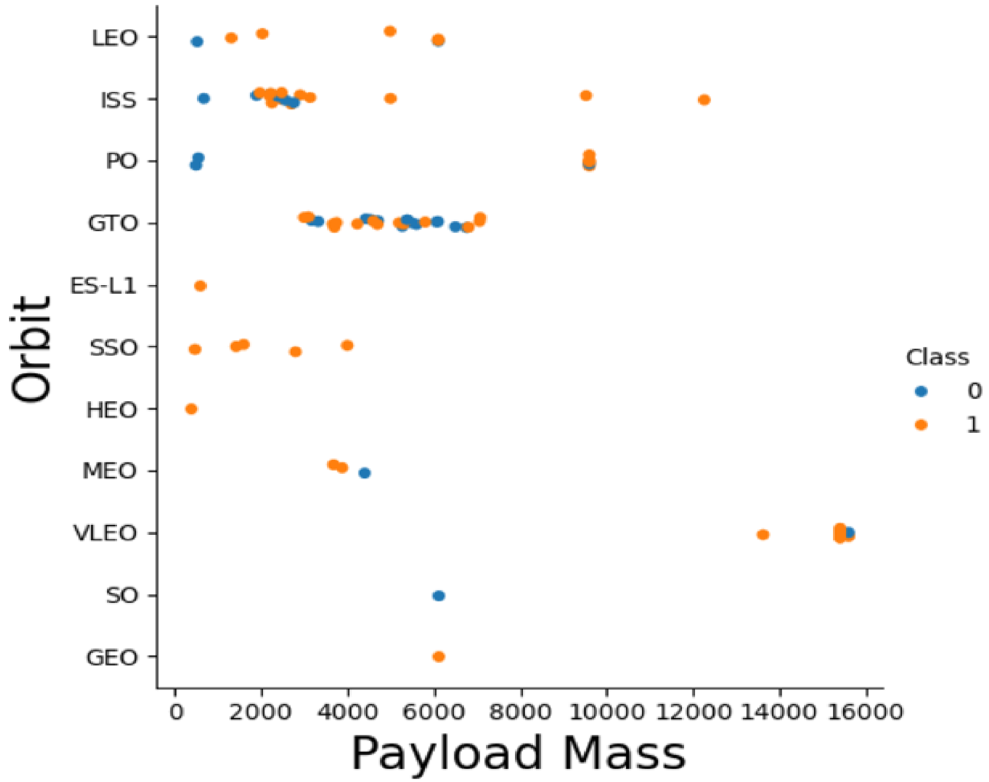
Flight Number Vs Orbit type

- Generally, over the period, as flights have increased the success rates for most orbits have increased too
- In the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success
- In the recent time, more launches have been dedicated to the VLEO orbit type, and they have been found to be all successful



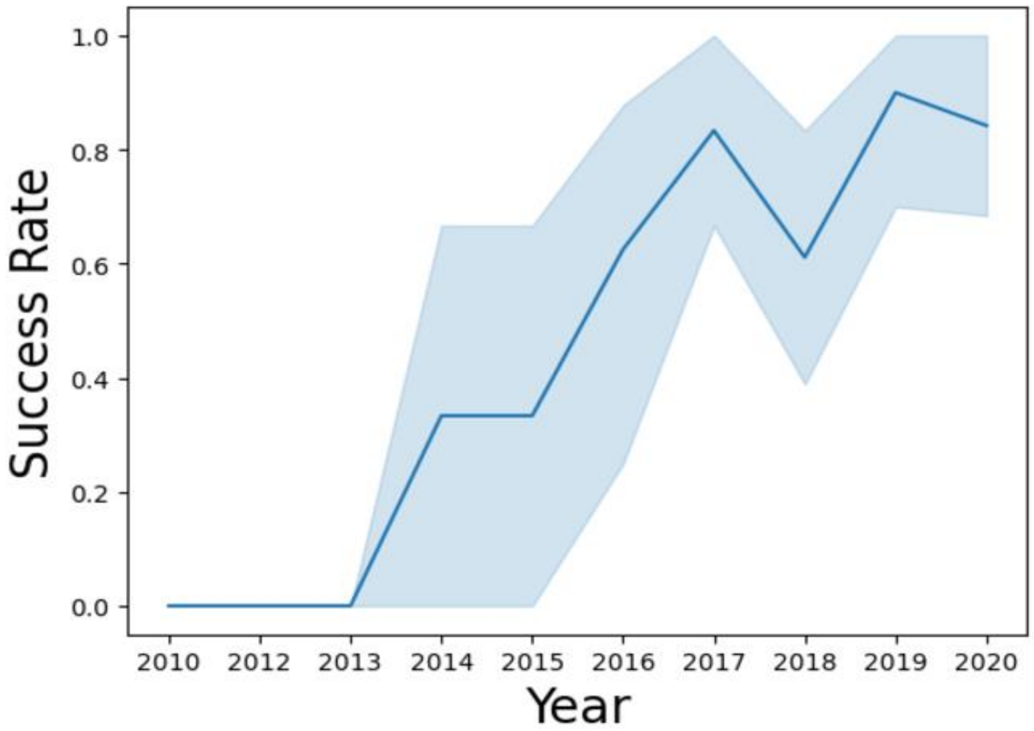
Payload Vs Orbit type

- Almost all the orbit types have launched more flights with lighter payload than a heavier one
- Most flights to an orbit type had a payload mass ranging between 0kg and 8000kg
- Only three orbit types, ISS, PO, and VLEO had flights launched with heavy payloads around 10000kg and above10000kg. And most of these launches were successful



Launch Success Yearly Trend

- From 2010 to 2013 the launch success rate was 0%
- Since year 2013, the success rate gradually increased up until year 2014 when the rate remained constant for a year, from 2014 to 2015.
- From year 2015 the rate again steadily increased, and after reaching 2017 there was a sharp decline in the success rate for 2018
- Since 2018, the success rate increased for the coming year, 2019, but yet again a slight decrease in success rate was observed for the year 2020



Launch Site Details

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

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

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

Following are the launch site names displayed using keyword ***DISTINCT***:

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

```
%sql SELECT * FROM SPACEXTBL 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

First 5 records were displayed for launch sites that begin with the string 'CCA', by using ***LIKE*** operator for pattern search and adding ***LIMIT*** clause for restricting to required number of records. The 2 launch sites that begin with the string 'CCA' are ***CCAFS LC-40*** and ***CCAFS SLC-40***

Payload Mass

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

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

SUM(PAYLOAD_MASS__KG_)
45596

Total payload carried by boosters from NASA (CRS) = ***45596kg***. The result was obtained by applying ***WHERE*** clause on the ***CUSTOMER*** column to filter for NASA (CRS)

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

Average payload mass carried by booster version F9 v1.1 = ***2928.24kg***. The result was obtained by applying ***WHERE*** clause on the ***BOOSTER_VERSION*** column to filter for F9 v1.1 version

Landing and Mission Information

```
%sql SELECT MIN(DATE) AS "DATE OF FIRST SUCCESSFUL LANDING OUTCOME IN GROUNG PAD" FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

* sqlite:///my_data1.db
Done.

DATE OF FIRST SUCCESSFUL LANDING OUTCOME IN GROUNG PAD
2015-12-22

Date of first successful landing on ground pad – ***22-12-2025***. The date was obtained by using ***MIN()*** function along with ***WHERE*** clause on ***LANDING_OUTCOME*** column to filter for ***'Success (ground pad)'***

```
%sql SELECT NUMBER_OF_SUCCESS_OUTCOMES, NUMBER_OF_FAILURE_OUTCOMES \
     FROM (SELECT COUNT(*) AS NUMBER_OF_SUCCESS_OUTCOMES FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%') SUCCESS_TABLE, \
     (SELECT COUNT(*) NUMBER_OF_FAILURE_OUTCOMES FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Failure%') FAILURE_TABLE;
```

* sqlite:///my_data1.db
Done.

NUMBER_OF_SUCCESS_OUTCOMES	NUMBER_OF_FAILURE_OUTCOMES
100	1

Total no. of successful mission outcomes = ***100***
Total number of failure mission outcomes = ***1***
The result was obtained by using 2 subqueries to determine the counts of success and failure outcomes. The subqueries used ***WHERE*** clause on the ***MISSION_OUTCOME*** column and ***LIKE*** operator for pattern search to get the respective counts

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

* sqlite:///my_data1.db
Done.

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

The boosters that had successful landing on drone ship and carried a payload mass between 4000kg and 6000kg are ***F9 FT B1022***, ***F9 FT B1026***, ***F9 FT B1021.2***, ***F9 FT B1031.2***. The result was obtained by applying ***WHERE*** clause on ***LANDING_OUTCOME*** column and ***PAYLOAD_MASS__KG_*** column to filter as per requirement

```
%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS Landing_Count \
     FROM SPACEXTBL \
     WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
     GROUP BY LANDING_OUTCOME \
     ORDER BY COUNT(LANDING_OUTCOME) DESC;
```

* sqlite:///my_data1.db
Done.

Landing_Outcome	Landing_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 landing counts for different landing outcomes between the years 04-06-2010 and 03-20-2017 were ranked in descending order. The landing outcomes in between these years include successful and unsuccessful ***drone ship***, ***ground pad***, ***parachute***, and ***ocean*** landings. The ranking was obtained by applying ***WHERE*** clause on the ***DATE*** column to filter for the required period, along with using ***GROUP BY*** for grouping different outcomes, and ***ORDER BY*** for ranking as per count of the landing outcome

Boosters

```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

The ***DISTINCT*** names of boosters that have carried the maximum payload mass are listed. The list was obtained by using a subquery within the ***WHERE*** clause, applied on the ***PAYLOAD_MASS__KG_*** column. And in order to get records for the maximum payload mass, the subquery used the ***MAX()*** function

