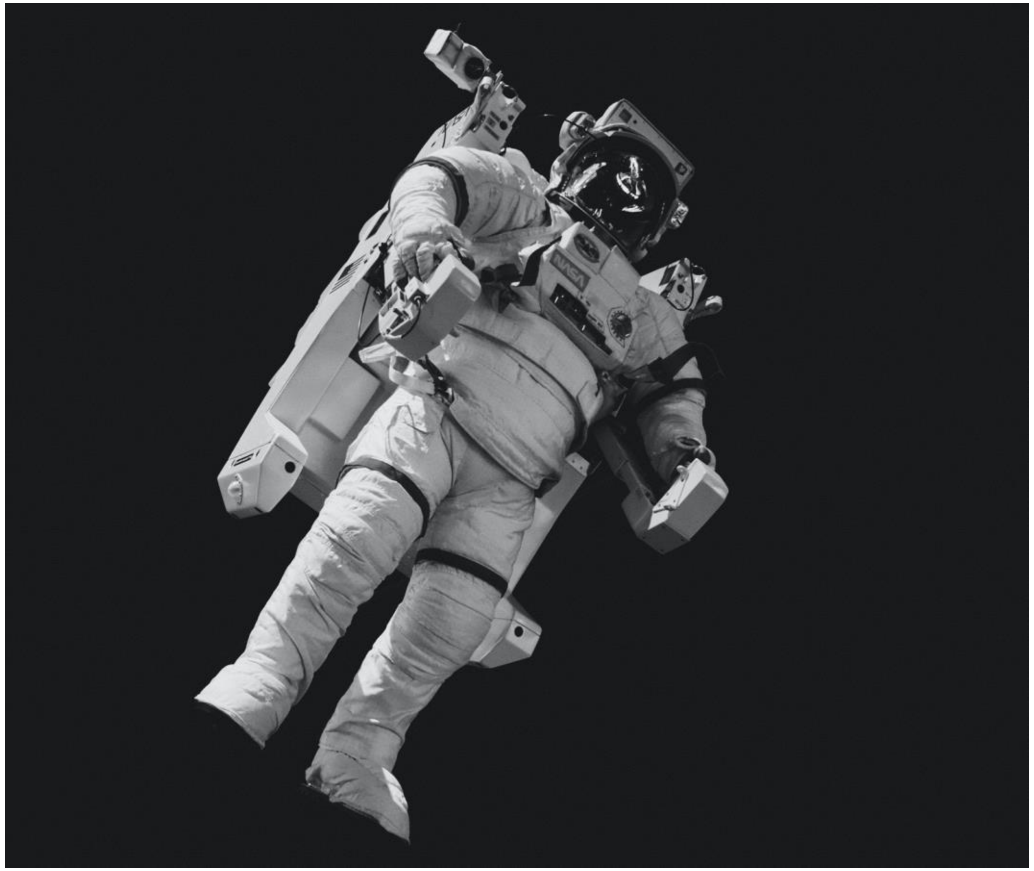
Data Science Capstone Project

SpaceX Falcon 9 First Stage Landing Prediction

Khadija Hammawa

26-08-2021

Outline



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

Executive Summary

- Summary of methodologies
  - Data Collection
    - Requests to the SpaceX API to acquire information on Falcon 9 rocket launches.
    - Web scraping Falcon 9 and Falcon Heavy launch records from Wikipedia.
  - Data Wrangling
    - Exploratory Data Analysis (EDA) to find patterns in SpaceX data related to the landing outcome of Falcon 9 first stage.
    - Convert landing outcomes into label for training supervised models.
  - Exploratory Data Analysis
    - Visualize detailed records for each launch site.
    - Select features that will be used in success prediction.
  - Interactive Visual Analytics and Dashboard
    - Build a dashboard to analyze launch records interactively with Plotly Dash.
    - Build an interactive map to analyze the launch site proximity with Folium.
  - Predictive Analysis
    - Split the data into training testing data and train different classification models (i.e., Logistic Regression, SVM, Decision Tree Classifier, and KNN)
    - Use GridSearchCV to determine best parameters
    - Calculate accuracy for each model to identify the most accurate model

