IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Anthonette Dianne Perez>
<2022/6/3>

Outline

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

Executive Summary

- Summary of methodologies
    - This project uses data science methods to predict the first stage landing success rate of the SpaceX Falcon 9 rocket using data collecting, data wrangling, exploratory data analysis, interactive visual analytics, and predictive analysis..
- Summary of all results
    - Out of the 90 landings of Falcon 9 rocket between the year of 2010 and 2020, the overall success rate is ≈ 66.67%
    - We observed that the success rates started to increase since 2013.
    - There are various relationships between the features including the flight number, launch site, payload, orbit, etc.
    - By appling machine learning method, we built classification models for the prediction and all classifiers have the same prediction accuracy of ≈ 83.33%

Introduction

- Project background and context
  - Space X offers Falcon 9 rocket launches at 62 million dollars on their website; other suppliers charge upwards of 165 million dollars apiece, with much of the savings due to Space X's ability to reuse the first stage. As a result, if we can figure out if the first stage will land successfully, we can figure out how much a launch will cost. If another business wishes to compete with Space X for a rocket launch, this information can be used..

- Problems you want to find answers
  - Whether or not the Falcon 9 first stage will land successfully?

Section 1

Methodology

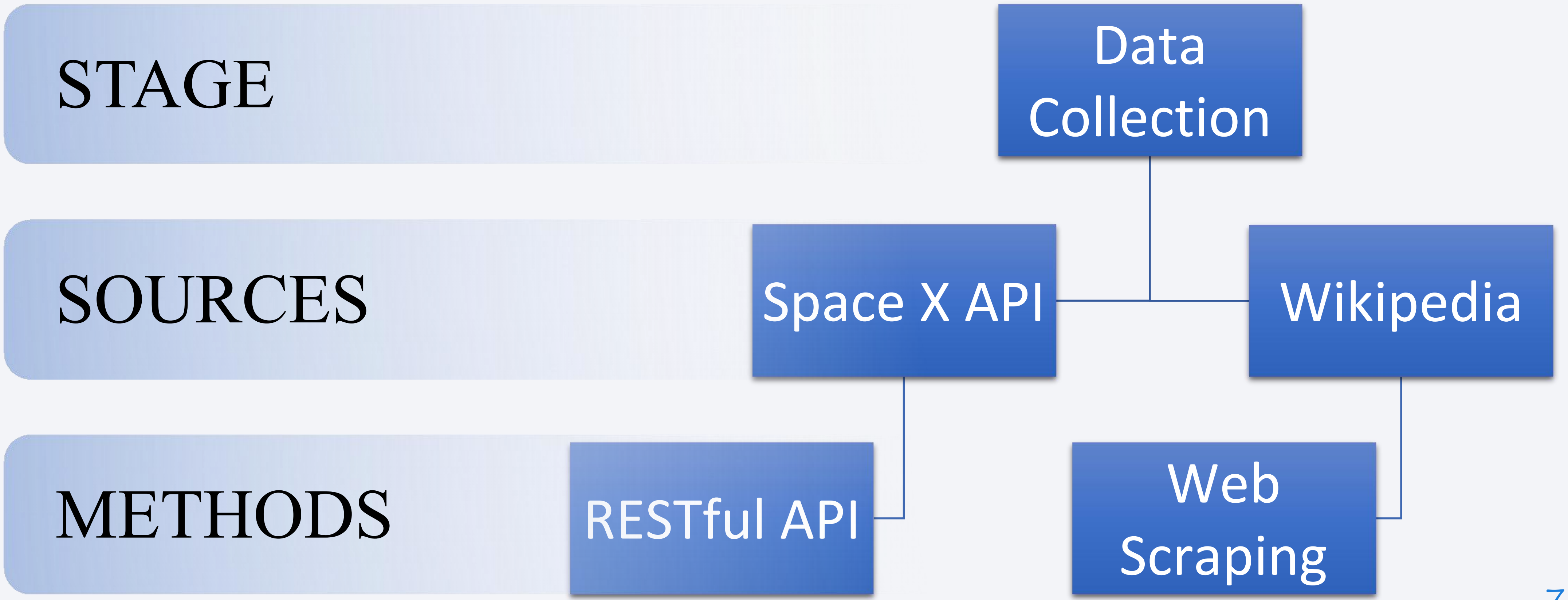
Methodology

Executive Summary

- Data collection methodology:
  - RESTful API and Web Scraping
- Perform data wrangling
  - Exploratory Data Analysis and Training Labels.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build, tune and evaluate classification models using Scikit-learn involving Train-Test Split and Grid Search.

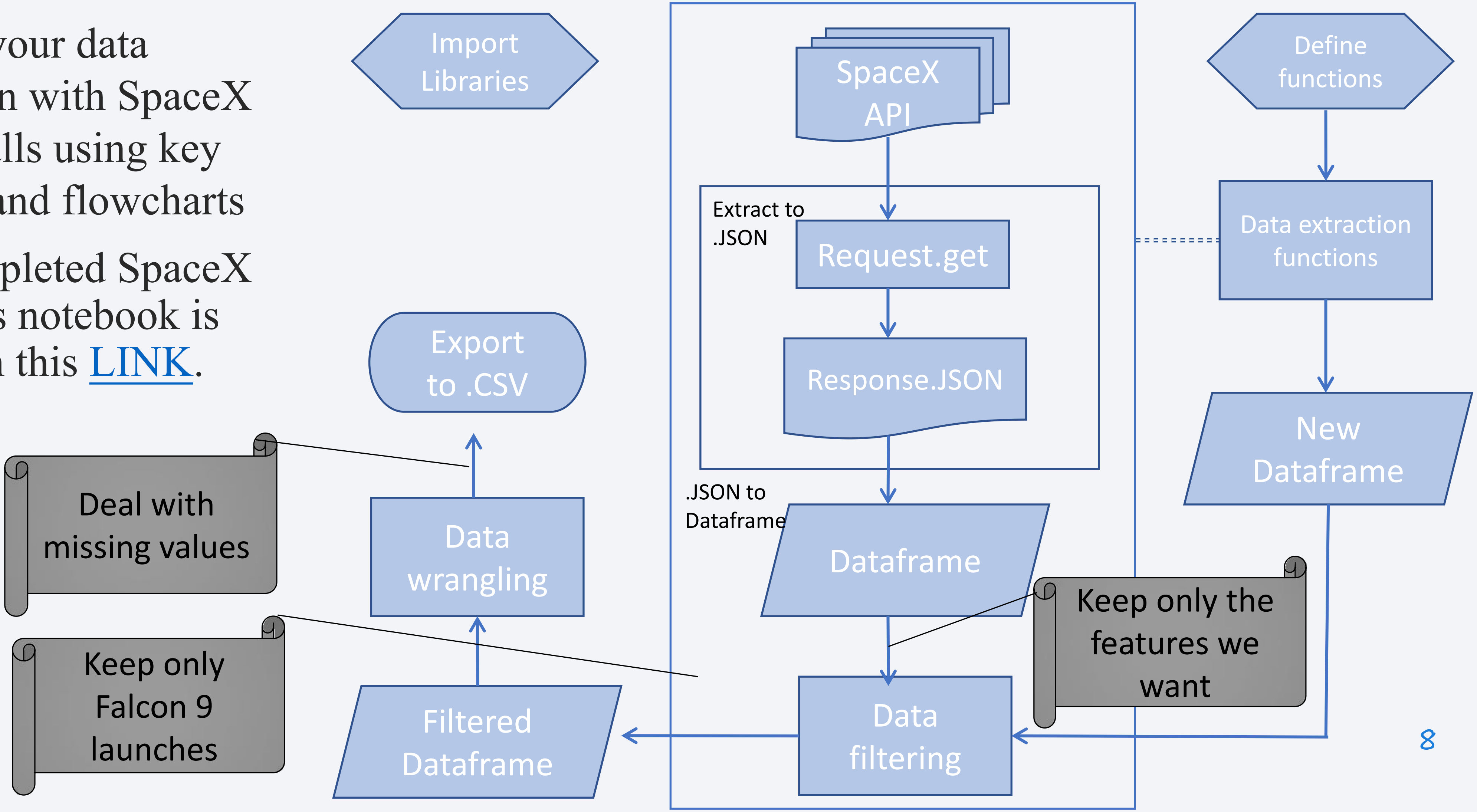
Data Collection

- Data sets were collected from two sources:



Data Collection – SpaceX API

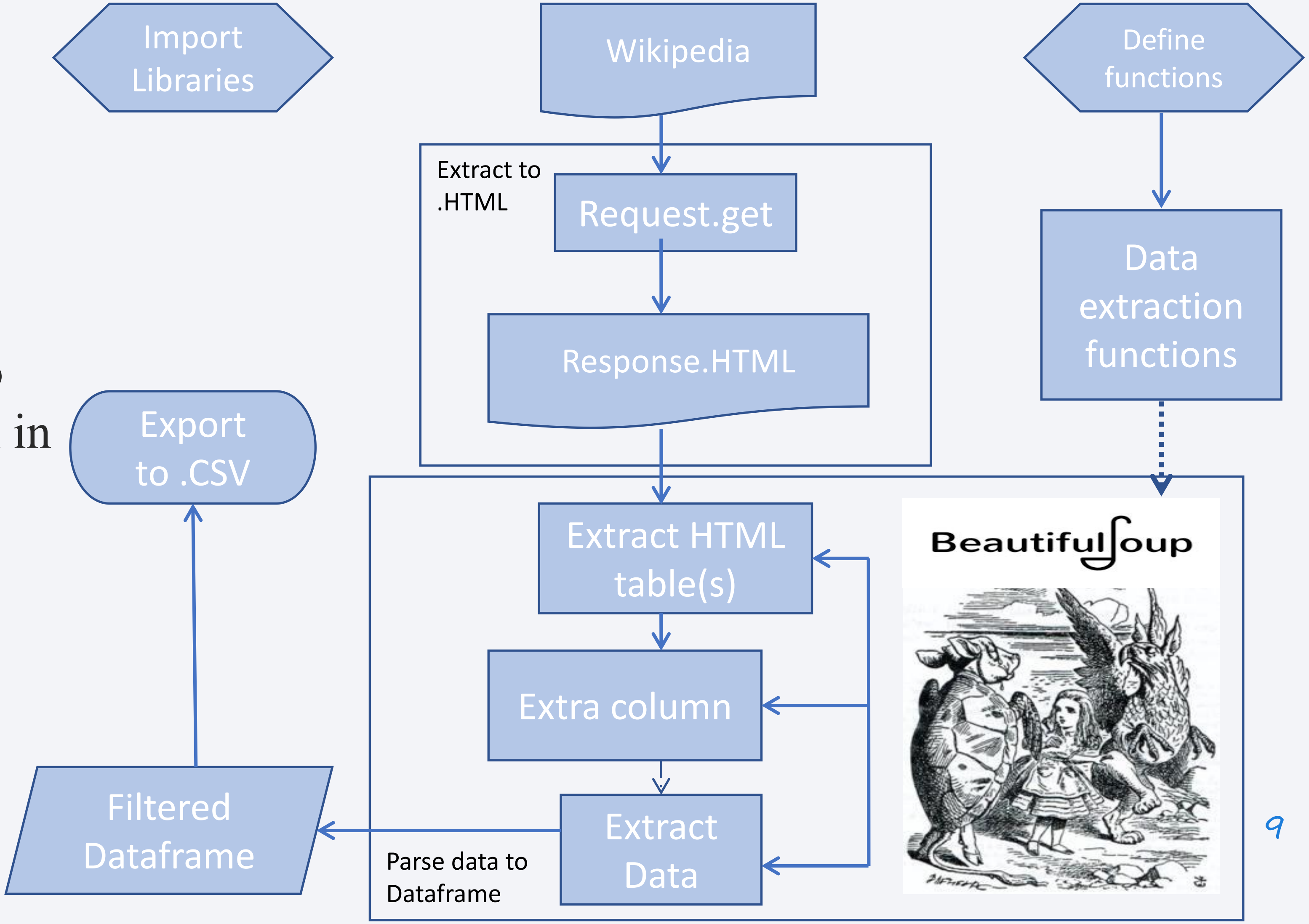
- Present your data collection with SpaceX REST calls using key phrases and flowcharts
- The completed SpaceX API calls notebook is shown in this LINK.



