IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Philip Rainsberger
January 22, 2023

Executive Summary

Summary of Methodologies

- Data collection
- Data wrangling
- Exploratory Data Analysis with Data Visualization
- Exploratory Data Analysis with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)

Summary of All Results

- ✓ Exploratory Data Analysis results
- ✓ Interactive Visualizations
- ✓ Predictive analysis results including best model

Table of Contents

Introduction

Project background and context

SpaceX is the most successful company of the commercial space age, making space travel affordable. The company 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. This information can then be used if an alternate company wants to bid against SpaceX for a rocket launch.

Questions we want to determine the answers to

- What factors have an impact on the successful landing of the first stage and to what degree?
- Is SpaceX seeing a better success rate over the years?
- What is the best machine learning algorithm to determine the success or failure in this case?

Methodology

- Data collection methodology
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform Data Wrangling
  - Filter data, removing data that is not needed, and cleansing missing values
  - Use One Hot Encoding to prepare the data for machine learning models
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Use scatter and bar graphs to show relationships between data
- Perform interactive visual analytics
  - Use Folium and Plotly Dash to create interactive visualizations to interact with, and analyze, data
- Perform predictive analysis using classification models
  - Build, tune, and evaluate classification models to determine the optimal model for prediction

Section 1

Methodology

Data Collection

The Data Collection phase included generating a request to retrieve data from the SpaceX REST API service as well as Web Scraping data from a table in SpaceX's Wikipedia web page.

Data Collected included:

- **SpaceX API**
  - FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude
- **SpaceX Wikipedia Web Page**
  - Flight No., Launch site, Payload, Payload mass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time

Data Collection – SpaceX API

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

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

Call SpaceX API to retrieve data

Normalize JSON data retrieved from API

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

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

Use custom functions to clean data

Create dictionary from prepared data

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass
```

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

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

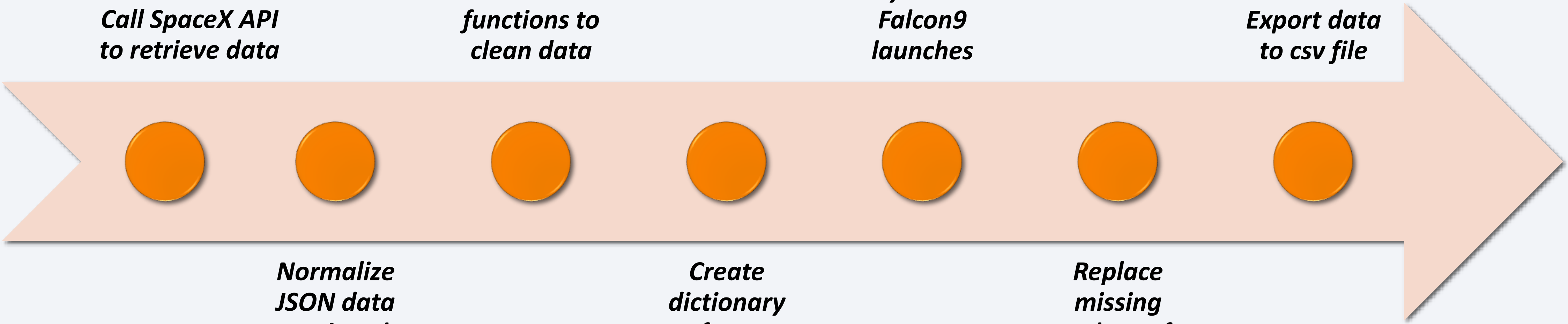
Filter data to only include Falcon9 launches

Replace missing values of Payload Mass column with average value for column

```
payloadmass_mean_value=data_falcon9['PayloadMass'].mean()
data_falcon9['PayloadMass'].fillna(value=payloadmass_mean_value, inplace=True)
```

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

Export data to csv file



GitHub URL: [SpaceX API Calls]

Data Collection – SpaceX API

Final data collected from SpaceX API:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561857
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561857
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829	34.632093
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561857
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89	86	2020-09-03	Falcon 9	15600.000000	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1060	-80.603956	28.608058
90	87	2020-10-06	Falcon 9	15600.000000	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	13	B1058	-80.603956	28.608058
91	88	2020-10-18	Falcon 9	15600.000000	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1051	-80.603956	28.608058
92	89	2020-10-24	Falcon 9	15600.000000	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	12	B1060	-80.577366	28.561857
93	90	2020-11-05	Falcon 9	3681.000000	MEO	CCSFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	8	B1062	-80.577366	28.561857

GitHub URL: [SpaceX API Calls]

Data Collection – Web Scraping

Get data from Falcon9 Wikipedia page

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

Create BeautifulSoup object to work with web page data

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

Find tables in web page

```
html_tables = soup.find_all("tr")
```

Using custom function, get column names from tables

```
column_names = []
for row in first_launch_table.find_all('th'):
    column_name = extract_column_from_header(row)
    print("Column_Name after extracting: ",column_name)

    if column_name is not None and len(column_name) > 0:
        column_names.append(column_name)
```

Create dictionary and append column names from keys

```
launch_dict= dict.fromkeys(column_names)
```

Create dataframe from dictionary

```
df1 = pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

Export data to csv file

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

GitHub URL: [SpaceX Web Scraping]

Data Collection – Web Scraping

Final data collected from SpaceX Web Scraping:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

GitHub URL: [SpaceX Web Scraping]

Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident:

- True Ocean means the mission outcome was successfully landed to a specific region of the ocean
- False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean
- True RTLS means the mission outcome was successfully landed to a ground pad
- False RTLS means the mission outcome was unsuccessfully landed to a ground pad
- True ASDS means the mission outcome was successfully landed on a drone ship
- False ASDS means the mission outcome was unsuccessfully landed on a drone ship.

In the Data Wrangling phase, we will convert the outcomes into Training Labels with `1` means the booster successfully landed `0` means it was unsuccessful. This will allow us to use the data with our machine learning algorithms and execute various prediction modelling routines.

