IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Isabella Winters
24.03.2024

Outline

Executive Summary

Challenges

- There are already several very successful players in the space industry
- To enter the market it is necessary to be able to compete with SpaceX
- Major cost driver is the first stage of the rocket, the prediction of successful landing is therefore essential

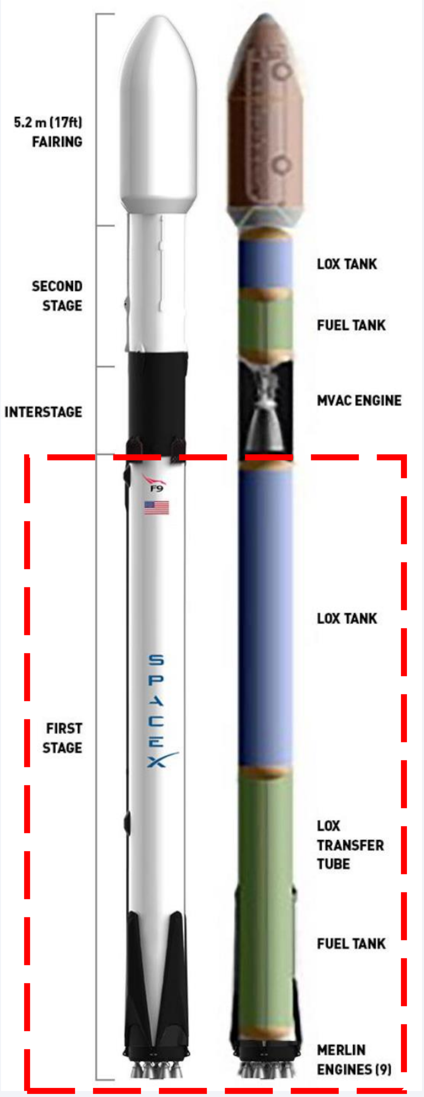
Results

- Based on the available Data it was possible to analyse the landing outcomes based on different parameters
- Through machine learning several predictive models were trained to compare their performance

Methodologies

- Collecting Data
  - SpaceX API
  - Web Scraping
- Data wrangling
- Exploratory Data Analysis (EDA)
  - Data Visualization using Pandas and Matplotlib
  - SQL Queries
- Visual Analytics
  - Interactive map with Folium
  - Dashboard with Plotly Dash
- Predictive analysis
  - Classification

Introduction



https://www.spacex.com/media/falcon-users-guide-2021-09.pdf

The ability to reuse the **first stage** of a Falcon 9 rocket gives SpaceX a big advantage over their competitors. Launches can be offered at a much lower price, because this stage is so large and expensive. Instead of costs upwards of 165 million dollars each, Falcon 9 rocket launches can be offered with a cost of 62 million dollars.

To determine the cost of a launch we have to find out if the first stage will land successfully or not. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

To gain this knowledge it is necessary to collect the available information for Falcon 9 rocket launches. With this data it is possible to train a machine learning model and predict if SpaceX will reuse the first stage.

We will also find out how different parameters like launch site, payload mass, orbit type and booster version influence the landing outcomes.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scraping Falcon 9 Launch Records
- Perform data wrangling
  - Landing outcomes converted into Training Labels (0 = not successful, 1 = successful)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression, Support Vector Machine, Decision Tree Classifier, K-nearest Neighbors

Data Collection

Data collected with **SpaceX REST API** (Endpoint for past launch data: https://api.spacexdata.com/v4/launches/past)

Request SpaceX launch data using the GET request → Turn it into a Pandas dataframe → Get additional information → Combine columns into a dictionary → Filter the dataframe for Falcon 9 launches → Deal with Missing Values → Export data to a CSV

Falcon 9 Launch Records collected with **Web Scraping** from Wikipedia

Request Launch data using the GET request → Create a BeautifulSoup object → Extract all column / variable names from the HTML table header → Parse the HTML tables to fill a dictionary → Create a dataframe → Export data to a CSV

Data Collection – SpaceX API

Link to GitHub: [Data Collection API.ipynb]

A

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

via SpaceX url

B

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API
response = requests.get(static_json_url)
results = json.loads(response.text)
data = pd.json_normalize(results)
```

with a static response object

C

```
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
LaunchSite = []
# Call getLaunchSite
getLaunchSite(data)
```

D

```
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}
```

E

```
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

F

```
# Calculate the mean value of PayloadMass column
avg_Payload = data_falcon9["PayloadMass"].astype("float").mean(axis=0)
avg_Payload

# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, avg_Payload, inplace=True)
data_falcon9.isnull().sum()
```

G

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

- A - Request data with GET method
- B - Decode the response content as a Json and turn it into a Pandas dataframe
- C - Get additional information for data represented as IDs with helper functions, e.g. for Launch Site
- D - Combine the columns into a dictionary
- E - Filter for Falcon 9 data only
- F - Fill empty values (mean used for feature PayLoadMass)
- G - Convert data to csv file

Data Collection - Scraping

Link to GitHub: [Web Scraping.ipynb]

A

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

B

```
soup = BeautifulSoup(data.content,'html.parser')
```

C

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

for x in range(len(all_th)):
    name = extract_column_from_header(all_th[x])
    if (name is not None and len(name) > 0):
        column_names.append(name)
```

D

```
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            launch_dict['Flight No.'].append(flight_number)
            print("Flight number:", flight_number)
            datatimelist=date_time(row[0])
```

E

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

F

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

A - Request data with GET method

B - Create a BeautifulSoup object from the response text content

C - Iterate through all html table header elements to extract the column/variable names

D - Parse the tables to fill a dictionary with launch data, e.g. for Flight Numbers

E - Create a dataframe

F - Convert data to csv file

Data Wrangling

Link to GitHub: [Data Wrangling.ipynb]()

A

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

B

```
df.isnull().sum()/len(df)*100
```

C

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

D

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

E

