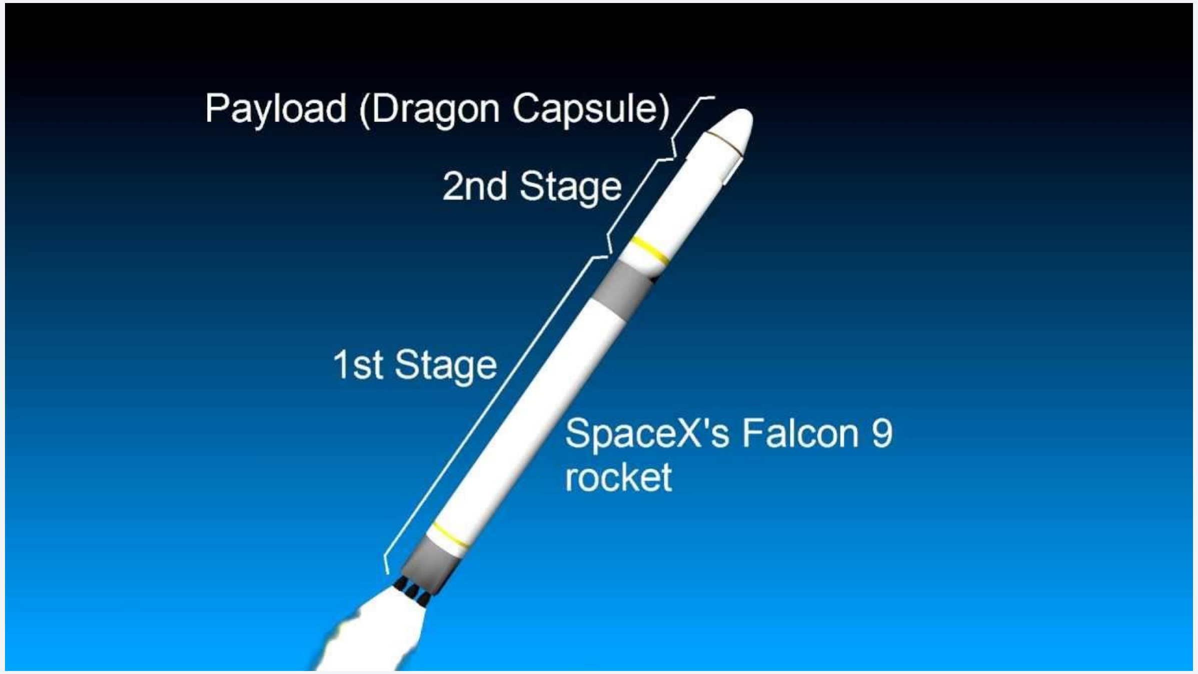
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Evan Alter
May 2023

Outline

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

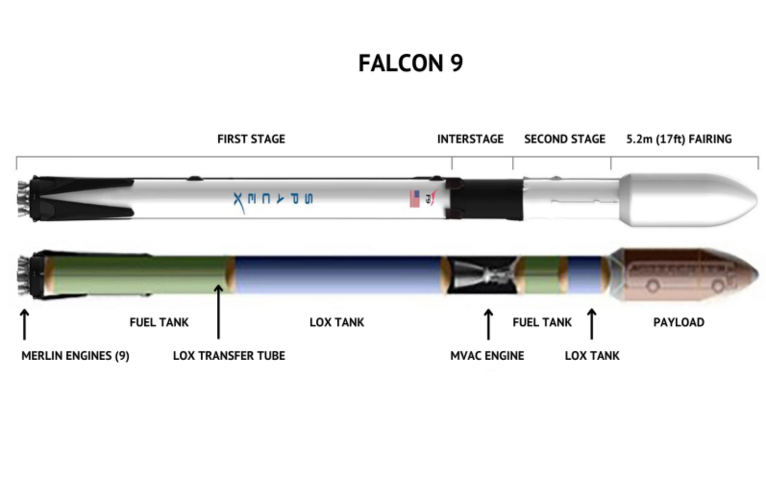


Executive Summary

- Overview
  - This analysis seeks to identify models that can be used to predict the successful return of the first stage of rockets. This can be used to plan profitability for Space Y as it emerges onto the market.
- Summary of methodologies
  - SpaceX REST API call and Web Scraping from Wikipedia with Beautiful Soup were used to collect data. The data was then cleaned and wrangled for Exploratory Data Analysis using Visualization and SQL. Next interactive visual analytics were performed using Folium Map and Plotly Dash. Lastly predictive analysis was performed using classification models (Logistic Regression, Support Vector Machine, Decision Tree, K Nearest Neighbor).
- Summary of all results
  - The Falcon 9 stage 1 has an overall successful return landing rate of 67%, but that varies greatly by launch site, booster version, orbit, and payload mass. The success rate trend has been improving from 2013-2020, and has reached 90% in 2019 and 84% 2020.
  - The Decision Tree method is the best option for predicting success rates. It had an R2 score of 89% in training and 83% in testing. The 3 other methods (Logistic Regression, Support Vector Machine, and K Nearest Neighbor) all performed similarly with a training R2 of 85% and testing R2 of 83%.

https://github.com/ealter19/Applied-Data-Science-Capstone

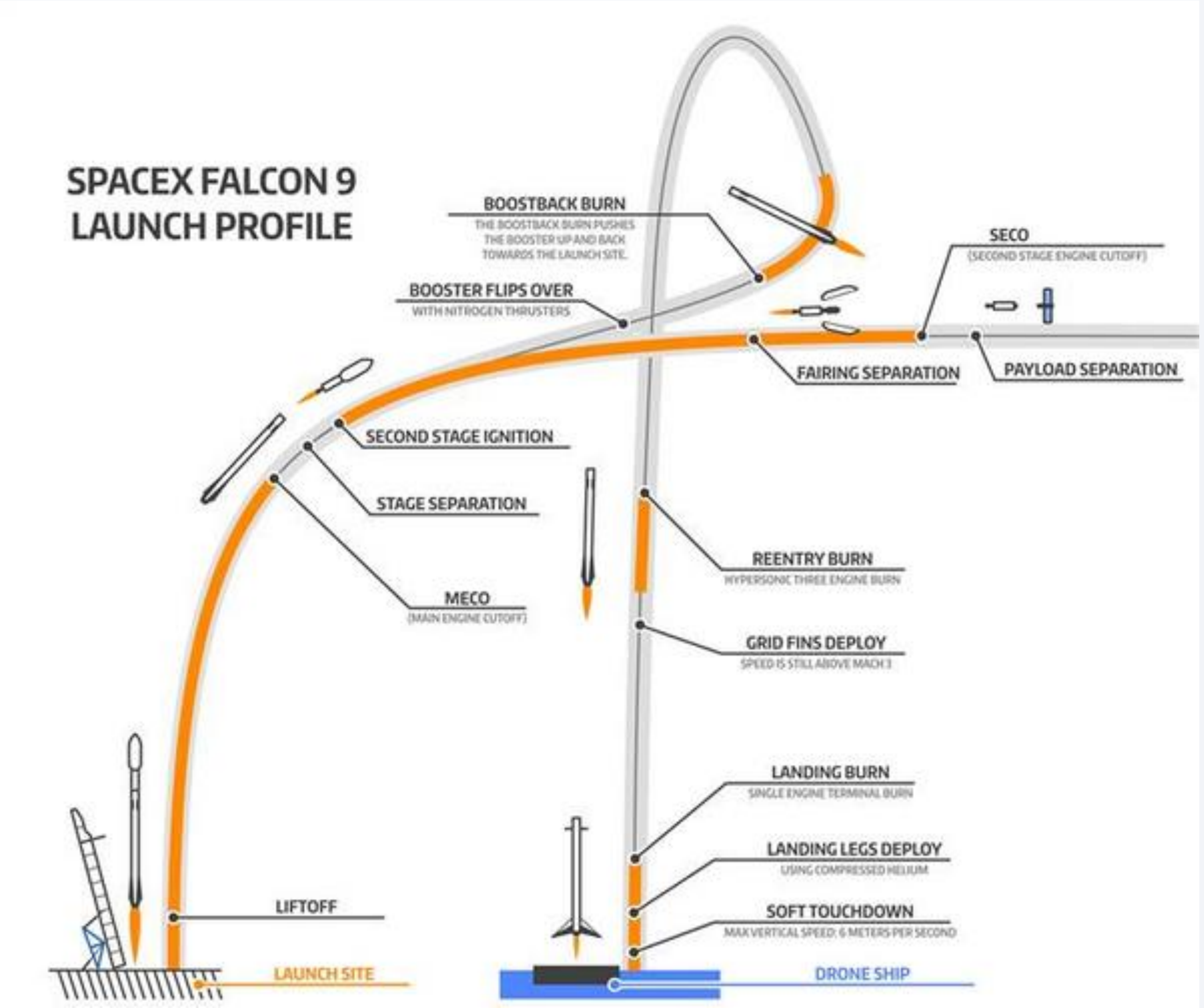
Introduction



- Project Background and context
  - Space Y is a new rocket company seeking to compete with Space X
  - Space Y needs to analyze the success of Space X to understand its challenges
  - The Falcon 9 is a Space X rocket that consists of two stages and a fairings.
    - Stage 1 is the lower portion of the Faclon 9 and has engines that launch it off the launch pad and into the upper atmosphere, where it disconnects from the rest of the rocket.
    - Stage 2 is in the middle of the rocket and has its own engines that fire after Stage 1 separates in order to position the payload in the necessary orbit around Earth. It then disconnects.
    - The fairings is the capsule at the top of the rocket that contains and protects the payload (e.g. satellites), and releases it into orbit when in the correct position.
  - Space X is unique in the space industry because Stage 1 is capable of returning to Earth for reuse in future missions. This can save a significant amount of money and increase Space X's profitability.
  - In this analysis we will examine the success rates of Falcon 9's Stage 1 safe return to Earth. We will build models to predict success rates based off of numerous factors.
  - These results will help Space Y business leaders decide how to best set up their business for success.
- Problems to answer
  - What are the Falcon 9 success rates under various conditions (payload mass, launch site, orbit, etc)?
  - What models can best predict Falcon 9 successful relandings?

GitHub Repository Link

Introduction – Falcon 9 Stages Visual



Section 1

Methodology

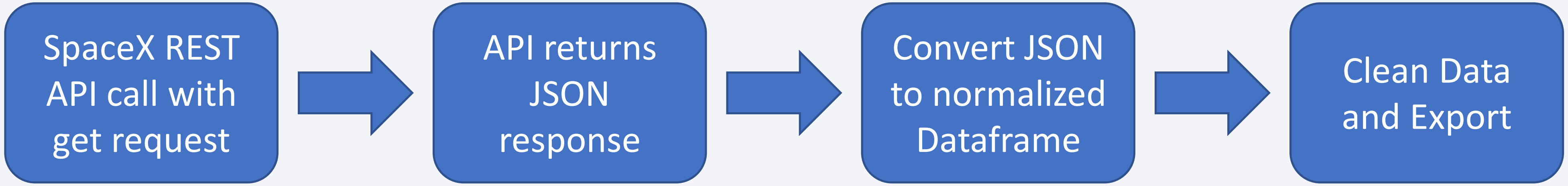
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Dropping unnecessary columns
  - One Hot Encoding for classification models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Publicly available information on Space X’s Falcon 9 program was collected from 2 sources:
  - SpaceX REST API (https://api.spacexdata.com/v4/)