Data Wrangling

Calculate the number of launches on each site

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

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

Calculate the number and occurrence of each orbit

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

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

Calculate the number and occurrence of mission outcome per orbit type

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

True ASDS	41
None None	19
True RTLS	14
False ASDS	6
True Ocean	5
False Ocean	2
None ASDS	2
False RTLS	1

Create a landing outcome label from Outcome column

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

Export data to csv file

```
df.to_csv('dataset_part_2.csv', index=False)
```

GitHub URL: [SpaceX Data Wrangling]

Data Wrangling

Final data collected from SpaceX Data Wrangling:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857	0
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857	1
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857	1
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857	0
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857	0

GitHub URL: [SpaceX Data Wrangling]

EDA with Data Visualization

We used a number of charts during our Exploratory Data Analysis phase to visualize the data. These charts included:

- **Scatter Plots**
  - A type of plot that displays values pertaining to typically two variables against each other. Usually it is a dependent variable to be plotted against an independent variable in order to determine if any correlation between the two variables exists.
  - Scatter plots used in our analysis included: Flight Number vs. Payload, Flight Number vs. Launch Site, Payload vs. Launch Site, Flight Number vs. Orbit Type, Payload vs. Orbit Type
- **Bar Graphs**
  - A type of plot where the length of each bar is proportional to the value of the item that it represents. It is commonly used to compare the values of a variable at a given point in time.
  - Bar Graphs used in our analysis included: Success Rate vs. Orbit Type
- **Line Charts**
  - A type of plot that show trends in data over time (time series).
  - Line Charts used in our analysis included: Launch Success Yearly Trend

GitHub URL: [SpaceX EDA Visualization]

EDA with SQL

A database is a repository of data - a program that stores data – and provides the functionality for adding, modifying, and querying that data. SQL is a language used for relational databases to query, add, delete, or update data in a database. During our analysis, we imported data into a Db2 database and used SQL to perform the following queries and analysis:

- 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 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 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
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for 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: [SpaceX EDA with SQL]

Build an Interactive Map with Folium

Folium is a powerful data visualization library in Python that was built primarily to help people visualize geospatial data. With Folium, you can create a map of any location in the world as long as you know its latitude and longitude values. You can also create a map and superimpose markers as well as clusters of markers on top of the map and create maps of different styles such as street level and stamen maps. For our analysis, we included the following map items:

- Add markers with Circle, Popup Label and Text Labels of all launch sites on a map to determine their proximity to the equator and coastlines
- Add colored markers to display the success (green) and failed (red) launches for each site on the map using a marker cluster. This will be used to gauge the success/fail rates for each launch site.
- Add colored distance indicator lines to display distances (in kilometers) between the launch Sites and their proximities to railways, Highways, coastlines, and closest cities.

GitHub URL: [SpaceX Data Visualization with Folium]

Build a Dashboard with Plotly Dash

Plotly is an interactive, open source plotting library that supports over 40 unique chart types. Dash is a Open-Source User Interface Python library for creating reactive, web-based applications. During our analysis we used Plotly Dash to create an application that included the following:

- A dropdown list to enable Launch Site selection to narrow results (default of "All Sites" launch sites)
- Pie Chart showing successful launches for "All Sites" and the Success vs. Failed counts for an individual site if selected from the drop down list
- Added a slider to set a range for the Payload Mass which allowed for further data inspection based on various Payload Mass values for all or specific sites
- Included a Scatter Chart of Payload Mass vs. Success Rate for the different Booster Versions which displayed the correlation between Payload Mass and Launch Success.

GitHub URL: [SpaceX Dashboard with Plotly Dash]

Predictive Analysis (Classification)

Load data and create Numpy Array with “Class” data

Standardize, fit, and transform the data

Split the data in to Train and Test datasets

Create GridsearchCV object to find optimal hyperparameters for each method

Execute each method and determine accuracy, review confusion matrix

Determine which method performs the best at predictions

Using the data we have collected and analyzed, we will use different methods on the data to predict which method is most accurate at predicting if the first stage will land successfully. Methods we used in our analysis include:

- Logistic Regression
- Support Vector Machine
- Decision Tree
- K Nearest Neighbor

After each method is created, we review the accuracy and confusion matrix. Finally we create a horizontal bar chart to display the Accuracy vs. Prediction Method to visually determine which model is most accurate.

GitHub URL: [SpaceX Predictive Analysis]

Results

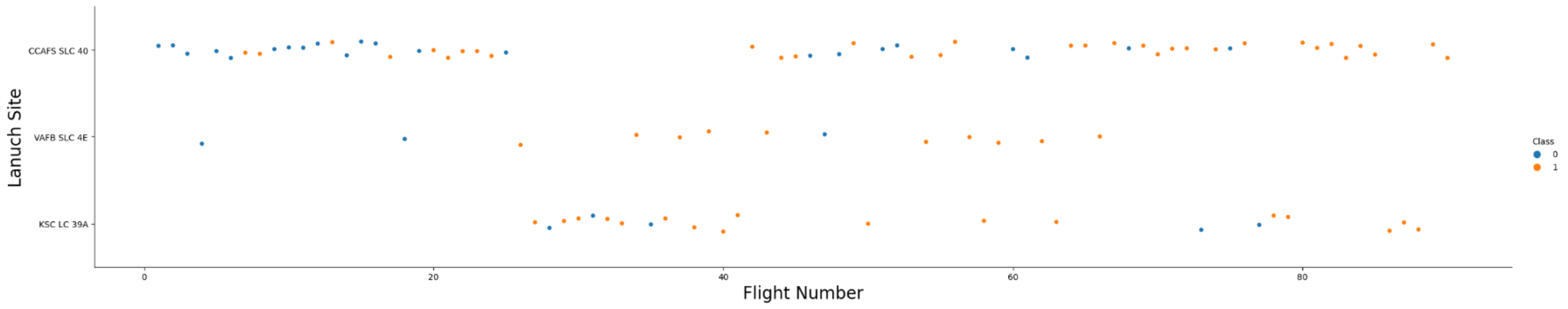
In the following sections we will provide detail for the following:

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

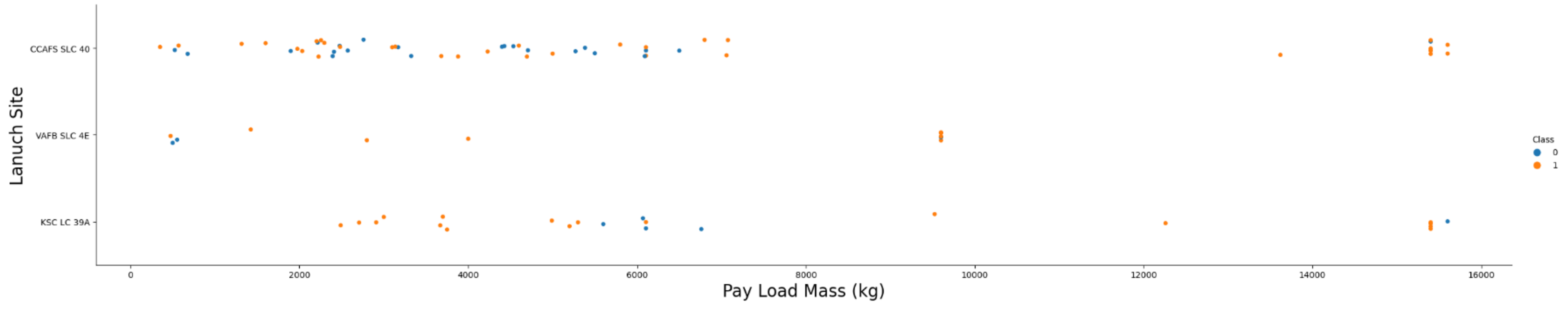
Flight Number vs. Launch Site



Explanation

- Early flights had a higher failure rate; as the number of flights increased, so did the success rate
- While the CCAFS SLC 40 launch site has the majority of launches, VAFB SLC 4E and KSC LC 39A have higher success rates
- As SpaceX continues with more launches, we can expect an increase in the success rate

