IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Boifang Nchengeti Khani

14/08/2024

Outline

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

Executive Summary

- The rationale of this research is to analyse data for SpaceX to determine if a launch will successfully land.
- To collect data, methods used are: 1. SpaceX API and Webscrapping.
- The data was then manipulated as follows:
- Filtered to have only falcon 9 data.
- Transformed and then launch outcomes categorized into success or failure.
- Different factors were evaluated to see how the affect launch outcome using visualization techniques.
- Results of data analysis, data visualization and sql queries are shared.
- Also, results of predictive models applied to predict the outcomes are shared with Decision Trees proving to be the best model.

Introduction

Project background and context

- SpaceX uses less money for it’s launches, around 62 million dollars, for which it can re-use the first stage. This means it is able to bid for less money than its competitors, which is advantageous.

Problems you want to find answers

- To determine if the first stage of launch by SpaceX will land successfully.
- To determine how different factors affect the outcome of landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected by two ways, using the SpaceX API and also by webscrapping from wikipedia.
- Perform data wrangling
  - Done so as to assign landing outcomes to either successful (1) or failure (0).
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Predictive modelling using SVM, Decision Trees, KNN and Logistic Regression to find which model best estimates the landing outcome.

Data Collection

- SpaceX API and Wescrapping used to collect the data

SpaceX API

Request and get data from SpaceX API → Extract required rows and columns → Filter for only falcon 9 launches

Webscraping

Request, get data and extract columns → Extract rows → Create Dataframe

Data Collection – SpaceX API

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

```
[18]: # Call getLaunchSite
      getLaunchSite(data)
```

```
[19]: # Call getPayloadData
      getPayloadData(data)
```

```
[20]: # Call getCoreData
      getCoreData(data)
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data2[(data2.BoosterVersion != 'Falcon 1')]
data_falcon9
```

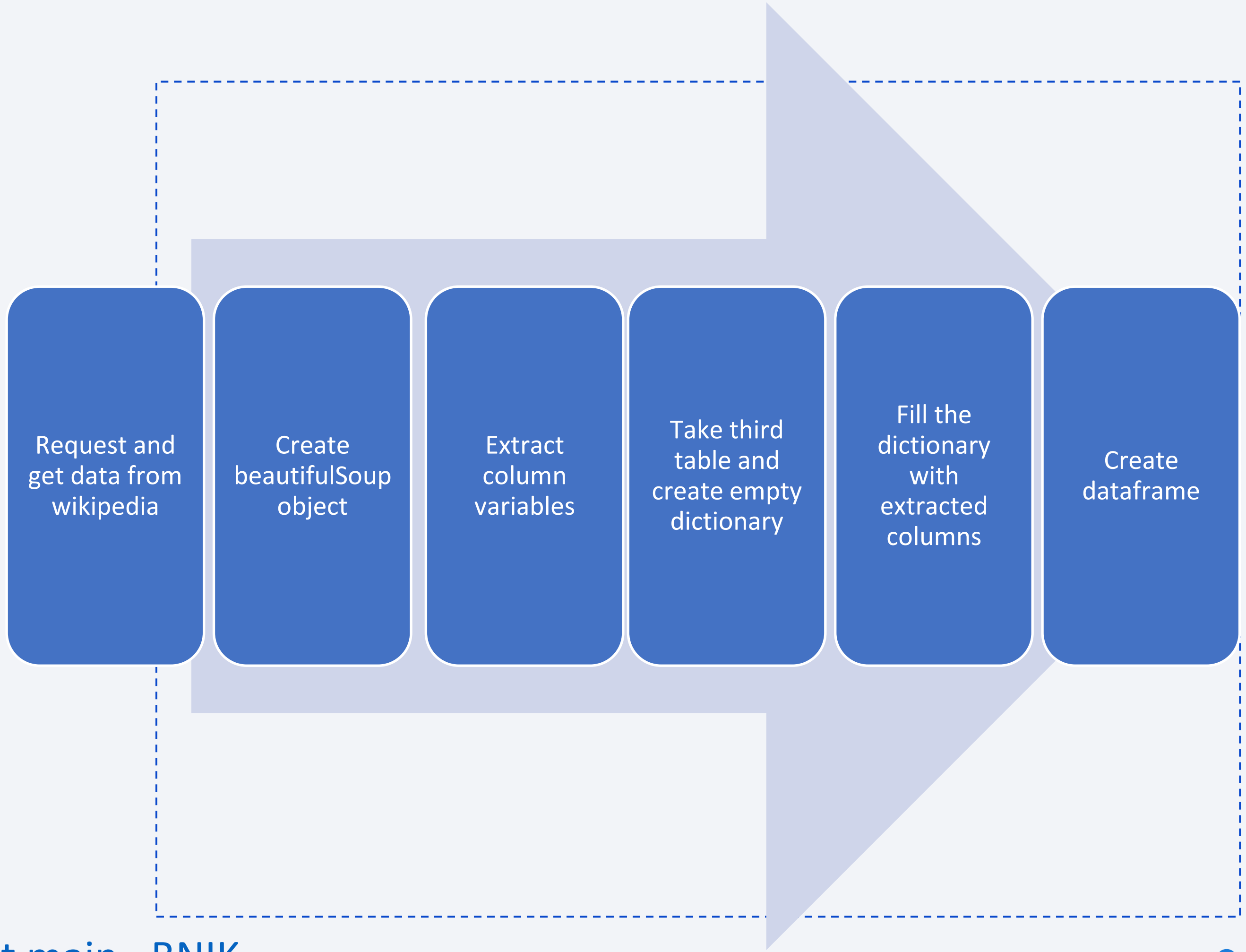
```
# Calculate the mean value of PayloadMass column
mean_PayLoadMass = data_falcon9["PayloadMass"].mean()

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

[BNIKdata3/jupyter-labs-spacex-data-collection-api (3).ipynb at main · BNIK-23/BNIKdata3 (github.com)]

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose



[BNIKdata3/jupyter-labs-webscraping (1).ipynb at main · BNIK-23/BNIKdata3 (github.com)]

Data Wrangling

Describe how data were processed

- Performed some Exploratory Data Analysis to find patterns in the data set.
- The number of launch sites and also the number of launches at each launch site were determined.
- The launches at each site were divided into orbit types.
- The occurrence of each orbit type at each launch site was determined.
- All failed landing outcomes were given label 0 in a Class column, whilst successful launches were labeled 1.

BNIKdata3/labs-jupyter-spacex-Data wrangling.ipynb at main · BNIK-23/BNIKdata3 (github.com)

Data Wrangling

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

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

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

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

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes`.Then assign it to a variable landing_outcomes.

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

Pyth

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

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

EDA with Data Visualization

Summary of charts plotted

- 1.Scatter plot: Plotted the following:

Relationship between Flight Number and Launch Site

Relationship between Flight Number and Orbit Type

- 2. Bar Chart:
- Commonly used to compare the values of a variable at a given point in time.
- Plotted following Bar chart to visualize:
- Relationship between success rate of each orbit type
- 3. Line Chart:

Commonly used to track changes over a period of time . It helps depict trends over

- Average launch success yearly trend

[BNIKdata3/edadataviz.ipynb at main · BNIK-23/BNIKdata3 (github.com)]

EDA with SQL

The following SQL queries were performed:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA
- Display the 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 succesful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- 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.

[BNIKdata3/jupyter-labs-eda-sql-coursera_sqllite (1).ipynb at main · BNIK-23/BNIKdata3 (github.com)]

Build an Interactive Map with Folium

Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map

folium.circle and folium.marker

These marked out and circled launch sites in a map.

Marker_cluster

To colour label the launch sites as failed or successful launches

MounsePosition()

To get coordinate of a place on the map using the mouse

Folium.PolyLine()

To get distance between launch site and coastline

These helped to calculate the distance between launch sites and some locations in the map.

- BNIKdata3/lab_jupyter_launch_site_location.ipynb at main · BNIK-23/BNIKdata3 (github.com)

Build a Dashboard with Plotly Dash

Dashboard helped answer following questions:

1.Which site has the largest successful launches?

2.Which site has the highest launch success rate ?

3.Which payload range(s) has the highest launch success rate?

4.Which payload range(s) has the lowest launch success rate?

5.Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate?

Created the following charts:

1.Added a Launch Site Drop-down Input component to the dashboard to provide an ability to filter by all launch sites or a particular launch site

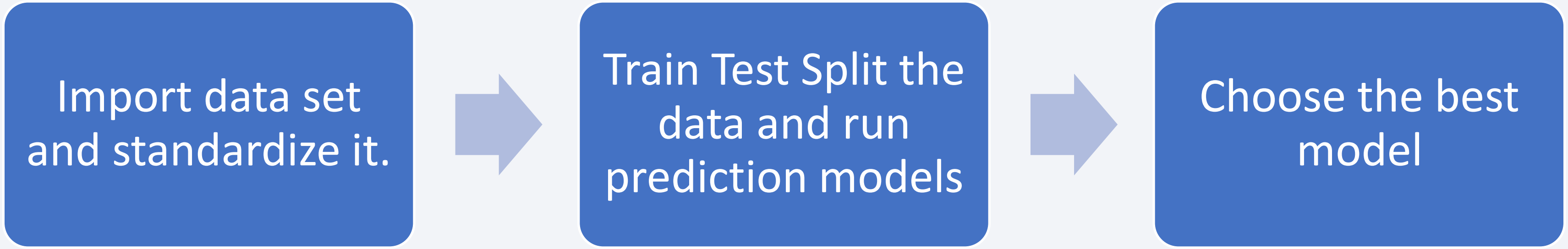
2.Added a Pie Chart to show total success launches when ‘All Sites’ is selected and show success and failed counts when a particular site is selected

3.Added a Payload range slider to the Dashboard to easily select different payload ranges to identify visual patterns

4.Added a Scatter chart to observe how payload may be correlated with mission outcomes for selected site(s). The color-label Booster version on each scatter point provided missions outcomes with different boosters

[BNIKdata3/ibm dash.ipynb at main · BNIK-23/BNIKdata3 (github.com)]()

Predictive Analysis (Classification)



[BNIKdata3/SpaceX_Machine Learning Prediction_Part_5 (1).ipynb at main · BNIK-23/BNIKdata3 (github.com)]

Predictive Analysis (Classification)