```
%%sql
SELECT
    CASE
        substr(Date, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS Month_Name, LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM
    SPACEXTBL
WHERE
    substr(Date, 0, 5) = '2015'
    AND LANDING_OUTCOME = 'Failure (drone ship)';
```

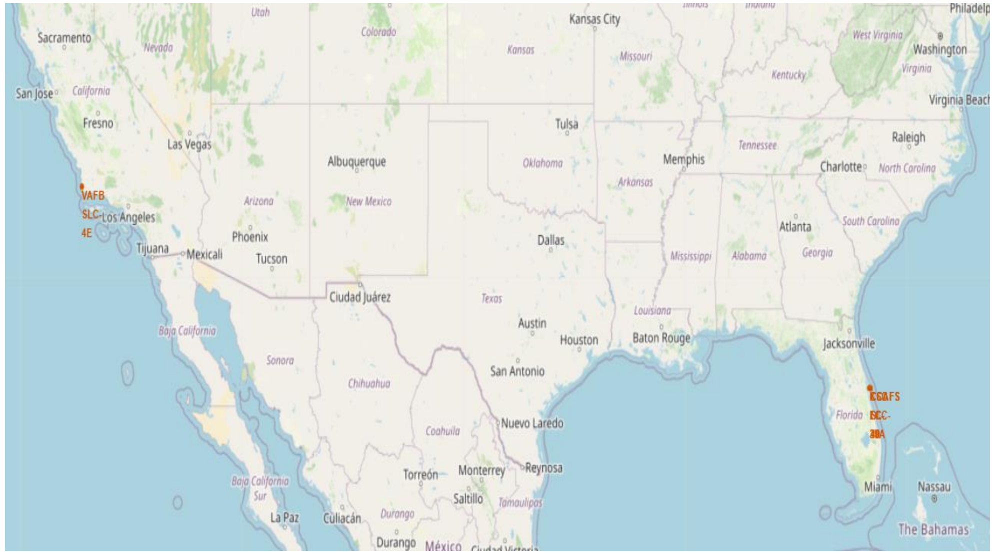
```
 * sqlite:///my_data1.db
Done.
```

Month_Name	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

The records displaying the month names, failed landing outcomes on drone ship, booster versions, and launch sites for the months in the year 2015 are listed. The list was obtained by applying the ***WHERE*** clause on the ***DATE*** and ***LANDING_OUTCOME*** columns to filter by the condition of failed landing outcome on drone ship in the year 2015. To display the month names for 2015 the CASE statement with ***substr()*** function was used. For access and display of months, ***substr(Date, 6,2)*** was used, while for the year, ***substr(Date,0,5)*** was applied and set to 2015

LAUNCH SITES PROXIMITIES ANALYSIS

Factors for selection of a geographic location for a launch site

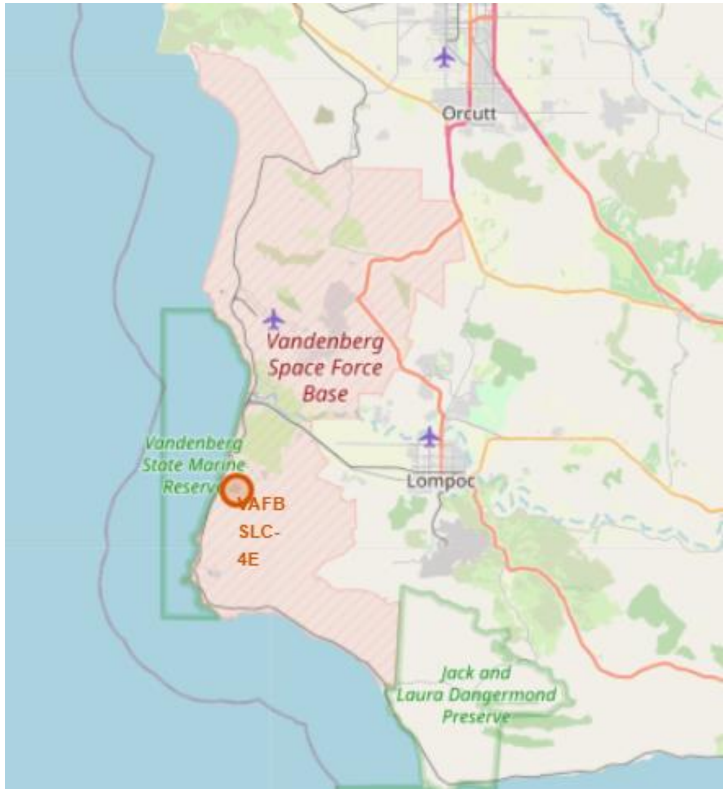


The 4 launch sites are:

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

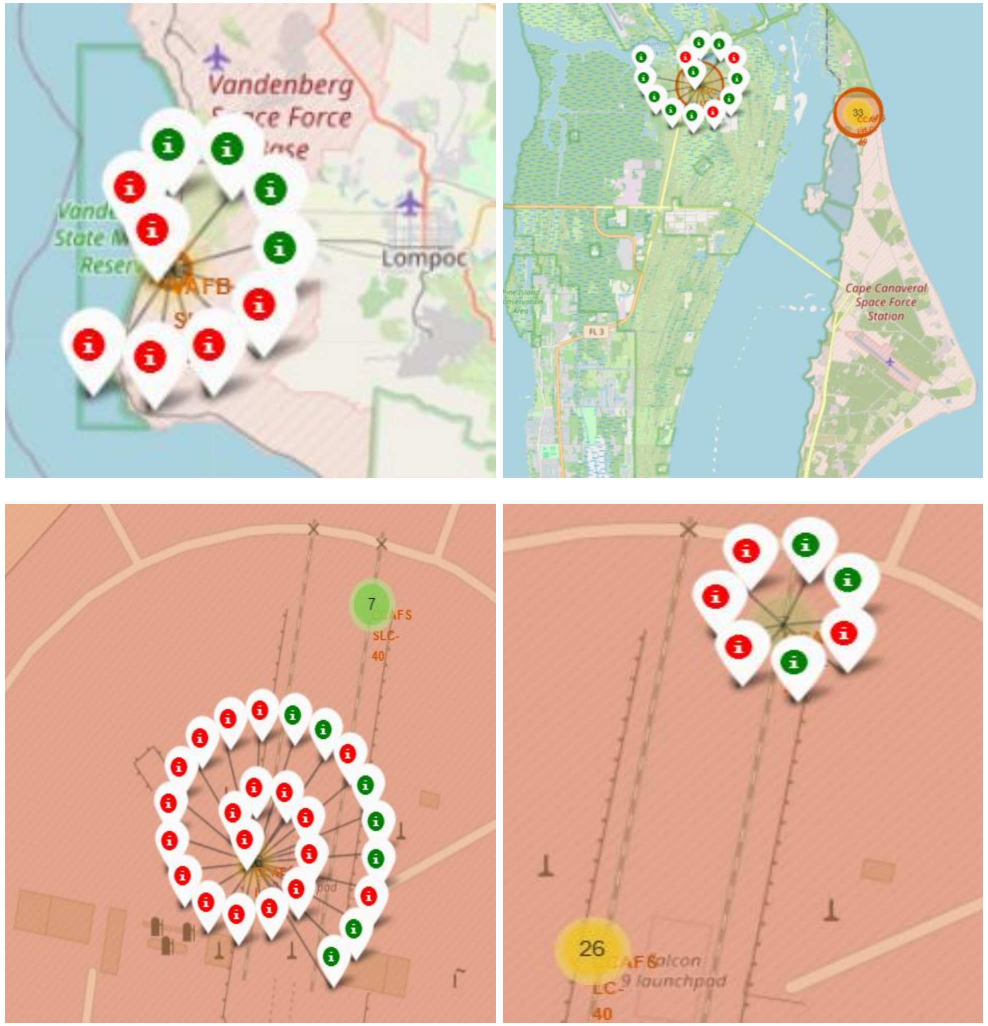
Proximity to the Equator- All the launch sites are in proximity to the equator line, making it viable to helping save the cost of putting in extra fuel and boosters as rockets launched from sites near the equator get an additional natural boost. This natural boost is due to earth's rotational speed being highest at the equator.

Proximity to the Coast- Launching a rocket from the east coast gives an additional boost due to the rotational speed of Earth. Also, if something goes wrong during the ascent, the debris will fall into an ocean instead of a densely populated area. As seen all the sites are located at very close proximity to the coast. Florida being prone to hurricanes is selected as a site location because of this. KSC LC-39 A is one of the sites in Florida.