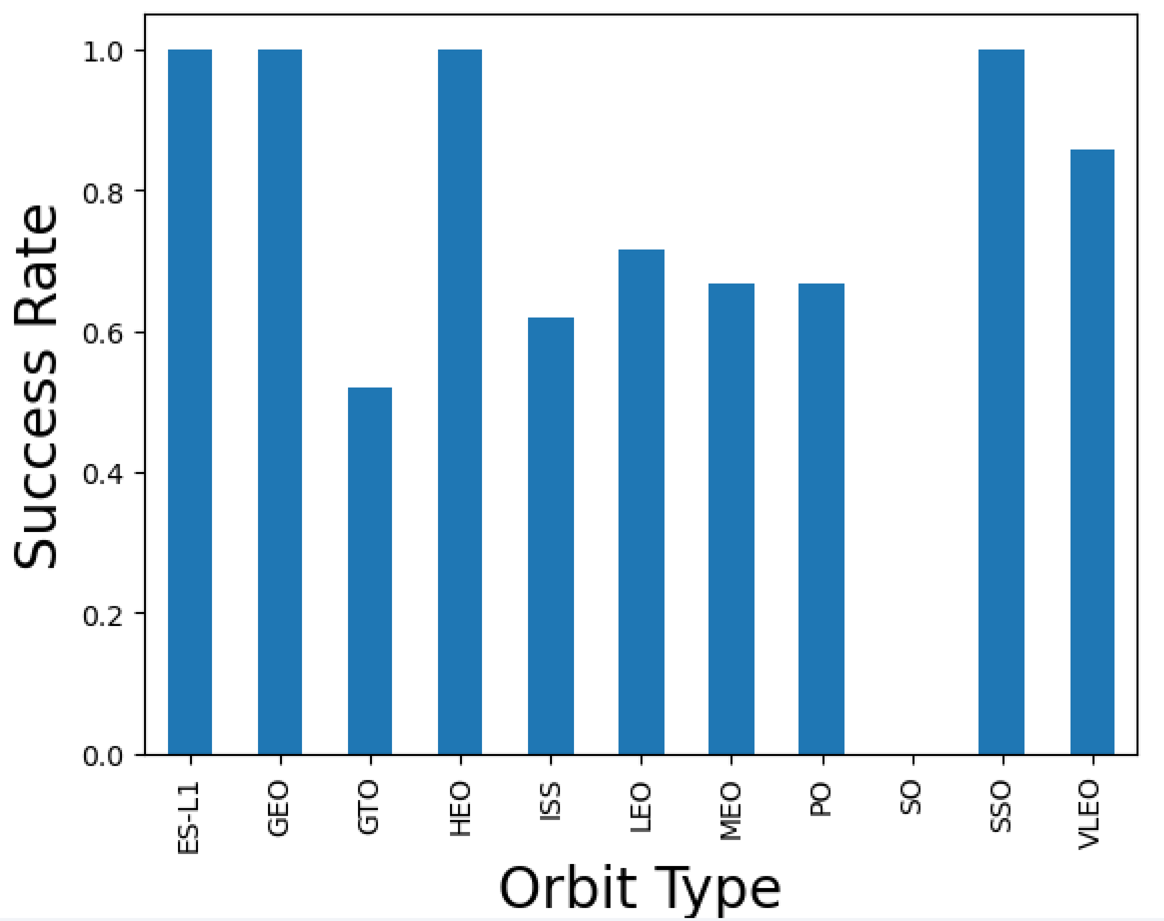
Payload vs. Launch Site



Explanation

- For each launch site, increased Payload Mass correlates to a higher success rate
- Majority of the launches with a Payload Mass over 7000kg were successful
- KSC LC 39A has a high success rate for smaller Payload Mass launches (i.e. under 5500kg)

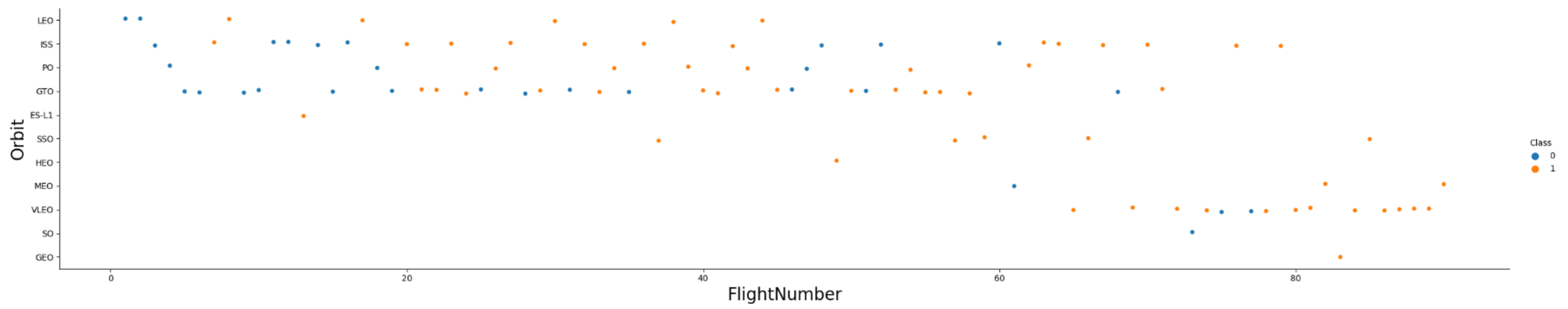
Success Rate vs. Orbit Type



Explanation

- ES-L1, GEO, HEO, and SSO orbit types had 100% success rate
- SO orbit type has 0% success rate
- Other orbit types had a success rate between 50%-70%