Data Collection – Scraping

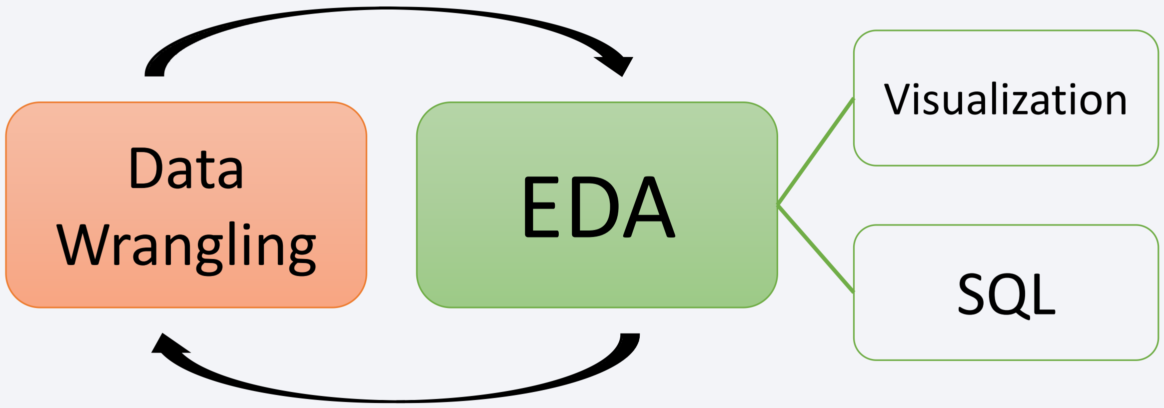
Web scraping process is illustrated in the flowchart

The completed SpaceX Web Scraping notebook is shown in this [LINK].

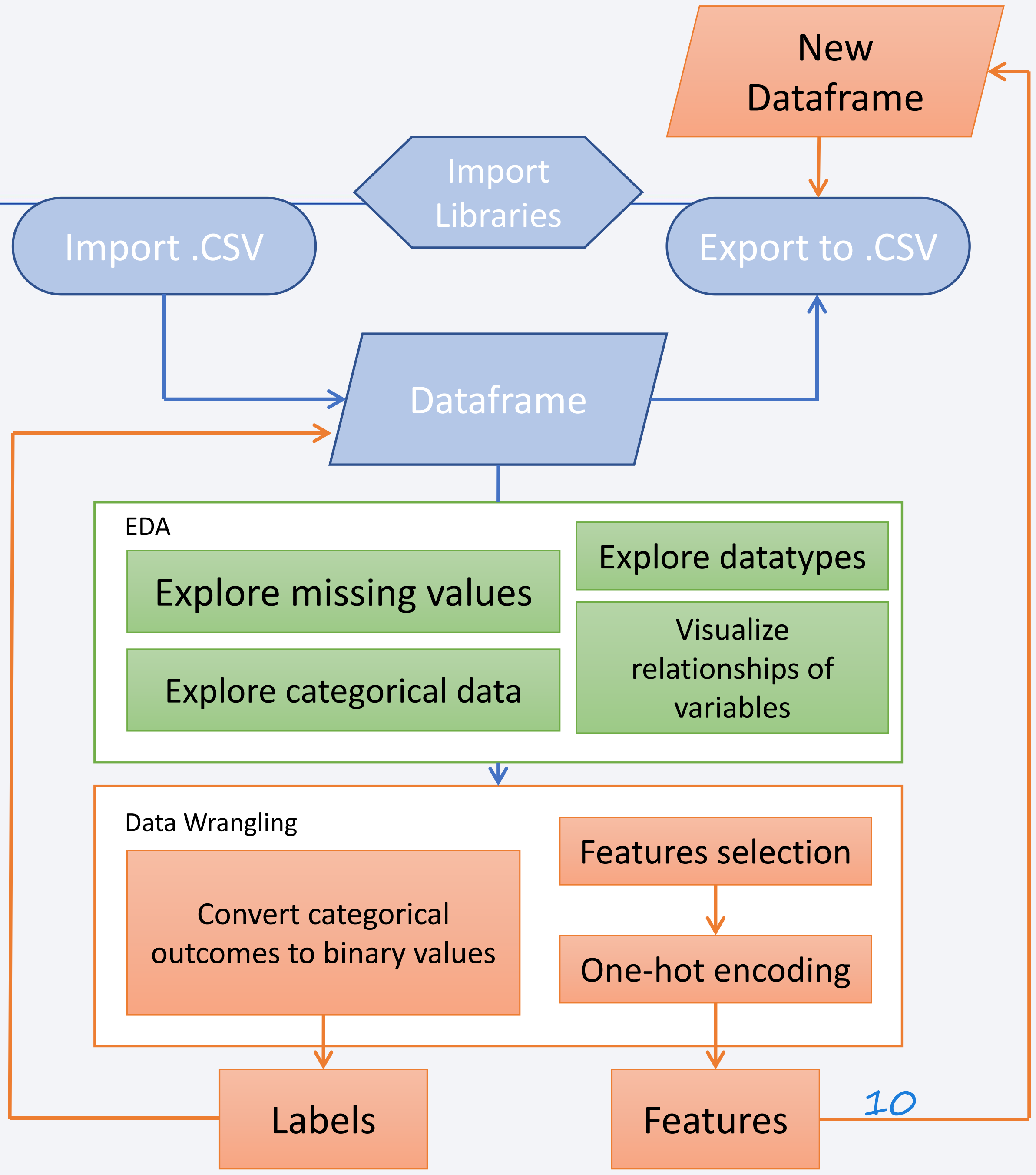


Data Wrangling

- To detect certain patterns in the data and define the label for training supervised models and prepare data feature engineering, exploratory data analysis (EDA) was used.



- The completed SpaceX Data Wrangling notebook is shown in this LINK.



EDA with Data Visualization

- Scatter charts (seaborn.catplot) were used to investigate how correlations between factors affected whether or not the landings were successful. A bar chart was used to determine the association between each variable's success rate. The yearly trend of success rate was determined using a line chart.

Scatterplot

- FlightNumber vs. PayloadMass
- FlightNumber vs. LaunchSITE
- Payload and LaunchSite
- FlightNumber and Orbit type
- Payload and Orbit type

Bar Chart

- Success rate of each orbit type

Line Chart

- Landing success yearly trend

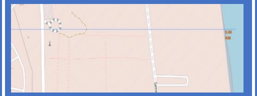
- The completed SpaceX EDA with Data Visualization notebook is shown in this LINK.

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 successful 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.
  - 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.
- The completed SpaceX EDA with SQL notebook is shown in this LINK.

Build an Interactive Map with Folium

- Map components such as circles, markers, and polylines were built and added to the folium map. MarkerClusters were also used to group together many markers with the same coordinate. A MousePosition was also introduced to acquire the coordinates for a mouse over a map point.
- The purposes of those objects added on the map are stated as table below:

Objects \ Map Objects	Circle 	Marker 	MarkerCluster 	Polyline 	MousePosition 
Task 1: Mark all launch sites on a map	✓	✓			
Task 2: Mark the success/failed launches for each site.		✓	✓		
Task 3: Distances between a launch site to its proximities.		✓		✓	✓

- The completed SpaceX Interactive Map with Folium notebook is shown in this LINK.