```python
from js import fetch
import io

URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part
resp1 = await fetch(URL1)
text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
data = pd.read_csv(text1)
```

Python

Standardize the data in X then reassign it to the variable X using the transform provided below.

+ Code + Markdown

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

TASK 3

+ Code + Markdown

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

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

we can see we only have 18 test samples.

TASK 4

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`.

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

```python
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()

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

Results

- Exploratory data analysis results

Orbits ES-LI, GEO, HEO, and SSO :

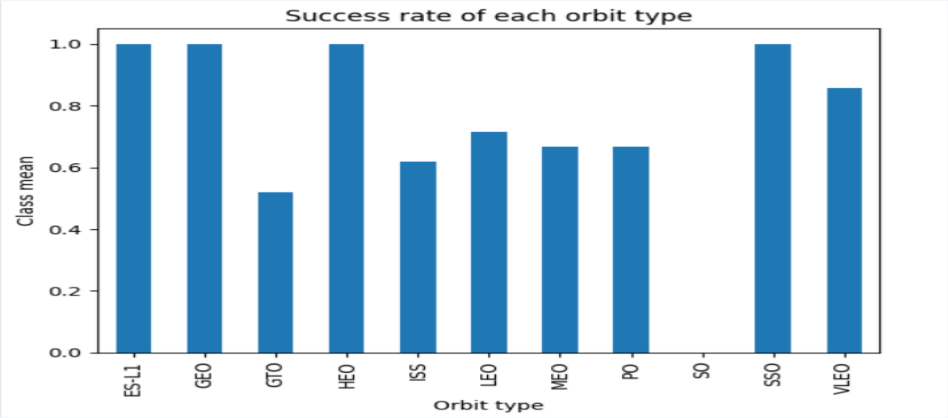
Have the highest succesees.

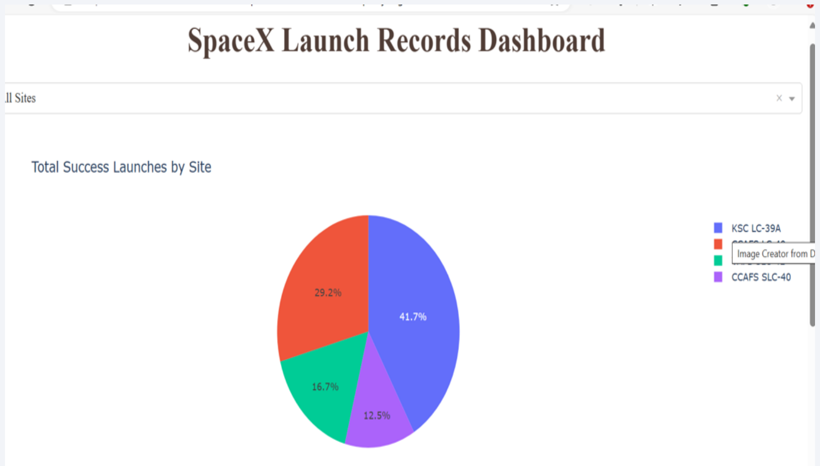
- Interactive analytics demo in screenshots