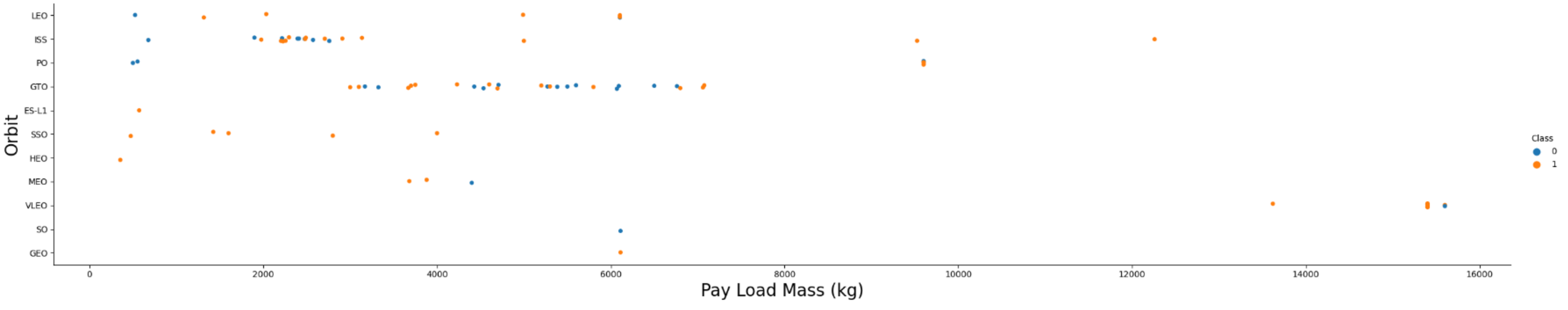
Flight Number vs. Orbit Type



Explanation

- The LEO orbit success rate increases with the number of flights
- There does not appear to be a correlation between GTO orbit and number of flights
- Additional flight data is needed for HEO, SO, and GEO as there is not data for flights 1 to ~50

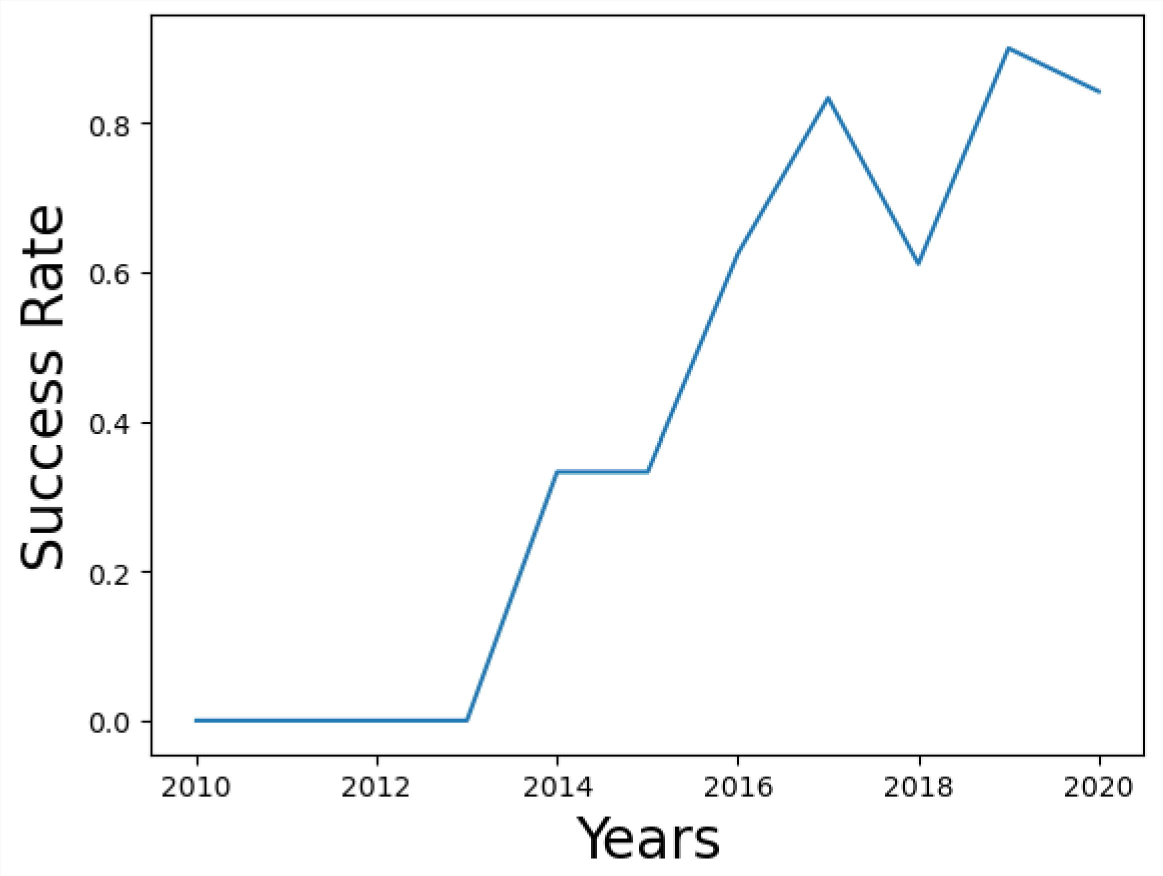
Payload vs. Orbit Type



Explanation

- Heavier payloads have a negative impact on success for GTO orbits
- Heavier payloads have a positive impact on success for ISS orbits

Launch Success Yearly Trend



Explanation

- Since 2013, the success rate has been increasing
- In 2018, we observe a 20% decline in success rate and another smaller decline in 2020, however, the success rate has generally been increasing since 2013

All Launch Site Names

```
%sql select unique(launch_site) from spacextbl
```

```
 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4859-88f0-a
Done.
```

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

Explanation

- SQL for querying a distinct list of Launch Site names
- Can use either unique() or the "distinct" keyword in the SQL Query: "select distinct launch_site from spacextbl"

Launch Site Names Begin with 'CCA'

```
%sql select * from spacextbl where launch_site like 'CCA%' limit 5
```

```
 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

Explanation

- Select all data for launch sites that begin with ‘CCA’
- Use “%” wildcard to select data (i.e. ‘CCA%’)
- Limit result set to 5 records (“limit 5”)

Total Payload Mass

```
%sql select sum(payload_mass__kg_) total_payload_mass from spacextbl
```

```
 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41
Done.
```

total_payload_mass
619967

Explanation

- Select the total payload mass across all flights for SpaceX

Average Payload Mass by F9 v1.1

```
%sql select avg(payload_mass__kg_) average_payload_mass \
from spacextbl \
where booster_version like 'F9 v1.1%'
```

```
 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4859-88f0-a6c3e6b4
Done.
```

average_payload_mass
2534

Explanation

- For Booster Version “F9 v1.1”, select the average payload mass for these flights

First Successful Ground Landing Date