Build a Dashboard with Plotly Dash

- The dashboard consists of two sections:

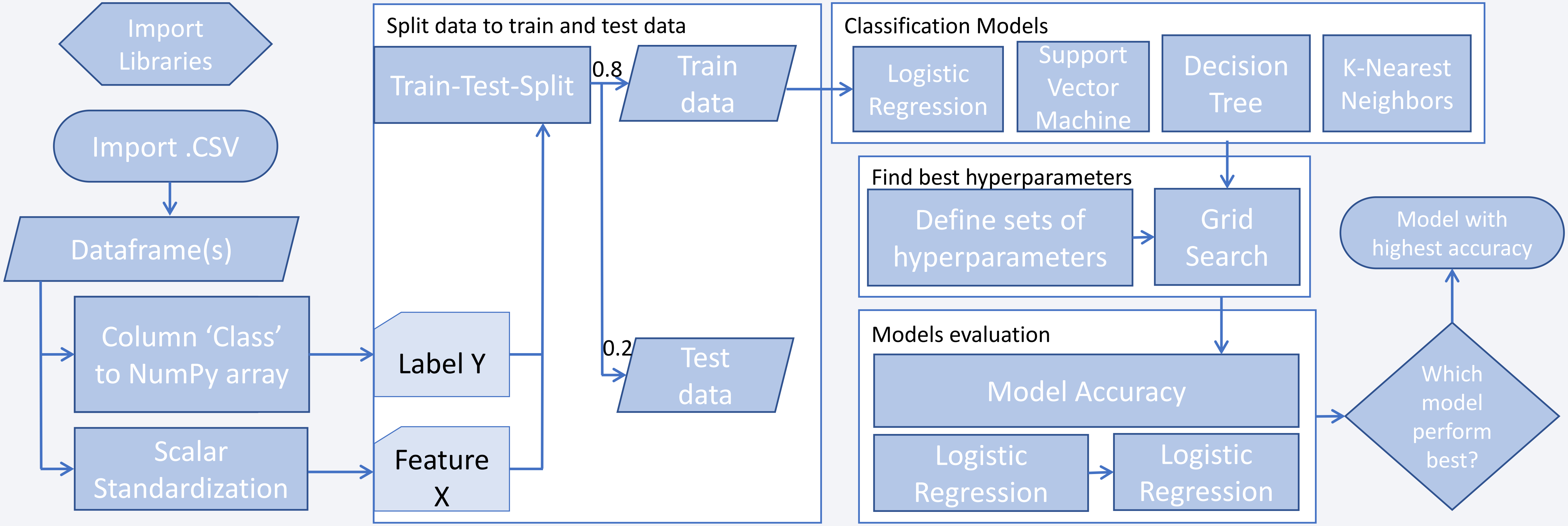
1. To begin, a dropdown menu of the four launch locations has been included, with the selected launch site displaying a pie chart depicting the launch success percentage (Success vs Failure).

2. Then, a range slider was added to pick a certain payload mass range, and a scatterplot display was added to demonstrate how payload may be connected to landing outcomes for the chosen launch location(s).

- The completed SpaceX Plotly Dash Laboratory is shown in this LINK.

Predictive Analysis (Classification)

- The best performing classification model was built, evaluated, improved, and found as flowchart below:



- The completed Predictive Analysis Laboratory notebook is shown in this LINK.

Results

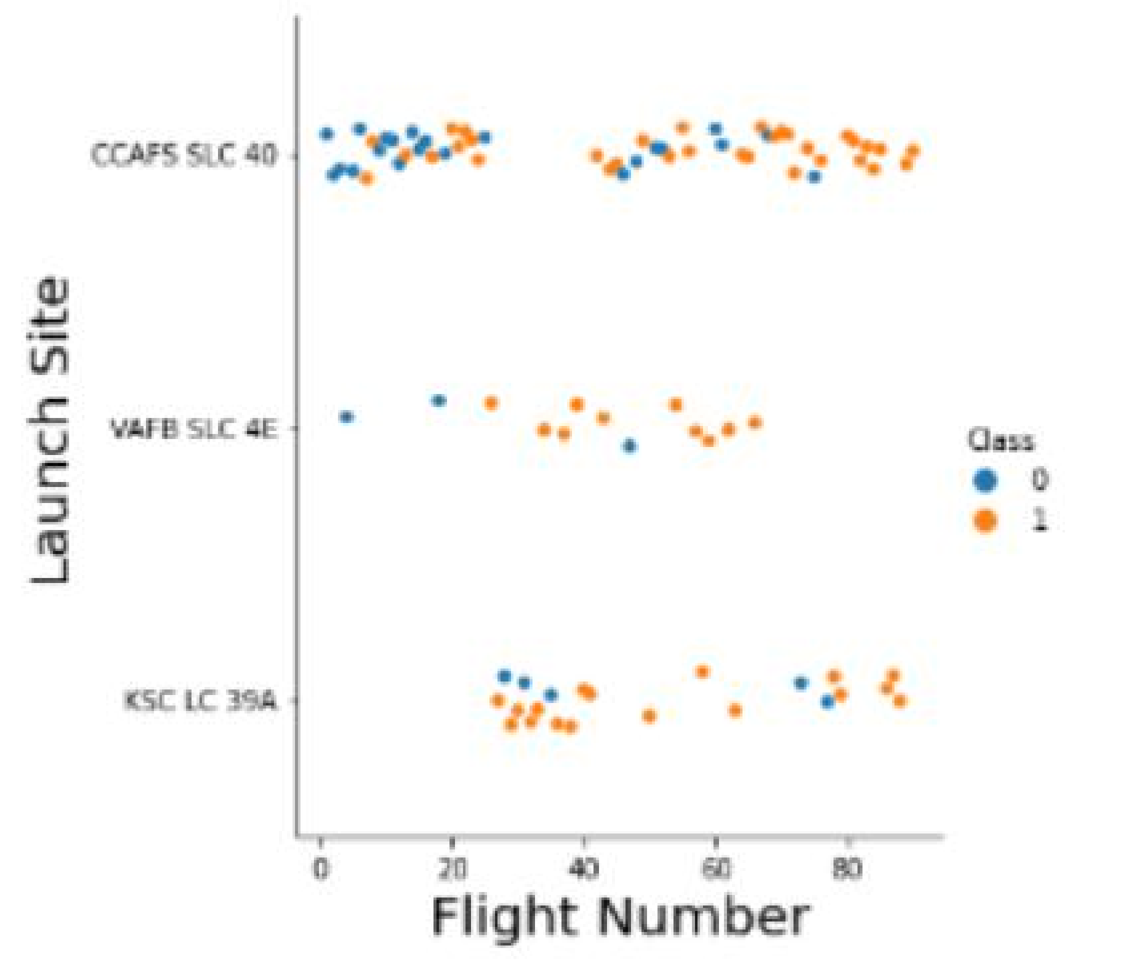
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

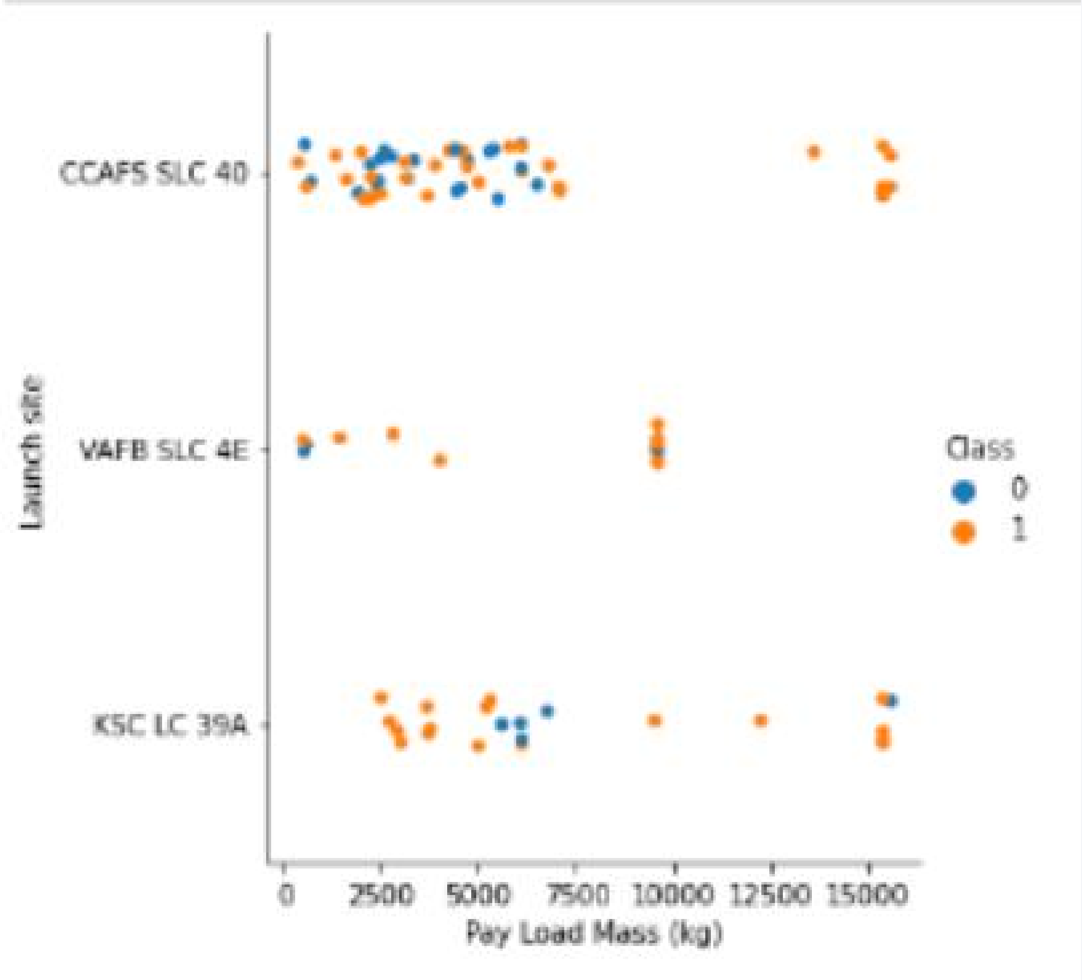
- Figure right shows a scatterplot of Flight Number vs. Launch Site.
- Overall, the first stage is more likely to land safely as the number of flights grows, notably at launch site CCAFS SLC 40. We also noticed that after the 70th flight for VAFB SLC 4E and before the 20th flight for KSC LC 39A, no rockets were fired.



Payload vs. Launch Site

- Figure on the right shows a scatterplot of Payload vs. Launch Site.