KSC-LC-39A has the most successful launches

- Predictive analysis results

The Decision Trees model Is the best model



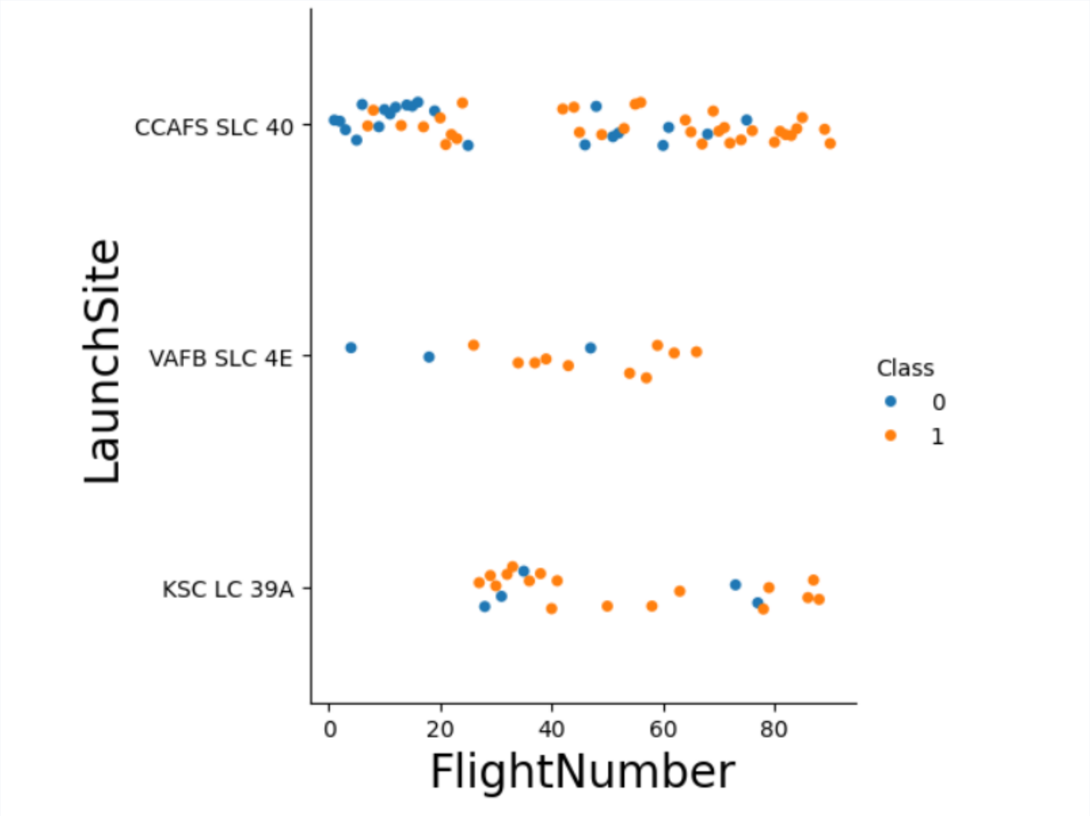


Section 2

Insights drawn from EDA

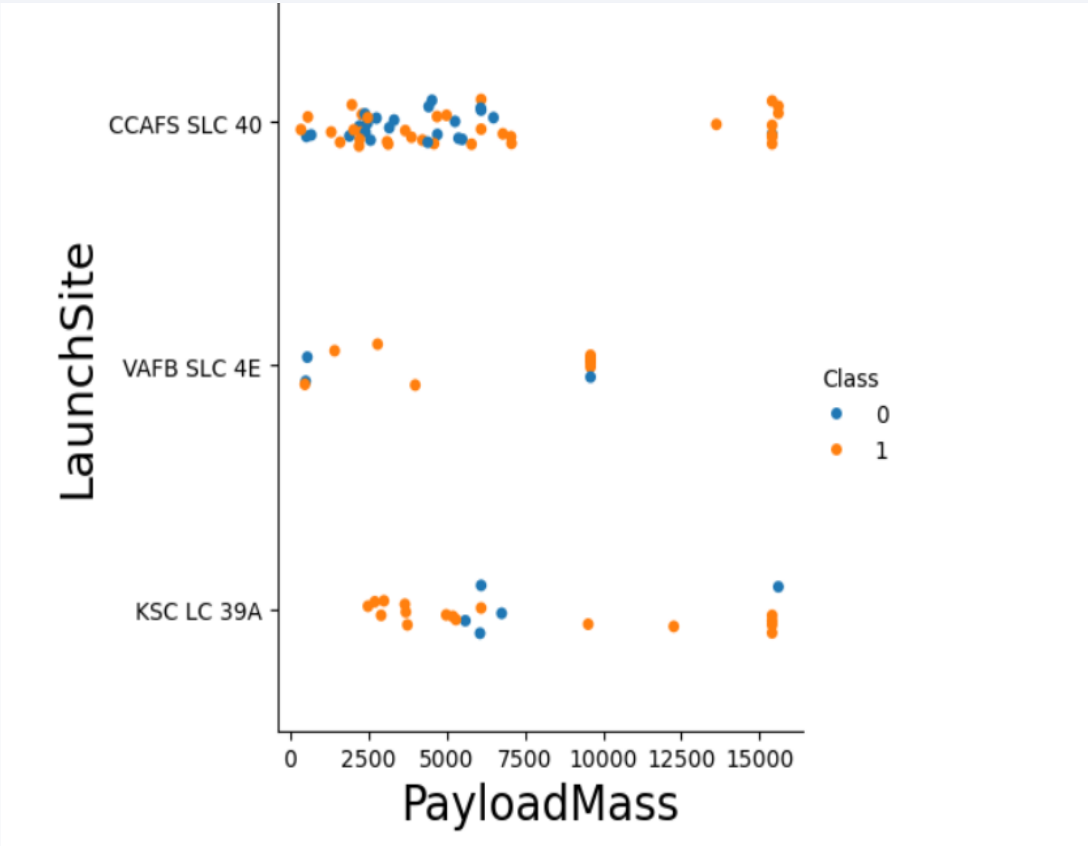
Flight Number vs. Launch Site

- Success rate increases with number of flights.
- For CCAFS SLC 40 data almost equally spread out.



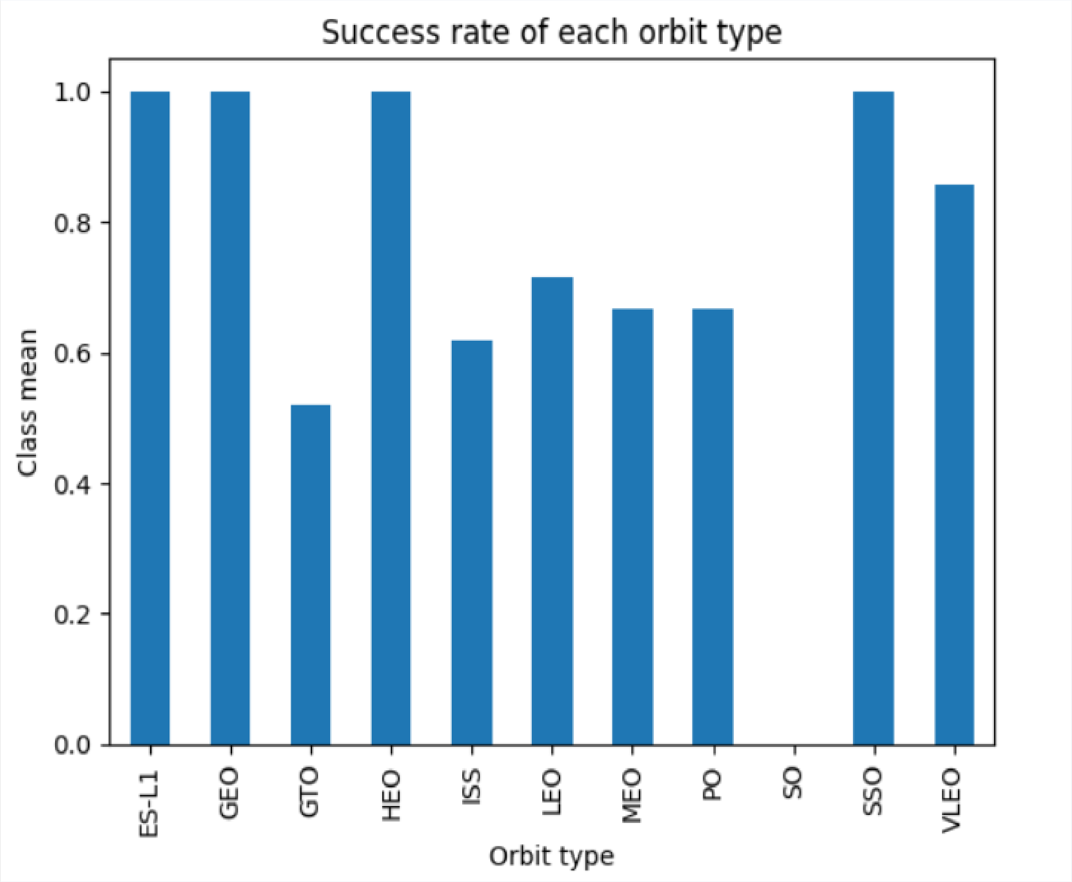
Payload vs. Launch Site

- For CCAFS SLC 40 success increases with payload mass.
- No launches for payload mass greater than 10000kg for VAFB SLC 4E