```
%sql select min(date) \
from spacextbl \
where lower(landing__outcome) like '%ground pad%'

 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4859-88f
Done.
```

1
2015-12-22

Explanation

- Use the “min” function to return the oldest date in the result set that was limited to “ground pad” landing outcomes

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select booster_version \
from spacextbl \
where lower(landing__outcome) like '%success%drone ship%' \
and payload_mass__kg_ between 4000 and 6000
```

```
 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b
Done.
```

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

Explanation

- List the Booster Version that has a payload mass between 4000kg and 6000kg
- Limit the result set to landing outcomes that have success in drone ship

Total Number of Successful and Failure Mission Outcomes

```
%sql select mission_outcome, count(*) \
from spacextbl \
group by mission_outcome
```

```
 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4
Done.
```

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Explanation

- List the count of each different type of outcome listed in our data set

Boosters Carried Maximum Payload

```
%sql select unique(booster_version) \
from spacextbl \
where payload_mass__kg_ in (select max(payload_mass__kg_) \
                              from spacextbl)
```

```
 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b9
Done.
```

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

Explanation

- List the booster version that have carried the heaviest payloads
- Use a subquery to find the value for the heaviest payload in our data

2015 Launch Records

```
%sql select landing__outcome, booster_version, launch_site \
from spacextbl \
where lower(landing__outcome) like '%fail%drone ship%' \
and to_char(date, 'YYYY') = '2015'
```

```
 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b9(
Done.
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Explanation

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Use the “to_char” function to convert the date year to a string for comparison

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

```
%sql select landing__outcome, count(*) count \
from spacextbl \
where date between to_date('2010-06-04', 'YYYY-MM-DD') and to_date('2017-03-20', 'YYYY-MM-DD') \
group by landing__outcome \
order by count(*) desc
```