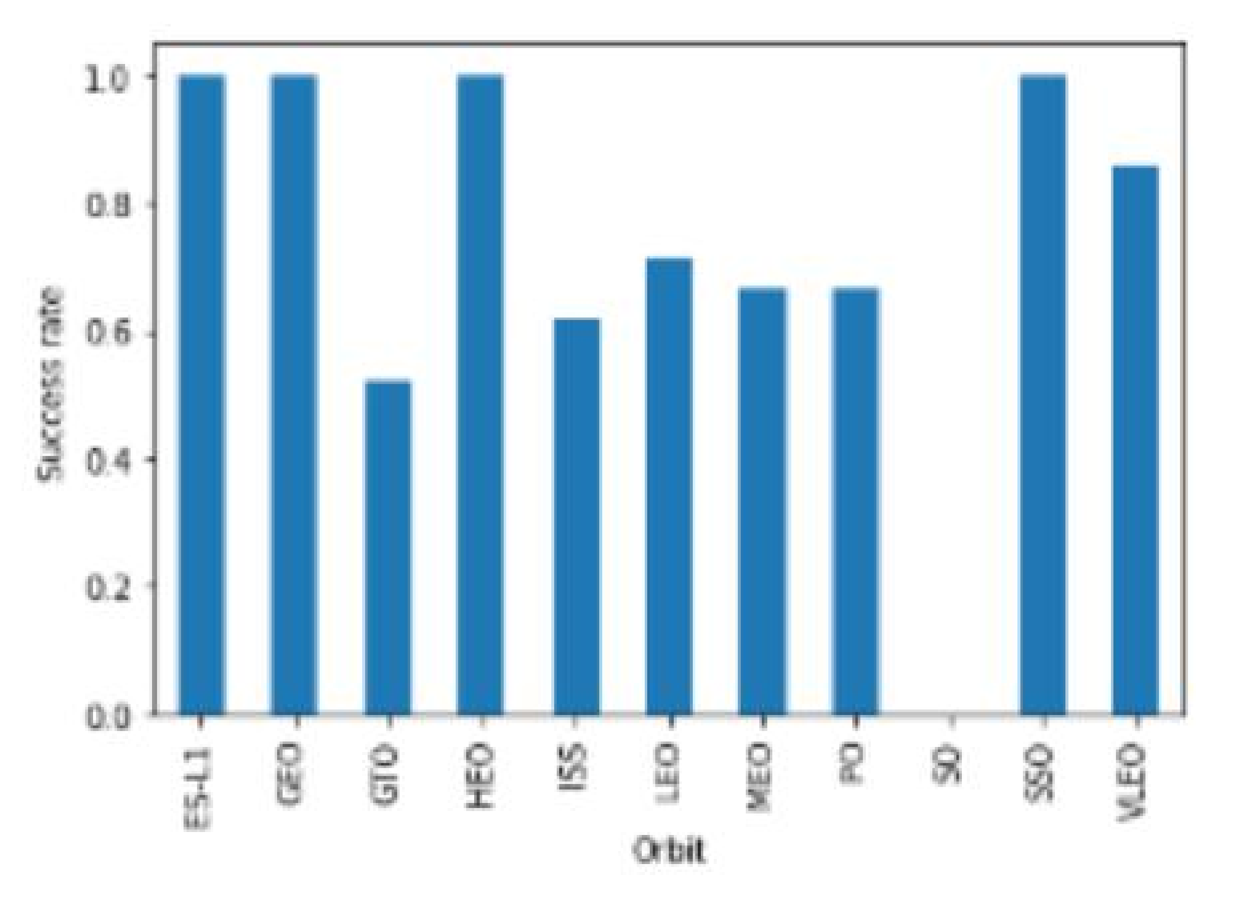
- There are no rockets launched for heavy payload mass at the VAFB SLC 4E launch site (greater than 10,000).



Success Rate vs. Orbit Type

- Bar chart on the right shows the success rate of each orbit type.

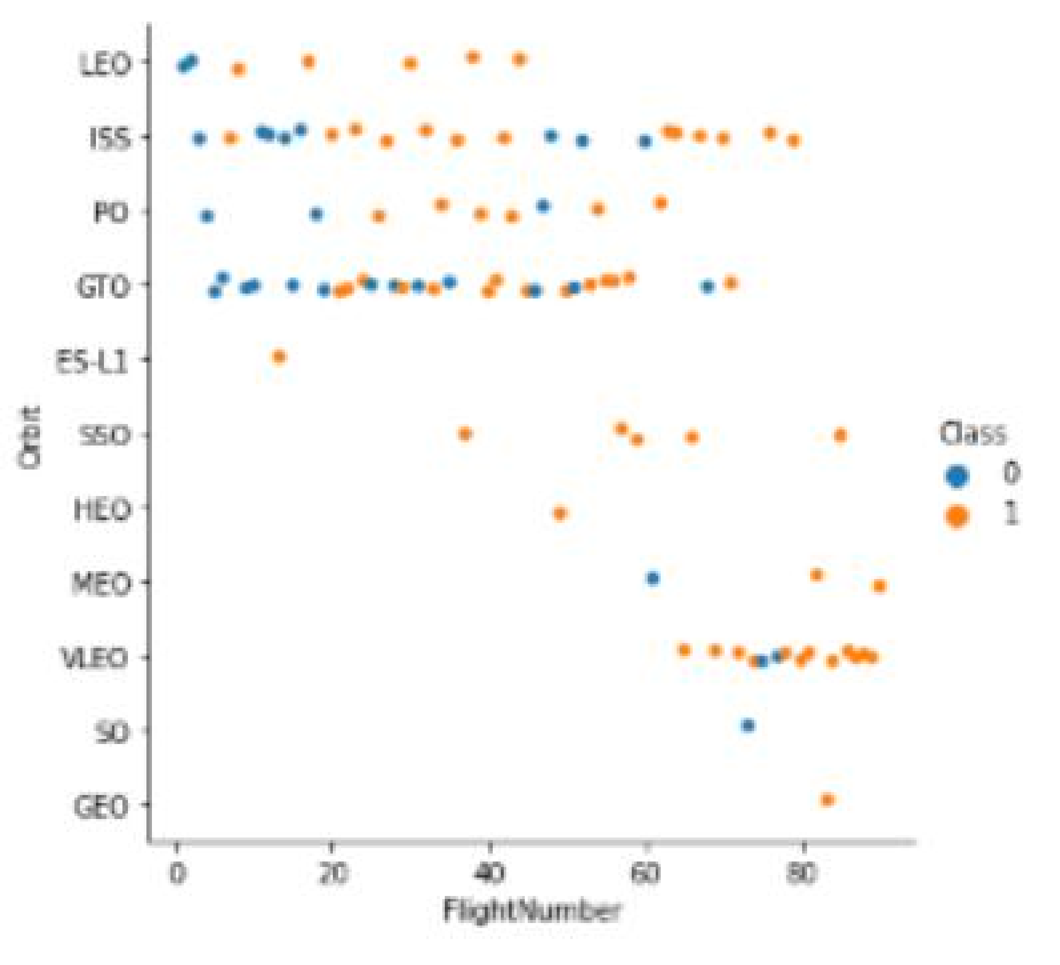
- We found that orbits ES-L1, GEO, HEO, and SSO all had 100% success rates, whereas orbit SO had no rockets successfully land.



Flight Number vs. Orbit Type

- Scatter point on the right illustrates the Flight Number vs. Orbit Type.

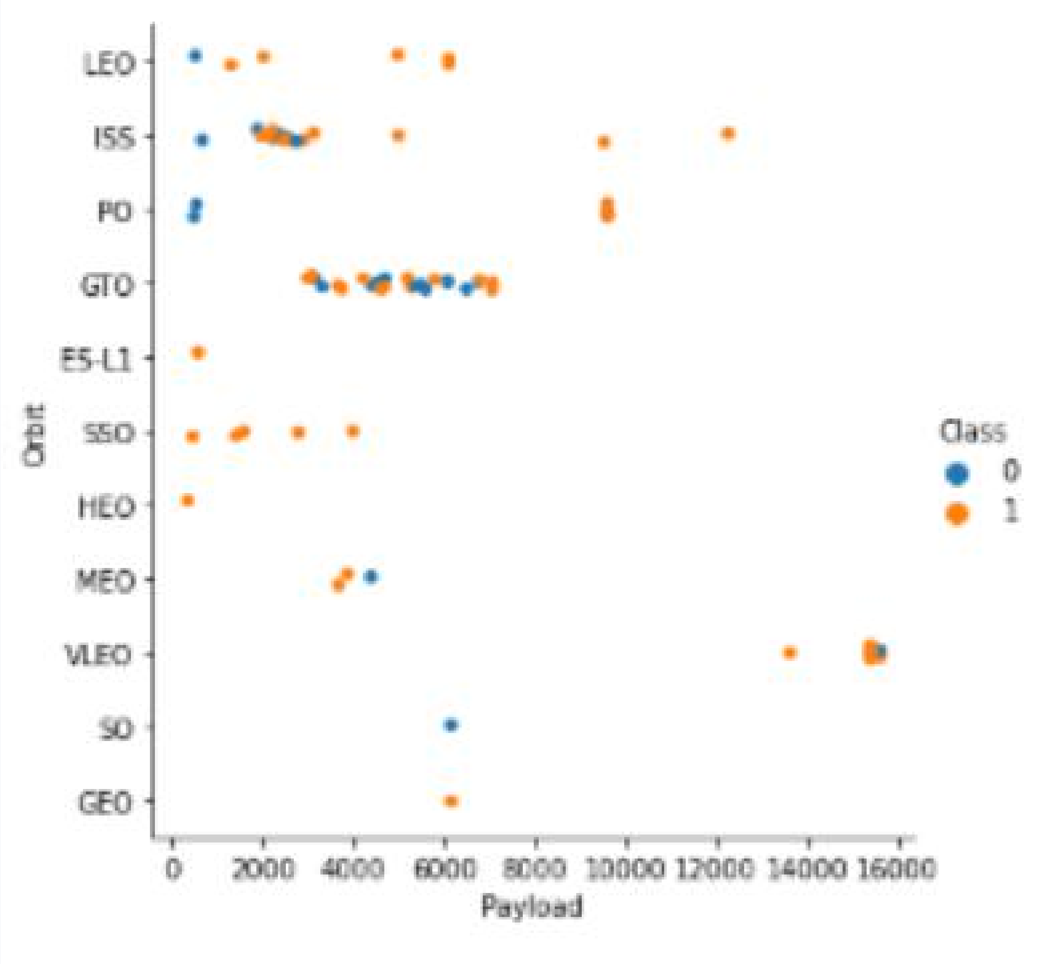
- In LEO orbit, success appears to be connected to the number of flights; however, in GTO orbit, there appear to be no connections between flight number and success.



Payload vs. Orbit Type

- Figure on the right illustrated a scatter point of Payload vs. Orbit Type.