```
df["Class"].mean()
```

A - Load dataset

B - Identify missing values

C - Investigate columns
- Launch Sites
- Orbit
- Landing Outcomes

D - Create landing outcomes label
- Class 0 for unsuccessful landings
- Class 1 for successful landings

E - Determine Success Rate

EDA with Data Visualisation

Link to GitHub: Data Visualisation.ipynb

It is often difficult to understand complex datasets. Data visualization offers an easily approach to comprehend these datasets. Trends and relationships can be highlighted that might be otherwise overseen.

In this project different types of plots are used:

Scatter plots to explore relationships between variables, helping us identify correlations or trends.

- Flight Number vs. Payload Mass
- Flight Number and Launch Site
- Payload and Launch Site
- Flight Number and Orbit type
- Payload and Orbit type

Bar plots to compare categories or groups, providing a visual comparison of their values.

- Success Rate of each orbit type

Line plots to capture trends and changes over time, allowing us to see patterns and fluctuations.

- Yearly Success Rate

EDA with SQL

Link to GitHub: [EDA with SQL.ipynb]

In this project SQL queries were used to answer following questions:

- 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 (with use of a subquery) 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.

Build an Interactive Map with Folium

Link to GitHub: Map with Folium.ipynb

Interactive visual analytics provide a lot of advantages, like:

- Explore and manipulate data in an interactive and real-time way
- Patterns could be found fast and effectively
- User engagement and interest is higher than with static graphs

For this project these map objects were used:

- **circles** and **marker** to add a highlighted circle area with a text label for launch sites
- **marker cluster** for markers with different colors to show launch successes and failures for a launch site
- **lines** between launch sites and landmarks or infrastructure to show the distance

(to calculate the distance the **mouse position** was included in the map to determine the necessary coordinates)

Build a Dashboard with Plotly Dash

Link to GitHub: [Plotly Dash.ipynb]

For this project these elements were added to the dashboard:

- **Dropdown List** to select Launch Sites, this enables to investigate them further. The other charts on a dashboard can be updated with this selection.
- **Pie Chart** to easily see how much each launch sites distributes to the total success
- **Slider** to select from a range of values. This enables to zoom in and see more details.
- **Scatter Chart** to show the correlation between payload and launch success. Booster versions are displayed in different colors in this scatter chart to provide more information.

Predictive Analysis (Classification)

Link to GitHub: [Predictive Analysis.ipynb]

A
```
URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv"
resp1 = await fetch(URL1)
text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
data = pd.read_csv(text1)
```

B
```
Y = data["Class"].to_numpy()
transform = preprocessing.StandardScaler()
X = transform.fit(X).transform(X)
```

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

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

E
```
logreg_cv.score(X_train, Y_train)
```

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

G
```
Report2 = {
    "Accuracy (on validation data)": [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_],
    }
df1 = pd.DataFrame(Report2)
df1.rename(index={0:'LogReg',1:'SVM',2:'Tree',3:'KNN'}, inplace=True)
print(df1)
```

A - Load dataframe

B - Create a NumPy array and transform it

C - Split the data into training and test data

D - Fit a GridSearchCV object to find the best parameters

E - Calculate the accuracy on the test data

F - Check predictions with Confusion Matrix

Steps D-F are done for
- Logistic Regression
- Support Vector Machine
- Decision Tree Classifier
- K-nearest neighbors

(code example on the left side given for Logistic Regression)

G - Compare accuracy and performance of methods to find the best method

Results

The results of this project are included in the next sections:

- EDA - Data Visualisation
- EDA - with SQL
- Interactive Map (Folium)
- Interactive Dashboard (Plotly Dash)
- Predictive Analysis (Classification)

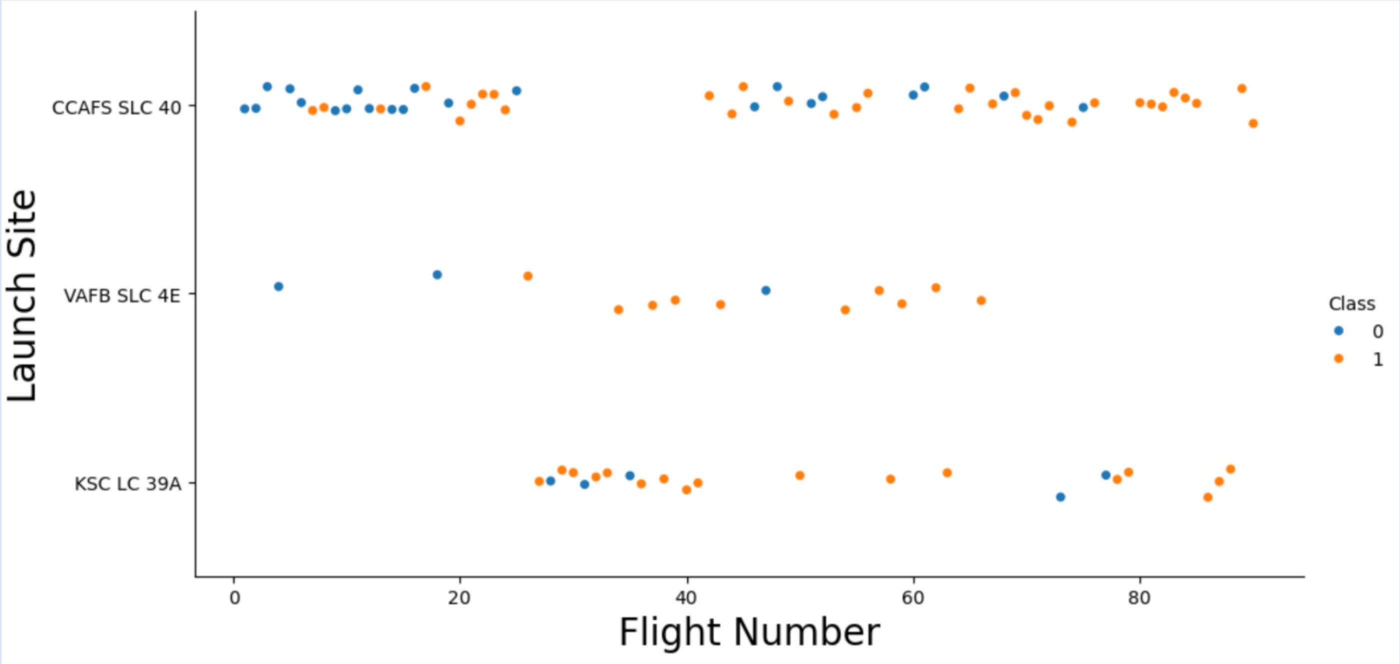
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

Class 0 (blue) = unsuccessful
Class 1 (orange) = successful

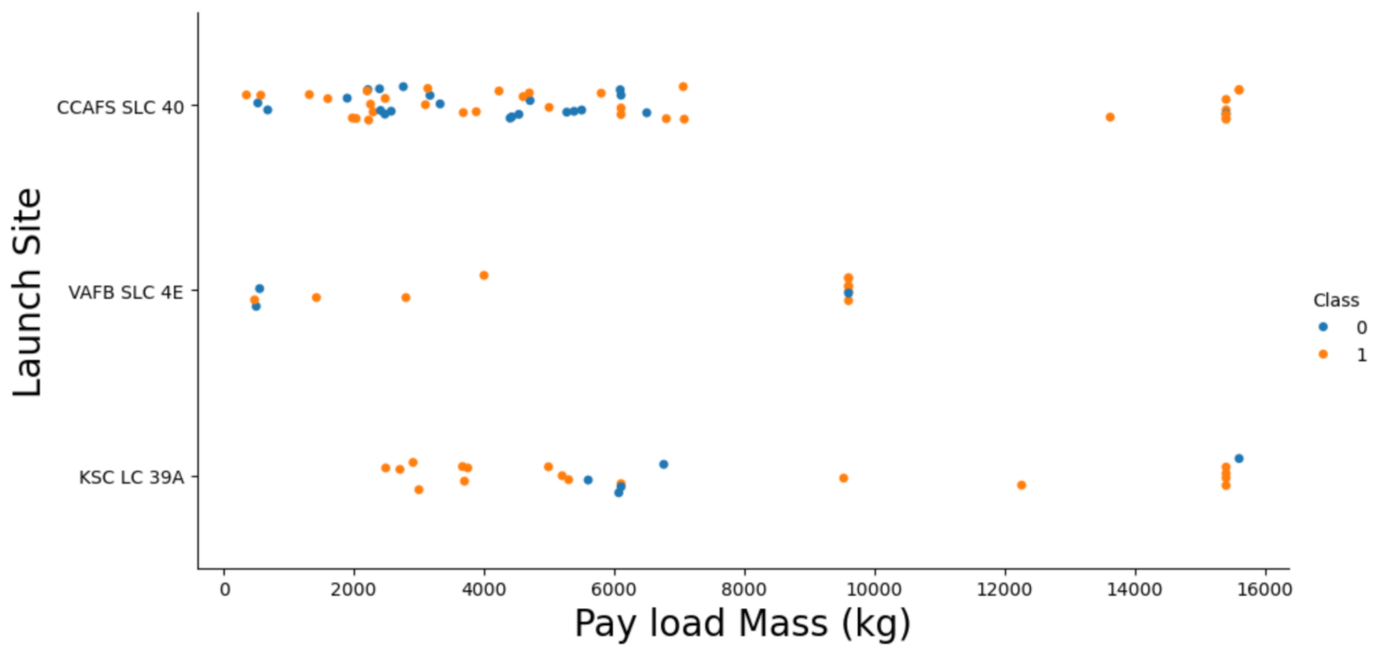