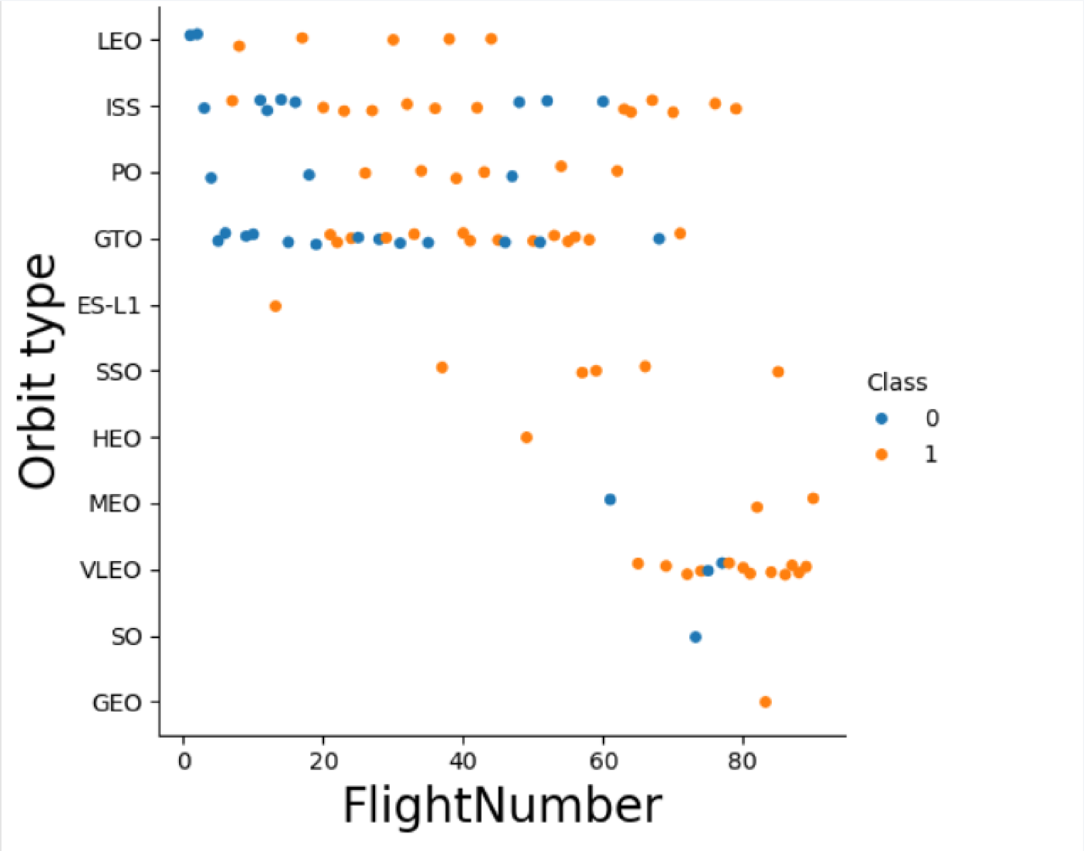
Success Rate vs. Orbit Type

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



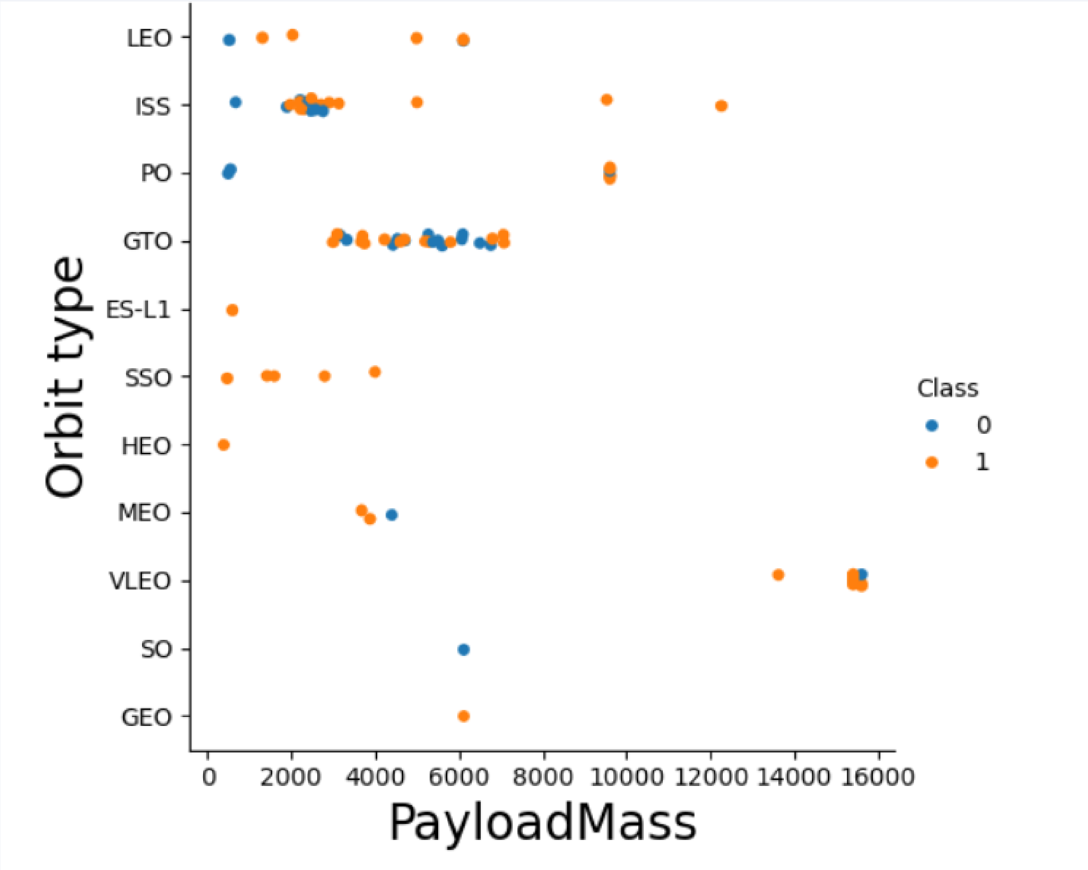
Flight Number vs. Orbit Type

- For orbit VLEO, first successful landing 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



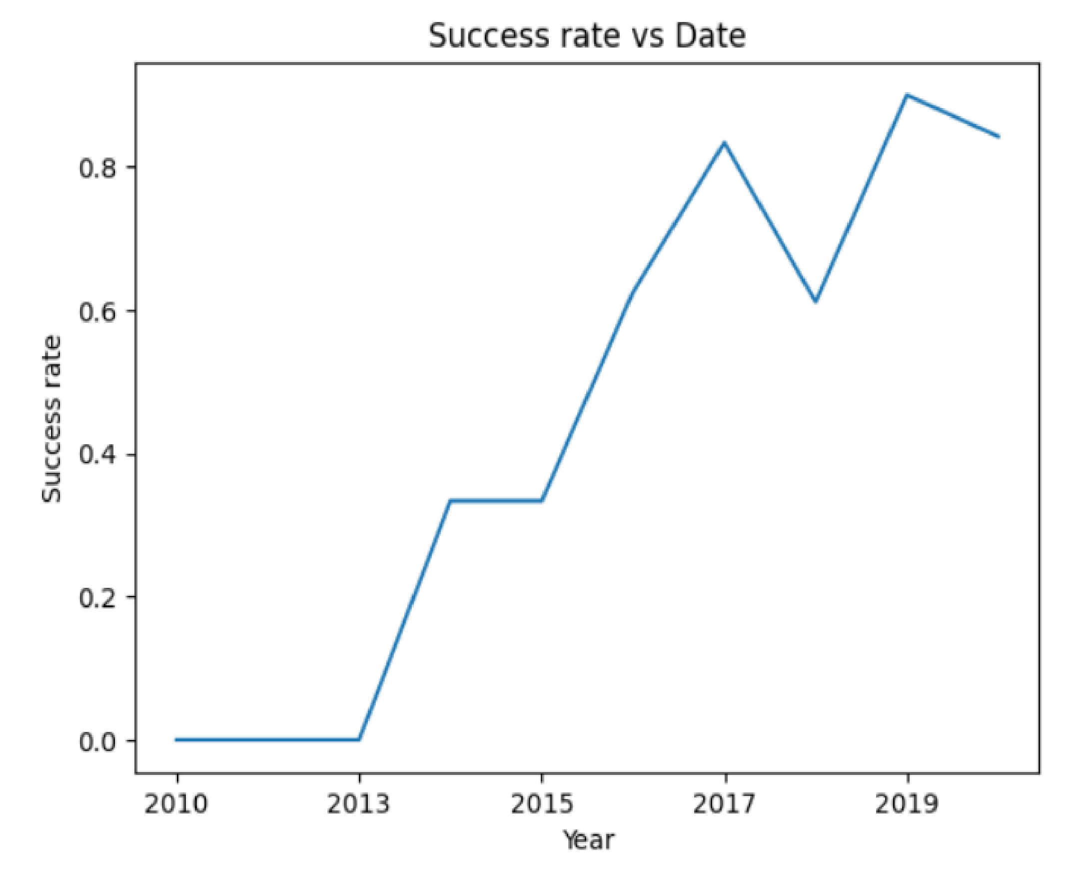
Payload Mass vs. Orbit Type

- Successful landing rates appear to increase with pay load for orbits LEO, ISS, PO, and SSO
- No correlation between payload and orbit type for GTO



Launch Success Yearly Trend

- Between 2013 and 2017, success rates increase to 80%
- Decline in success rate from 2017 till around 2018, then in increase again



All Launch Site Names

- Distinct ensures launch site names are given without repetition of those appearing many times.