```
 * ibm_db_sa://vrn61236:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.a
Done.
```

]:

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

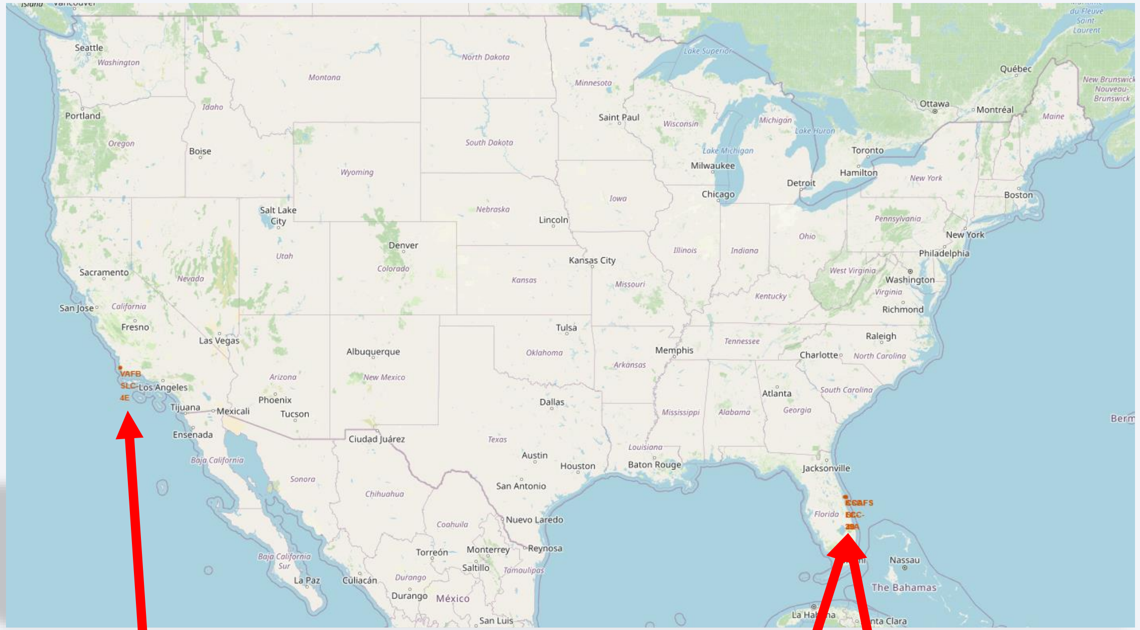
Explanation

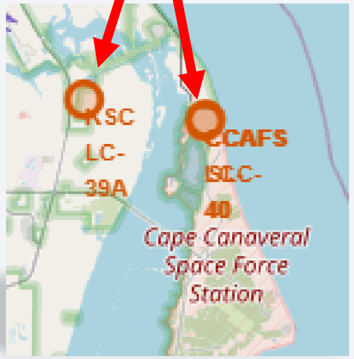
- Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order
- Use the “to_date” function to convert the date strings to dates for comparison operators
- Sort the list in descending order based on the count of each landing outcome

Section 3

Launch Sites Proximities Analysis

Interactive Map – All Launch Sites

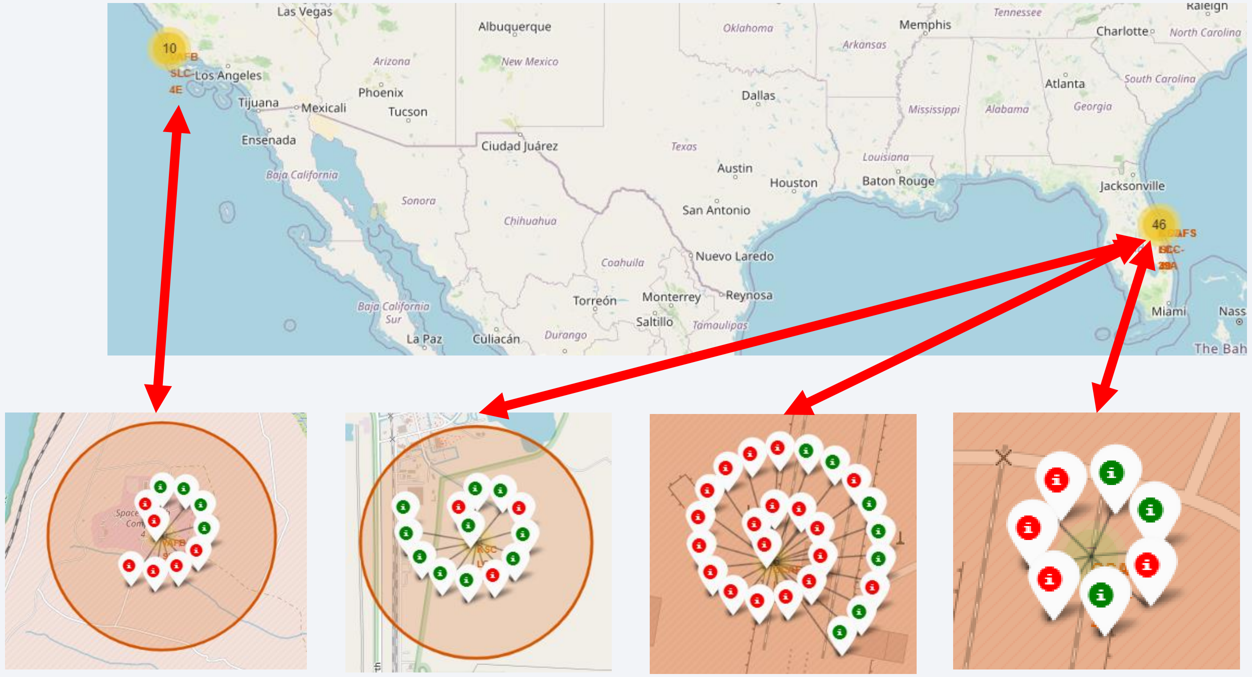




Using Folium, we are able to place a marker on the map, using the launch sites' latitude and longitude coordinates, which depicts the location for each launch sites.

We see that the launch sites are all near the oceanic coasts which provides for less risk and exposure should there be a failure. Additionally, all sites are about equidistant from the Earth's equator.

Interactive Map – Launch Outcomes



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

The green markers ( )on the map show successful launches while red markers( ) shows failed launches.

By visually inspecting the ratio of red-to-green for each site, we can determine that KSC LC-39A has the best probability for a successful launch.

Interactive Map – Launch Site Proximities to Landmarks

Using the KSC LC-39A launch site as an example, we can locate the distance to various landmarks:

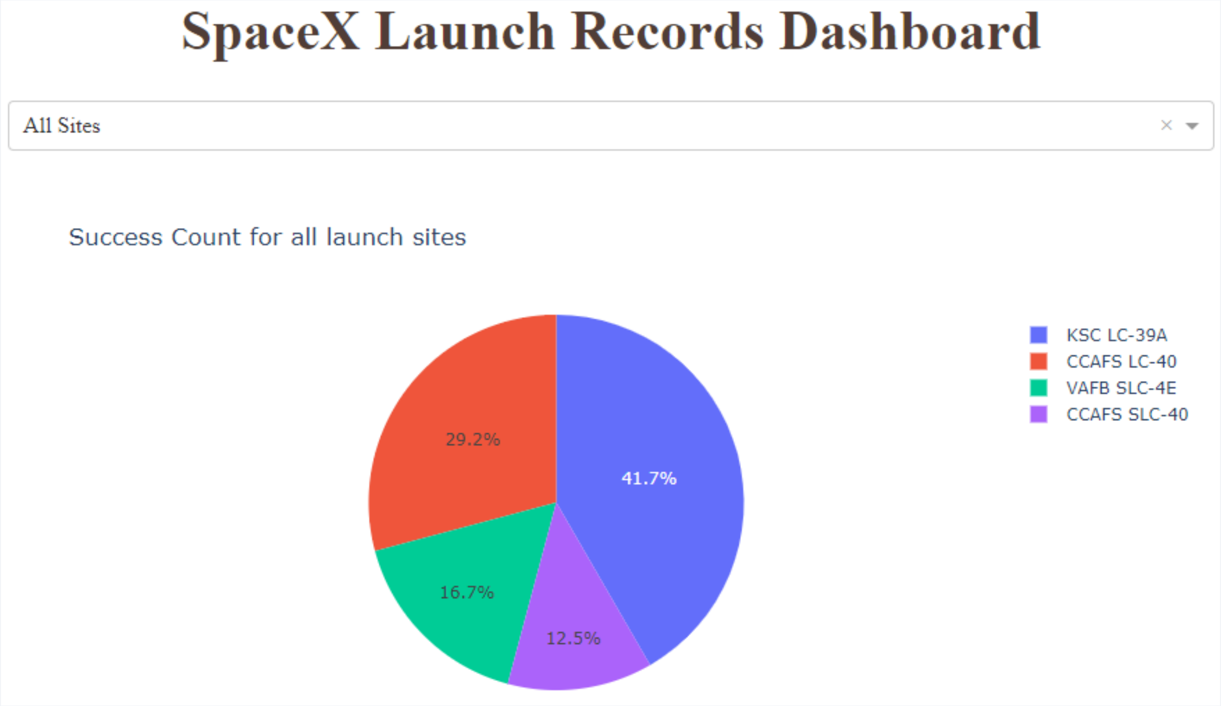
- Closest City: 16.32km (Red)
- Closest Coastline: 14.99km (Orange)
- Closest Railway: 15.23km (Blue)
- Closest Highway: 20.28 (Green)