- For each site most of the earlier flights failed, while the higher flight numbers were more successful
- Site VAFB SLC 4E was used much less than the others, although it had only few unsuccessful outcomes



Payload vs. Launch Site

Class 0 (blue) = unsuccessful
Class 1 (orange) = successful

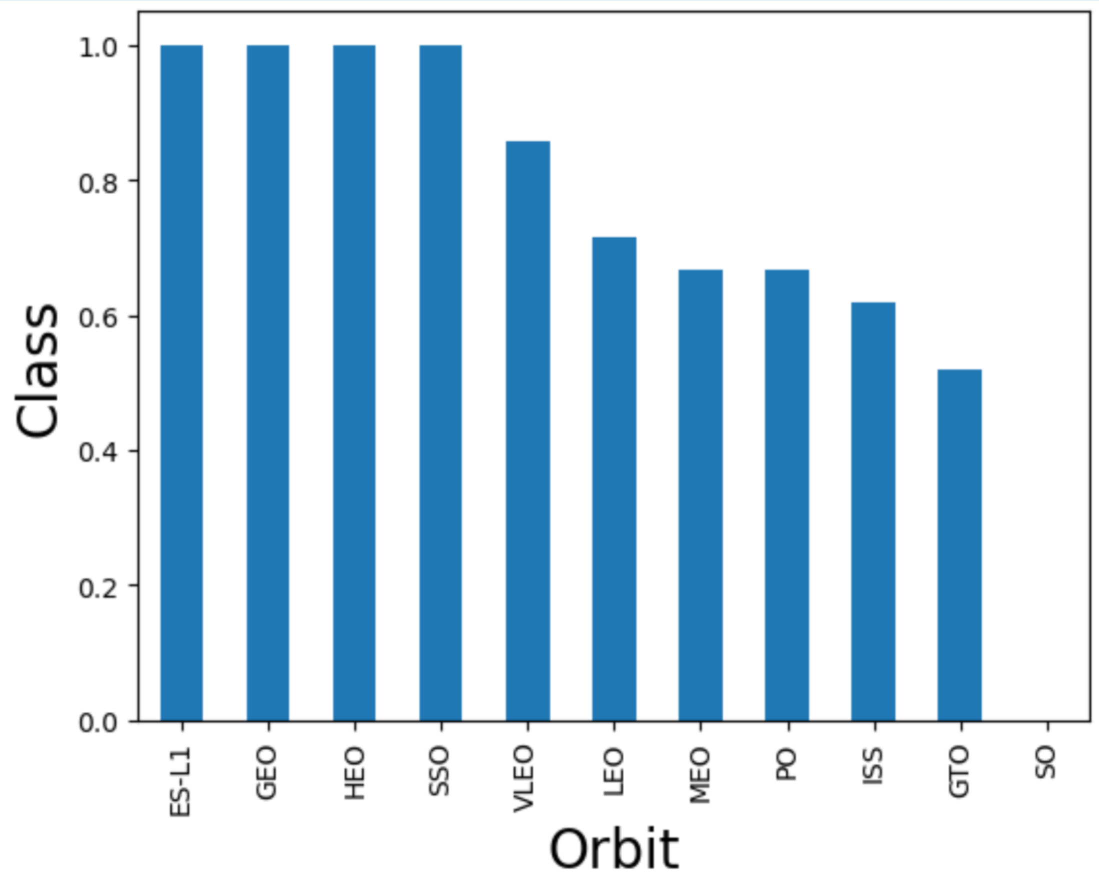
- Most of the launches with high payloads were successful
- Site VAFB SLC 4E was not used for payloads over 10.000 kg
- While transporting low payloads the site CCAFS SLC 40 had much more failures than the other sites



Success Rate vs. Orbit Type

The y-axis shows the mean success rate for orbits

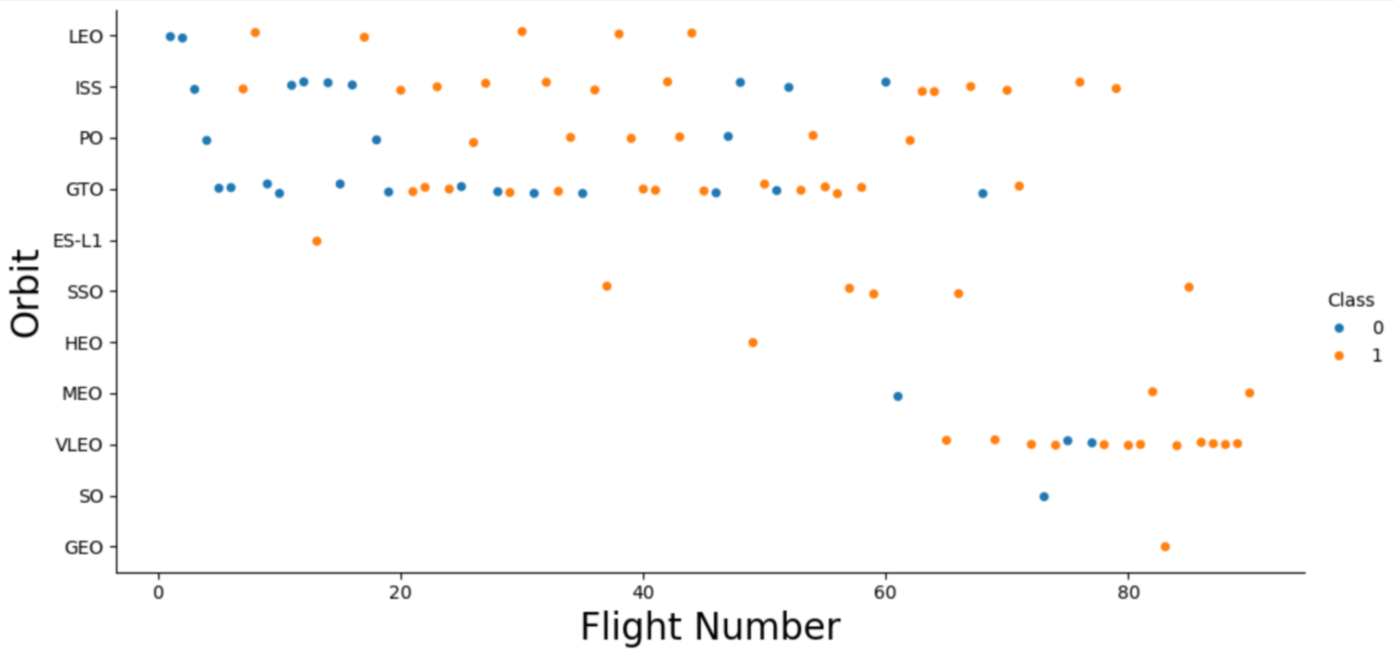
- Orbits ES-L1, GEO, HEO, and SSO have 100% success
- GTO has the lowest success rate
- For orbit SO no data is available



Flight Number vs. Orbit Type

Class 0 (blue) = unsuccessful
Class 1 (orange) = successful

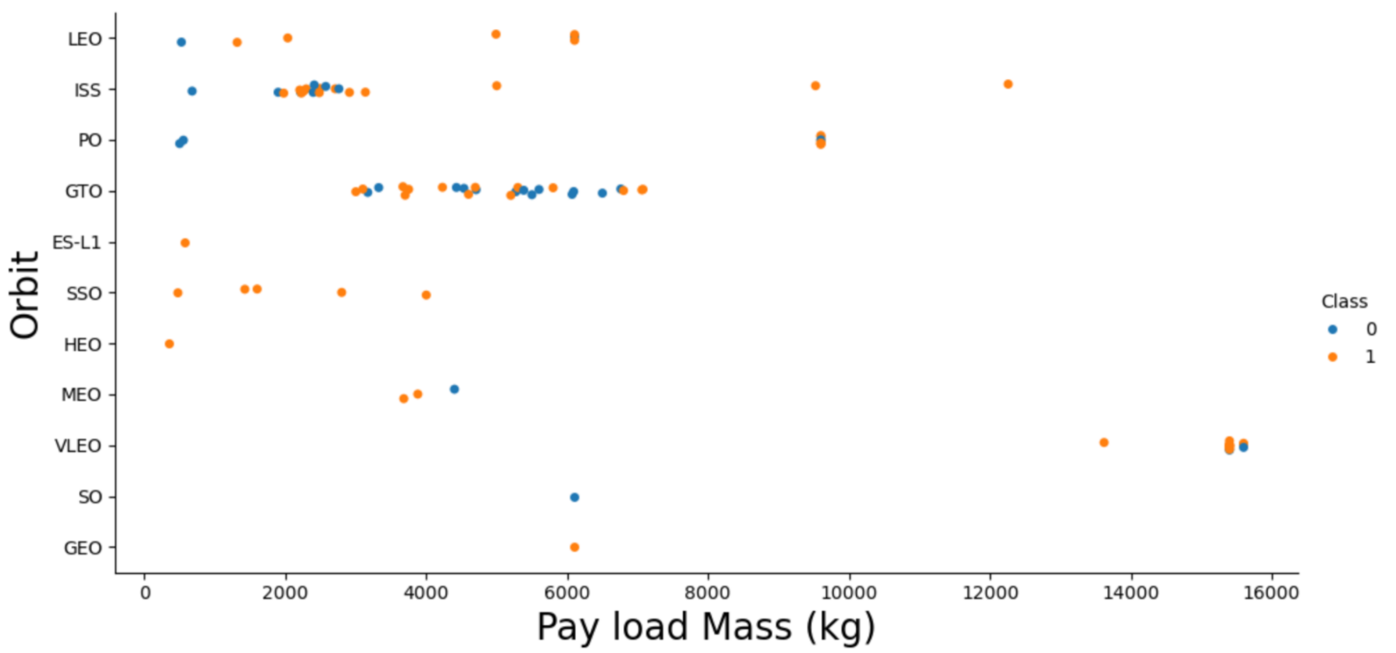
- GTO orbits show mixed outcomes
- LEO, ISS, MEO, and VLEO have improved outcomes with later flight numbers
- Some orbits were used only once (ES-L1, HEO, SO, GEO)
- SSO is the orbit that was used more than once with 100% success



Payload vs. Orbit Type

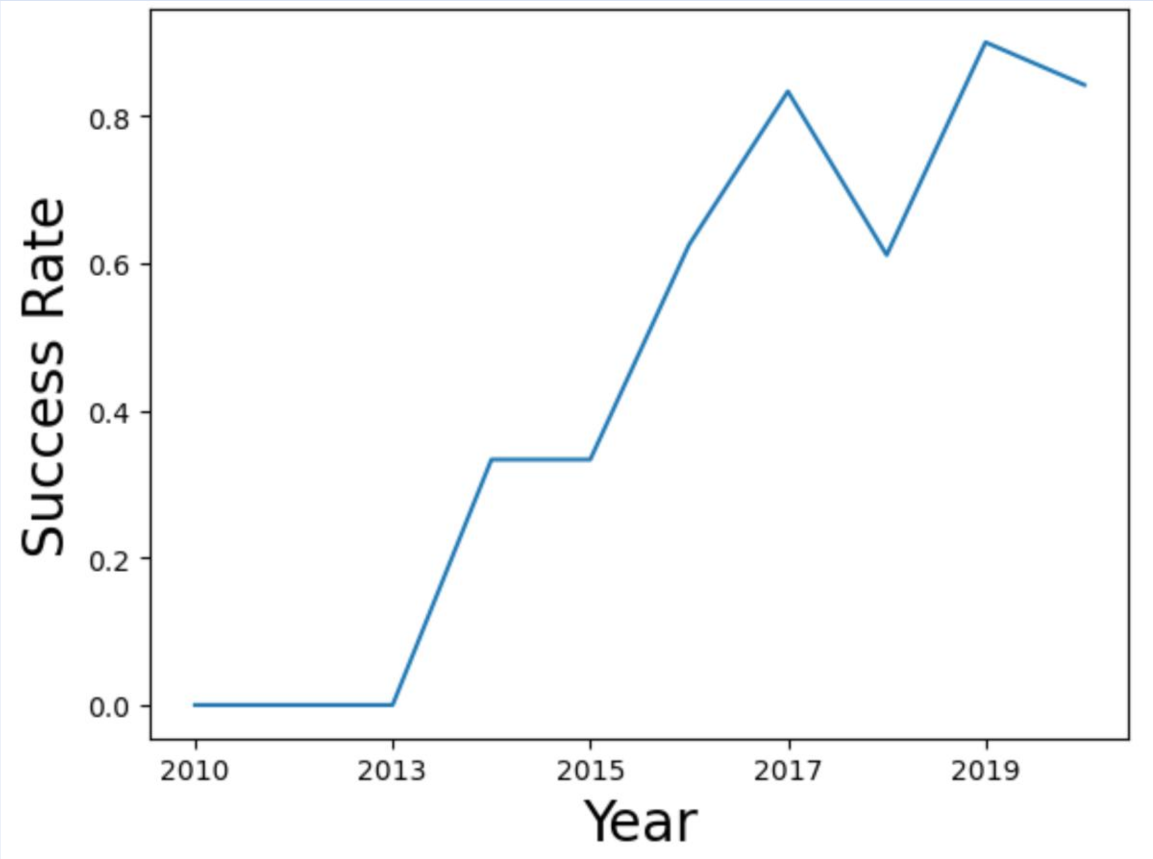
Class 0 (blue) = unsuccessful
Class 1 (orange) = successful

- 4 orbits were used for payloads over 10.000 kg
- For low payloads ES-L1, SSO, HEO, and GEO had 100% success
- Payload cannot be used as an indicator if an GTO orbit will have a positive outcome or not



Launch Success Yearly Trend

- From 2010 until 2013 all outcomes were not successful
- With exception of 2018 and 2020 the success rate increases each year
- 100% success is not yet achieved



All Launch Site Names

- Used query for the names of the launch sites:

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

to get only unique results

table with SpaceX data

- outcome

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

CCAFS = Cape Canaveral Space Force Station

- Launch Complex LC-40
- Space Launch Complex SLS-40

Vandenberg Air Force Base

- Space Launch Complex 4E

Kennedy Space Center

- Launch Complex 39A

Launch Site Names Begin with 'CCA'

- Used query for the launch sites beginning with 'CCA':

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%'
```