Executive Summary

- Summary of all results
  - Cape Canaveral Space Launch Complex 40 (CCAFS SLC 40) had the most launches
  - Most launches were geosynchronous orbit (GTO)
  - Success of first stage launches across all launch sites was 66.67%
  - Total payload mass carried by boosters launched by NASA (CRS) was 111,268 kg’s
  - Average payload mass carried by booster version F9 v1.1 was 2,534 kg
  - The first successful landing outcome in ground pad was achieved on December 22nd, 2015
  - 12 Booster Versions were determined to have carried the maximum payload mass
  - When payload is over 10,000 kg's almost all the first stage landings are successful
  - ES-L1, GEO, HEO, and SSO Orbits had the highest success rates
  - Heavy payloads have a negative influence on GTO orbits and positive influence on GTO and Polar LEO (ISS) orbits
  - Success rate of launches have continually increased since 2013
  - Flight Number, Payload Mass, Orbit, Launch Site, Flights, Grid Fins, Reused Legs, Landing Pad, Block, and Reused Count were determined the be the features more important in predicting landing outcome
  - The Kennedy Space Center Launch Complex 39A was shown to have the highest success ratio for landing outcomes
  - All models performed the same with an accuracy score of 83.33%

Introduction

- Project Background
  - The age of commercial space travel is no longer a far distant dream for humanity, instead, companies are now making space travel affordable for everyone. Companies such as Virgin Galactic, Rocket Lab, and Blue Origin are at the forefront of these suborbital spaceflight developments. One company that has had exceptional success is SpaceX. SpaceX advertises Falcon 9 rocket launches for 62 million USD while other companies charge upwards of 165 million USD per launch. These savings are largely due to SpaceX's ability to reuse the first stage; the first stage is the first rocket engine to engage, providing the initial thrust to send the rocket skyward, this stage is typically bigger than the next stage(s) because it must transport not only itself but the rest of the stages.

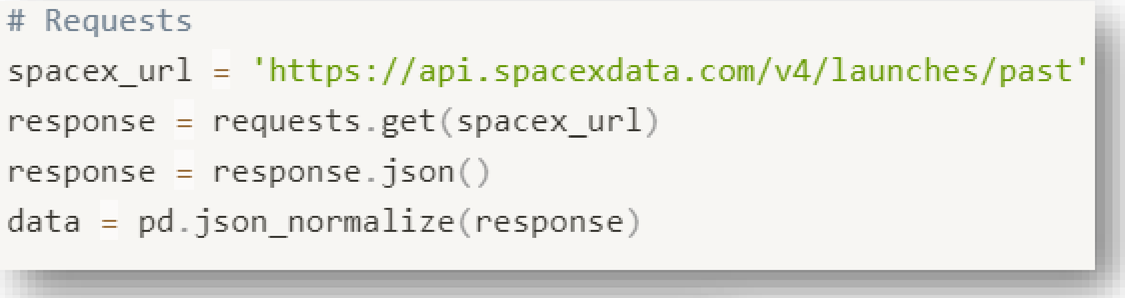
- Problem
  - In this capstone, we will predict if the Falcon 9 first stage will land successfully. Provided that we can determine whether the first stage will land, we can also calculate the cost of a launch.

Methodology

```
            end
        end
        synthDataCell(i) = synthDataTmp;
    end

    synthData = cell2mat(synthDataCell);
end


function y = interpolate(x)
    % interpolate by a factor of 3
    y = interp(x, 3);

    % remove random values until y is the same size as
    y(randperm(width(y), width(y) - width(x)
end

function y = decimate_(x)
    % decimate by 2
    if ~isa(class(x), 'double')
        y = decimate(double(x), 2, 'FIR');
        y = cast(y, 'like', x);
    else
        y = decimate(x, 2, 'FIR');
    end
```

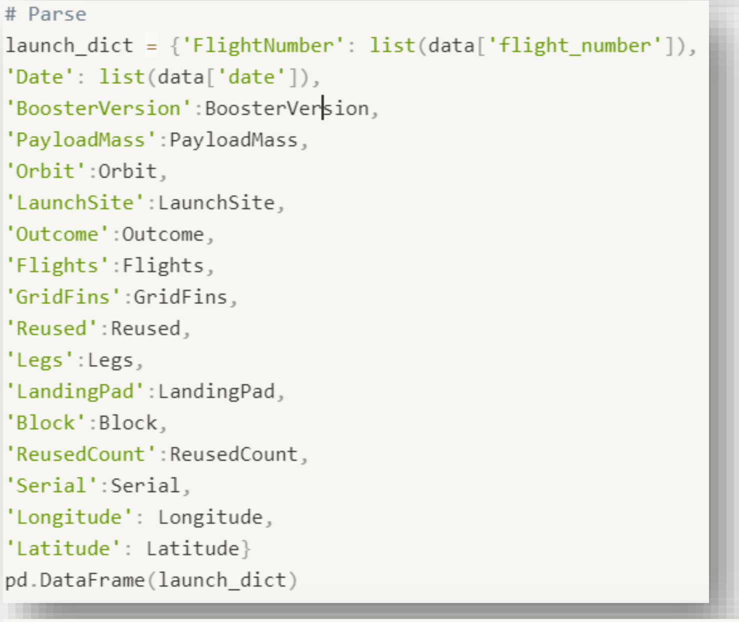
Data collection – SpaceX API

1. Request SpaceX launch data using the GET request
   - The response will return a json file which we will convert to a pandas DataFrame for easy data manipulation and analysis