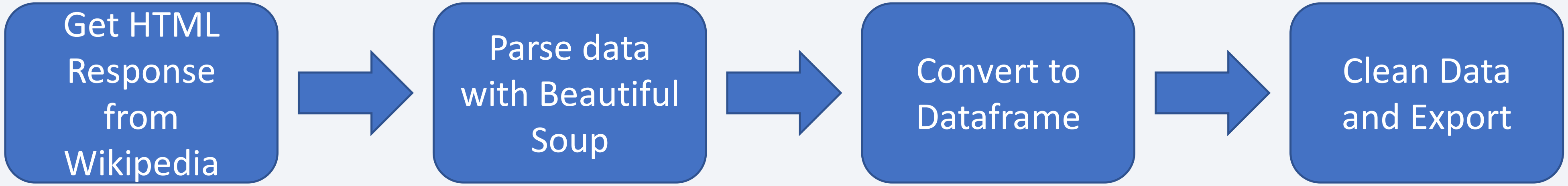
SpaceX REST API call with get request

API returns JSON response

Convert JSON to normalized Dataframe

Clean Data and Export

  - Web Scraping html tables from Wikipedia.org



Get HTML Response from Wikipedia

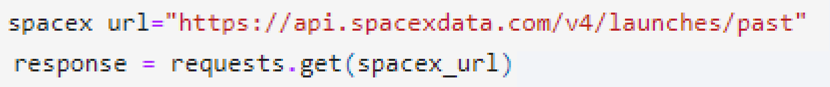
Parse data with Beautiful Soup

Convert to Dataframe

Clean Data and Export

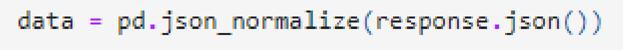
Data Collection – SpaceX API

1. SpaceX REST API Call



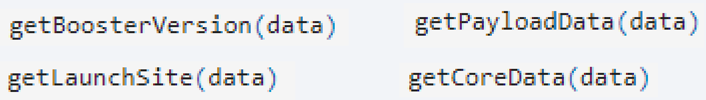
```
spacex url="https://api.spacexdata.com/v4/launches/past"
 response = requests.get(spacex_url)
```

2. Convert reponse to JSON file and create Dataframe



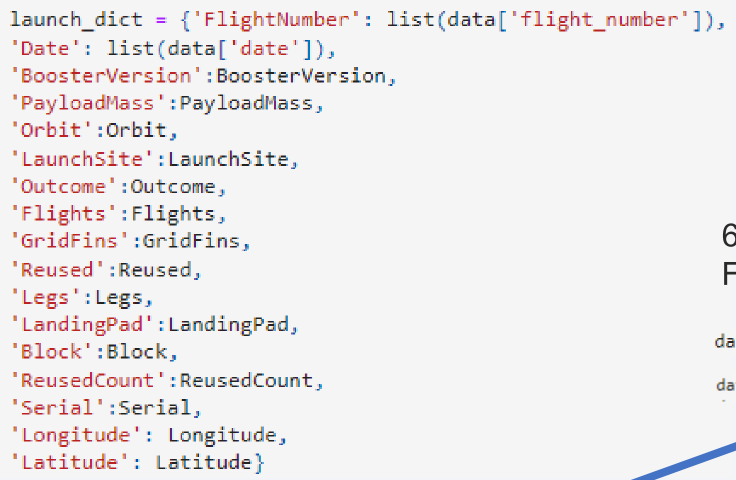
```
data = pd.json_normalize(response.json())
```

3. Apply custom functions to clean the data



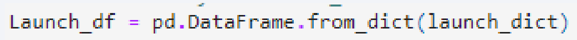
```
getBoosterVersion(data)      getPayloadData(data)

getLaunchSite(data)          getCoreData(data)
```

4. Construct dataset by coimbining columns into a dictionary



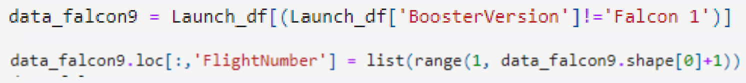
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5. Create Dataframe



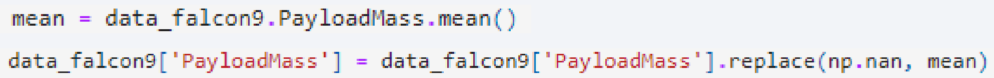
```
Launch_df = pd.DataFrame.from_dict(launch_dict)
```

6. Filter Dataframe to include only Falcon 9, reset Flight Number



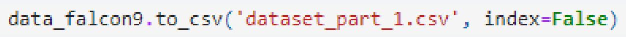
```
data_falcon9 = Launch_df[(Launch_df['BoosterVersion']!='Falcon 1')]

data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

7. Replace missing PayloadMass values with the mean



```
mean = data_falcon9.PayloadMass.mean()
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, mean)
```

8. Export to CSV file

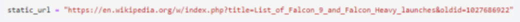


```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

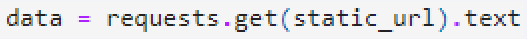
GitHub URL of completed SpaceX API calls notebook

Data Collection - Scraping

1. Request HTTP Response

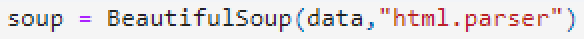


```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```



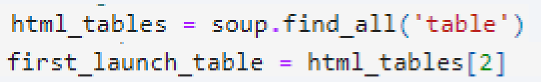
```
data = requests.get(static_url).text
```

2. Parse Data with Beautiful Soup



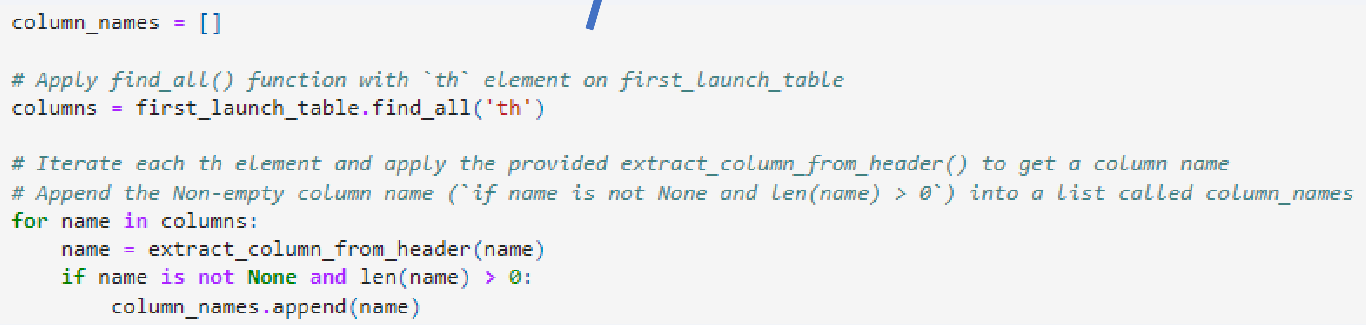
```
soup = BeautifulSoup(data,"html.parser")
```

3. Find all tables, select relevant table



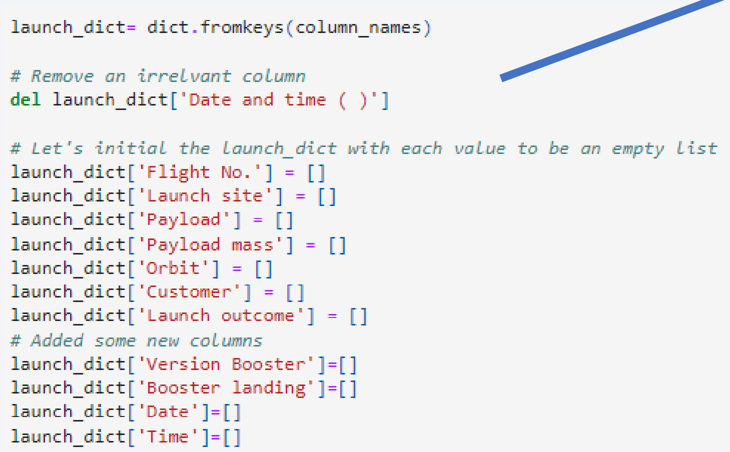
```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
```

4. Extract column names



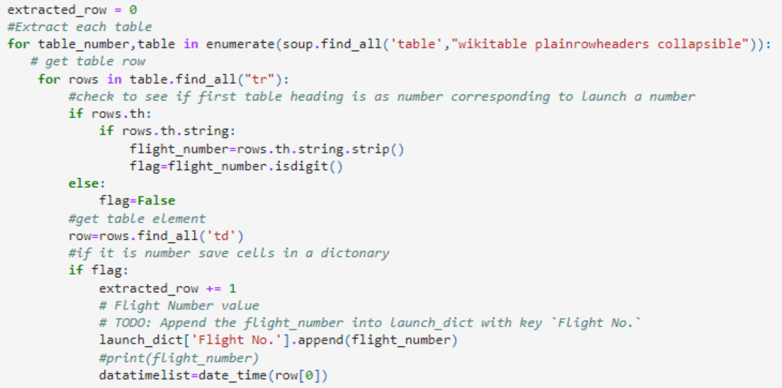
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
columns = first_launch_table.find_all('th')

# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for name in columns:
    name = extract_column_from_header(name)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

5. Create dictionary



```
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']=[]
```

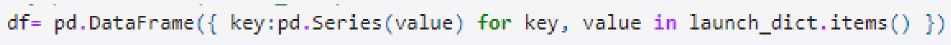
6. Add data to keys



```
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])
```

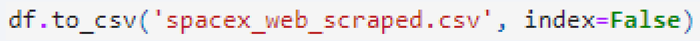
Refer to notebook for remainder of code

7. Create Dataframe



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

8. Export to CSV



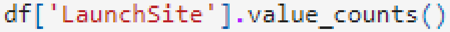
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

GitHub URL of completed web scraping notebook

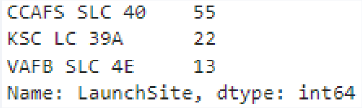
Data Wrangling

Goal: Transform string variables into binary categorical values where 1 = success and 0 = failure

1. Calculate number of launches on each site

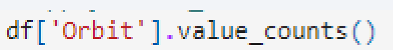


```
df['LaunchSite'].value_counts()
```

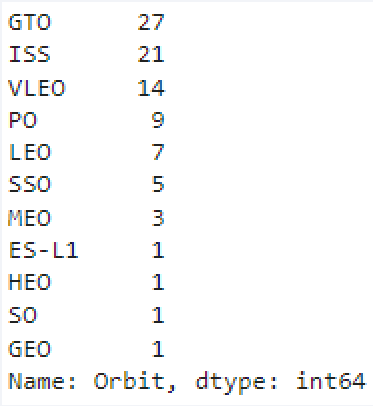


```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

2. Calculate number and occurrence of each orbit

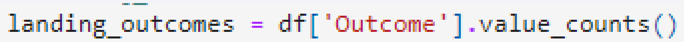


```
df['Orbit'].value_counts()
```

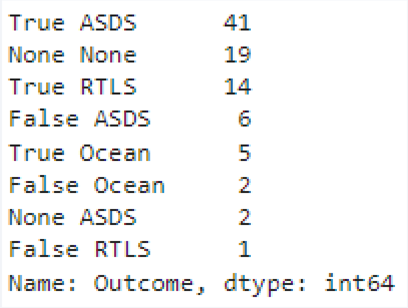


```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

3. Calculate number and occurrence of mission outcome per orbit type

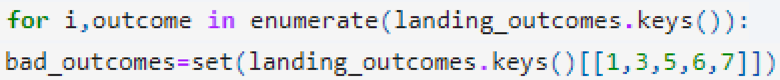


```
landing_outcomes = df['Outcome'].value_counts()
```



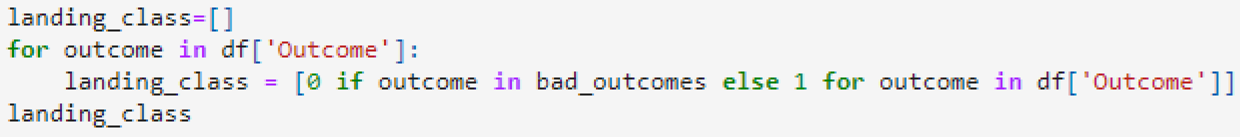
```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

4. Create outcomes lists



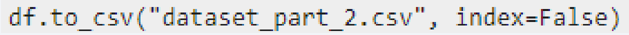
```
for i,outcome in enumerate(landing_outcomes.keys()):
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
```

5. Create a landing outcome label from outcome column



```
landing_class=[]
for outcome in df['Outcome']:
    landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
landing_class
```