Launch outcomes at each site

- **Green** marker – Success outcome
- **Red** marker – Failure outcome



Number of launch outcomes at each site:

- VAFB SLC-4E - ***10***
- KSC LC-39A - ***13***
- CCAFS LC-40 - ***26***
- CCAFS SLC-40 - ***7***

Success rate (in %) = Number of success outcomes for a site x 100 / Total number of outcomes for a site

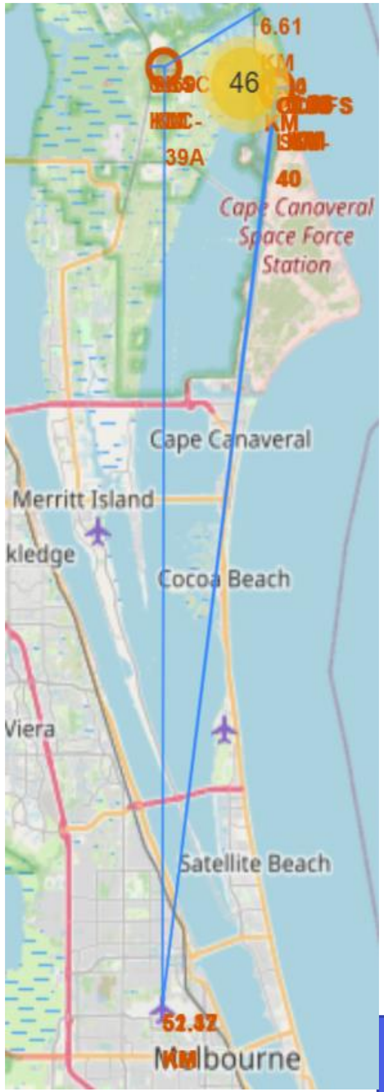
$$\text{Success rate (in \%)} = \frac{\text{Number of success outcomes for a site}}{\text{Total number of outcomes for a site}} \times 100$$

- VAFB SLC-4E – ***Success rate = 4/10 x 100 = 40%***
- KSC LC-39A – ***Success rate = 10/13 x 100 = 76.92%***
- CCAFS LC-40 – ***Success rate = 7/26 x 100 = 26.92%***
- CCAFS SLC-40 – ***Success rate = 3/7 x 100 = 42.86%***

Distances between a launch site to its proximities

Distance between a site to its proximities is represented by the ***blue*** line

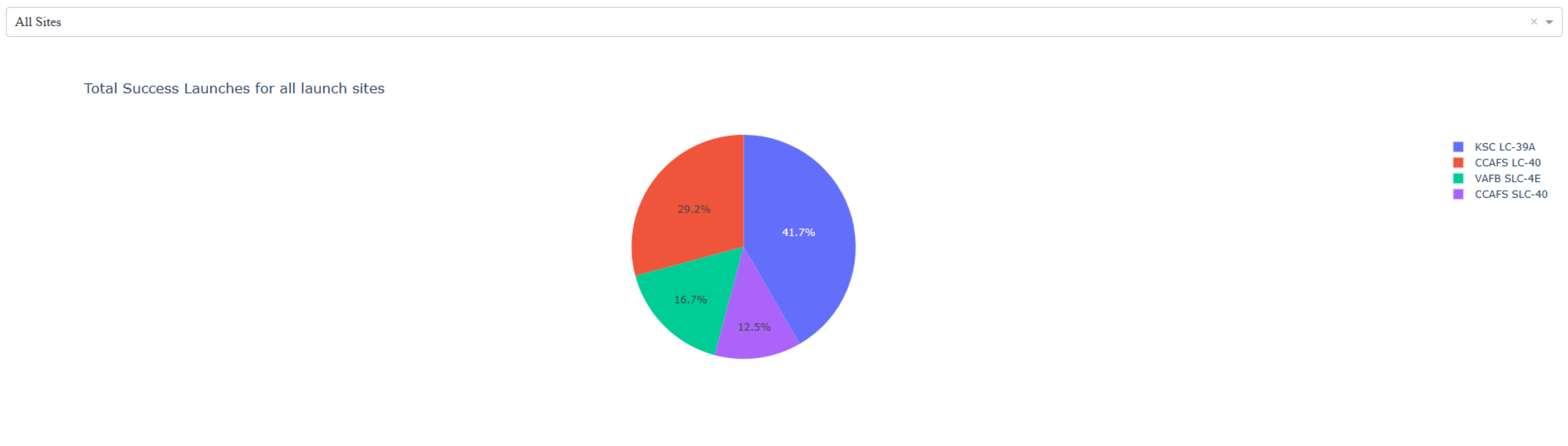
Proximity to cities – The launch sites have been situated not in close proximity to cities. The nearest city in case of CCAFS LC-40 is Melbourne and is at distance of ***52.82km*** away from the site. This factor is crucial while considering the selection of a launch site and should take into account several concerns like safety of populated areas, noise effects, and compliance to zoning regulations.



Proximity to railways, highways and coastline – Mostly the launch sites are in proximity to either railways or highways and are in close proximity to coastline. For example, the nearest railway and highway from VAFB SLC-4E is at distances of ***12.62km***, and ***8.85km*** respectively. However, nearest coastline distance is ***1.35km***. This ensures that, in the event of dropped stages along the launch path or an unexpected failure during launch or mid-flight, debris or rocket parts do not fall on local population or property. That being said, the site should remain close enough to roads, railways, or docks to facilitate the transport of people and materials for launch activities.

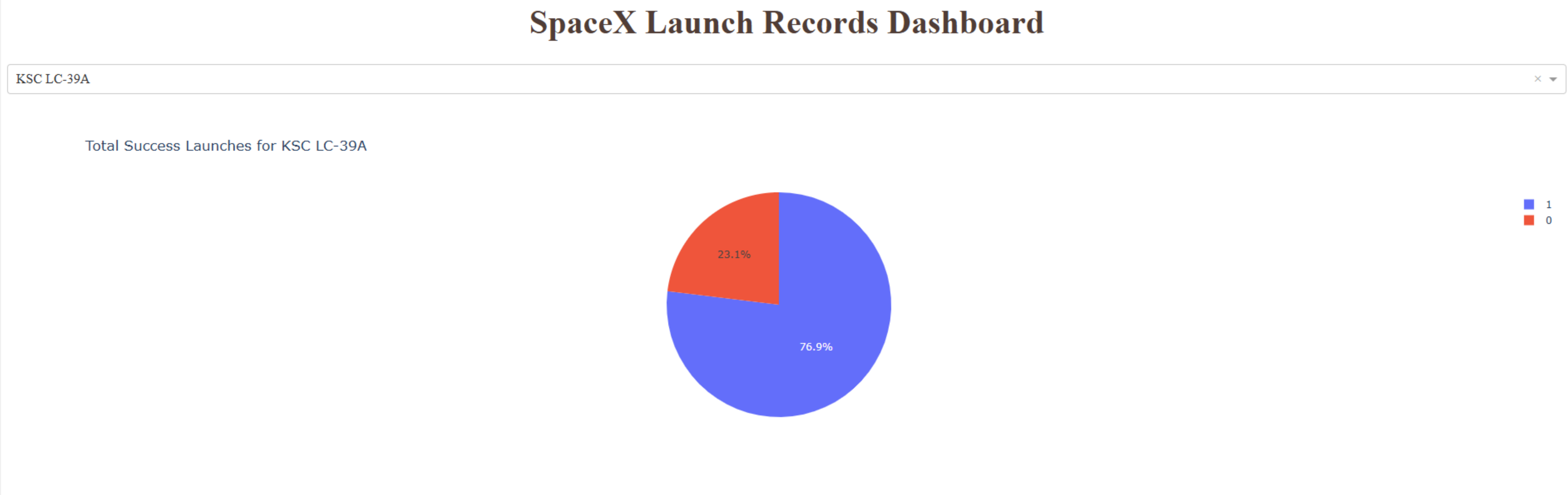
DASHBOARD WITH PLOTLY DASH

Success launches by site



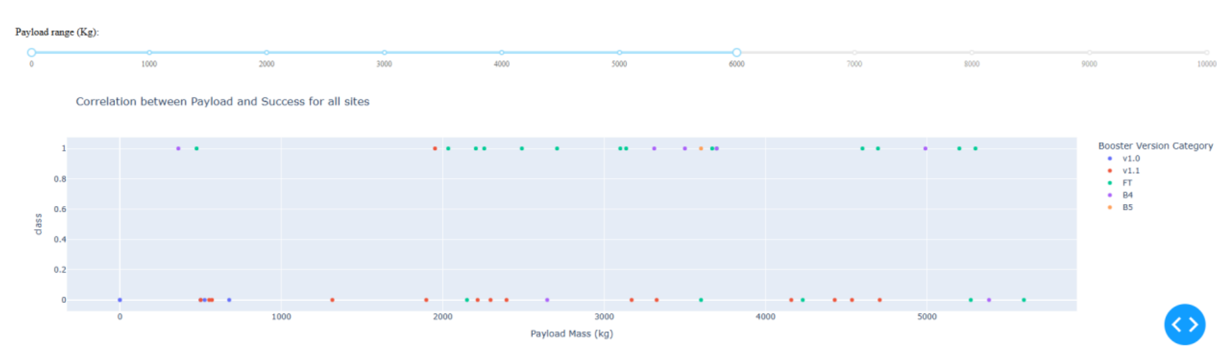
KSC LC-39A has the most success launches amongst all launch sites with percentage of ***41.7%***

Launch Success Ratio (KSC LC-39A)

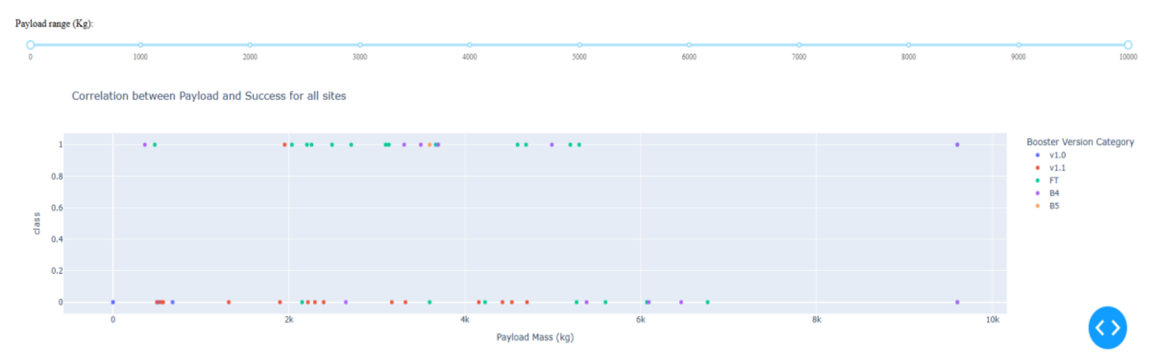


Launch success ratio or alternatively known as launch success rate is highest for ***KSC LC-39A.***
Out of the 13 launches at KSC LC-39A launch site, 10 launches were successful and only 3 were unsuccessful.
The launch success ratio or launch success rate in percentage for KSC LC-39A is ***76.9%***

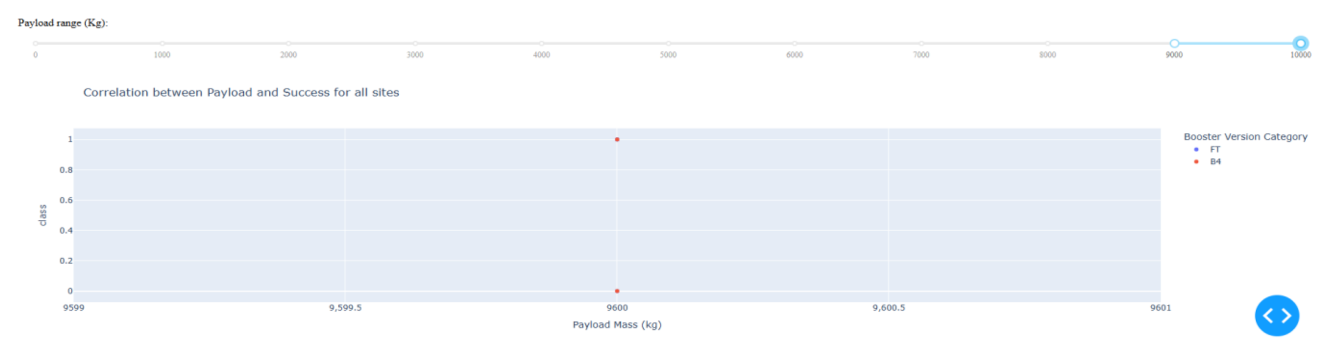
Payload Vs Launch Outcome for all sites



Payload range having highest launch success rate is 0kg to 6000kg



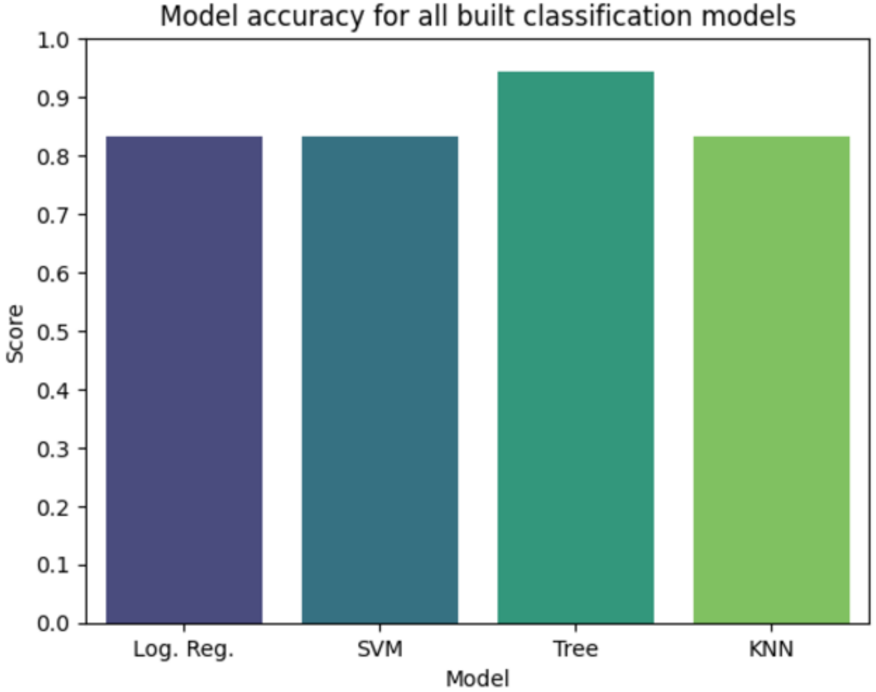
Out of the different booster versions for F9 rocket like v1.0, v1.1, FT, B4, and B5, the ***booster version that has highest launch success rate is FT***. For finding this the entire payload range from 0kg to 10000kg was considered



Payload range having lowest launch success rate is 9000kg to 10000kg

PREDICTIVE ANALYTICS

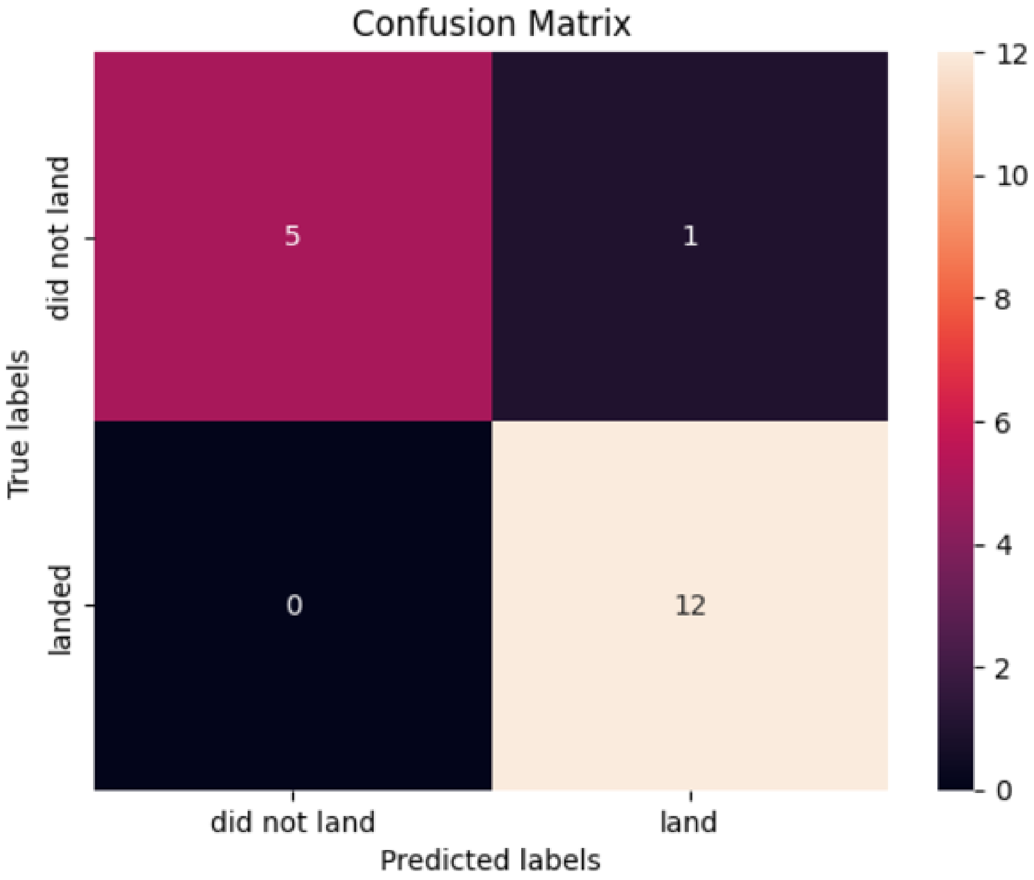
Classification Accuracy



	Logistic Regression	SVM	Decision Tree	KNN
Jaccard Score	0.8	0.8	0.923	0.8
Accuracy	0.833	0.833	0.944	0.833

- According to the Jaccard score and classification accuracy results for all the built classification models, the model that best performs out of the 4 is the ***Decision Tree Classifier***. This classifier had an ***accuracy of 94.4%*** with a ***Jaccard score of 0.923***.