Display the names of the unique launch sites in the space mission

```
%%sql
select Distinct "Launch_Site" from SPACEXTABLE
```

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

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

Launch Site Names Begin with 'CCA'

- Limit 5 ensures that only 5 records are given.
- Like ‘CCA%’ takes all records where Launch site begins with CCA

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

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Sum() gives the sum of values in a column.
- The sum is 45596 Kg

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

```
%%sql
select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where "Customer" = 'NASA (CRS)';
```

* sqlite:///my_data1.db
Done.

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Avg() gets the average of selected column
- Where is predicated to limit results

Display average payload mass carried by booster version F9 v1.1

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

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

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Min(Date) selects first or oldest date from date column.
- Where is predicate to limit which results we take based on landing outcome column

```
%%sql
select MIN("Date") from SPACEXTABLE where "Landing_Outcome" = 'Success (ground pad)';
```

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

MIN("Date")
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Conditions given after where predicate limit results of booster version to specified conditions.
- The four booster versions found are given.

```
%%sql
select "Booster_Version" from SPACEXTABLE 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

Total Number of Successful and Failure Mission Outcomes

- The query groups the data by Mission_Outcome column, then counts the number of occurences of failure or success.

```
%%sql
select "Mission_Outcome",count("Mission_Outcome") from SPACEXTABLE GROUP BY "Mission_Outcome";
```

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

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

Boosters Carried Maximum Payload

- Subquery executes first to give maximum payload mass. Then booster version for which payload mass is maximum is querried.

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%%sql
select "Booster_Version" from SPACEXTABLE where PAYLOAD_MASS__KG_ = ( select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

```
 * 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

2015 Launch Records

- The query lists landing outcome, booster version, and the launch site where landing outcome is failed in drone ship and the year is 2015

```
%%sql
select SUBSTR(Date,6,2) as Month, "Landing_Outcome", "Booster_Version", "Launch_Site" from SPACEXTABLE where SUBSTR(Date,0,5) = '2015' and "La
```

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

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- The query groups data by Landing outcome, counts the number of times each outcome occurs and ranks the results in descending order.

```
%%sql
select "Landing_Outcome",count("Landing_Outcome") from SPACEXTABLE where "Date" between '2010-06-04' and '2017-03-20' group by "Landing_Outcome
```

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

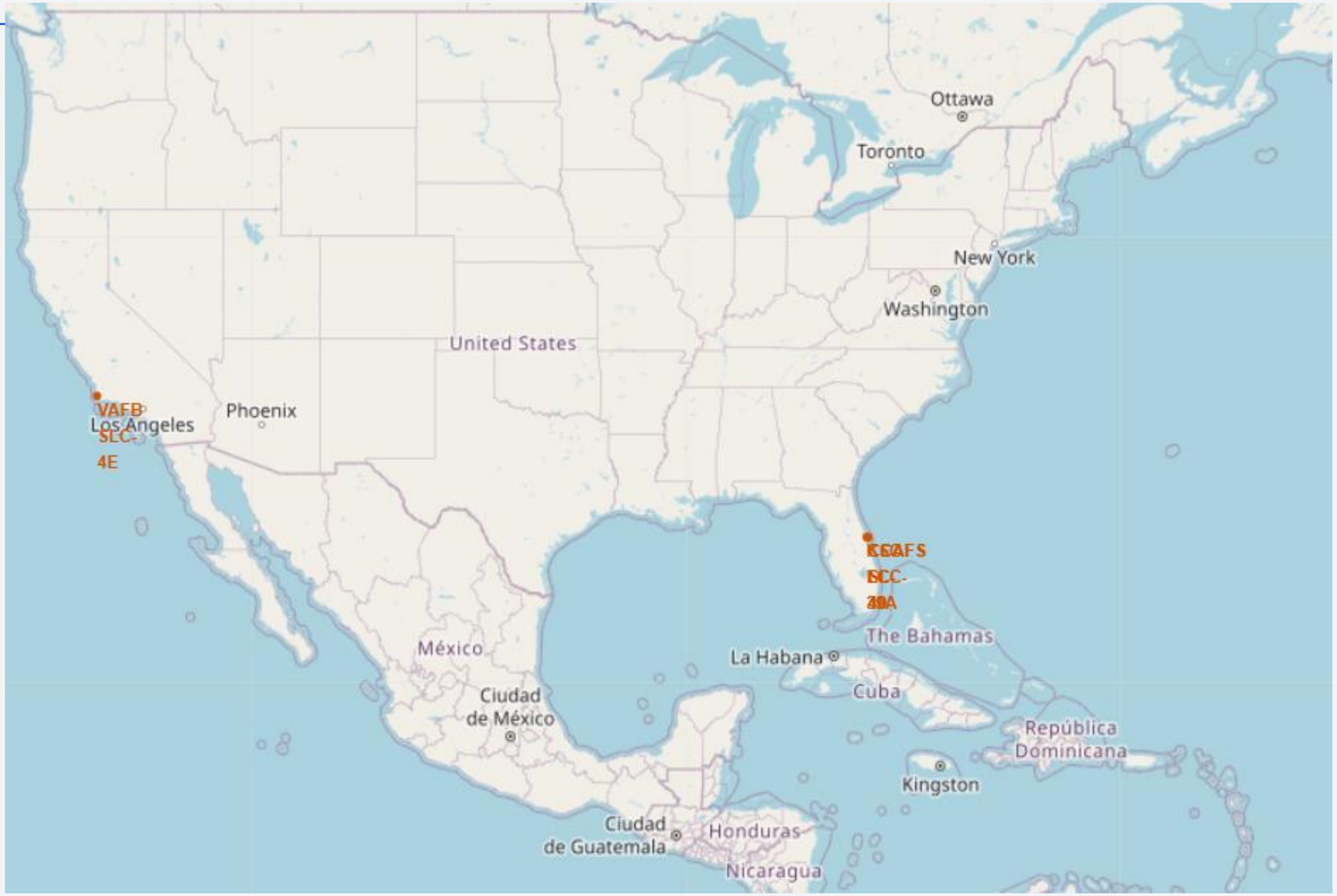
Landing_Outcome	count("Landing_Outcome")
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