6. Export to file

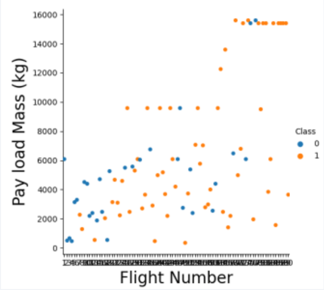


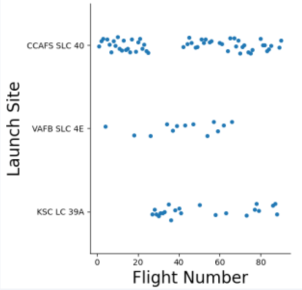
```
df.to_csv("dataset_part_2.csv", index=False)
```

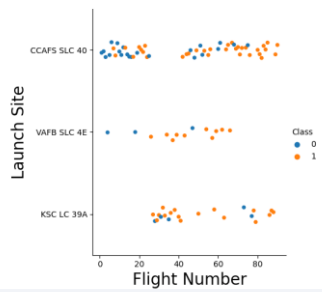
GitHub URL of completed data wrangling notebook

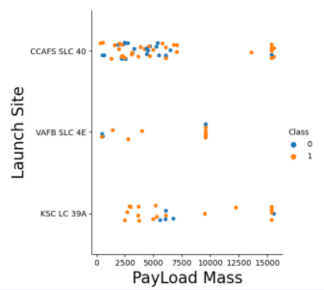
EDA with Data Visualization

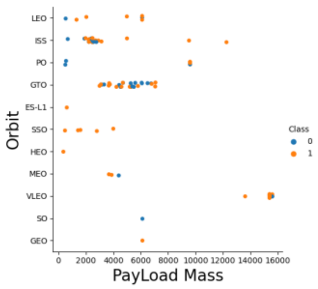
Scatter Graphs (visualize patterns and correlations between numeric variables)

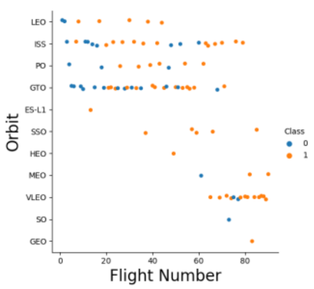


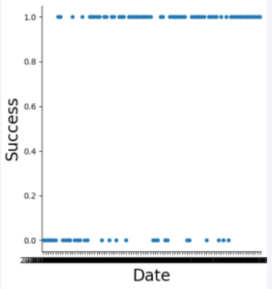




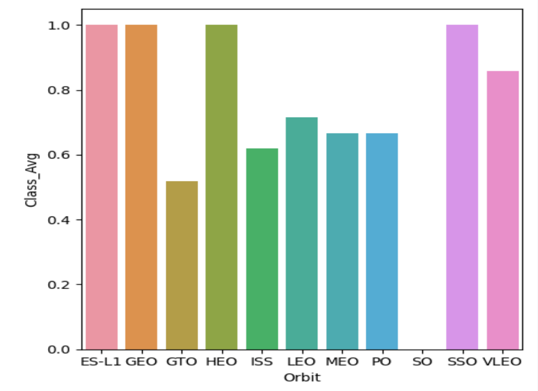




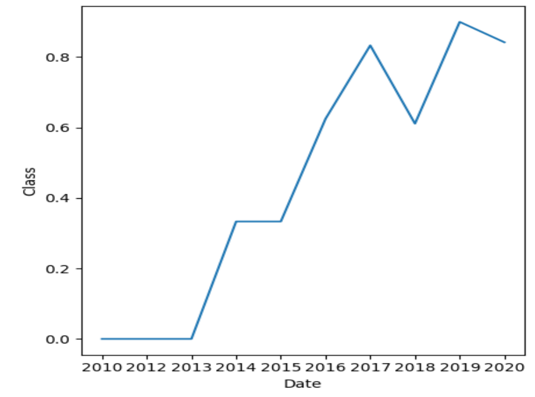




Bar Graphs (visualize relationship between numeric and categorical variables)



Line Graphs (visualize trend of numeric data over time)



GitHub URL of completed EDA with data visualization notebook

EDA with SQL

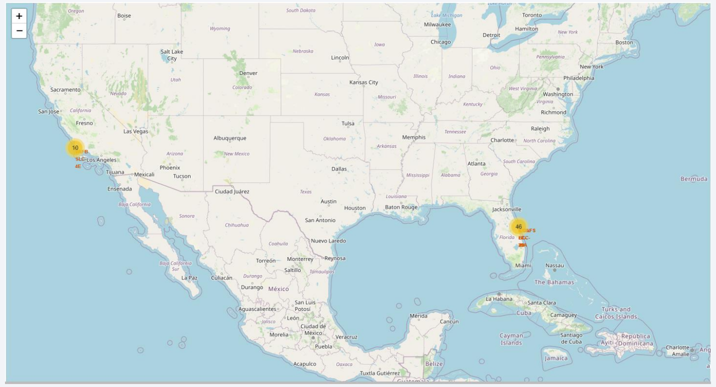
- SQL queries performed:

  1. Display the names of the unique launch sites in the space mission
  2. Display 5 records where launch sites begin with the string 'CCA‘
  3. Display the total payload mass carried by boosters launched by NASA (CRS)
  4. Display average payload mass carried by booster version F9 v1.1
  5. List the date when the first successful landing outcome in ground pad was acheived.
  6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  7. List the total number of successful and failure mission outcomes
  8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
  9. List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  10. Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order

GitHub URL of completed EDA with SQL notebook

Build an Interactive Map with Folium

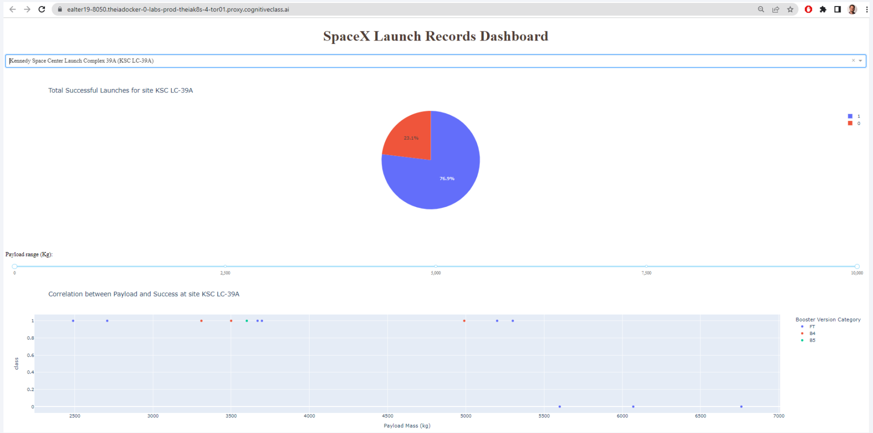
- Folium Map Objects created:
  - Circles: created a red circle at NASA Johnson Space center and created a red circle at each Launch Site using Latitude and Longitude
  - Markers: created markers for each launch and color coded them based on success/failure (green/red). Displayed popup with Launch Site names. Grouped markers in cluster when launches occurred at the same location.
  - Lines: Drew a line between a launch site and the coast to determine distance from the coast to the launch site



GitHub URL of completed interactive map with Folium map

Build a Dashboard with Plotly Dash

- The Dashboard includes the following:
  - Dropdown: Allows user to select Launch Site or all sites
  - Pie Chart: Displays percentage of successful launches vs failed launched for the selected launch site
  - Range Slider: Allows user to select payload mass (kg) range to view in chart. They can then observe changes among high/low mass payloads.
  - Scatter Plot: Allows user to see the correlation between Payload Mass, Launch Outcome (success or failure), and Booster Version.



ealter19-8050.theiadocker-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai

SpaceX Launch Records Dashboard

Kennedy Space Center Launch Complex 39A (KSC LC-39A)

Total Successful Launches for site KSC LC-39A

Payload range (Kg):

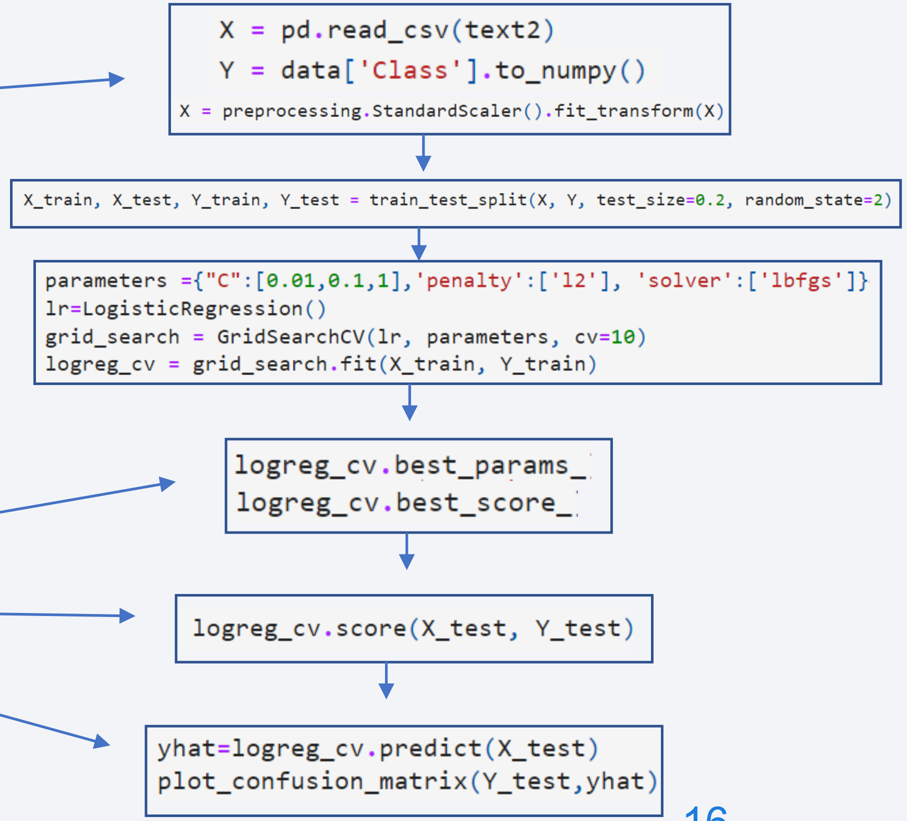
Correlation between Payload and Success at site KSC LC-39A

GitHub URL of completed Plotly Dash lab

Predictive Analysis (Classification)

- Predictive Analysis Methodology
  - Build
    - Preprocessing (Load dataframe, Standardize data)
    - Split data into Training and Testing sets
    - Set possible method parameters (varies by method: Logistic Regression, Support Vector Machine, Decision Tree Classifier, and K Nearest Neighbors). Define method, find best parameters using training data.
  - Evaluate
    - Identify best parameters, calculate accuracy (R2) based on training data
    - Calculating the accuracy (R2) for each method based on the testing data
    - Assess Confusion Matrices to determine what problems each method has
    - Determine the method that performs best

Flowchart example for Logistic Regression



```
X = pd.read_csv(text2)
Y = data['Class'].to_numpy()
X = preprocessing.StandardScaler().fit_transform(X)
```

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

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
lr=LogisticRegression()
grid_search = GridSearchCV(lr, parameters, cv=10)
logreg_cv = grid_search.fit(X_train, Y_train)
```