search in the column Launch_Site

filter the results
- starting with CCA
- after that % is used as a wildcard

- outcome

Launch_Site
CCAFS LC-40
CCAFS SLC-40

Total Payload Mass

- Used query for the calculation of total payload mass carried by boosters launched by NASA (CRS):

```
%sql SELECT CUSTOMER, sum(PAYLOAD_MASS__KG_) as total_payload_mass from SPACEXTBL where CUSTOMER LIKE 'NASA (CRS)%';
```

calculate the sum of the payload   column name in the outcome table   filter for NASA (CRS) as customer

- outcome

Customer	total_payload_mass
NASA (CRS)	48213

Average Payload Mass by F9 v1.1

- Used query for the calculation of average payload mass carried by F9 v1.1:

```
%sql SELECT Booster_Version, avg(PAYLOAD_MASS__KG_) as average_payload_mass from SPACEXTBL where Booster_Version LIKE 'F9 v1.1%';
```

calculate the average of the payload

column name in the outcome table

filter the booster version
(screenshot for query was cut off on the right hand side due to limited cell width in Jupiter Labs)

- outcome

Booster_Version	average_payload_mass
F9 v1.1 B1003	2534.6666666666665

First Successful Ground Landing Date

- Used query to find the first successful landing outcome on ground pad:

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome IS 'Success (ground pad)';
```

select the lowest value in column Date

filter for successful landings on ground pad

- outcome

MIN(Date)
2015-12-22

December 22, 2015

Successful Drone Ship Landing with Payload between 4000 and 6000

- Used query to find names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000:

```
%sql SELECT DISTINCT Booster_Version FROM SPACEXTBL WHERE ((Landing_Outcome IS 'Success (drone ship)') AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000));
```

filter successful drone ship landing

filter payload in defined range

combine two criteria to filter the query result

- outcome

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

Total Number of Successful and Failure Mission Outcomes

- Used query to calculate the total number of successful and failure mission outcomes:

```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) AS "total number" FROM SPACEXTBL GROUP BY TRIM(Mission_Outcome);
```

calculate total number

remove space after string

group result by Mission_Outcome

- outcome

Mission_Outcome	total number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

without trim "Success" and "Success " (with empty space after string) would be counted separately

Boosters Carried Maximum Payload

- Used query to list the names of the booster which have carried the maximum payload mass:

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

select unique values for booster

select maximum payload mass

define condition to filter query results

- outcome

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

- Used query to list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015: (one line query split up in 2 screenshots here)

```
%sql SELECT Date, Landing_Outcome, Booster_Version, Launch_Site, substr(Date,0,5) AS YEAR, substr(Date,6,2) AS MONTH FROM SPACEXTBL
```

select columns to be shown in result

to get months and years from format YYY-MM-DD

```
WHERE (Landing_Outcome IS 'Failure (drone ship)') AND (substr(Date,0,5) IS '2015');
```

filter failures for drone ship

combine two filter criteria

filter for year

- outcome

Date	Landing_Outcome	Booster_Version	Launch_Site	YEAR	MONTH
2015-01-10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015	01
2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015	04

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

- Used query 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: (one line query split up in 2 screenshots here)

```
%sql SELECT Landing_Outcome, Count(Landing_Outcome) FROM SPACEXTBL WHERE (Date BETWEEN '2010-06-04' AND '2017-03-20')
```

calculate number of landing outcomes

filter result for Date within a defined period of time