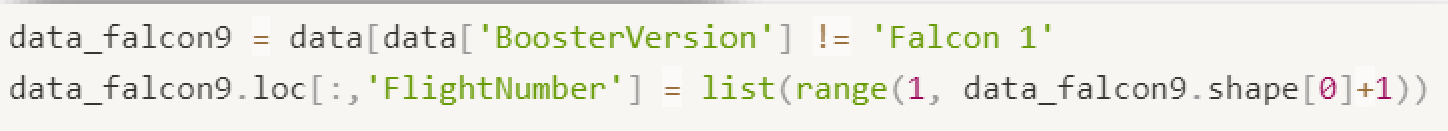
```
# Requests
spacex_url = 'https://api.spacexdata.com/v4/launches/past'
response = requests.get(spacex_url)
response = response.json()
data = pd.json_normalize(response)
```

2. Parse SpaceX launch data



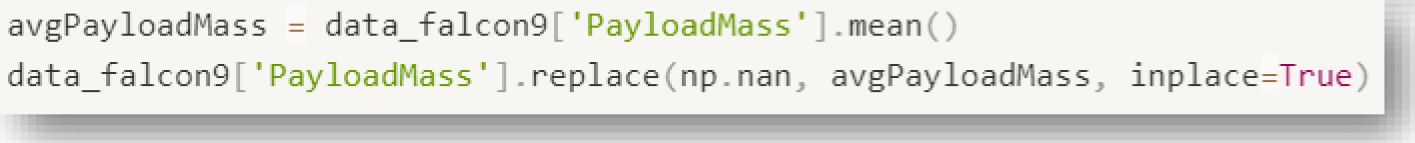
```
# Parse
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,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

3. Filter the DataFrame to only include Falcon 9 launches



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

4. Replace missing Payload Mass values with average Payload Mass

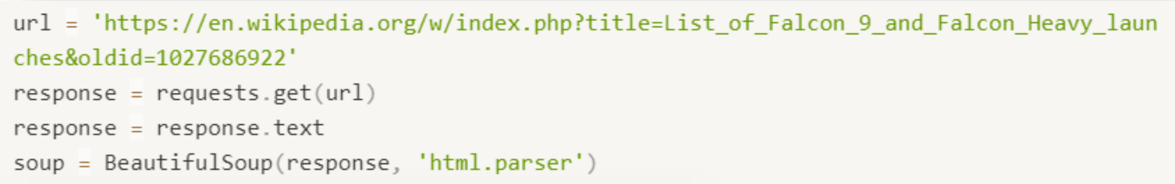