```
logreg_cv.best_params_
logreg_cv.best_score_
```

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

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

Replace logreg_ with svm_,
tree_, or knn for other methods
and adjust parameters

[GitHub URL of completed predictive analysis lab]

Results

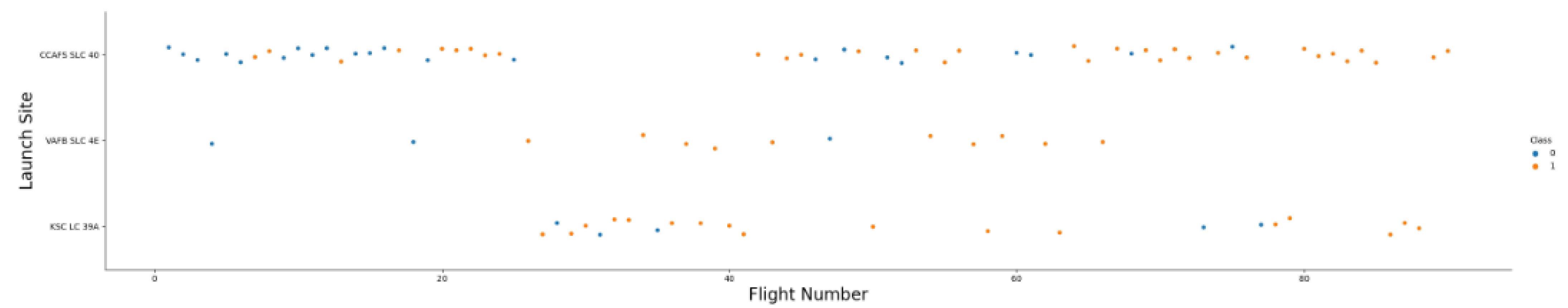
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

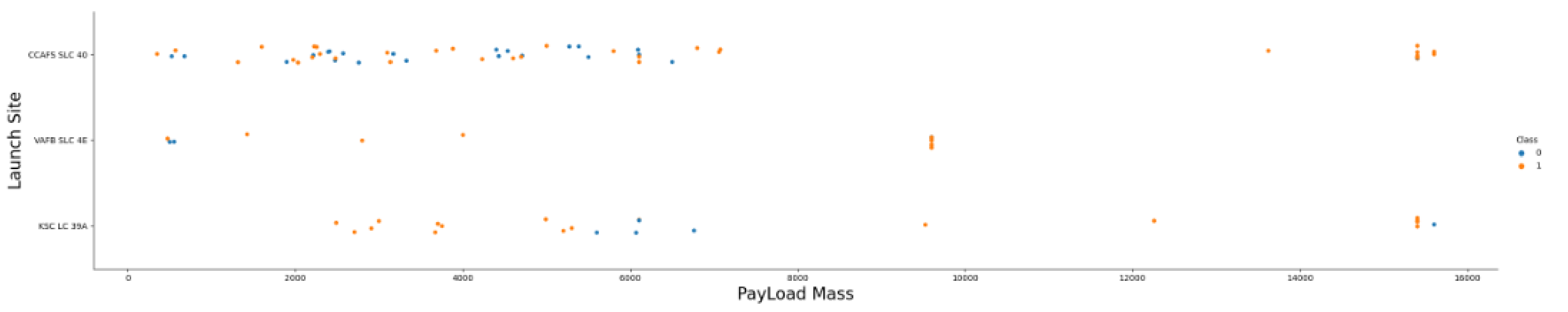
- Show a scatter plot of Flight Number vs. Launch Site



- For each Launch Site, there seems to be more successful launches as Flight Number increases

Payload vs. Launch Site

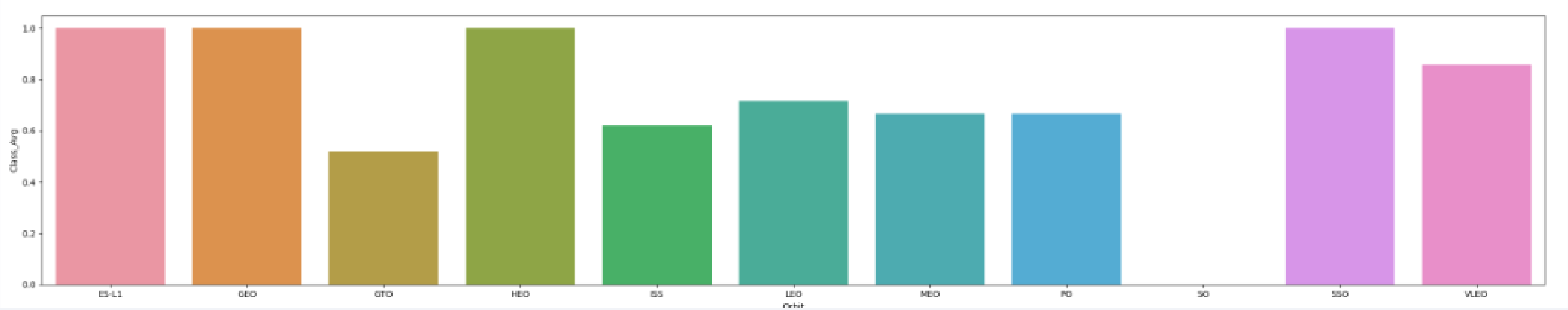
- Show a scatter plot of Payload vs. Launch Site



- For each Launch Site, there were more failures with lighter payloads, though heavier payloads may be more recent and have better success due to other learnings

Success Rate vs. Orbit Type

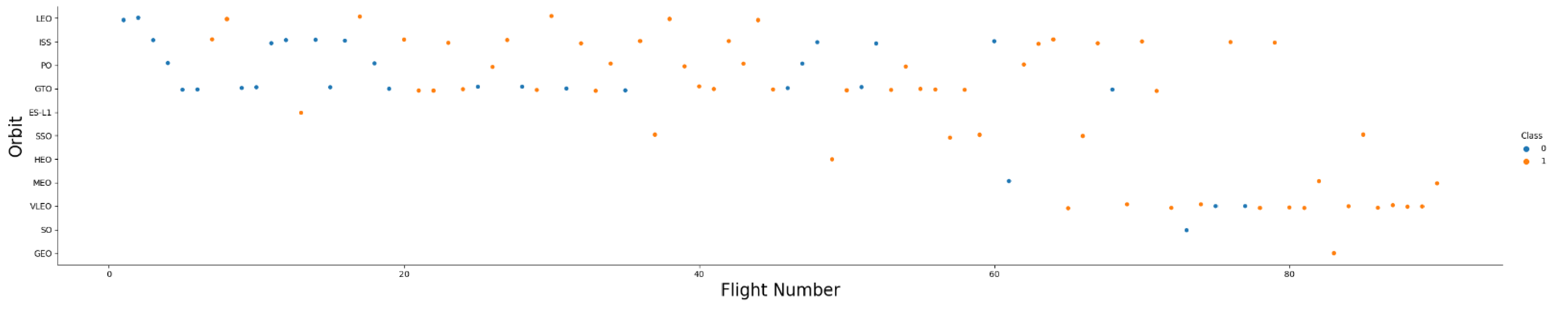
- Show a bar chart for the success rate of each orbit type



- ES-L1, GEO, HEO, and SSO have the highest success rates

Flight Number vs. Orbit Type

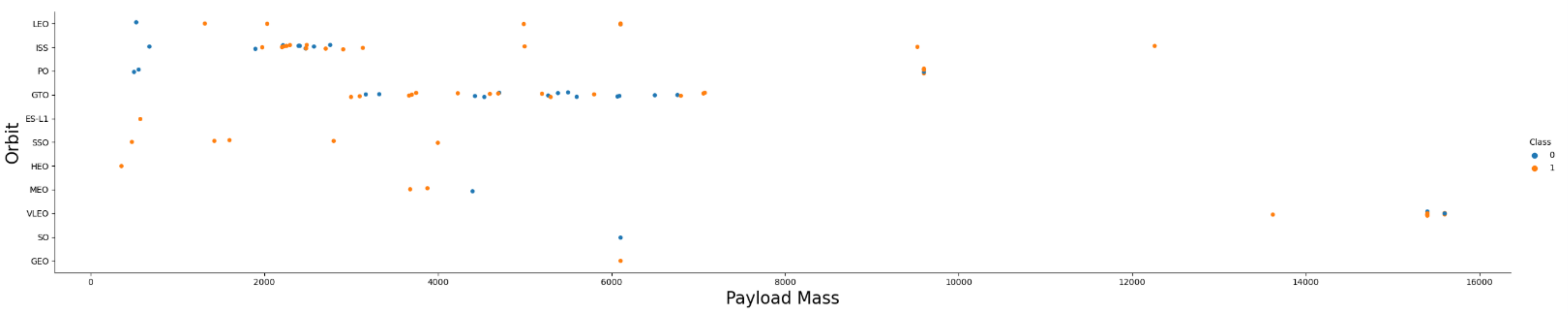
- Show a scatter point of Flight number vs. Orbit type



- Success rates seem to improve for each Orbit type as Flight Number increases. This could be due to the SpaceX team learning more with each successive flight.

Payload vs. Orbit Type

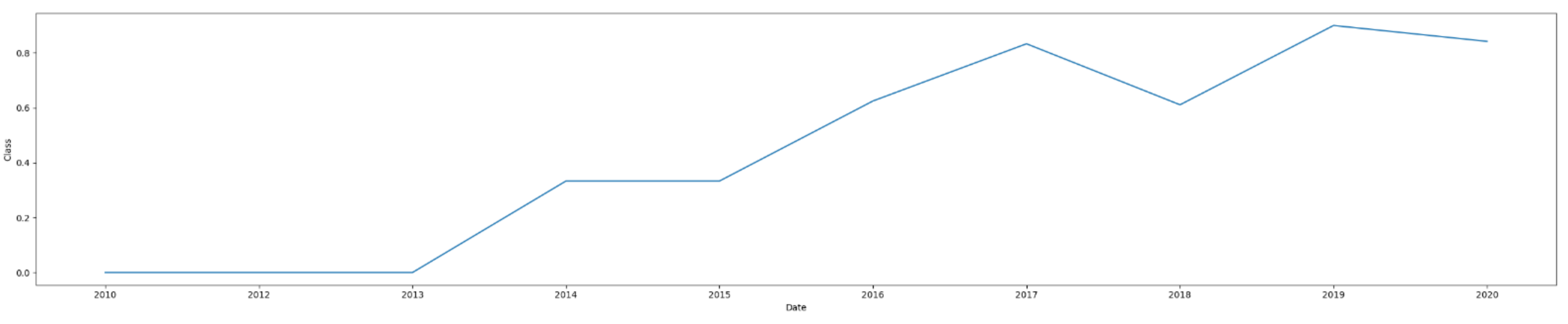
- Show a scatter point of payload vs. orbit type



- For some orbits, lighter payloads have been correlated with more failures, but that could be attributed to other factors such as experience

Launch Success Yearly Trend

- Show a line chart of yearly average success rate

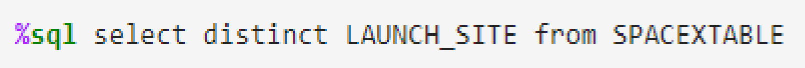


Year	Avg Success Rate
2010	0%
2012	0%
2013	0%
2014	33%
2015	33%
2016	63%
2017	83%
2018	61%
2019	90%
2020	84%

- Success rate has been trending up from 2013-2020, and has reached 90% in 2019 and 84% in 2020.

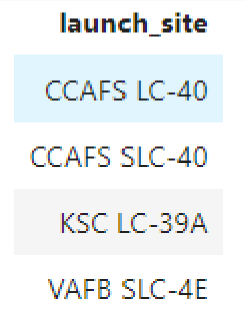
All Launch Site Names

SQL Query



```
%sql select distinct LAUNCH_SITE from SPACEXTABLE
```

Results



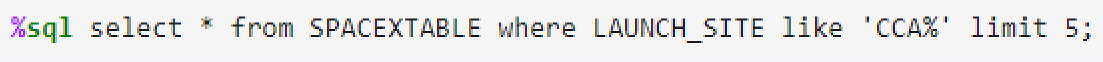
launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Explanation

Distinct returns unique names from the Launch_Site column of the SpaceXTable

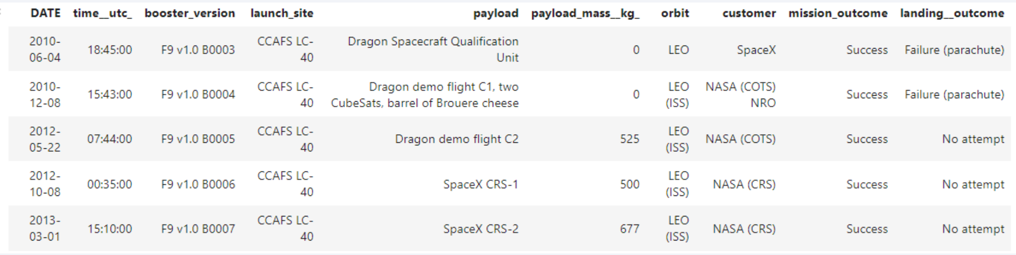
Launch Site Names Begin with 'CCA'

SQL Query