```
GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC;
```

sort the result in descending order for number of landing outcomes

- outcome

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

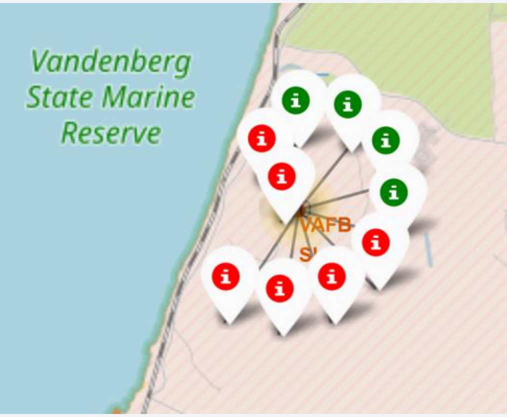
Locations of Launch Sites

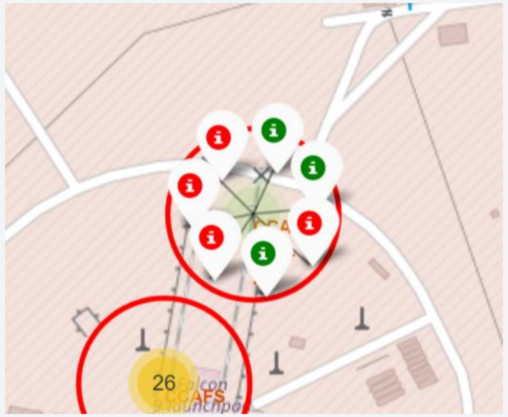
- All launch sites are located in the United States – one near the west coast and three near the east coast
- The launch sites are located near the Equator line

Launch Outcomes

Site VAFB SLC-4E

- More than half of the launch outcomes were not successful



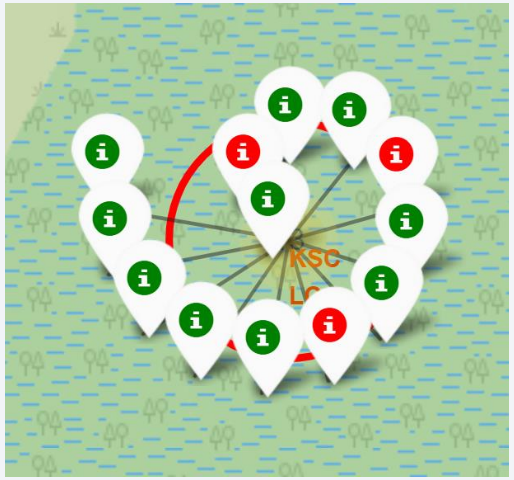


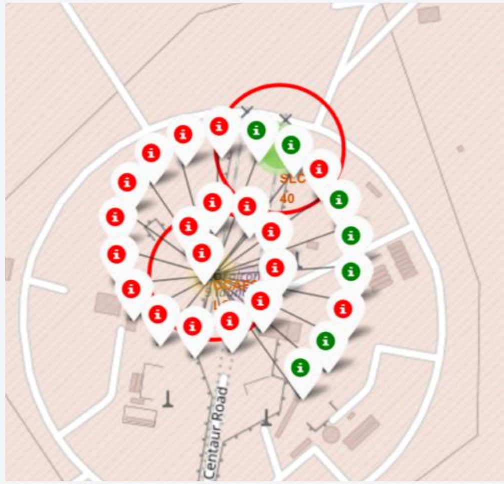
Site CCAFS SLC-40

- A low number of launches and a mixed result of launch outcomes

Site KSC LC-39A

- Most launch outcomes were successful, only 3 were not successful





Site CCAFS SC-40

- Highest number of launches, but a low number of successful outcomes

Marker in green = successful outcome
Marker in res = unsuccessful outcome

Proximities to Landmarks and Infrastructure

Launch sites are built far away from cities, but near to infrastructure like highways or railways.

Distance to next city: 14.76 km

Distance to next highway: 0.84 km

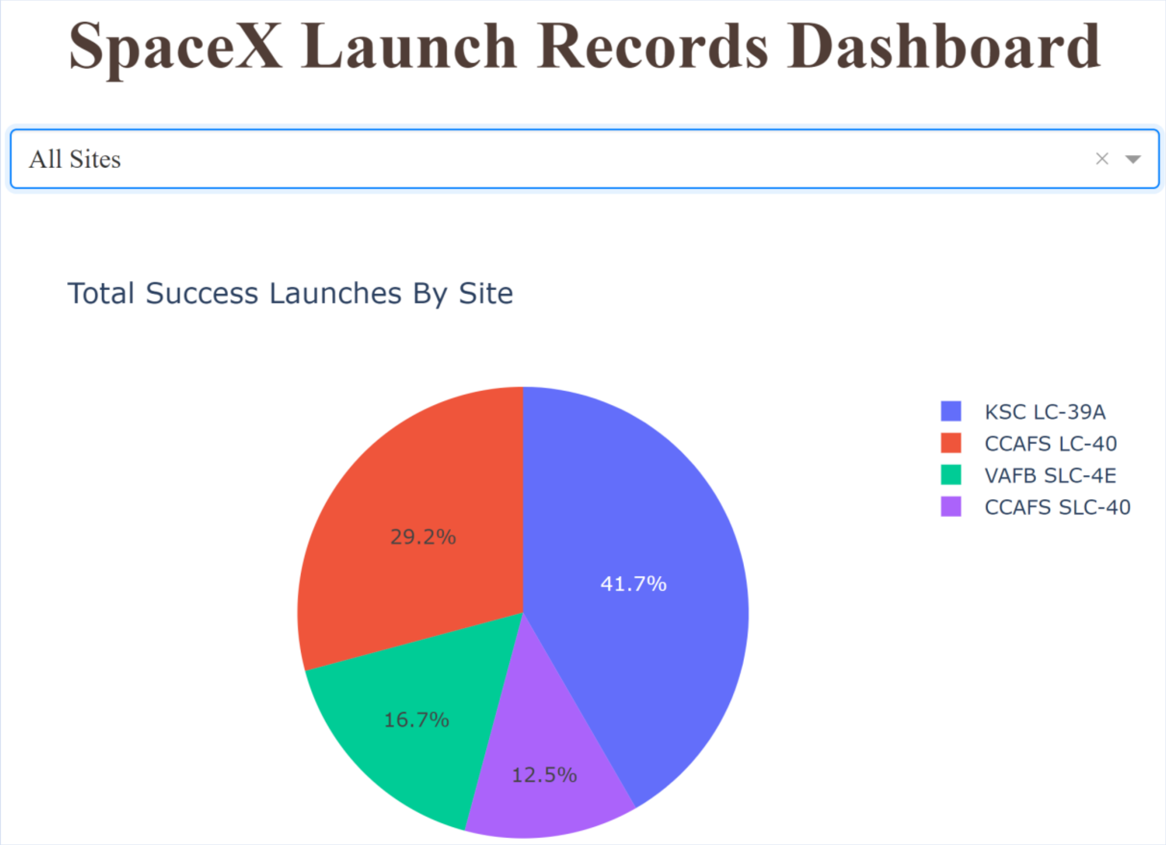
Distance to next railway: 0.69 km