Distance proximities to various landmarks are an important factor to consider when select a launch site as there could be catastrophic consequences with a failed landing. Ensuring an appropriate distance from the landmarks ensures safety.

Section 4

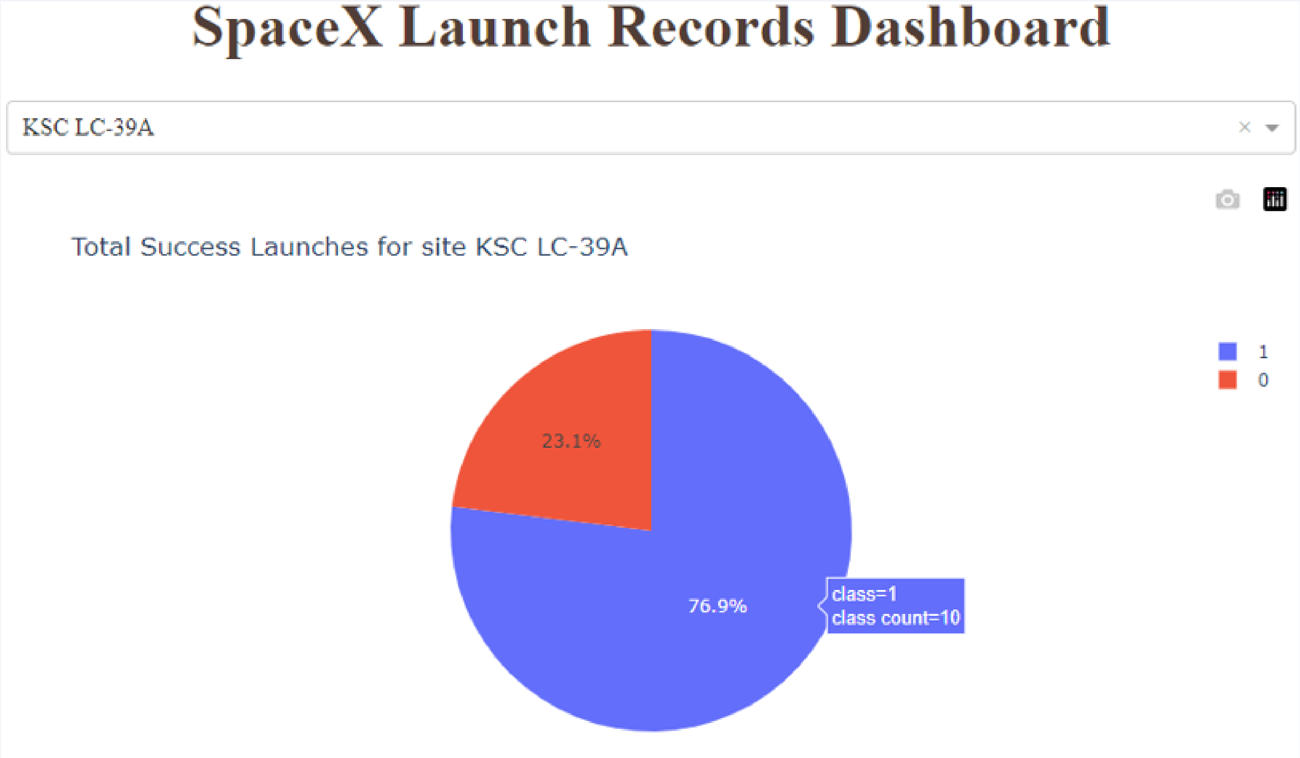
Build a Dashboard with Plotly Dash

Dash App - Launch Successes for All Sites



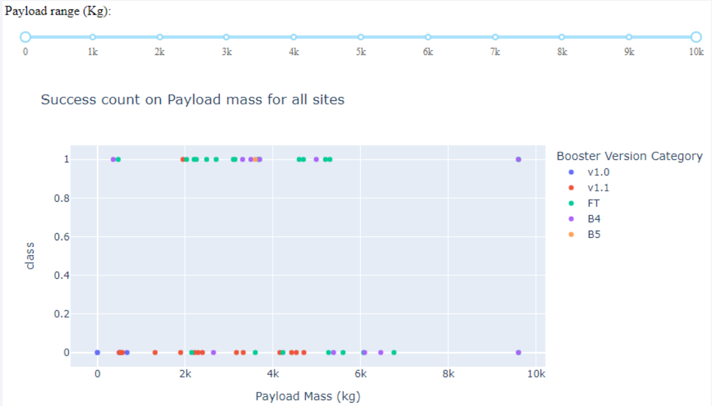
From the rendered pie chart, we can clearly see that launch site KSC LC-39A has the highest success rate at 41.7% with CCAFS SLC-40 having the least success with only 12.5%.

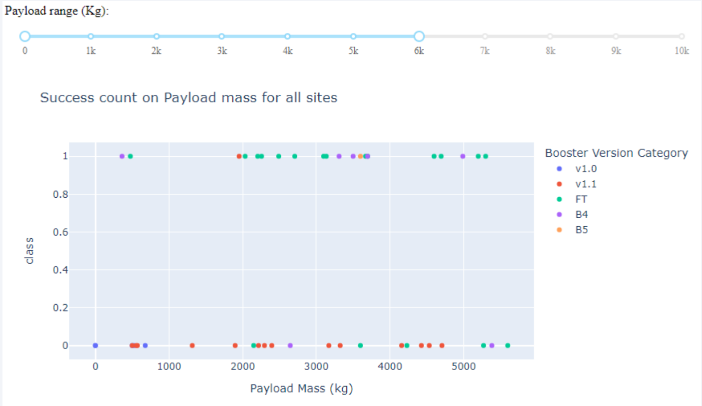
Dash App – Launch Site with Highest Success Rate

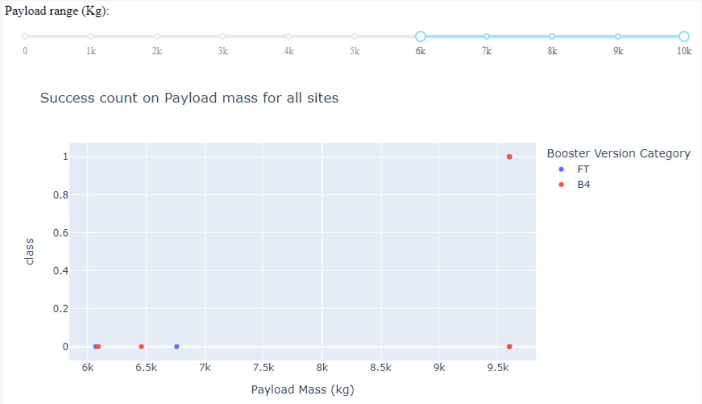


For the launch site with the highest success rate, KSC LC-39A, we see that 76.9% of the launches are successful while only 23.1% are unsuccessful. Additionally, by hovering the mouse over each section in the pie chart, we can see that there were 10 successful launches and 3 failures.

Dash App – Payload vs. Launch Outcome







- Using the slider created for the scatter plots, we can conclude that the higher the payload mass (> 6000kg) the greater chance of failure, however, only FT and B4 booster versions have data supporting this conclusion.

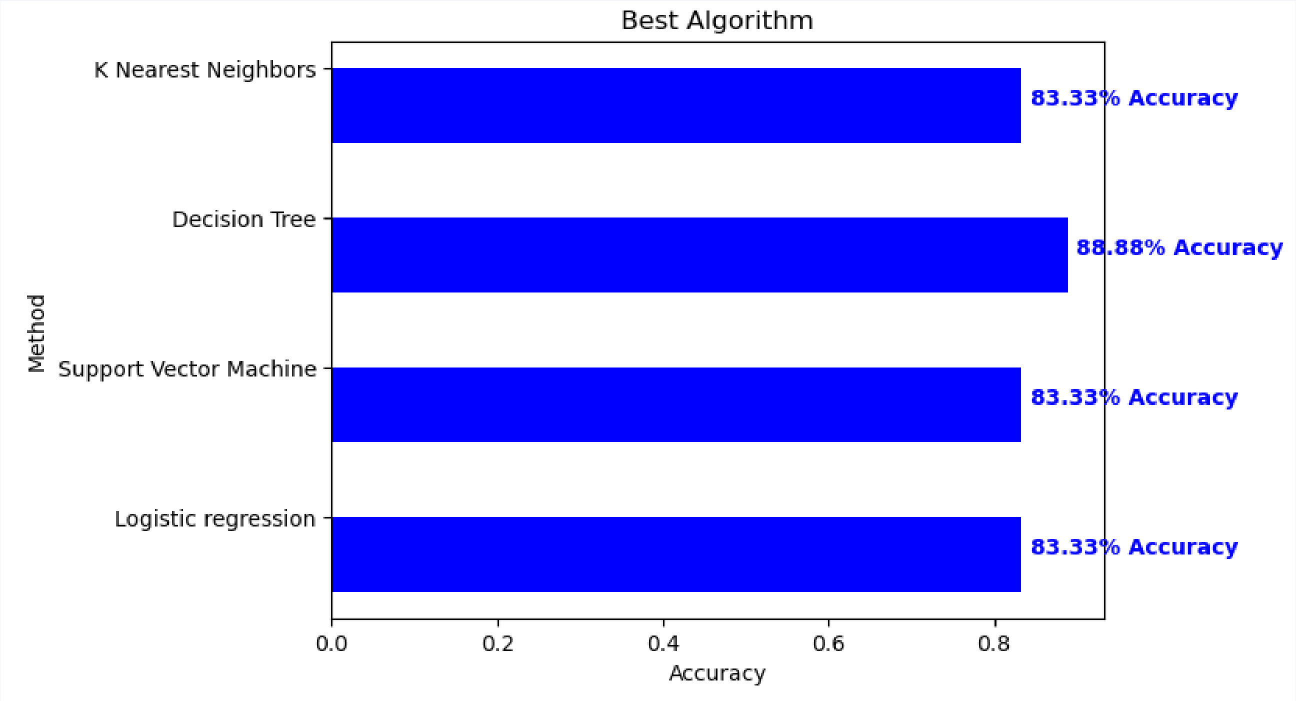
- We can also see that booster version v1.1 has a very low success rate having only been tested with payloads up to 4707kg

- FT booster version appears to have the highest success rate carrying successful payloads from 475kg up to 5300kg.

Section 5

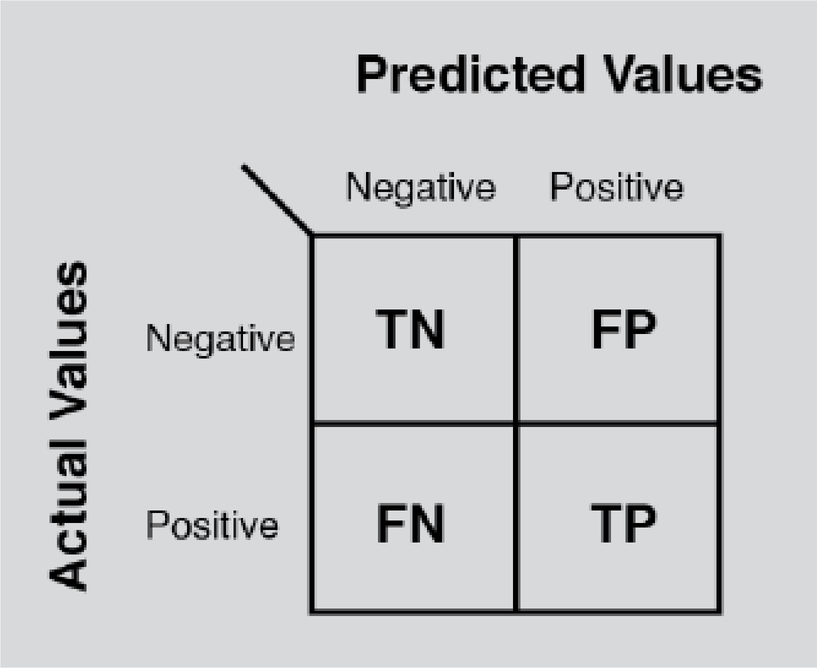
Predictive Analysis (Classification)

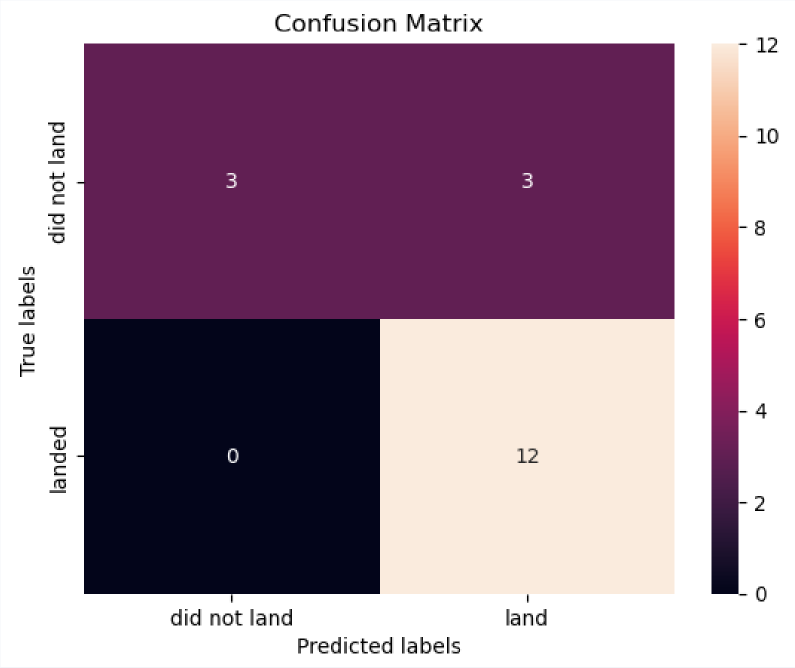