```
%sql select * from SPACEXTABLE where LAUNCH_SITE like 'CCA%' limit 5;
```

Results



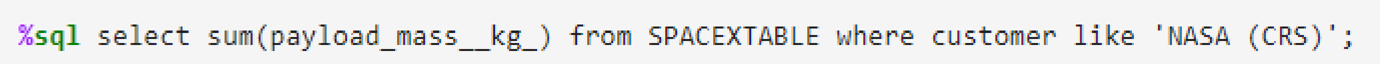
DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight 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

The asterisk selects all entries that meet the rest of the criteria. “where” and “like” ‘CCA%’ filters entries that being with ‘CCA’ in LAUNCH_SITE. The ‘%’ indicates anything may follow. “limit 5’ selects only the first 5 entries.

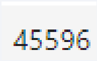
Total Payload Mass

SQL Query



```
%sql select sum(payload_mass__kg_) from SPACEXTABLE where customer like 'NASA (CRS)';
```

Results



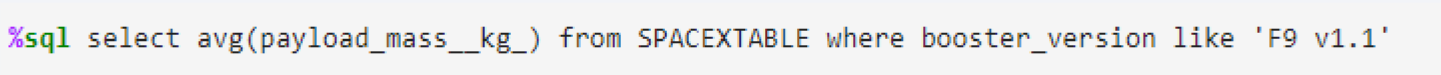
```
45596
```

Explanation

This sums all payload masses where the customer is NASA (CRS)

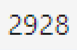
Average Payload Mass by F9 v1.1

SQL Query



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

Results



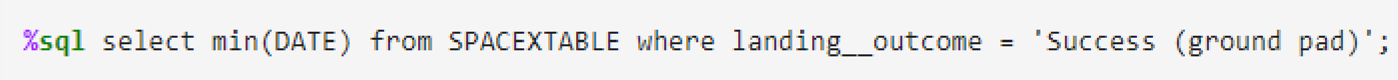
```
2928
```

Explanation

This averages payload mass for the specific booster version indicated

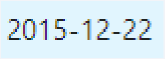
First Successful Ground Landing Date

SQL Query



```
%sql select min(DATE) from SPACEXTABLE where landing__outcome = 'Success (ground pad)';
```

Results



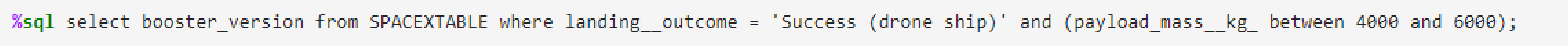
```
2015-12-22
```

Explanation

Returns earliest date Ground Pad has sucess

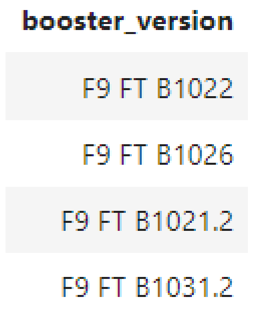
Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query



```
%sql select booster_version from SPACEXTABLE where landing__outcome = 'Success (drone ship)' and (payload_mass__kg_ between 4000 and 6000);
```

Results



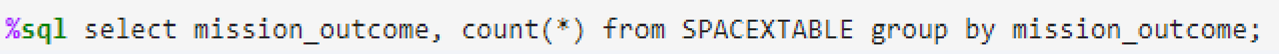
booster_version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Explanation

The “where” and “and” terms filter the returns. The “between” term filters to values between 4000 and 6000.

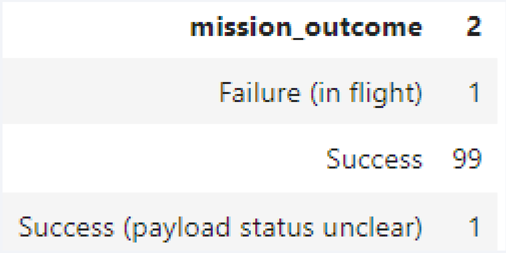
Total Number of Successful and Failure Mission Outcomes

SQL Query



```
%sql select mission_outcome, count(*) from SPACEXTABLE group by mission_outcome;
```

Results



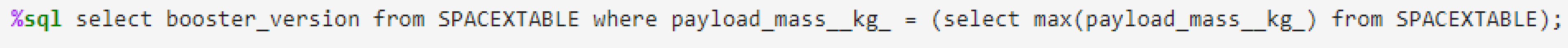
mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Explanation

Returns the count of each possible mission outcome

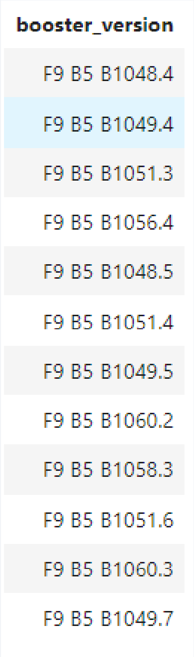
Boosters Carried Maximum Payload

SQL Query



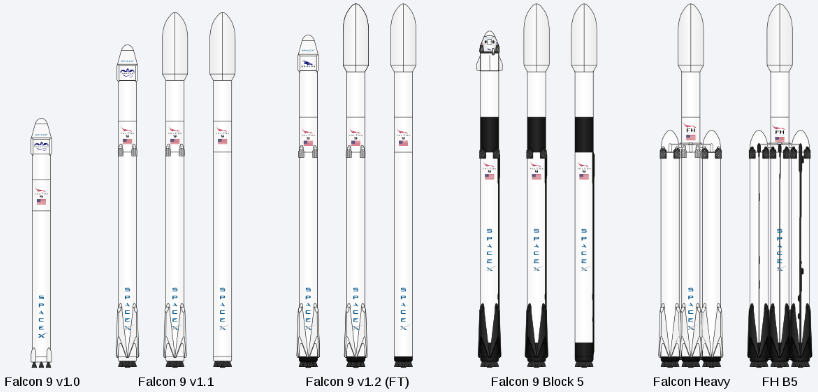
```
%sql select booster_version from SPACEXTABLE where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTABLE);
```

Results



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

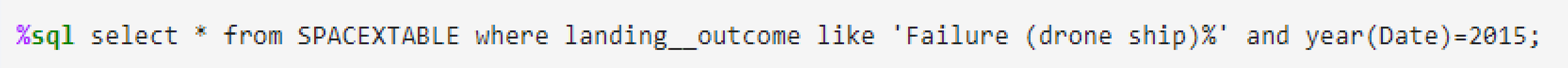
Explanation



“max” term finds the highest payload mass. Parenthesis show subquery. Max Payload is 15600kg and 12 booster versions carried that weight.

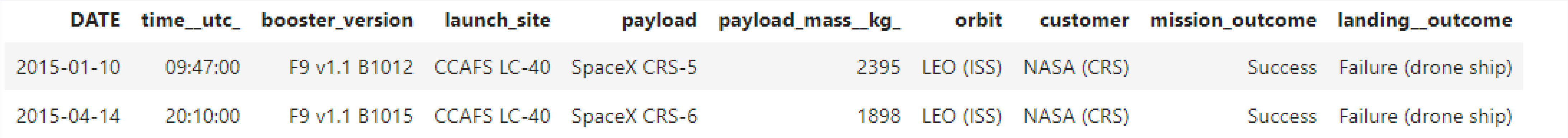
2015 Launch Records

SQL Query



```
%sql select * from SPACEXTABLE where landing__outcome like 'Failure (drone ship)%' and year(Date)=2015;
```

Results



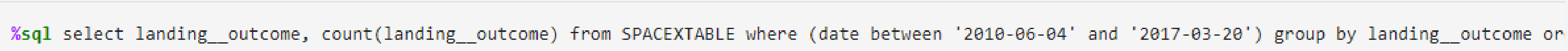
DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2015-01-10	09:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)

Explanation

Returns failed landing outcomes for the Drove Ship in 2015 only

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

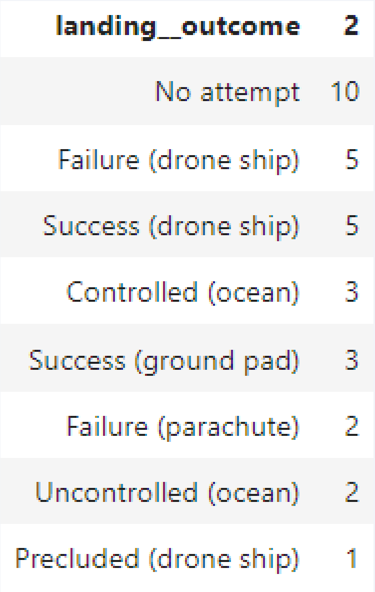
SQL Query