- The other models also performed much significantly, and the results of their accuracy and score clearly depict the same. Also, the bar chart representation is a great indicator of this highly comparative score results. On close observation, it can also be seen that the scores and accuracies for the rest of the models were the same, which could likely be due to the small sample size of test data, 20% of the entire dataset, that was considered.

Confusion Matrix



- ***False Negative (FN) = 5***
- ***False Positive (FP) = 1***
- ***True Negative (TN) = 0***
- ***True Positive (TP) = 12***

- A confusion matrix helps summarize the performance of a classification model as a visual representation. In all four classifiers that were considered, it was observed that only the decision tree classifier stood out, while the rest were identical.
- From the confusion matrix of the decision tree classifier shown here, it can be seen that there is only 1 False Positive. This is the reason for the high accuracy achieved. Type 1 error (False Positives) is always one to look out for while comparing model accuracies.
- Out of the 6 samples in the 'did not land' category, only 1 was incorrectly predicted. And in case of the other 3 models, there were 3 samples each incorrectly predicted. While for the 'landed' category, all the models correctly predicted all 12 samples in this category.
- Evaluation metrics from the confusion matrix for the decision tree classifier:
  - ***Precision*** = TP/(TP+FP) = 12/(12+1) = 12/13 = ***0.92***
  - ***Recall*** = TP/(TP+FN) = 12/(12+5) = 12/17 = ***0.70***
  - ***F1-Score*** = 2 x (Precision x Recall)/(Precision + Recall)
    = 2 x (0.92 x 0.70)/(0.92 + 0.70) = 2 x (0.64)/(1.62)
    = ***0.79***
  - ***Accuracy*** = (TP+TN)/(TP+TN+FP+FN) = (12+0)/(12+0+1+5)