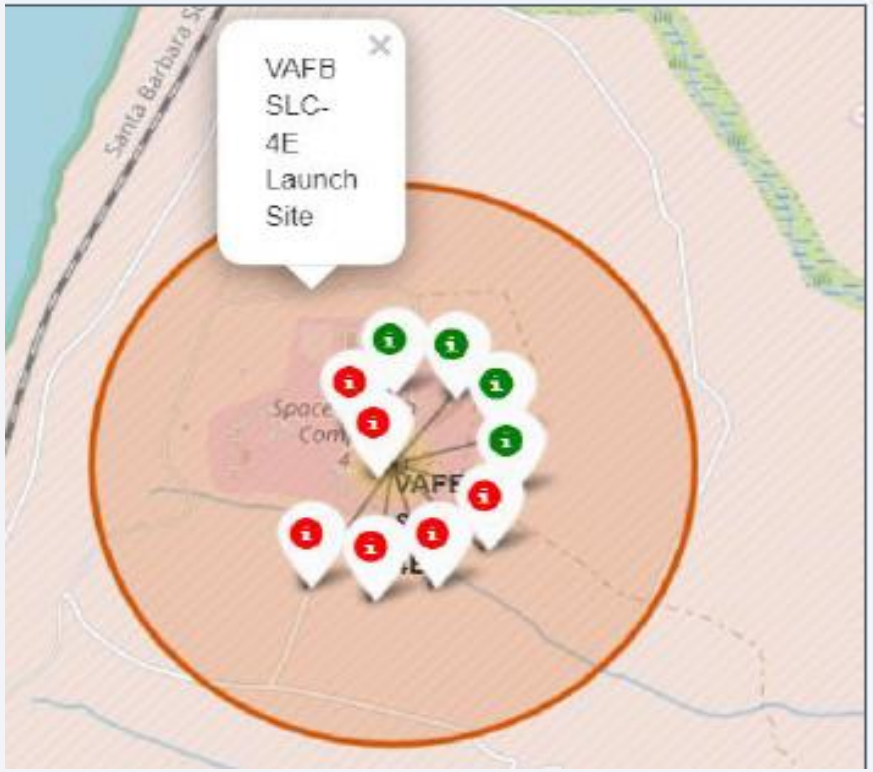
Launch Sites Locations

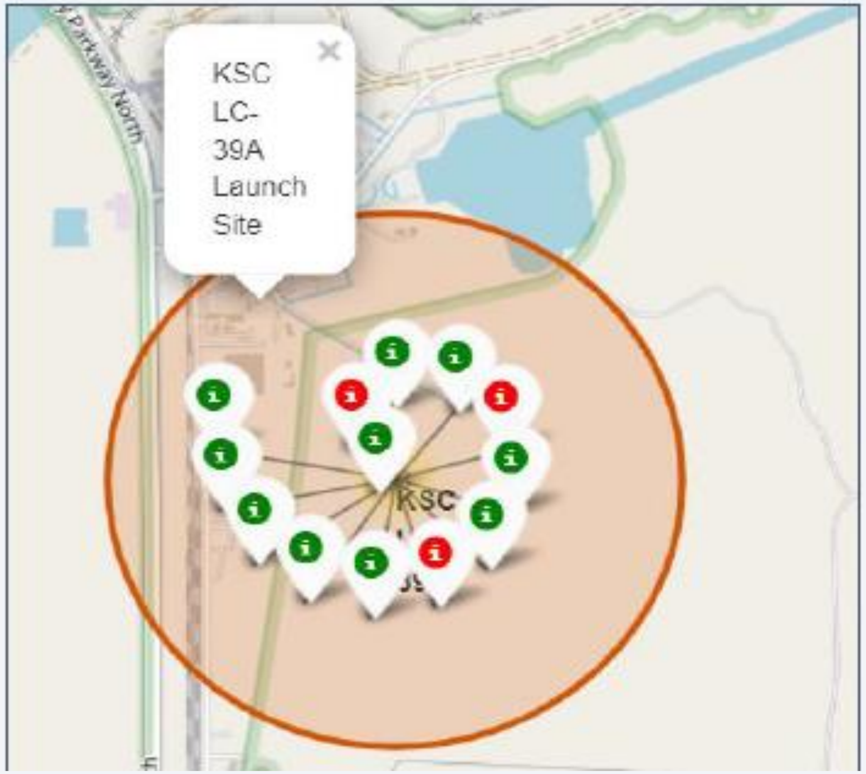
Map shows launch site locations marked on a USA map.



Succes vs Failure for all Launch Sites

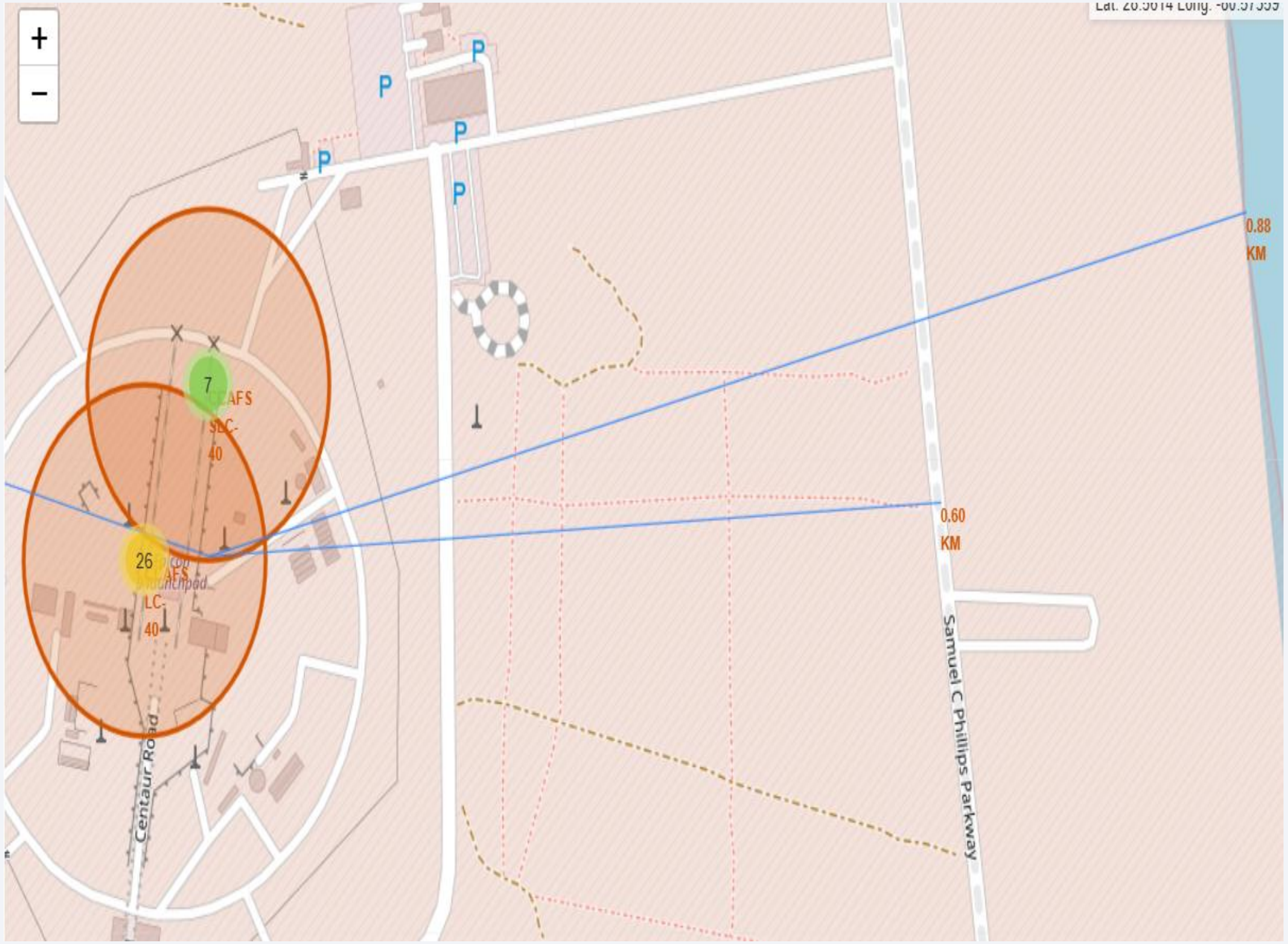
Figure shows KSC LC-39A has more successful results.





The launch site is much closer to highway than coastline.