```
%sql select landing__outcome, count(landing__outcome) from SPACEXTABLE where (date between '2010-06-04' and '2017-03-20') group by landing__outcome or
```

%sql select landing__outcome, count(landing__outcome) from SPACEXTABLE where (date between '2010-06-04' and '2017-03-20') group by landing__outcome order by count(landing__outcome) desc;

Results



landing__outcome	2
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

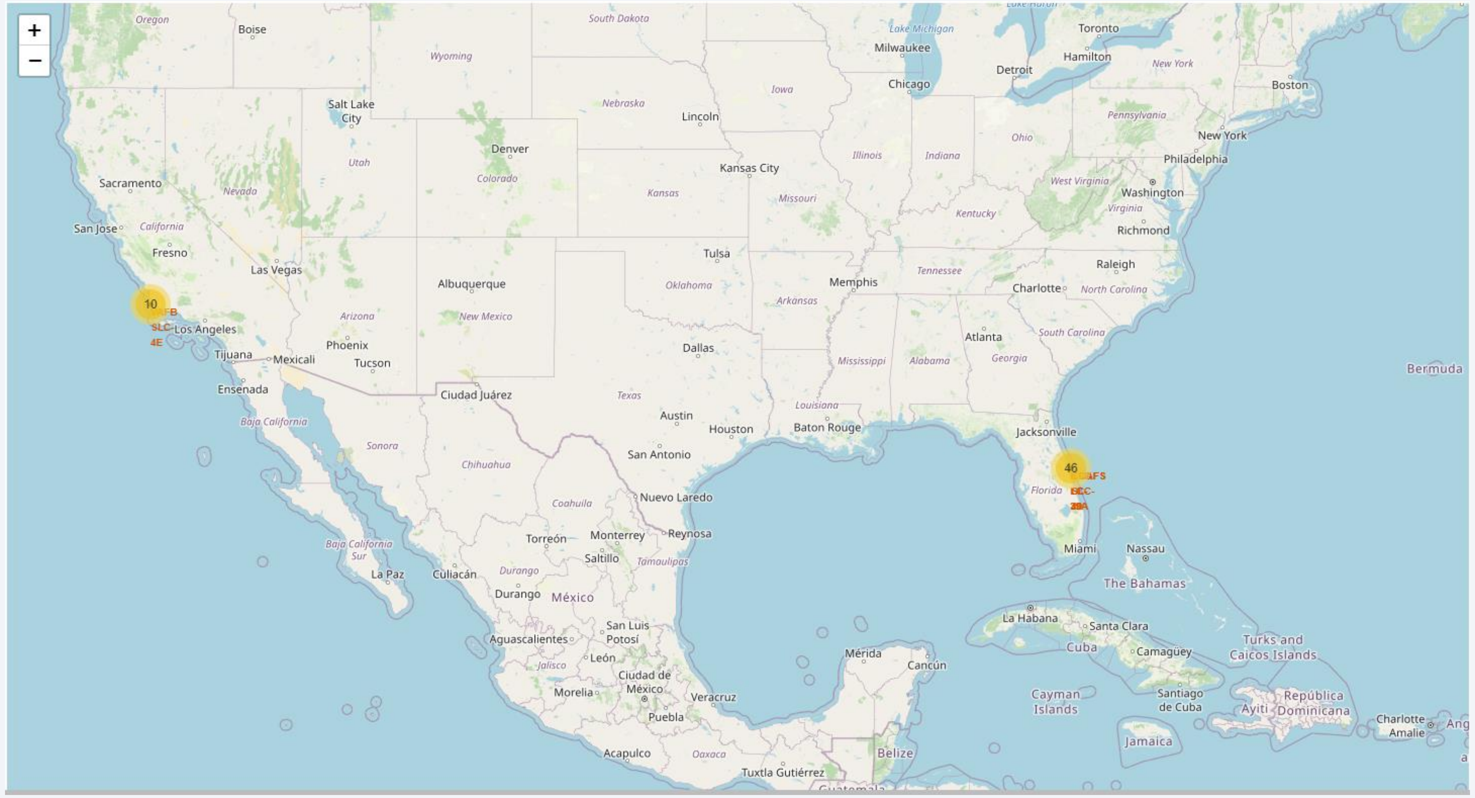
Explanation

The “Group By” term groups results by landing outcome

Section 3

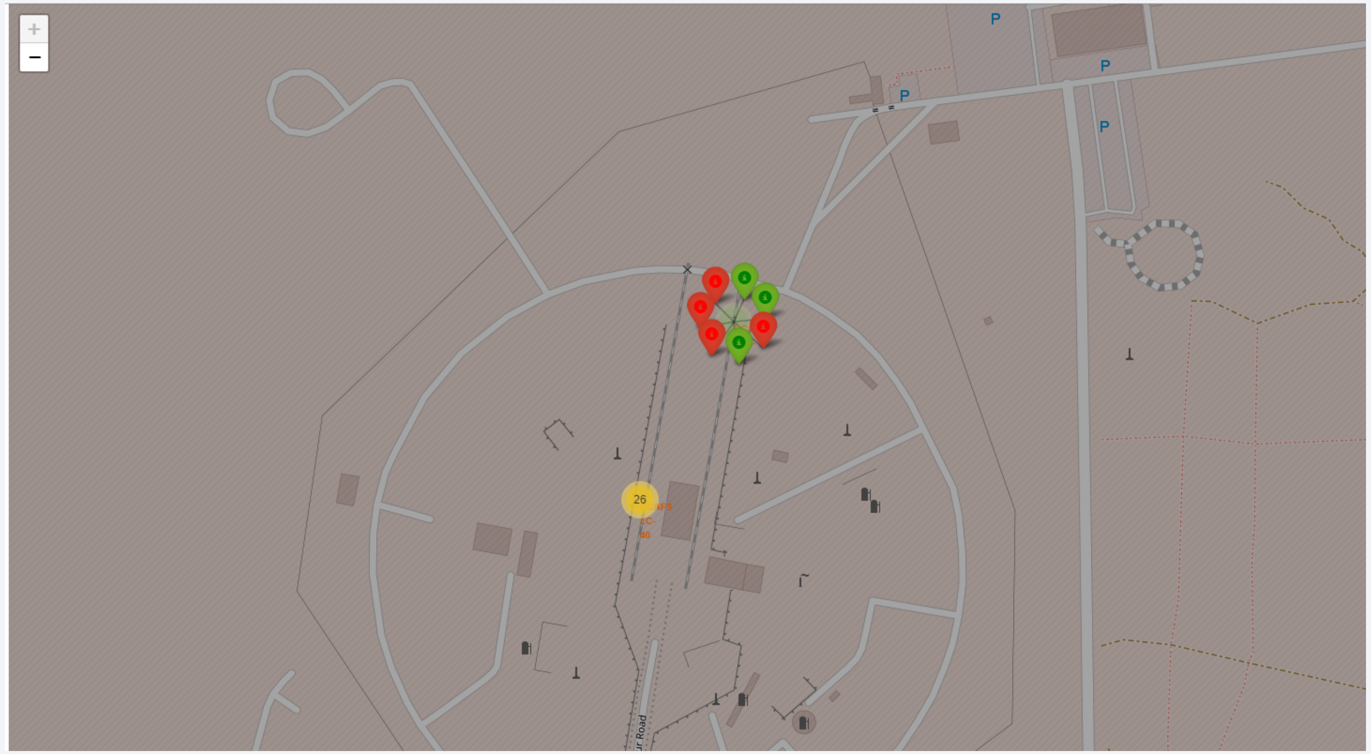
Launch Sites Proximities Analysis

Folium Map of Launch Sites



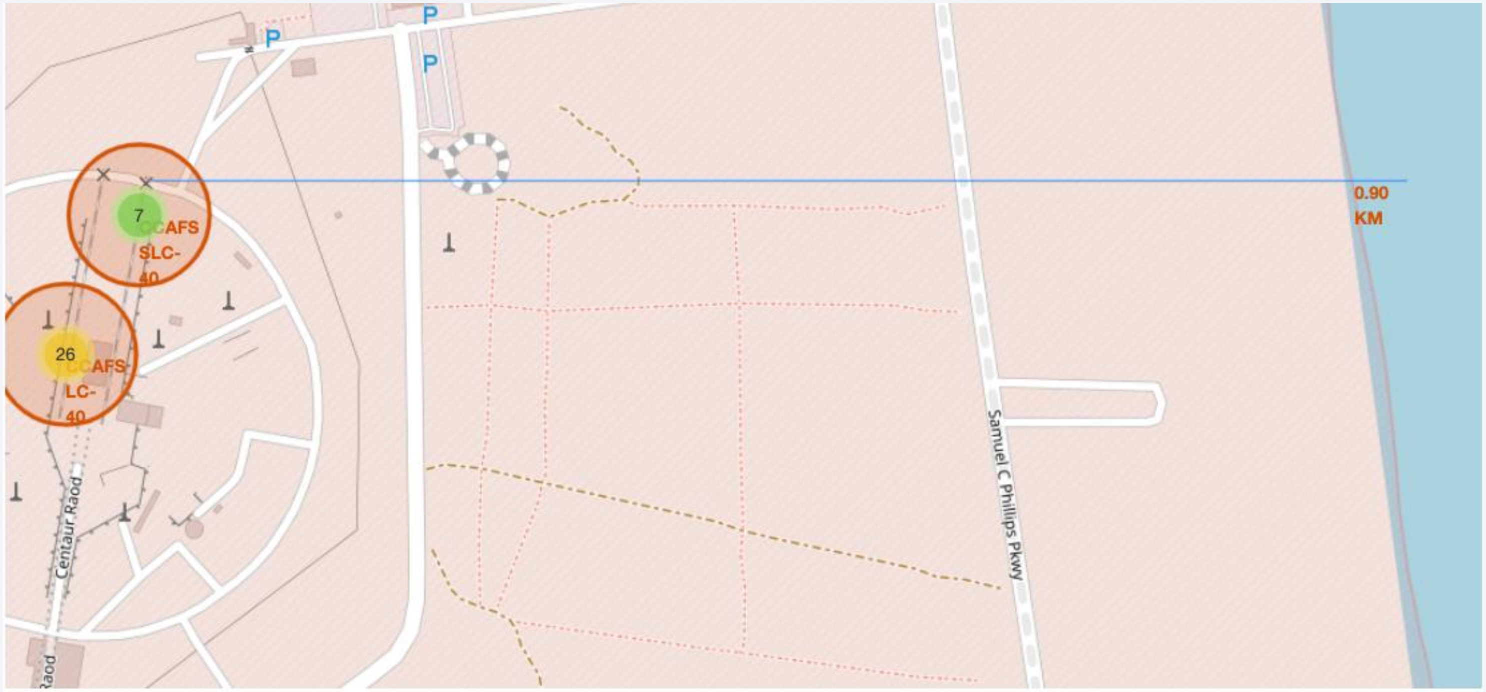
This map shows all launch sites, and clusters together those that are in close proximity with one another

Folium Map Marker Clusters



This map is zoomed in on the Cape Canaveral Launch Sites. The yellow circle with “26” indicates that 26 launches happened at that spot0, and are clustered together for ease of viewing. To the north east of that circle there was a yellow circle marked with a “7”, indicating that 7 launches happened at that spot. In the screenshot above, I’ve clicked on that circle to reveal the 7 individual launches as their own markers. Each marker is colored either green for success or red for failure.

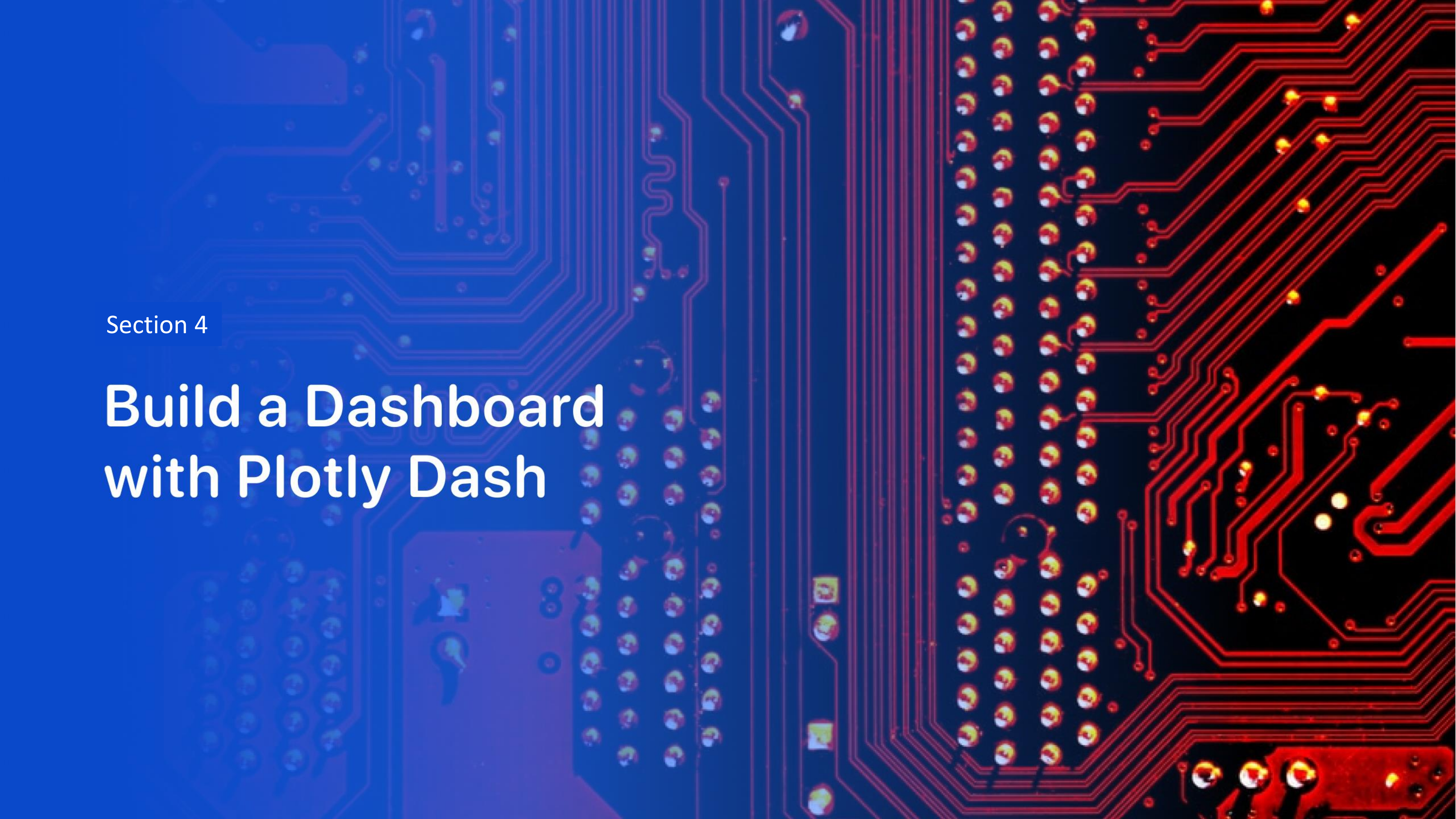
Folium Map Distances



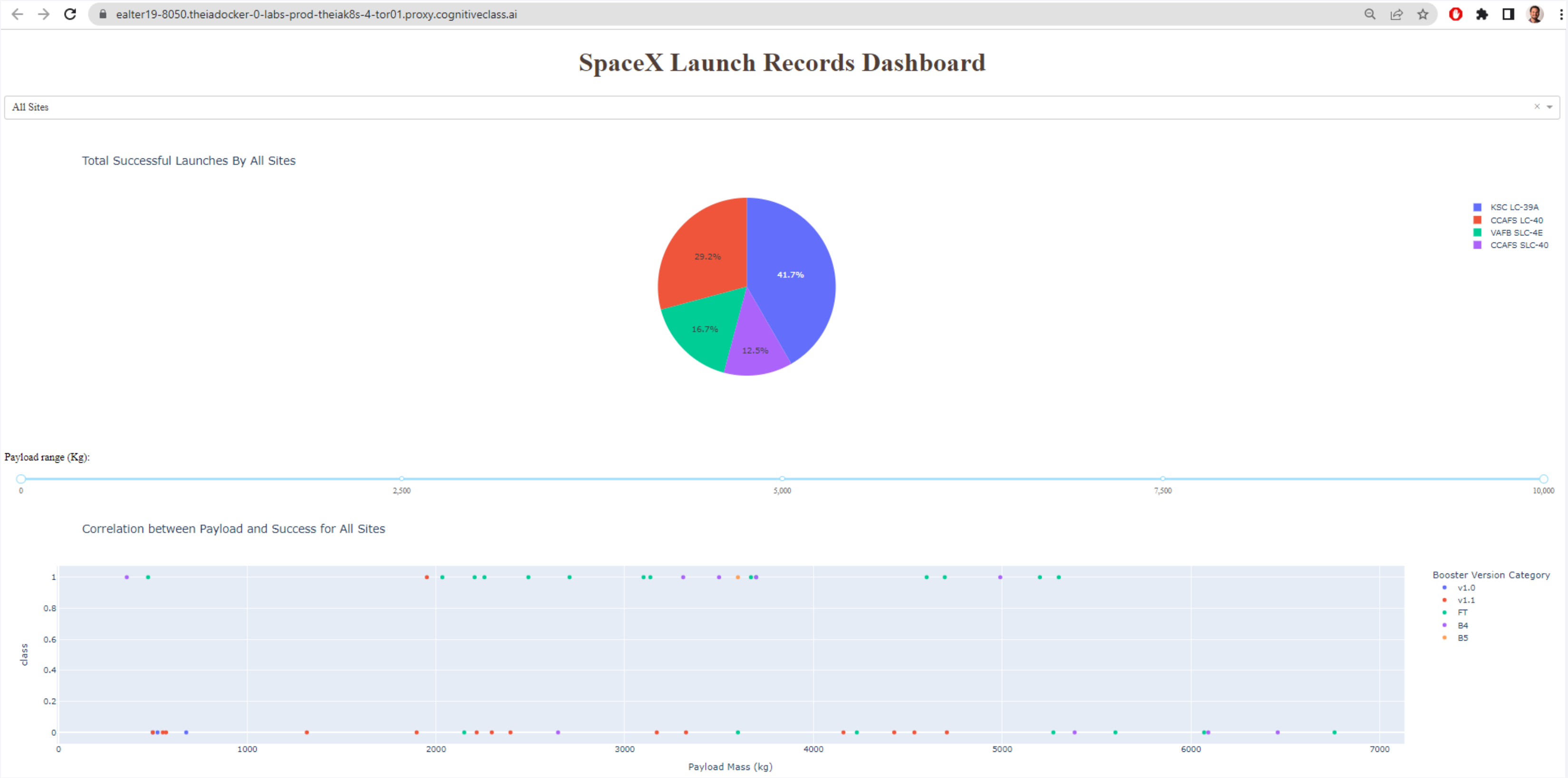
This screenshot show two clusters of launches on the left, and a line shows the distance between the CCAFS SLC-40 site and the coast, and indicates that it is 0.90 KM away.

Section 4

Build a Dashboard with Plotly Dash



Successful Launches by Launch Site in Dashboard



ealter19-8050.theiadocker-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai

SpaceX Launch Records Dashboard

All Sites

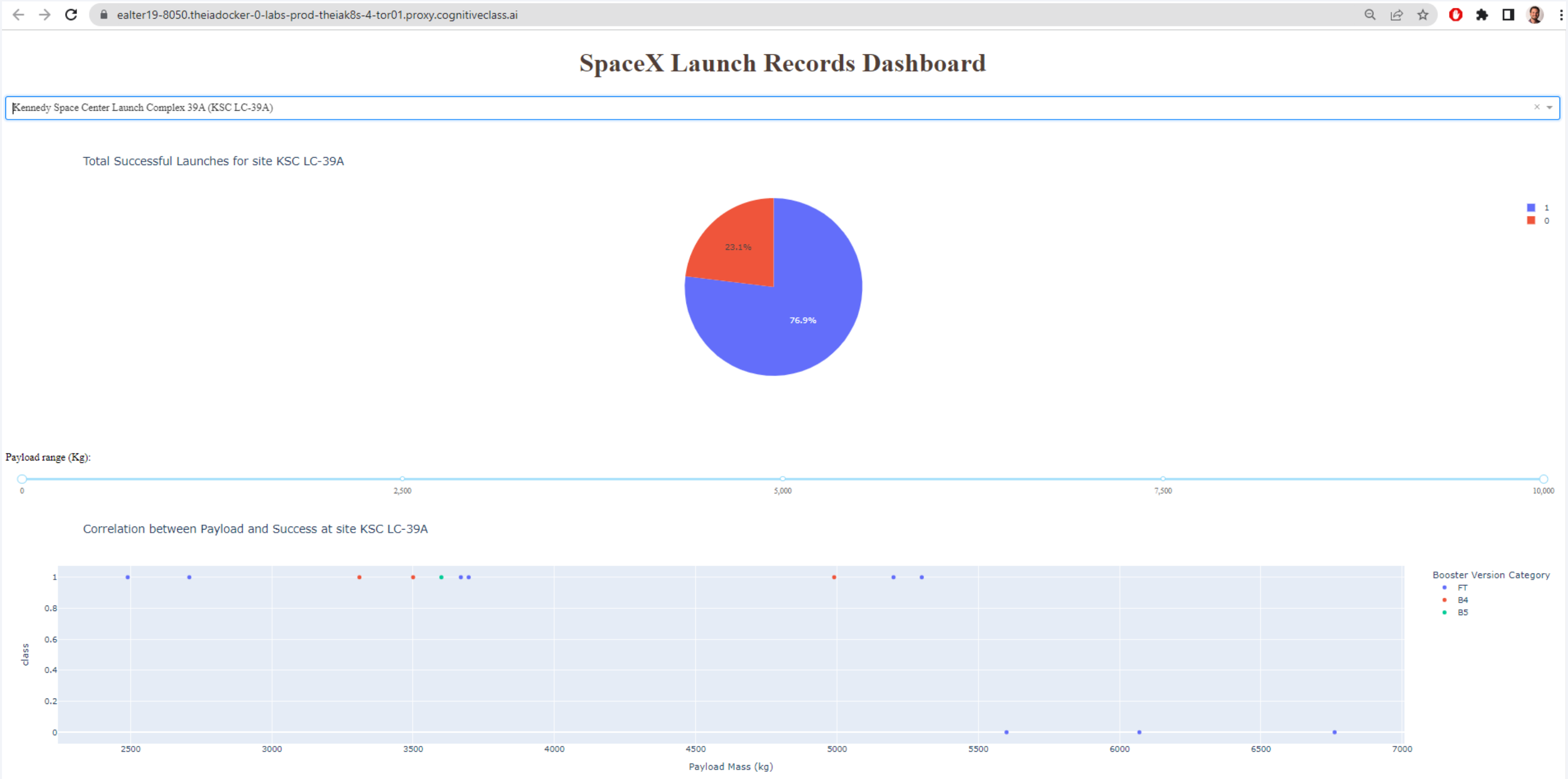
Total Successful Launches By All Sites

Payload range (Kg):