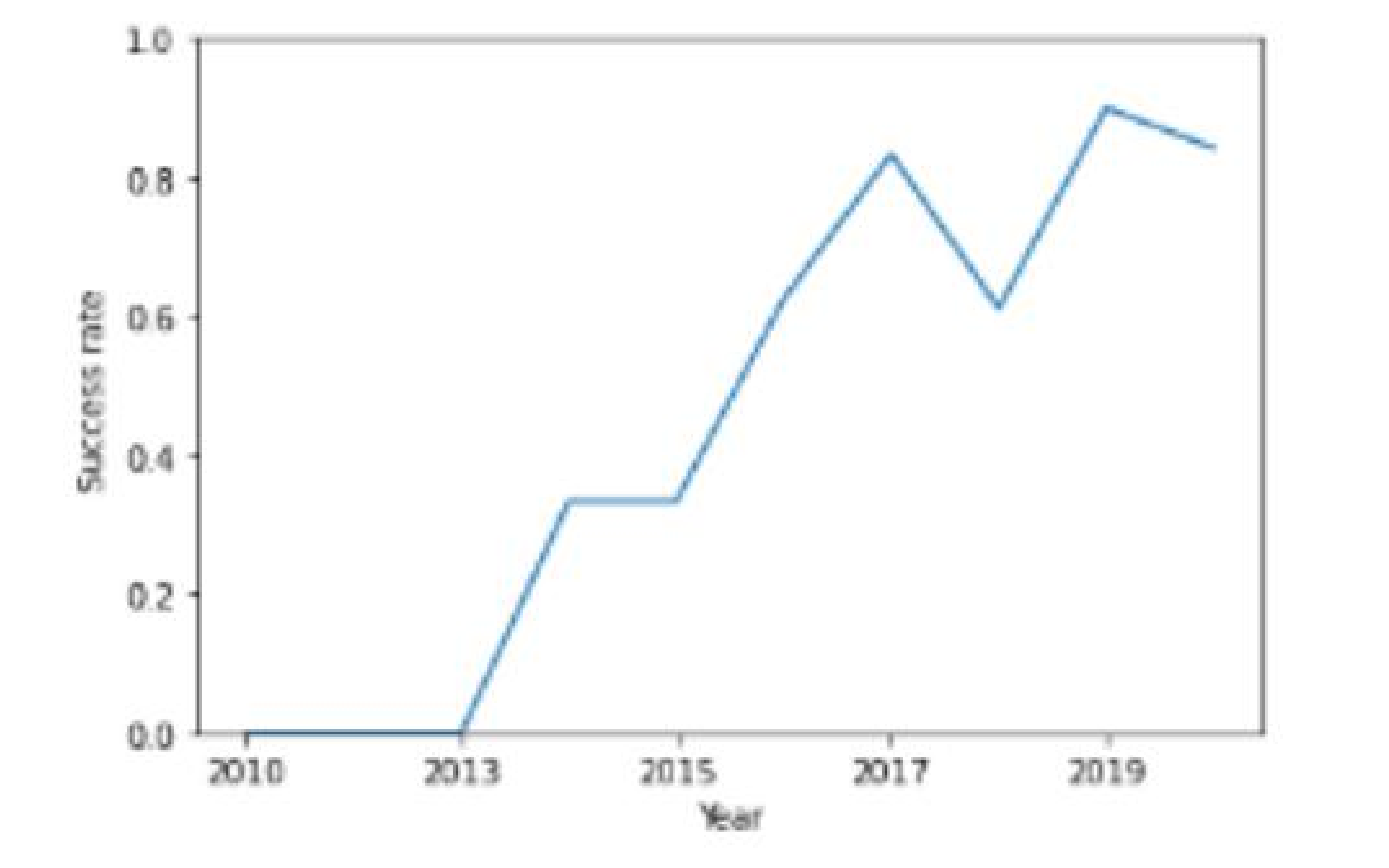
- Polar, LEO, and ISS are more likely to land successfully with big payloads. However, we can't tell the difference in GTO since both positive and negative landing rates happen.



Launch Success Yearly Trend

- The figure shown on the right is a line chart of Yearly Average Success Rate.

- In general, the success rate since 2013 kept increasing til' 2020.



All Launch Site Names

- Find the names of the unique launch sites

```
%sql SELECT UNIQUE(LAUNCH_SITE) FROM SPACEXDATASET
```

```
   ibm_db_sa://kq109780:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

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

- There are 4 launch sites for SpaceX Falcon 9 rocket within the past 10 years.

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

```
%sql SELECT * FROM SPACEXDATASET \
    WHERE LAUNCH_SITE LIKE '%CCA%' \
    LIMIT 5
```

```
   ibm_db_sa://kq109780:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:31249/bludb
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/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

- The query displays a list of records and information with launch site names begin with CCA.

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

```
%sql SELECT SUM(payload_mass__kg_) AS total_payload_mass FROM SPACEXDATASET \
    WHERE customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

7]:

total_payload_mass
45596

- The total payload mass carried by boosters from NASA is 45596 kg.

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(payload_mass__kg_) AS average_payload_mass FROM SPACEXDATASET \
    WHERE booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

8]:

average_payload_mass
2928

- The average payload mass carried by booster version F9 v1.1 is 2928 kg.

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
%sql SELECT MIN(DATE) FROM SPACEXDATASET \
    WHERE landing__outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

9]:

1
2015-12-22

- The first successful landing outcome on ground pad happened on 2015 December 22nd.

Successful Drone Ship Landing with Payload between 4001 and 5999

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%sql SELECT booster_version FROM SPACEXDATASET \
    WHERE (landing__outcome = 'Success (drone ship)') \
            AND (payload_mass__kg_ BETWEEN 4001 AND 5999)
```

```
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.a
ppdomain.cloud:32731/bludb
Done.
```

0]:

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

- There are 4 boosters that satisfied the queries, namely F9 FT B1022, F9 FT B1026, F9 FT B1021.2, and F9 FT B1031.2.

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
%sql SELECT mission_outcome, COUNT(mission_outcome) AS total_number FROM SPACEXDATASET \
    GROUP BY mission_outcome
```

```
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

1]:

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

- There is only 1 failure mission outcome and 100 success mission outcomes, with one uncleared payload status.

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
%sql SELECT booster_version, payload_mass__kg_ FROM SPACEXDATASET \
    WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXDATASET)
```

```
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

2]:

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

- The result shows a list of 12 booster versions that carried the maximum payload mass of 15,600 kg.

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql SELECT booster_version, launch_site, landing__outcome FROM SPACEXDATASET \
   WHERE (landing__outcome = 'Failure (drone ship)') AND (DATE BETWEEN '2015-01-01' AND '2015-12-31'
```

```
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.a
ppdomain.cloud:32731/bludb
Done.
```

3]:

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

- In 2015, booster version F9 v1.1 B1012 and B1015 have failed landing in drone ship both on launch site CCAFS LC-40.

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

- 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

```
%sql SELECT landing__outcome, COUNT(landing__outcome) AS count_outcome FROM SPACEXDATASET \
    WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
    GROUP BY landing__outcome \
    ORDER BY count_outcome DESC
```

```
 * ibm_db_sa://wxd04714:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

4]:

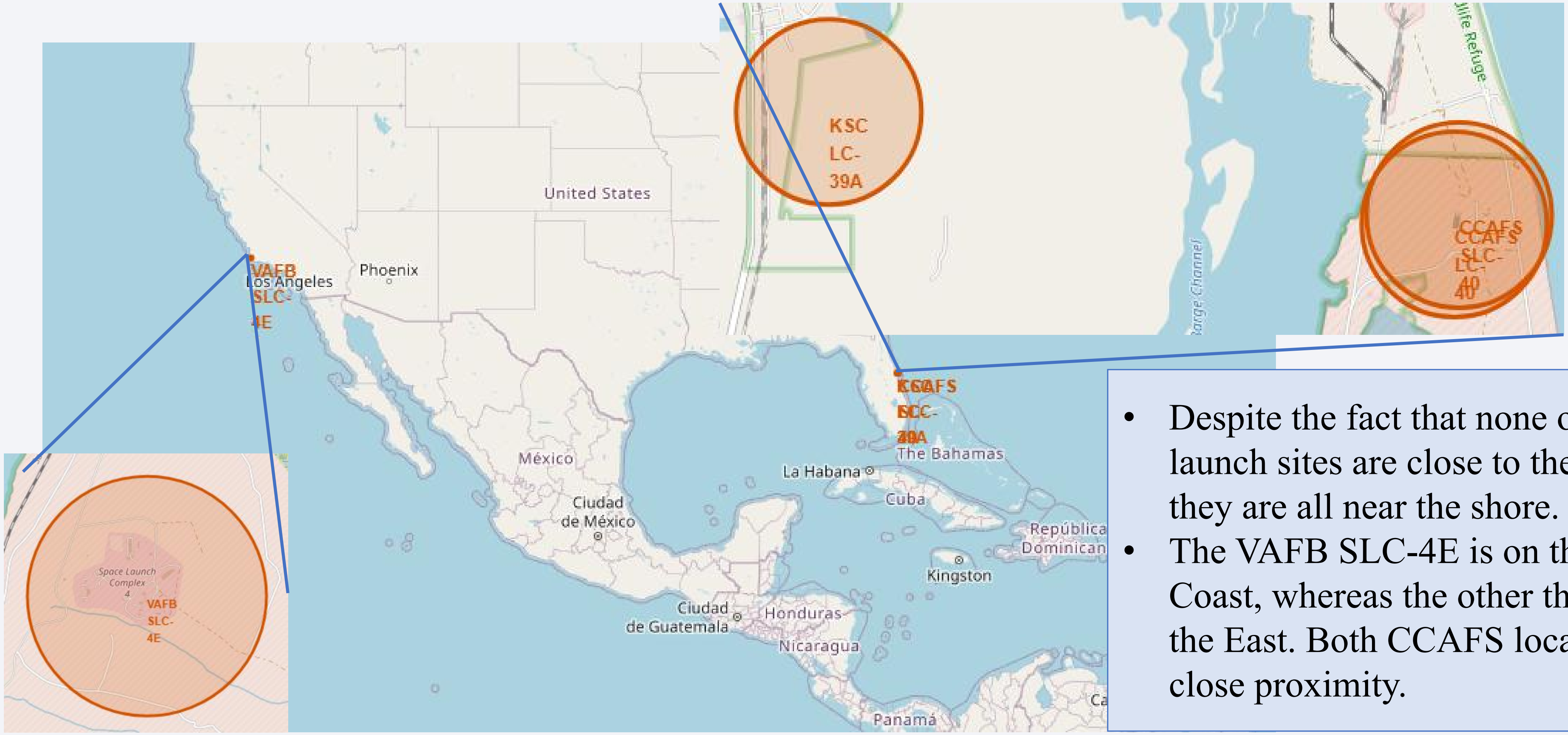
landing__outcome	count_outcome
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

- Within the specified dates, most of the landing outcomes were no attempt, followed by both success and failure in drone ship.