Section 4

Build a Dashboard with Plotly Dash

Launch Success Count for all sites

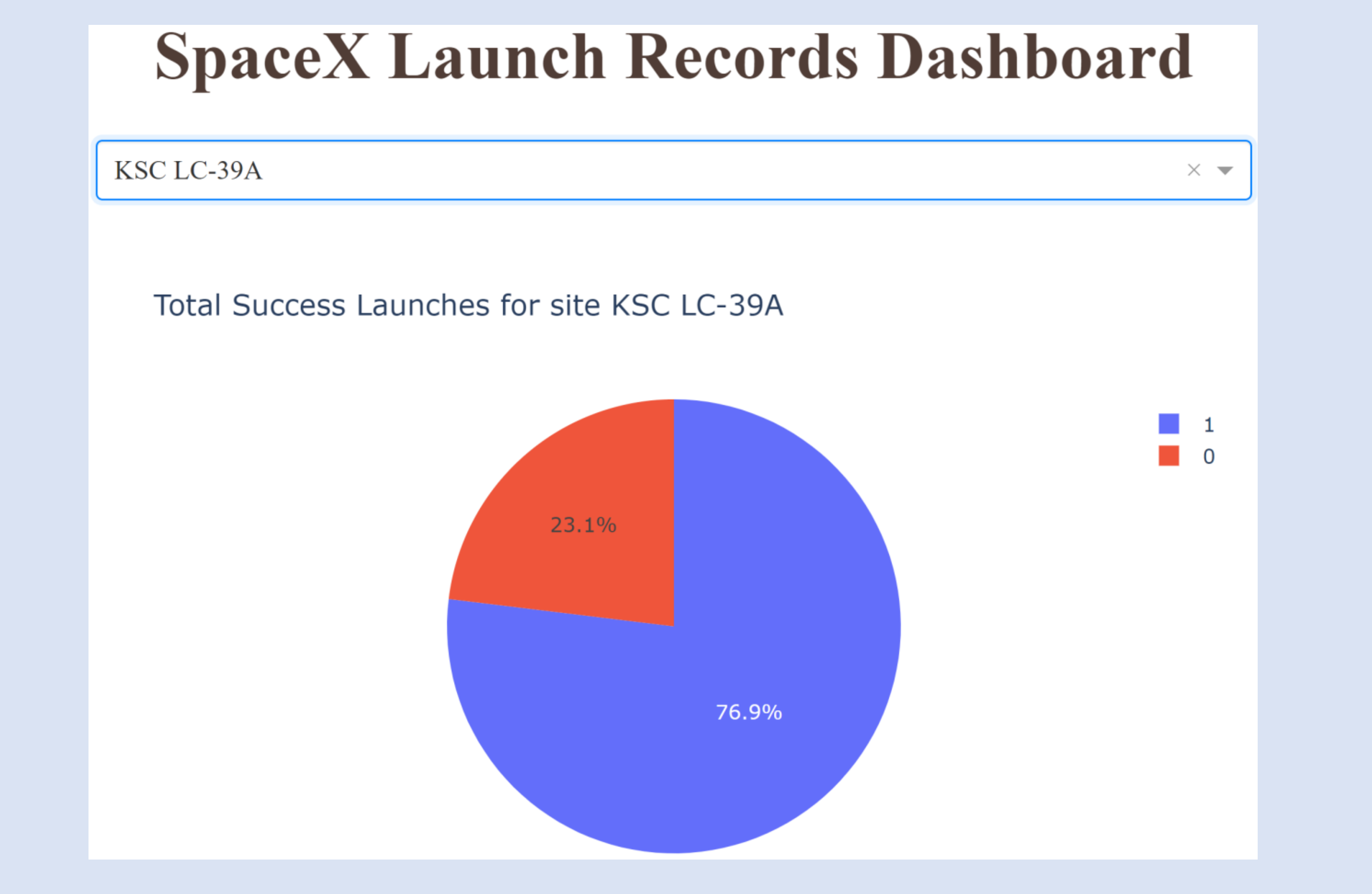
- Nearly half of all successful launches are coming from launch site KSC LC-39A
- The site with least success is CCAFS SLC-40



Launch Site with highest Launch Success

Class 1 (blue) = successful
Class 2 (red) = unsuccessful

- Site KSC LC-39A is the site with the highest percentage of successful launches



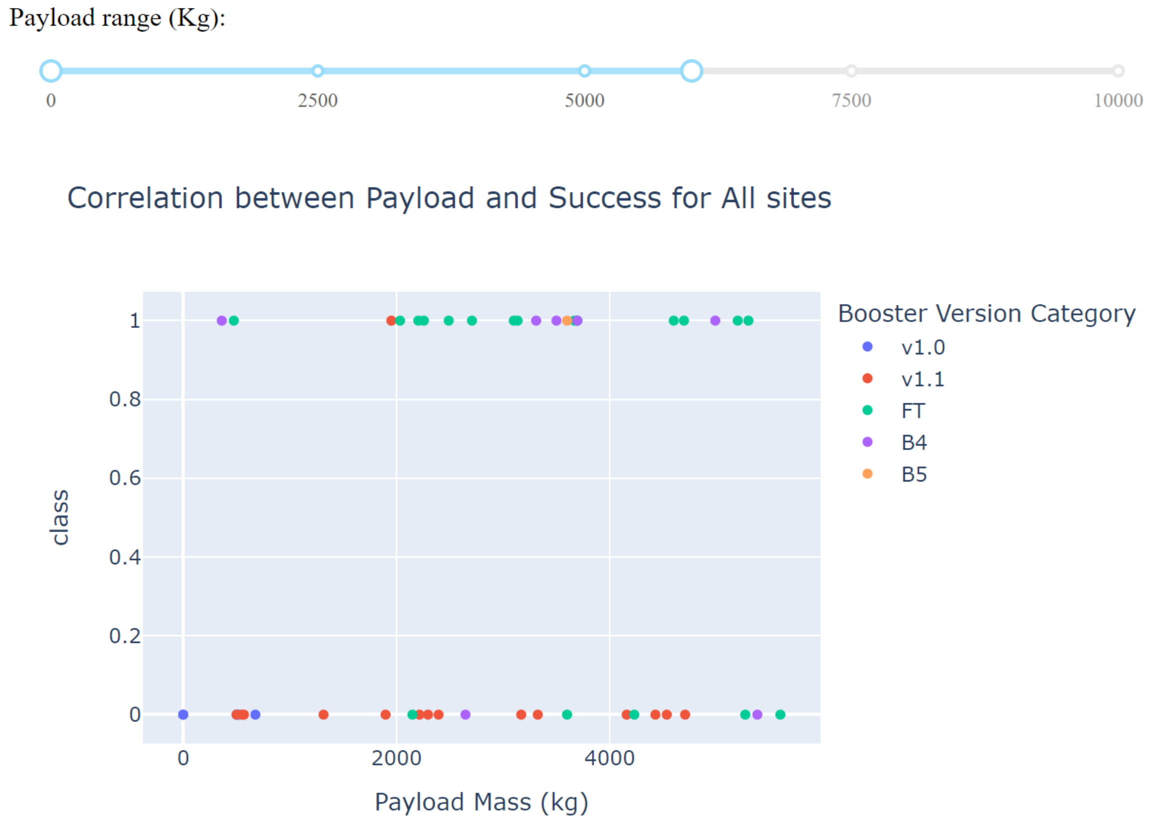
Launch Outcomes based on different Payloads

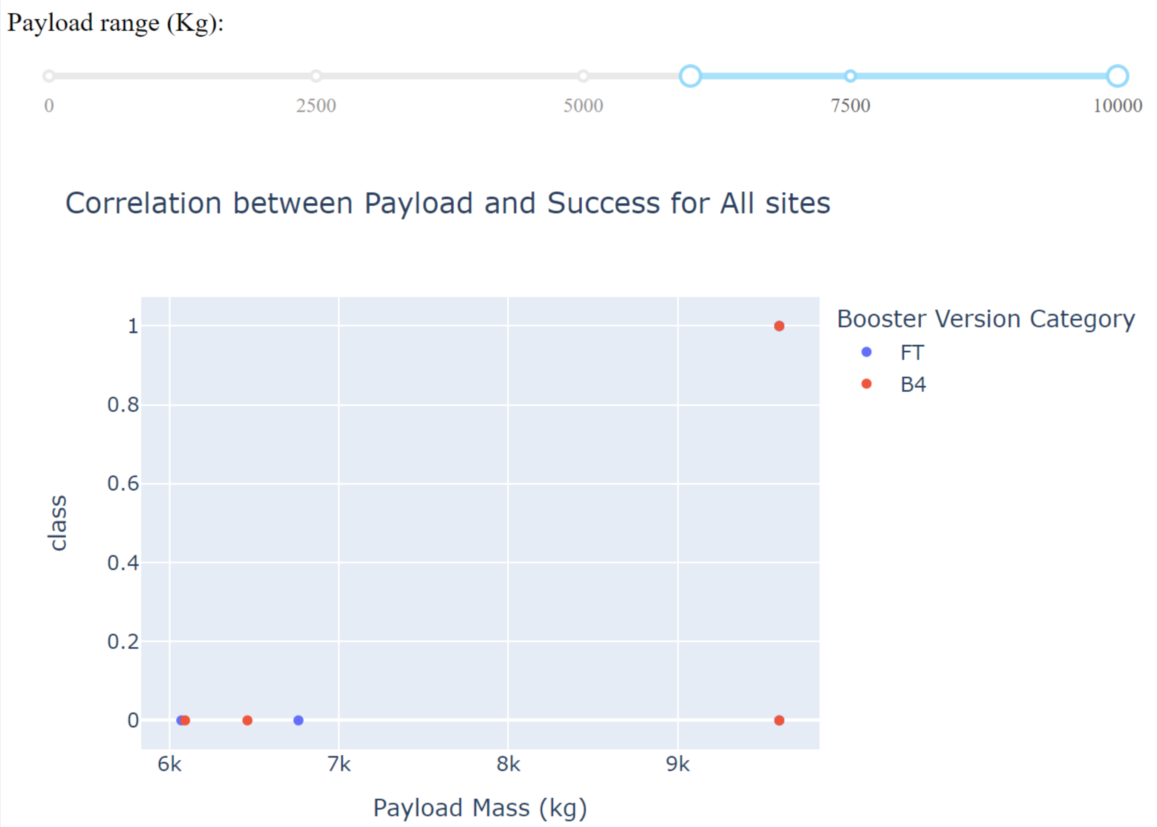
Low Payload range selected: **0 – 6.000** kg

- Booster Versions FT and B4 were more successful
- Booster Version v1.1 had the most failures

High Payload range selected: **6.000 – 10.000** kg

- Only Booster Versions FT and B4 were used
- Only 1 Launch outcome was successful



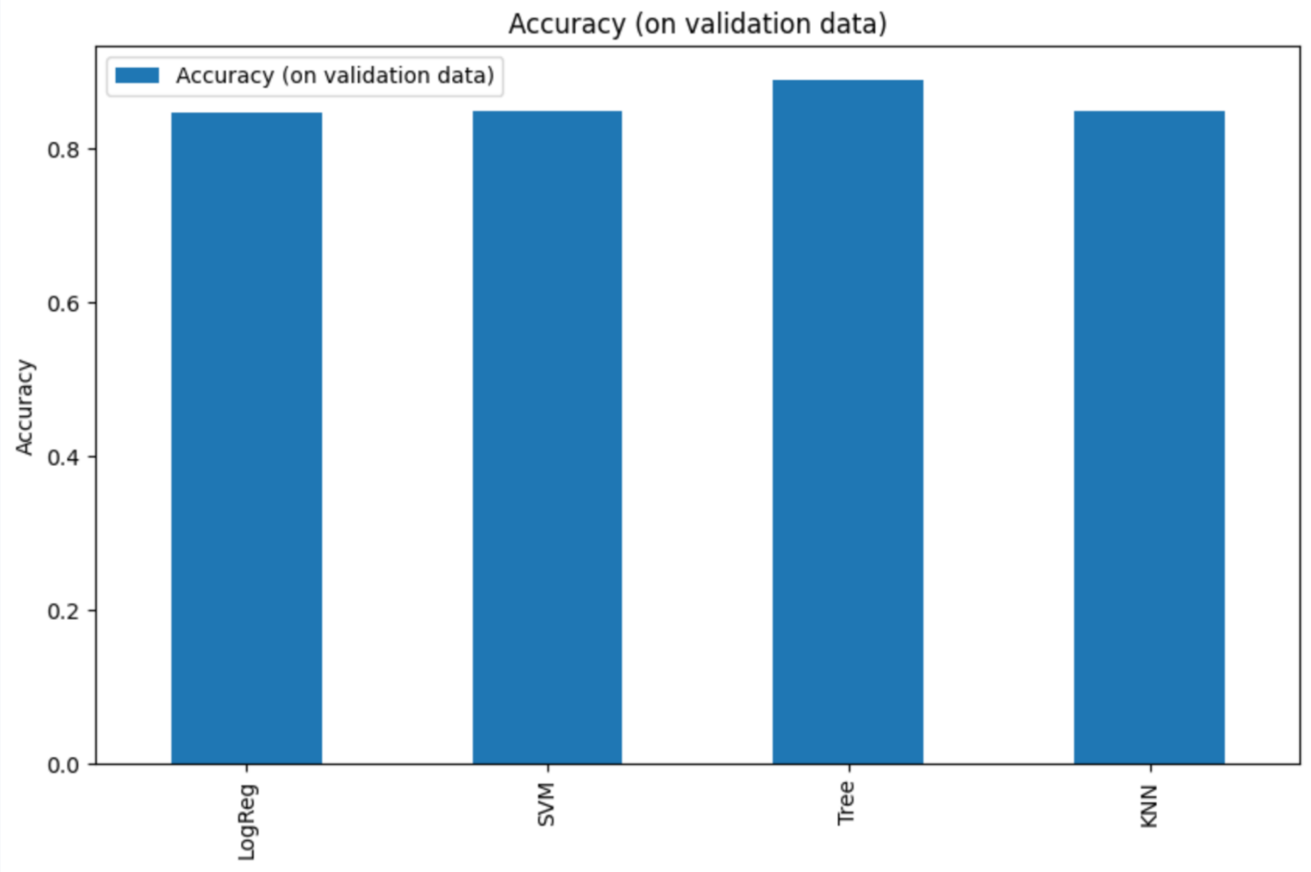


Class 0 = unsuccessful
Class 1 = successful

Section 5

Predictive Analysis (Classification)

Classification Accuracy



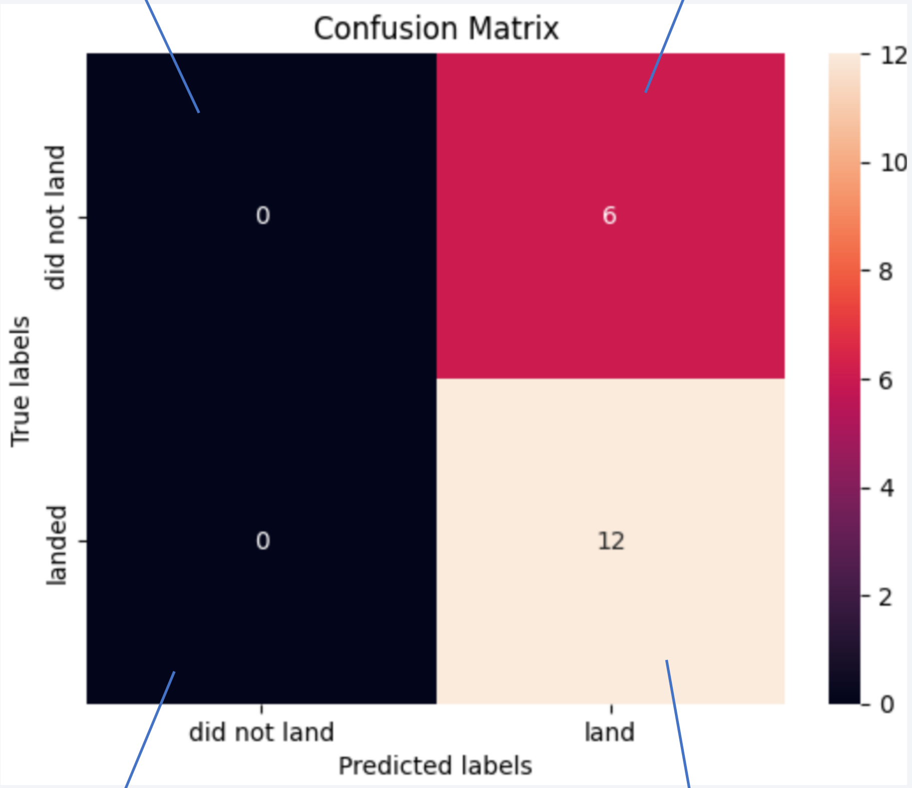
Decision Tree has the highest classification accuracy

	Accuracy (on validation data)
LogReg	0.846429
SVM	0.848214
Tree	0.889286
KNN	0.848214

Confusion Matrix

True negative

False positive



False negative

True positive

The Confusion Matrix shows the corrected and wrong predictions in comparison with the actual labels.

In this example for the **decision tree** model you can see that the landing was correctly predicted. However, there are still some false positive results for the landing.

Conclusions

- Higher flight numbers and launches with high payloads were more successful
- Launch sites are built far away from cities, but near to infrastructure like highways or railways
- The site CCAFS SC-40 had the highest number of launches, but a low number of successful outcomes
- The site KSC LC-39A is the site with the highest percentage of successful launches
- The orbit GTO has the lowest success rate. No correlation with payload can be found for this orbit.
- The orbit SSO was only used for low payloads, but has a success rate of 100%
- Booster versions FT and BF were used for all payload ranges. For low payloads they are more reliable than other booster versions, but for payloads over 6.000 kg only one launch was successful
- Launch success was zero from 2010 until 2013 and improved during most years after that
- Compared with the other models in this project the Decision Tree Model has the highest accuracy and is therefore the best learning algorithm in this case

Thank you!