Correlation between Payload and Success for All Sites

In this dashboard, the dropdown is filtered to “All Sites” and the Pie Chart is showing the breakdown of successful launches across the 4 Launch Sites, for the entire Payload Range

KSC LC-39A has the highest success rate



ealter19-8050.theiadocker-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai

SpaceX Launch Records Dashboard

Kennedy Space Center Launch Complex 39A (KSC LC-39A)

Total Successful Launches for site KSC LC-39A

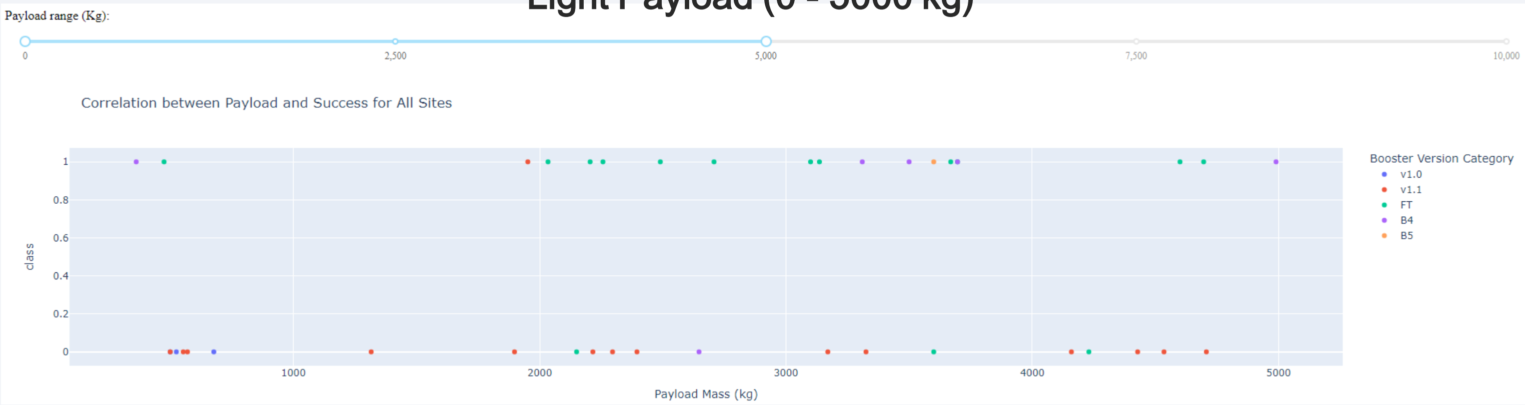
Payload range (Kg):

Correlation between Payload and Success at site KSC LC-39A

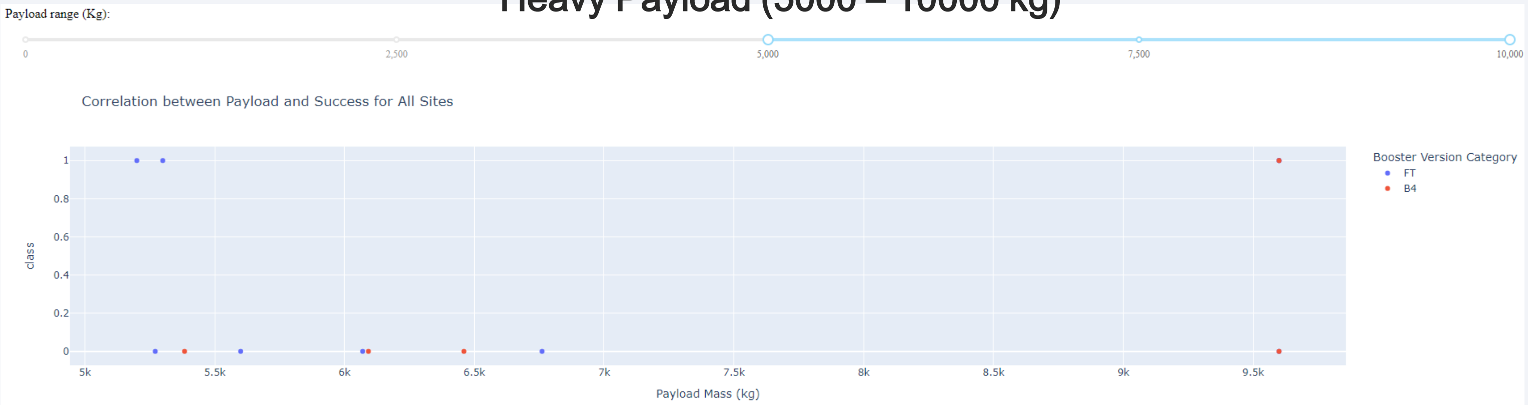
Kennedy Space Center Launch Complex 39A (KSC LC-39A) had a success rate of 76.9%, which is higher than any other launch site measured in this analysis.

Payload vs Launch Outcomes for all sites

Light Payload (0 - 5000 kg)



Heavy Payload (5000 – 10000 kg)



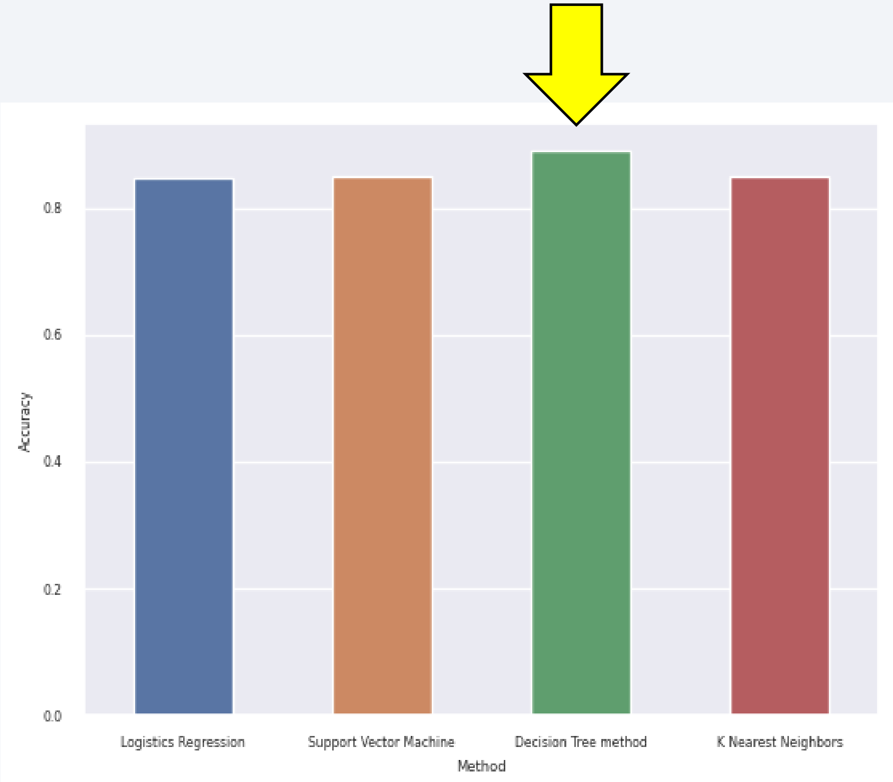
The top chart uses the range slider to show payload mass 0-5000 kg for all booster types. The bottom chart show the same but for 5000-1000 kg. For lighter payloads, FT seems to have a very high success rate, while v1.1 has a low success rate. For heavy payloads, both booster types have low success rates.

Section 5

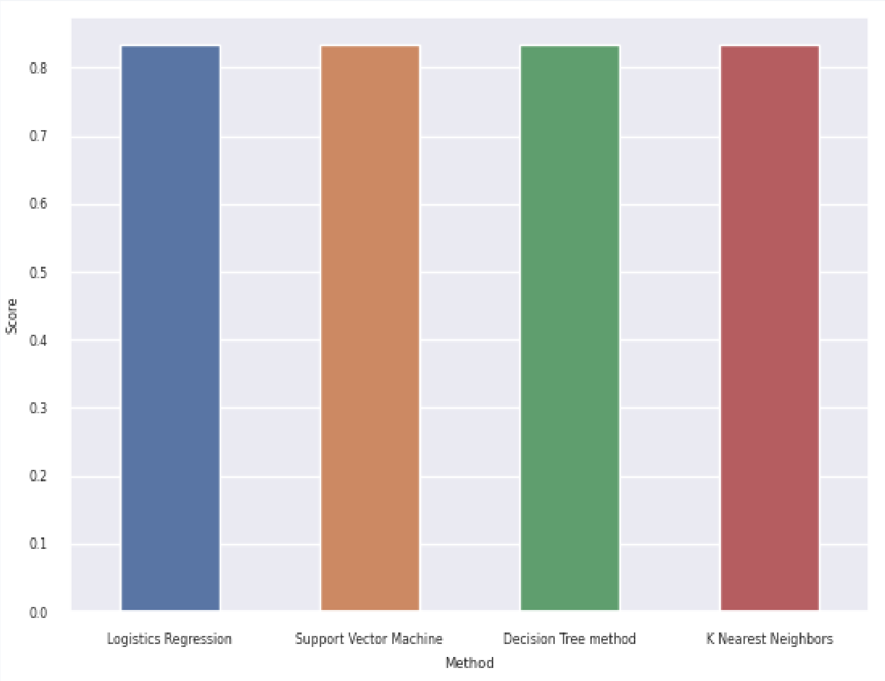
Predictive Analysis (Classification)

Classification Accuracy

Training Accuracy across Methods



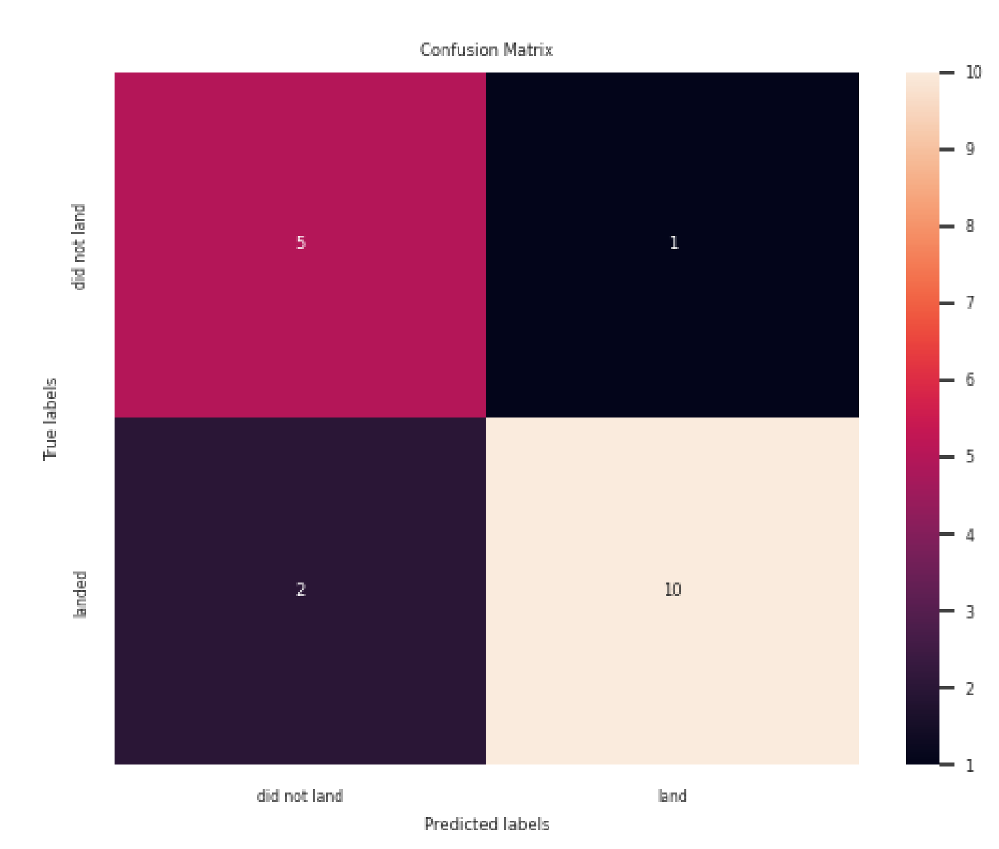
Testing Accuracy across Methods



The Decision Tree method had the highest accuracy when training the model. It had an R2 of 89%, while the other 3 models had an R2 of 85%. All 4 models had the same R2 in the testing phase of 83%.

Confusion Matrix

- The Decision Tree Method is good at detecting True Positives in the lower left quadrant and True Negatives in the upper left quadrant. However, it did register a small number of False Positives in the upper right quadrant and False Negatives in the lower left quadrant.