Section 3

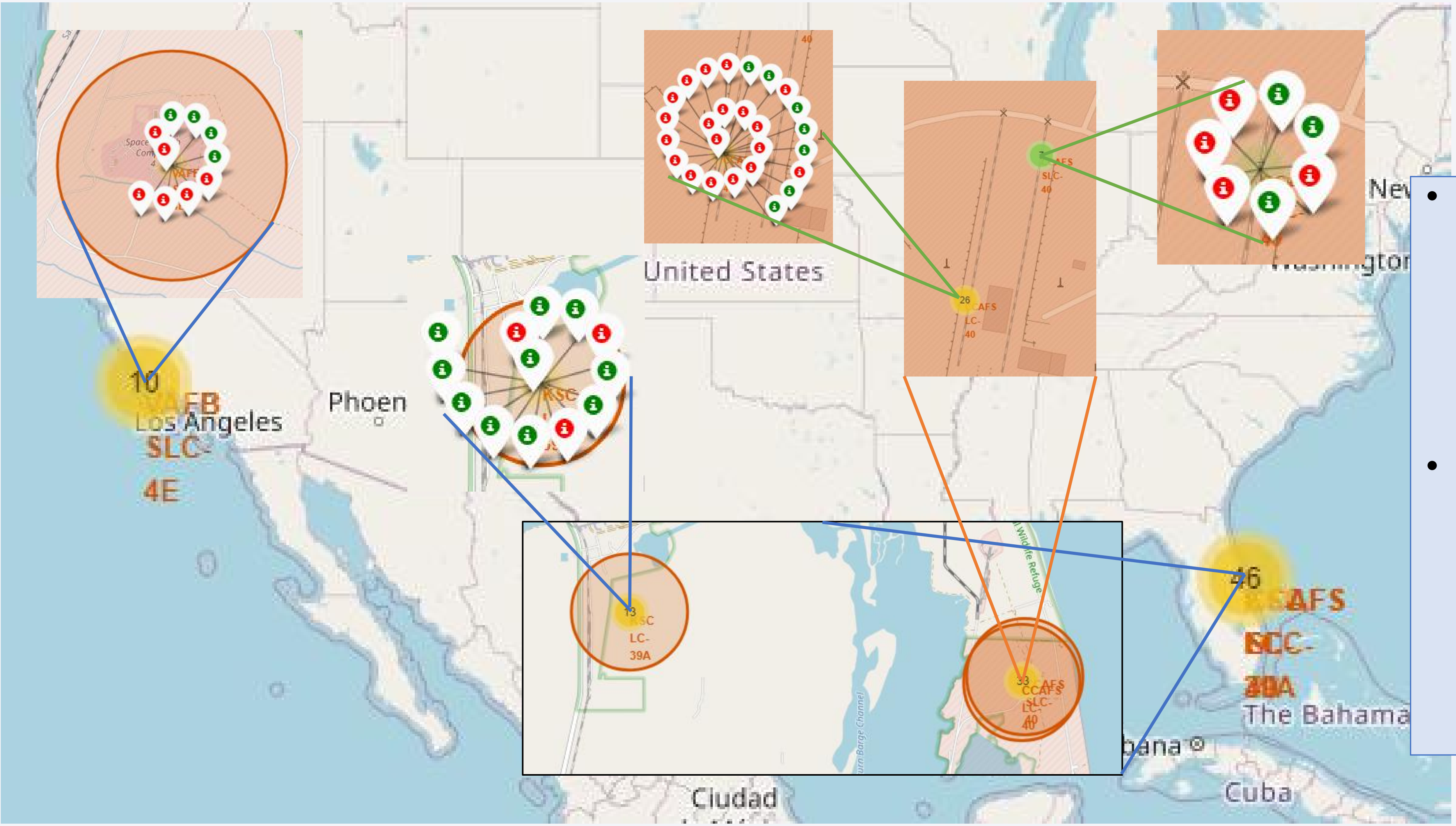
Launch Sites Proximities Analysis

Locations of All Launch Sites



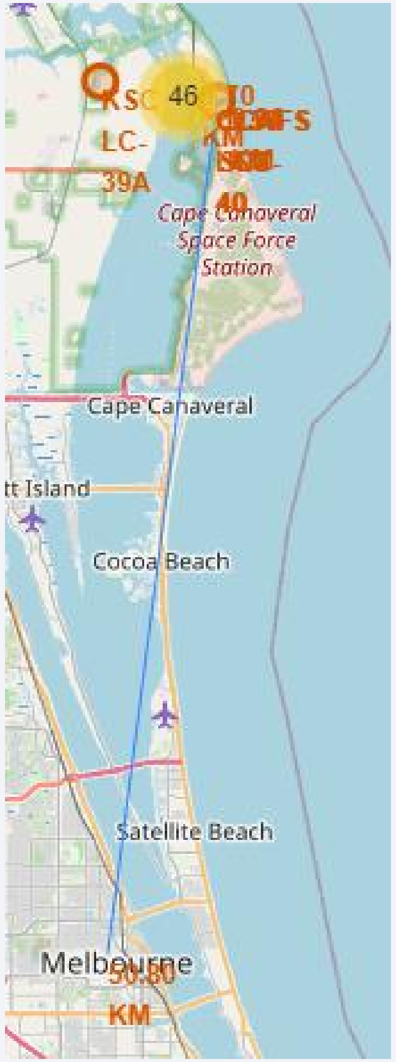
- Despite the fact that none of the launch sites are close to the Equator, they are all near the shore.
- The VAFB SLC-4E is on the West Coast, whereas the other three are on the East. Both CCAFS locations are in close proximity.

Success or Failed Launches for each Site



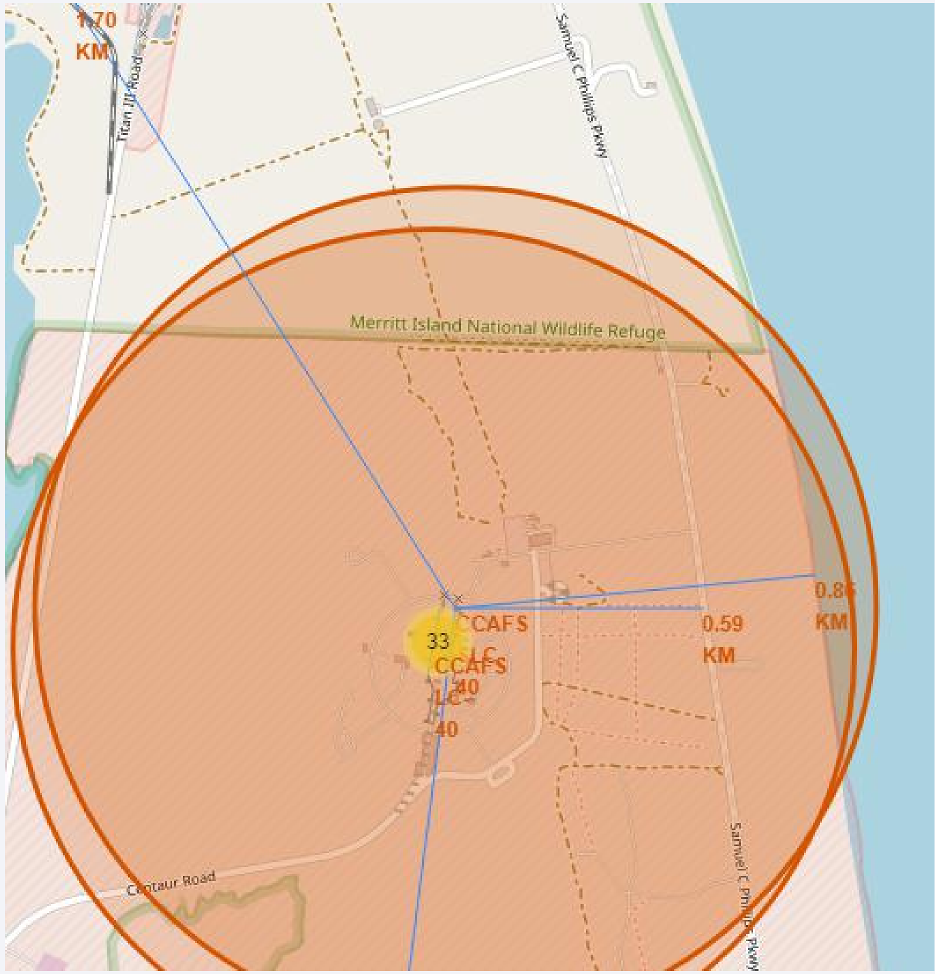
- Green marks indicate successful launches, while red markers indicate unsuccessful launches.
- We can see that KSC LC-39A has a pretty good success rate when comparing the colored markers for different launch locations.

Distances between a Launch Site to its Proximities



- The maps show the distances between site CCAFS SLC-40 and its nearby cities, railways, highways, and shoreline.
- The results are summarizes as table shown below:

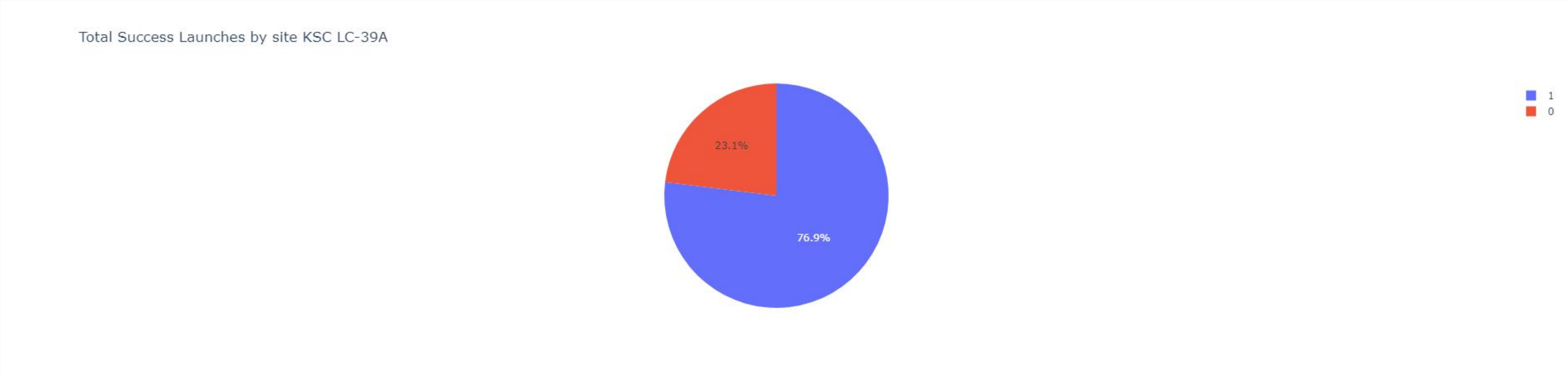
Proximities	Distances (KM)
Nearest City	50.80
Railway	1.70
Highway	0.59
Coastline	0.86