```
avgPayloadMass = data_falcon9['PayloadMass'].mean()
data_falcon9['PayloadMass'].replace(np.nan, avgPayloadMass, inplace=True)
```

SpaceX API Notebook URL

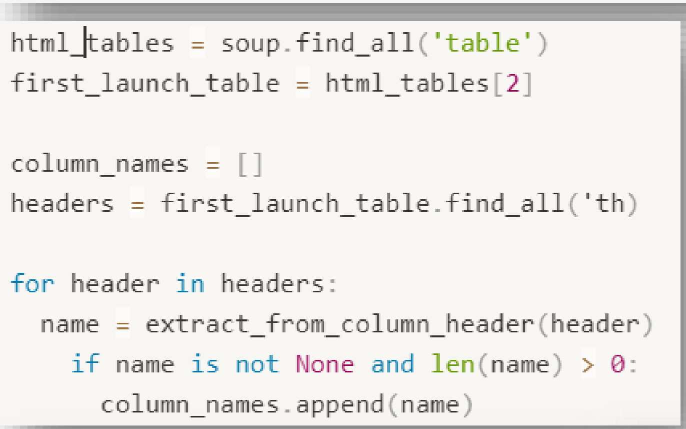
Data collection – Web scraping

1. Request the Falcon 9 Launch Wiki page from its URL



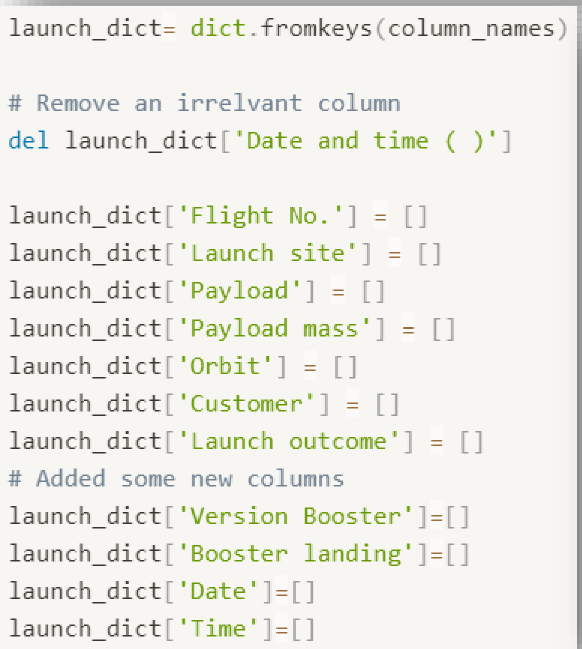
```
url = 'https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_laun
ches&oldid=1027686922'
response = requests.get(url)
response = response.text
soup = BeautifulSoup(response, 'html.parser')
```

2. Extract all column/variable names from the HTML table header
   - iterate through the <th> elements and apply the provided extract_column_from_header() to extract column name one by one

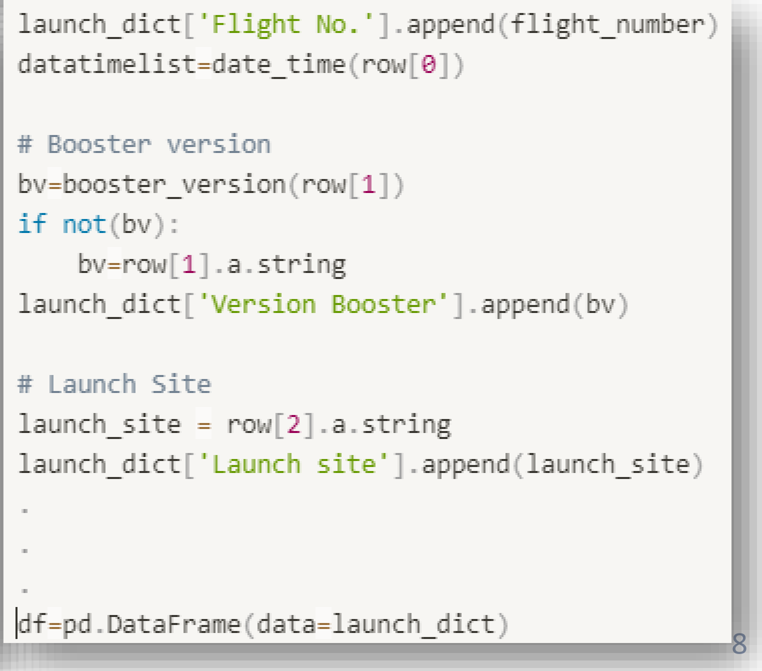


```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]

column_names = []
headers = first_launch_table.find_all('th)

for header in headers:
  name = extract_from_column_header(header)
    if name is not None and len(name) > 0:
      column_names.append(name)
```

3. Create a data frame by parsing the launch HTML tables
   - Initialize dictionary keys with empty lists
   - Append values to each dictionary key (code snippet only shows Booster Version and Launch Site, however, all keys were appended)



```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

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']=[]
```

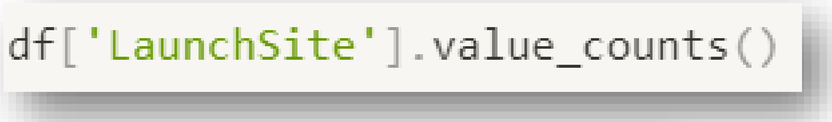


```
launch_dict['Flight No.'].append(flight_number)
datatimelist=date_time(row[0])

# Booster version
bv=booster_version(row[1])
if not(bv):
    bv=row[1].a.string
launch_dict['Version Booster'].append(bv)