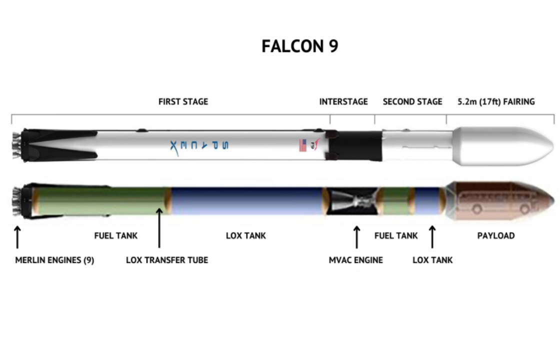
Conclusions

- The Decision Tree method is the best option for predicting success rates. It had an R2 score of 89% in training and 83% in testing. The 3 other methods (Logistic Regression, Support Vector Machine, and K Nearest Neighbor) all performed similarly with a training R2 of 85% and testing R2 of 83%.
- The Kennedy Space Center Launch Complex 39A (KSC LC-39A) is the launch site with the highest success rate by far (76.9%). Cape Canaveral Launch Complex 40 (CAFS LC-40) has the lowest success rate (26.9%).
- Vandenberg Air Force Base Space Launch Complex (VAFB SLC-4E) (42.9%) and Cape Canaveral Space Launch Complex 40 (CCAFS SLC-40 (40.0%) are similar in success rates (42.9% and 40.0% respectively).
- Heavy Payloads (>5,000kgs) have a low success rate at 35.7% compared to Lighter Payloads (<5,000kg) at 45.2%.
- Booster Version FT has a high success rate (66.7%). Booster Versions v1.0 and v1.1 had very low success rates (5.0% combined). B4 and B5 combined had a 58.3% success rate.
- SpaceX Falcon 9 Success Rates improve each year as their team learns new lessons. Their success rate reached up to 90% in 2019 and 84% in 2020.
- KSC LC-39A and VAFB SLC-4E have had more success with heavy payloads (>5000kg) than CCAFS LC-40 and CCAFS SLC-40.
- The orbits ES-L1, GEO, HEO, and SSO have the highest success rates
- The success rate of landing on a drone ship (74%) is similar to landing on a ground pad (75%)

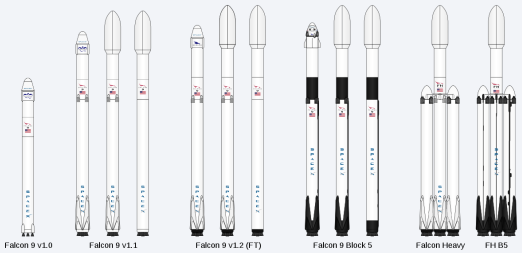
Appendix

- GitHub Repository Link: https://github.com/ealter19/Applied-Data-Science-Capstone

- Applied your creativity to improve the presentation beyond the template
  - Displayed additional graphics to help reader understand rockets



- Displayed any innovative insights
  - The success rate of landing on a drone ship (74%) is similar to landing on a ground pad (75%)



Thank you!