Section 4

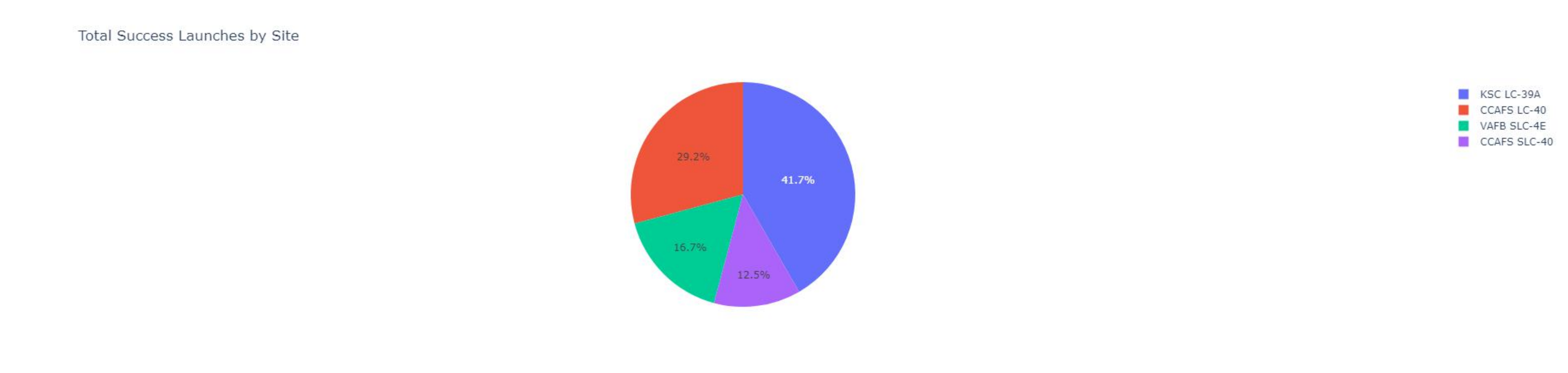
Build a Dashboard with Plotly Dash

Launch Success Ratio for Launch Site



- In the SpaceX Launch Record Dashboard, pie charts on the right show the success ratio (1 = success vs 0 - failed).
- KSC LC-39A site has the highest success ratio of 76.9%; while the other sites have < 50% of success ratio (Refer to Appendix 3).

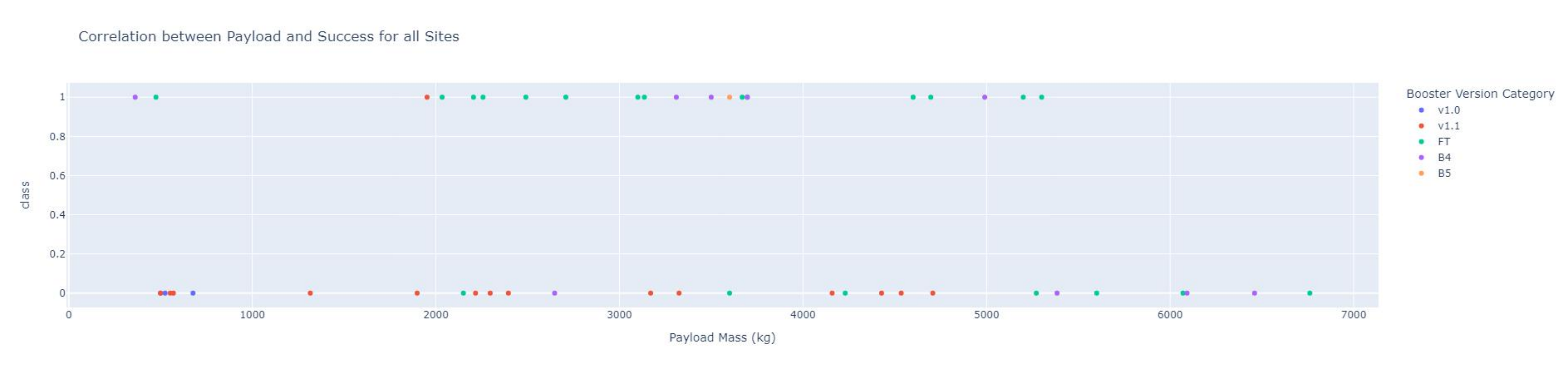
Launch Success Counts for All Sites



- Based on the SpaceX Launch Record Dashboard, pie chart on the right displays the total counts of success for all sites.

- Launches on KSC LC-39A site has the highest success counts, with 41.7% success launches among all.

Payload vs. Launch Outcome for All Sites



- With different booster variants, the scatterplots show the correlations between payload and launch results. In general, a larger payload bulk lowers the success rate.
- However, after 7,000 kg, there are certain outliers in the payload range. As a result, we discovered that booster v1.1 has a very low success rate in the region of 0 to 7,000 kg, but boosters FT and B4 have a greater success rate with lower payload mass.

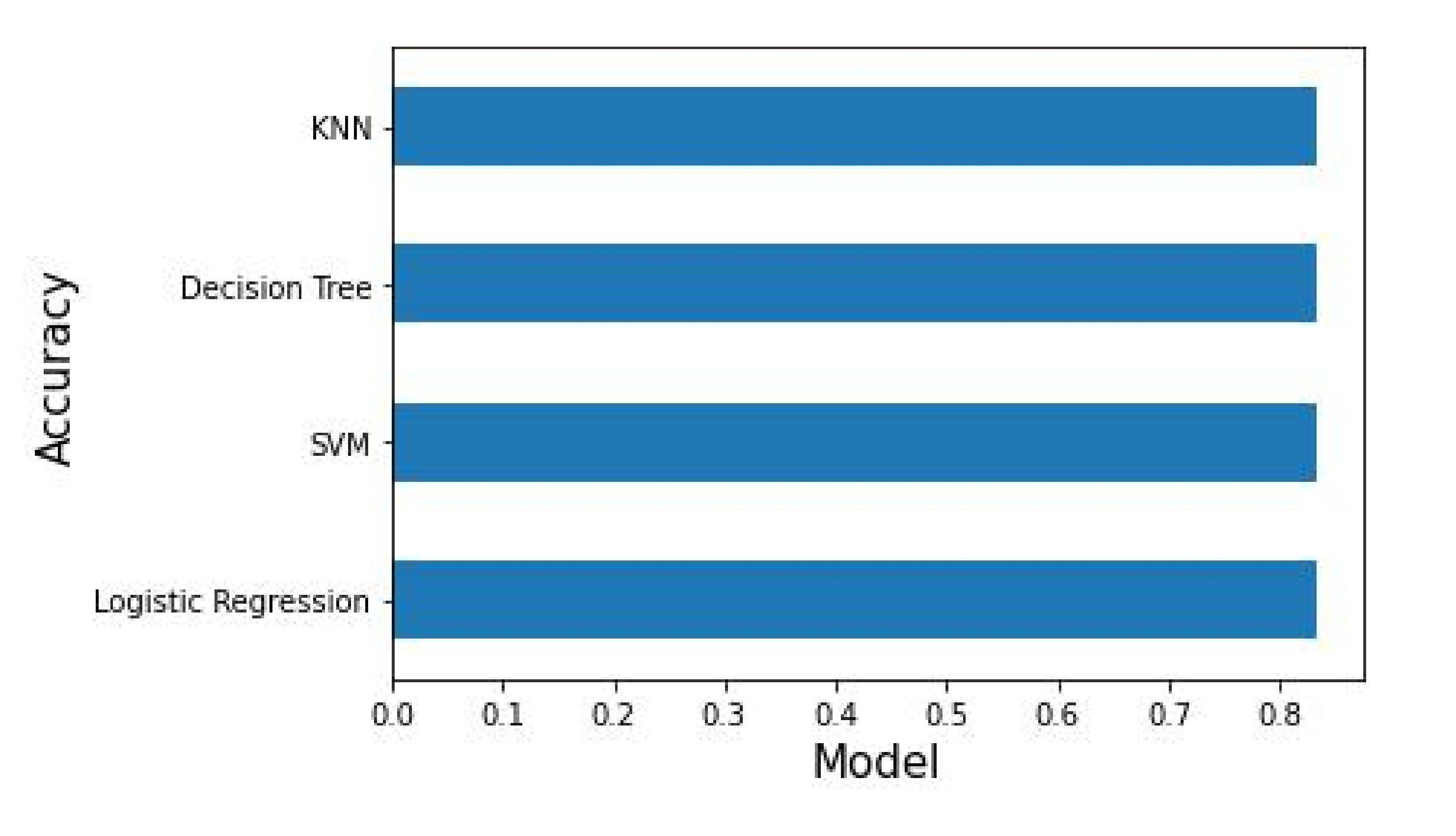
Section 5

Predictive Analysis (Classification)

Classification Accuracy

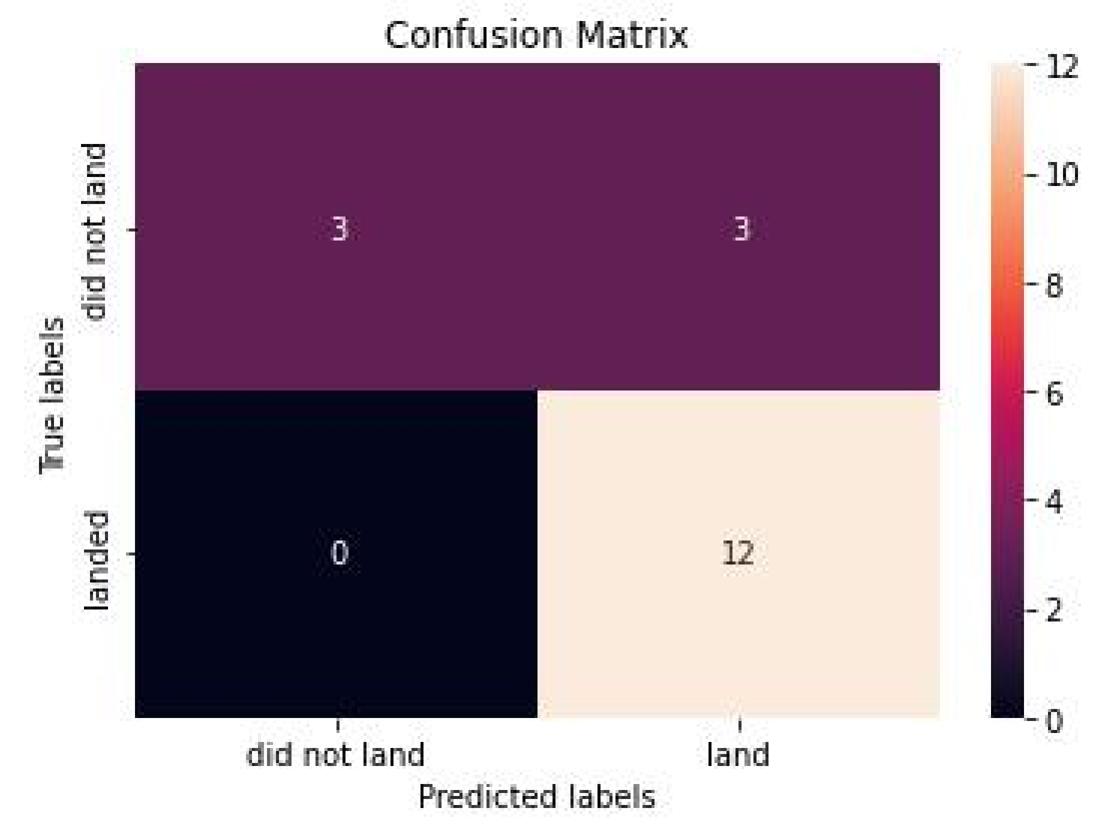
- Bar Chart on the right visualizes the built model accuracy for all built classification models.