    = 12/18 = ***0.66***

CONCLUSION

- ***Best performing model*** – Decision Tree Classifier
- ***Payload mass*** – The launches were not as successful with heavier payload masses as they were with lighter payload masses
- ***Launch success*** – Kept increasing over the years. This could be a testament to the effort that goes behind for continuous improvement
- ***Launch sites to proximities connectivity*** – The launch sites were at close proximity to coastline and not quite far away from logistic infrastructure including, railways and highways

- ***Equatorial launch site*** – All the launch sites were near the equator, and this ensured to take advantage of the natural boost from Earth, thus making it economically viable too

- ***KSC LC-39A*** – Has the highest launch success ratio and most success launches among all sites. The site has a 100% success ratio for launches carrying payload less than 5500kg. All the 10 success launches from the site come in this range

- ***Orbits*** – ES-L1, GEO, HEO, and SSO orbit types have a 100% launch success ratio, conversely, SO has a 0% launch success ratio. Recently, more launches were dedicated to the VLEO orbit type, and all of them were successful as well

EXPANDING HORIZONS: FUTURE SCOPE AND IDEAS

1) Dataset expansion

Collecting ***larger and more diverse*** dataset to test generalizability across different conditions ensures it performs consistently and reliably across varied scenarios

2) Advanced Feature Analysis

Applying ***Principal Component Analysis (PCA)*** or other dimensionality reduction techniques can identify the most influential features, reduce noise, and potentially improve model accuracy. This way while retaining critical information both model performance and interpretability can be enhanced

3) Model Optimization