Distance to coastline is 0.88km, to highway it’s 0.60km

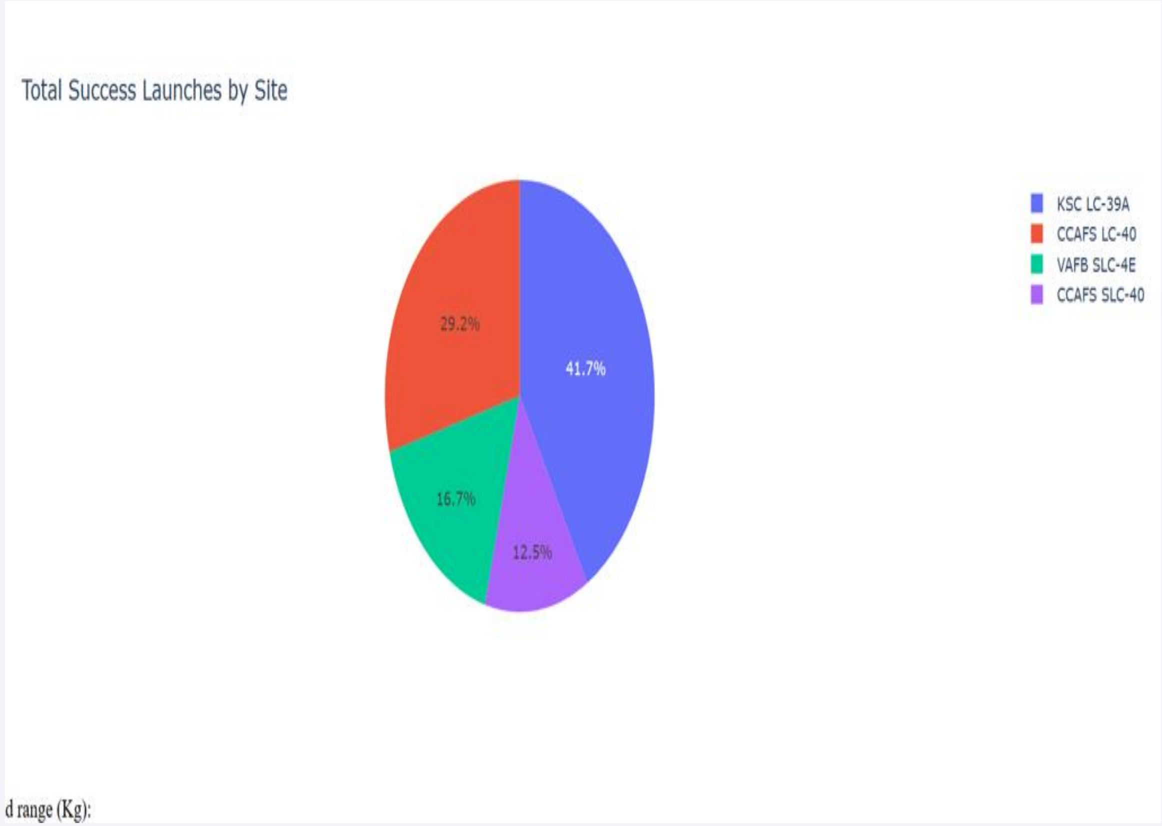


Section 4

Build a Dashboard with Plotly Dash

Launches by Site Chart

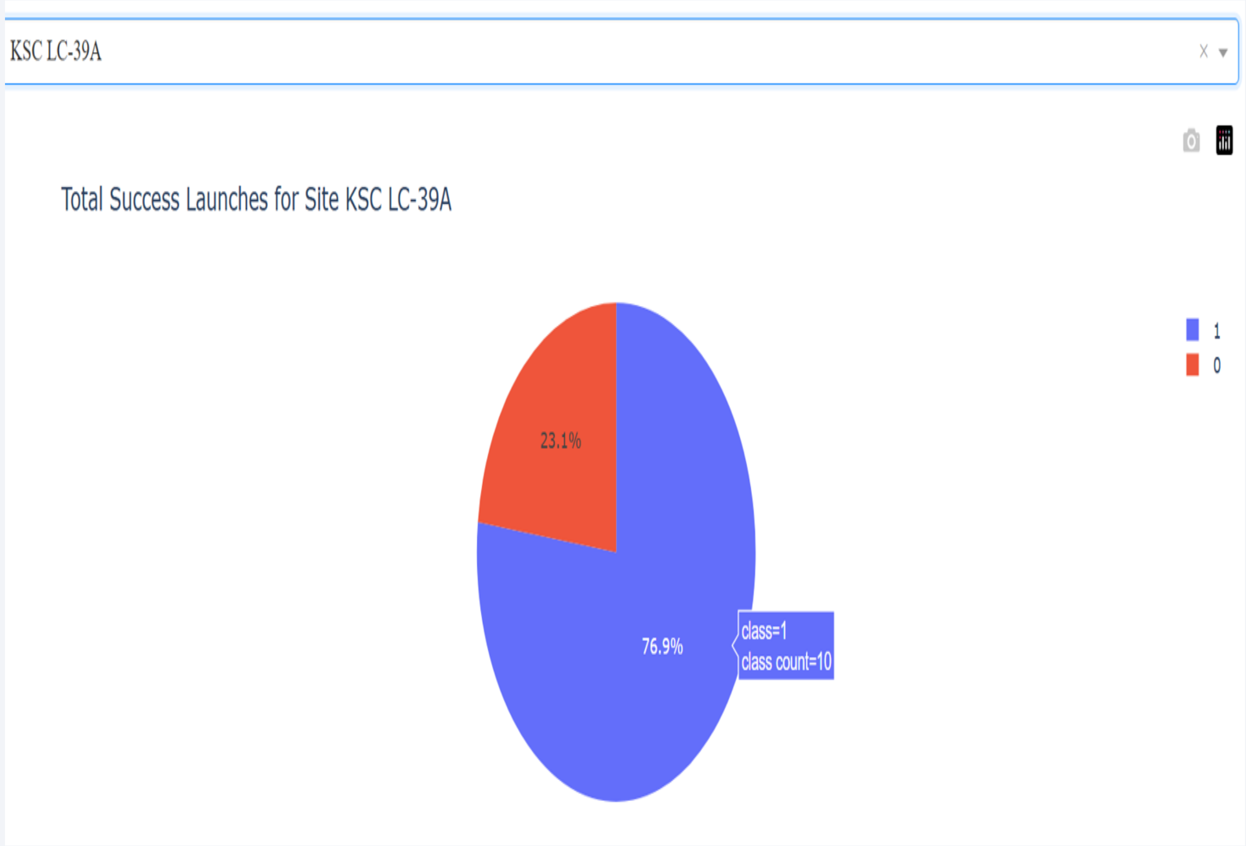
- KSC LC 39A has more successful launches compared to all sites
- CCAFS SLC-40 has the least successful launches