Classification Accuracy



Based on our training and testing analysis of for classification methods, a Decision Tree model is the most accurate model in predicting a successful or failed launch using our test data. We achieved an accuracy of 88.88% which is approximately 5.5% higher than the other methods tested.

Confusion Matrix





For all four methods used, we received the same confusion matrix which does not assist us in drawing any conclusions on the accuracy of the various models. Although the majority were predicted as "True Positives" and "True Negatives", and there were no "False Negatives", we do have three "False Positives" that were predicted incorrectly.

Conclusions Drawn from Analysis

- There is a positive correlation between number of flights and success rate. As with most exercises, the more practice (i.e. more launces) you perform, the better ratio of your successful outcomes.
- Launches with a low payload mass were often more successful than launches with a larger payload mass.
- KSC LC-39A has the highest success rate of the launches from all the sites. Additional analysis should be performed to determine the reasons for their success and model future launch sites after KSC LC-39A.
- Orbit types ES-L1, GEO, HEO, and SSO have a 100% success rates.
- Decision Tree model was the optimal model of the four tested to predict success and failed launches.

Appendix

All code and assets can be found in my [GitHub repository]() for reference.

I want to thank Coursera, IBM, and all of the instructors who worked to make this class possible. I will highly recommend to my peers as a solid foundation to learning Python and data science.

Thank you!