- It seems that all four models have the similar classification accuracy of ≈83.33%.



Confusion Matrix

- Since the accuracy for all models perform the same, their confusion matrix are also similar as shown in the figure.
- With a precision of 80% and a sensitivity of 100%, we can see that the models accurately predicted all of the landed outcomes, however there were a few "did not land" events that were mis-predicted as landing by the models.



Conclusions

- The goal of this study was to anticipate if the first stage of SpaceX Falcon 9 rockets will successfully land. As a result, classification models were developed and tested for this purpose.

- Logistic Regression, SVM, Decision Tree, and K-Nearest Neighbors are all appropriate models with good model accuracy (greater than (>) 80 percent).

- Our findings also show that where the rockets launch and which booster types are employed have a significant impact on successful landings. The higher the success rate, the lighter the payload.

Appendix

- Appendix 1: Example of Python code and outputs of Data Collection API

```
# Show the head of the dataframe
data_launch.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

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

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366

Now that we have removed some values we should reset the FlgihtNumber column

```
pd.options.mode.chained_assignment = None
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
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
```

Before we can continue we must deal with these missing values. The `LandingPad` column will retain None values to represent when la

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace n the mean you calculated.

```
pd.options.mode.chained_assignment = None
# Calculate the mean value of PayloadMass column
PayloadMass_mean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PayloadMass_mean)
data_falcon9.isnull().sum()
```

```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
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
```

Appendix

- Appendix 2: Example of Python code and outputs of Data Collection with Web Scraping

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
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']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.

Usually, HTML tables in Wiki pages are likely to contain unexpected annotations and other types of noises, such as reference links `B0004.1[8]`, missing values `N/A [e]`, inconsistent formatting, etc.

To simplify the parsing process, we have provided an incomplete code snippet below to help you to fill up the `launch_dict`. Please complete the following code snippet with TODOs or you can choose to write your own logic to parse all launch tables:

```
extracted_row = 0
#Extract each table
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
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)

            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            #print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
```


```
            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            #print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
```


```
            launch_dict['Version Booster'].append(bv)
            #print(bv)


            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)
            #print(launch_site)

            # Payload
            # TODO: Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)
            #print(payload)

            # Payload Mass
            # TODO: Append the payload_mass into launch_dict with key `Payload mass`
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)
            #print(payload)

            # Orbit
            # TODO: Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)
            #print(orbit)

            # Customer
            # TODO: Append the customer into launch_dict with key `Customer`
            customer = row[6].text.strip()
            launch_dict['Customer'].append(customer)
            #print(customer)

            # Launch outcome
            # TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome'].append(launch_outcome)
            #print(launch_outcome)

            # Booster landing
            # TODO: Append the launch_outcome into launch_dict with key `Booster landing`
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing'].append(booster_landing)
            #print(booster_landing)
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

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

	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 (COTS)\nNRO	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

Appendix

- Appendix 3: Example of Python code and outputs of Data Wrangling

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()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

TASK 3: Calculate the number and occurence of mission outcome per orbit type

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
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `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 to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

```
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
```

We create a set of outcomes where the second stage did not land successfully:

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

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

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
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

```
df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude
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
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
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
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
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

We can use the following line of code to determine the success rate:

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

```
0.6666666666666666
```

We can now export it to a CSV for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

```
df.to_csv("dataset_part\_2.csv", index=False)
```

Appendix

- Appendix 4: Dataset stored in IBM Db2 Database

IBM Db2 on Cloud

Load Data | Load History | **Tables** | Views | Indexes | Aliases | MQTs | Sequences | Application objects

WXD04714.SPACEXDATASET

Back

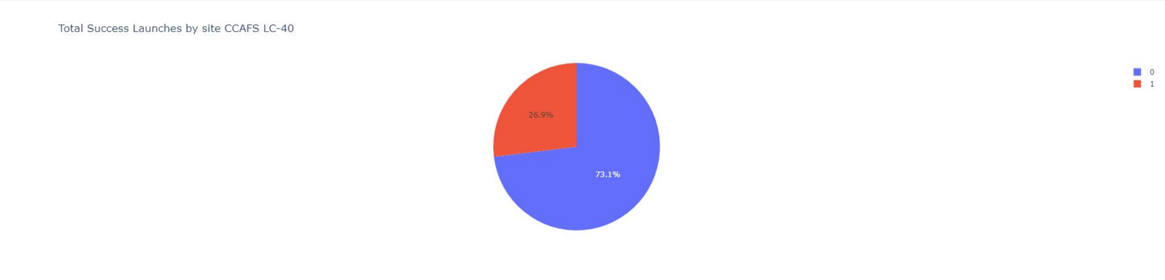
Export to CSV

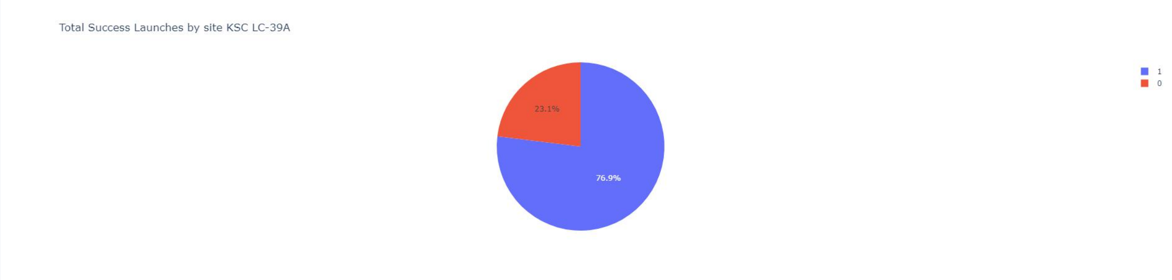
DATE	TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD	PAYLOAD_MASS__KG_	ORBIT	CUSTOMER	MISSION_OUTCOME	LANDING__OUTCOME
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight 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
2013-09-29	16:00:00	F9 v1.1 B1003	VAFB SLC-4E	CASSIOPE	500	Polar LEO	MDA	Success	Uncontrolled (ocean)
2013-12-03	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt
2014-01-06	22:06:00	F9 v1.1	CCAFS LC-40	Thaicom 6	3325	GTO	Thaicom	Success	No attempt
2014-04-18	19:25:00	F9 v1.1	CCAFS LC-40	SpaceX CRS-3	2296	LEO (ISS)	NASA (CRS)	Success	Controlled (ocean)
2014-07-14	15:15:00	F9 v1.1	CCAFS LC-40	OG2 Mission 1 6 Orbcomm-OG2 satellites	1316	LEO	Orbcomm	Success	Controlled (ocean)
2014-08-05	08:00:00	F9 v1.1	CCAFS LC-40	AsiaSat 8	4535	GTO	AsiaSat	Success	No attempt
2014-09-07	05:00:00	F9 v1.1 B1011	CCAFS LC-40	AsiaSat 6	4428	GTO	AsiaSat	Success	No attempt

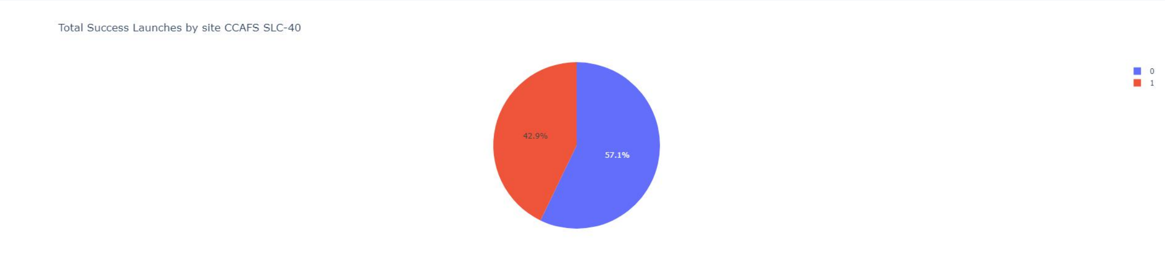
Items per page: 50 | 1–50 items | 1 page 1

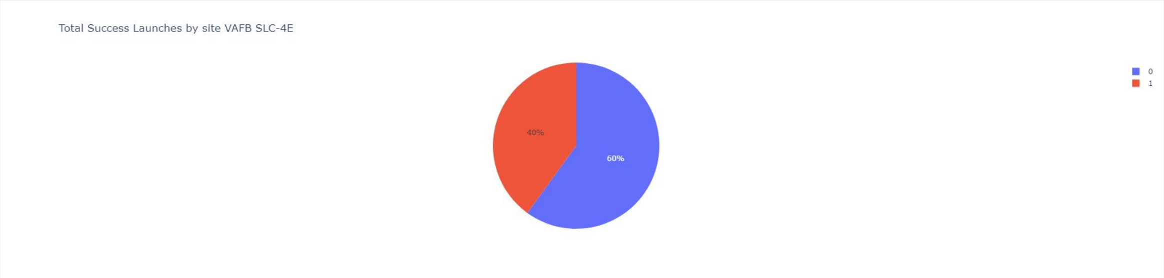
Appendix

- Appendix 5: Pie Charts of Success Ratio for Other Sites.









Thank you!