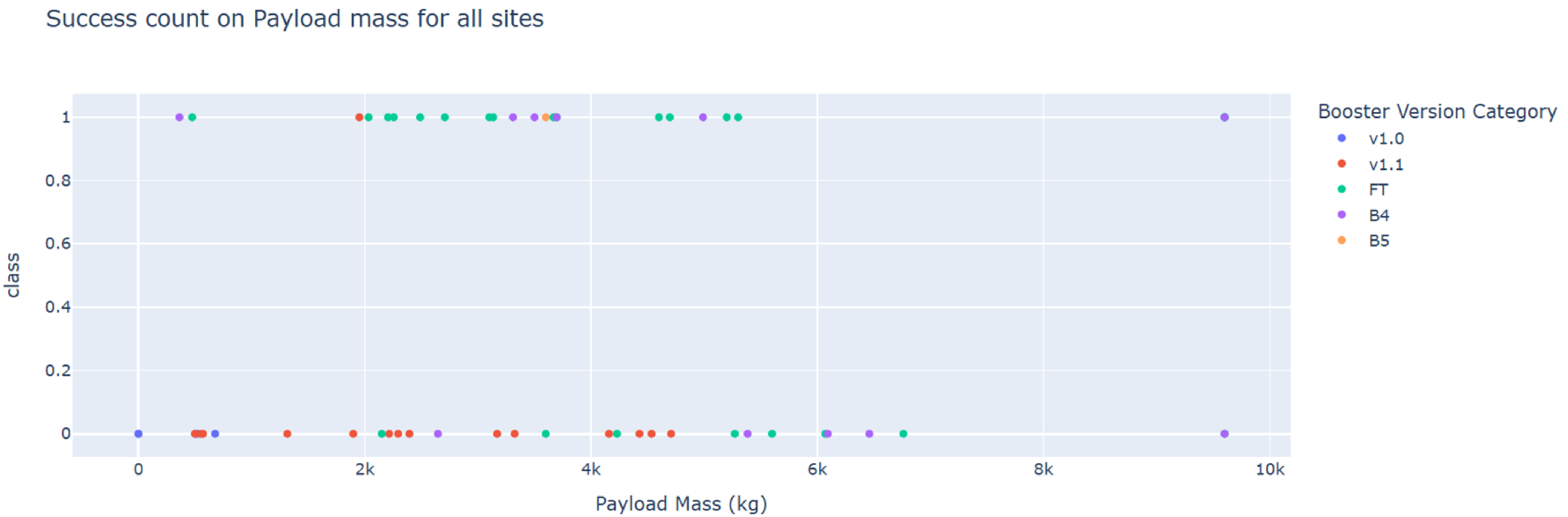
KSC LC-39A Success rate

KSC LC-39A has a 76.90% success rate.

23.1% launches are failed.



Scatter Plot for Launch Outcome vs Payload Mass

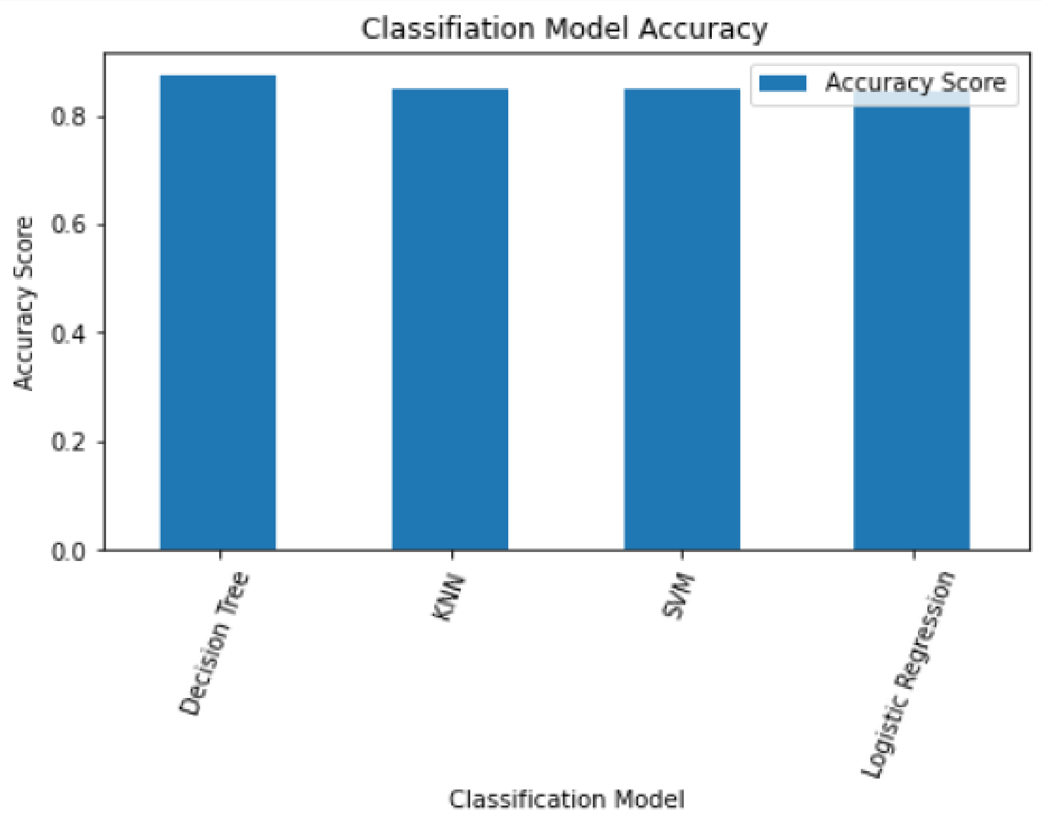


Section 5

Predictive Analysis (Classification)

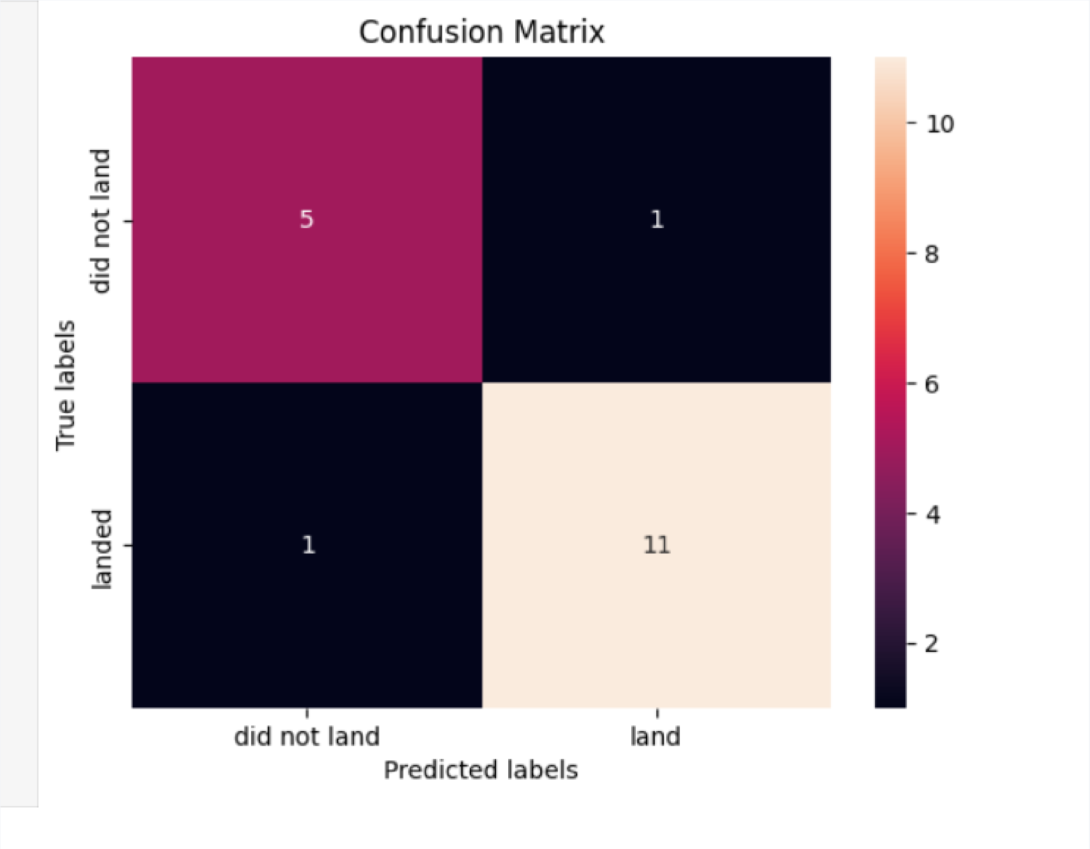
Classification Accuracy

- Decision Trees have the highest accuracy score.



Confusion Matrix

- 18 predictions made.
- 11 predictions for successful landing correctly predicted.
- 1 successful prediction incorrectly predicted as failed.
- 5 failed outcomes correctly predicted, 1 incorrectly classified as successful.
- 88% Accuracy



Conclusions

- 80% increase in success rate from 2013 to 2017.
- More successes with increased flight number.
- KSC LC-39A has more launch successes.
- Orbits ES L1, GEO, HEO, and SSO have the highest launch success rates and orbit GTO the
- Launch sites are closer to coastlines and railways than they are to cities
- Decision Trees model is the best predictor of launch outcomes.

Appendix

- Included relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets created during this project

Thank you!