XGBoost is a powerful ensemble algorithm known for its efficiency and accuracy in classification tasks. Incorporating this model could offer valuable insights and potentially outperform existing models. Ultimately, helping increase robustness and reduce overfitting with fine-tuned hyperparameters

4) Real-time and Operational Enhancements

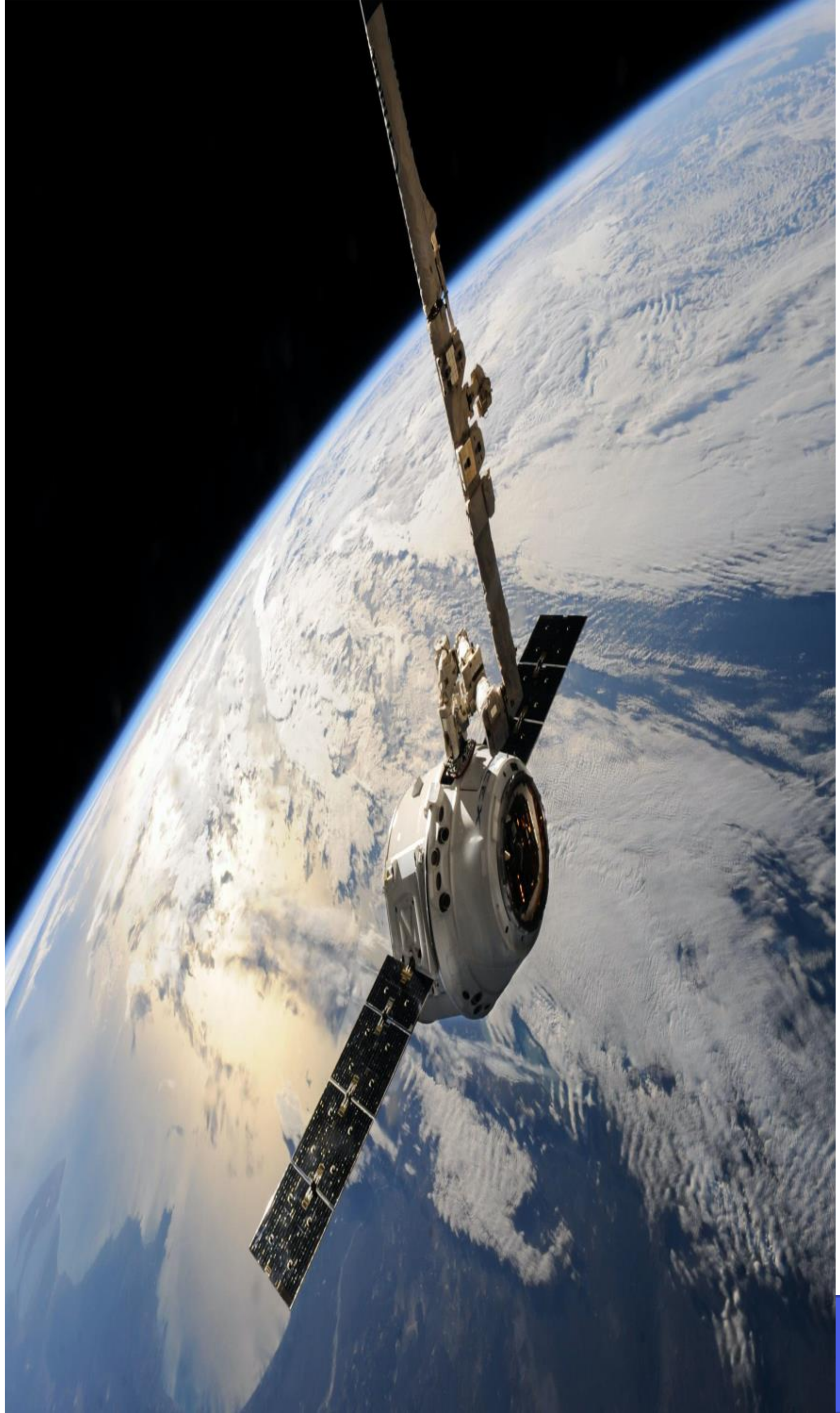
Integrating ***real-time weather and telemetry*** data for launches provide enhanced mission control with dynamic prediction of outcomes. Also, accommodating an ***Automated Decision Support System*** by developing an application that integrates predictive results with visual analytics assists engineers in optimizing launch configurations

5) Extending Predictive Capabilities

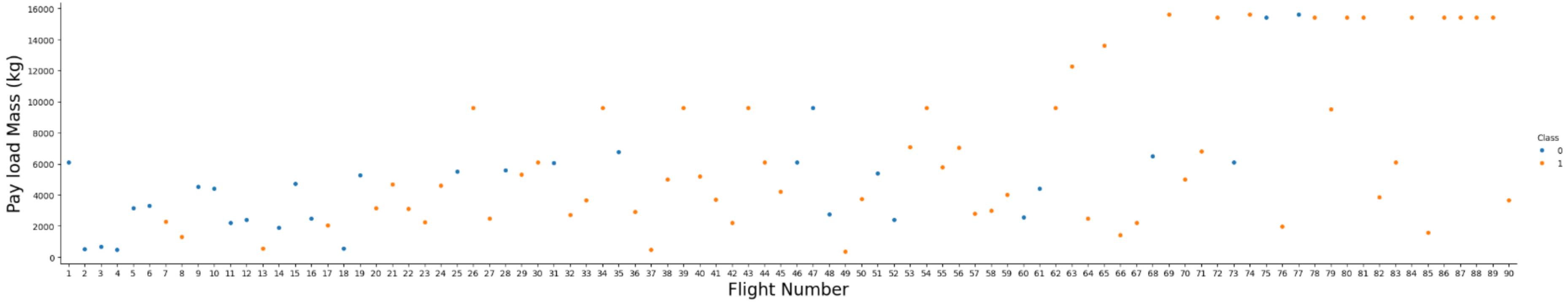
By expanding the model to predict additional metrics like payload success or fuel efficiency, and exploring sequential dependencies in launches using ***Recurrent Neural Networks (RNNs) or Transformers*** for time-series analysis

6) Ethical and Sustainability Considerations

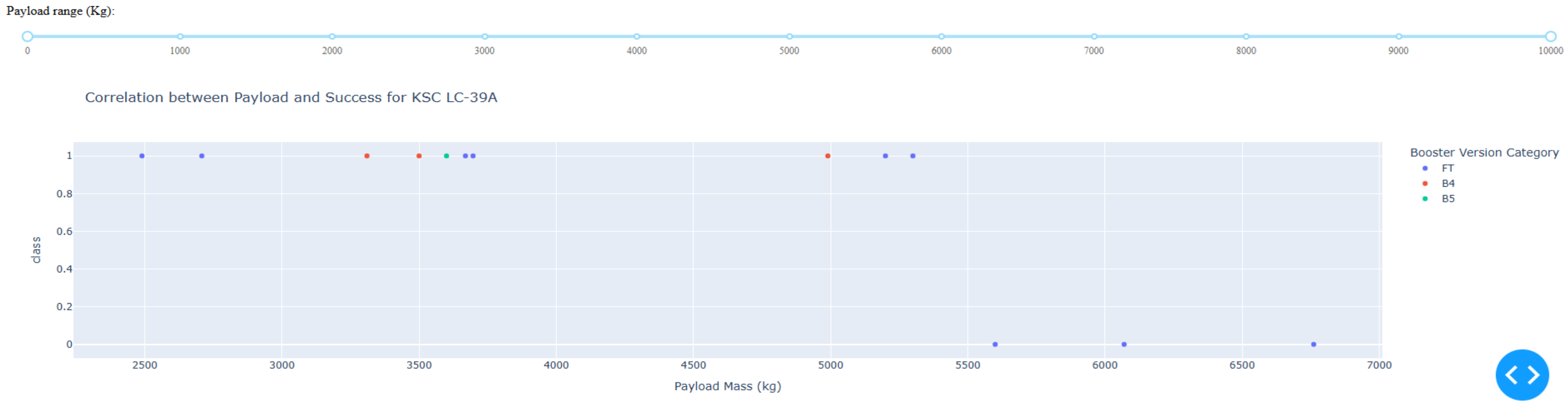
Using prediction to optimize rocket reuse minimizes environmental impact and promotes sustainable space exploration. For example, incorporating features like carbon footprint per launch or material recovery efficiency could be considered



APPENDIX



For more recent flights the first stage landing was found to be more successful compared to the much earlier flights. It can also be observed that flights carrying heavier payload masses were often successful as well



Out of the 13 launches at the KSC LC-39A launch site, 10 successful launches carried payload masses of less than 5500kg. In addition, the 3 failed launches corresponded to the FT booster version, in contrast to the successful launches, which included the FT, B4, and B5 booster versions


THANK YOU