# Launch Site
launch_site = row[2].a.string
launch_dict['Launch site'].append(launch_site)
.
.
.
df=pd.DataFrame(data=launch_dict)
```

SpaceX Web scraping Notebook URL

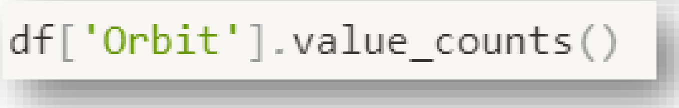
Data Wrangling

1. The number of launches at each site
   - This can help us determine which sites have the best outcomes for launches



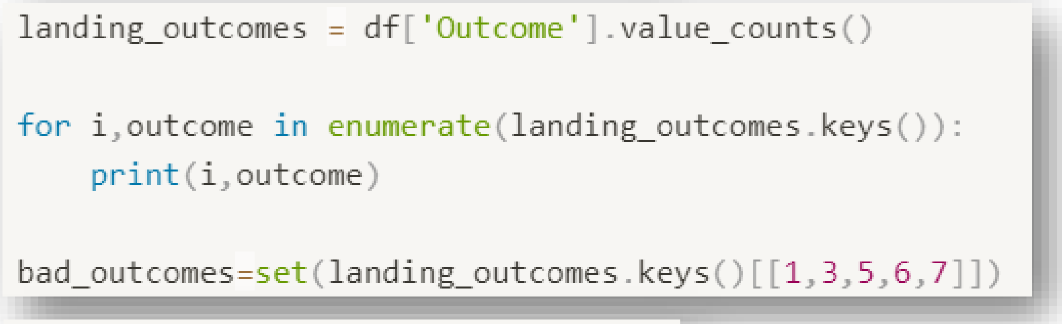
```
df['LaunchSite'].value_counts()
```

2. The number and occurrence of each orbit
   - Calculating the number and occurrence of each orbit can help us identify which orbit launches are most successful



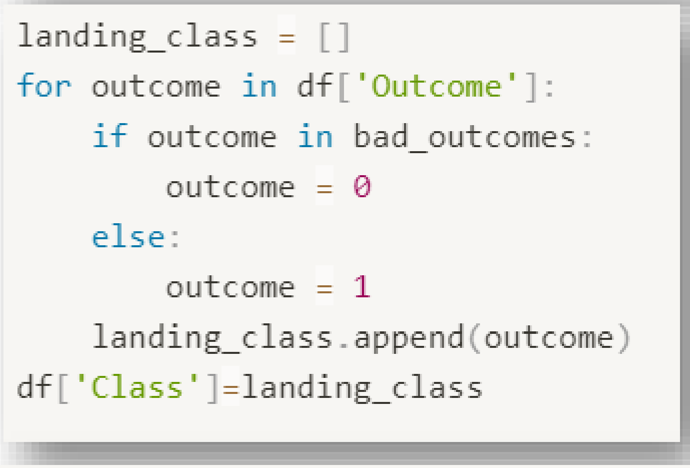
```
df['Orbit'].value_counts()
```

3. The number and occurrence of mission outcome per orbit type
   - Landing outcomes can be True or False (i.e., successful or not successful) and ASDS, RTLS, or Ocean represent ground pad, drone ship, or ocean landing, respectively. By understanding these variables, we can create a set of bad outcomes where the second stage did not land successfully and where this landing occurred.



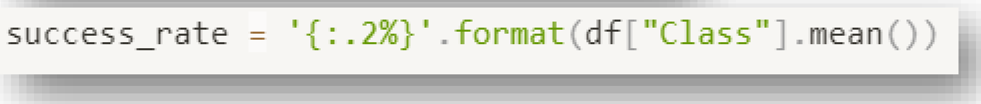
```
landing_outcomes = df['Outcome'].value_counts()

for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)

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

4. Create a landing outcome label from Outcome column



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

5. Calculate the success rate across all launch sites



```
success_rate = '{:.2%}'.format(df["Class"].mean())
```

Data Wrangling Notebook URL

EDA with data visualization

- For our EDA with data visualizations six plots were charted:
  1. In this first plot we created a scatter point chart that looked at the relationship between Flight Number and Launch Site. The points on the graph were either blue or orange corresponding to their class; Class 0 = unsuccessful and Class 1 = successful launch, respectively.
  2. In our second chart we also created a scatter point graph to illustrate the relationship between Payload and Launch site, similarly, points were color-coded according to their class. This graph highlights the Launch sites which appeared to have the most successful launches of unsuccessful launches as well as the typical mass of the payload launched.
  3. Third, we created a bar chart for success rate of each orbit. The bar chart easily identifies which orbit's had high success rates.
  4. Next, we wanted to see if there was any relationship between Flight Number and Orbit type, therefore, we created a scatter point plot to demonstrate this.
  5. Similarly, we also created a scatter point chart to plot the relationship between Payload Mass and Orbit type. This chart provides a clear visualization of how a Payload's Mass can either positively or negatively influence a certain orbit type.
  6. Lastly, we created a line chart to demonstrate the launch success yearly trend. In this graph, average success rate was the dependent variable while the years 2010 to 2017 were plotted as the independent variable.

SpaceX EDA with Pandas & Matplotlib URL

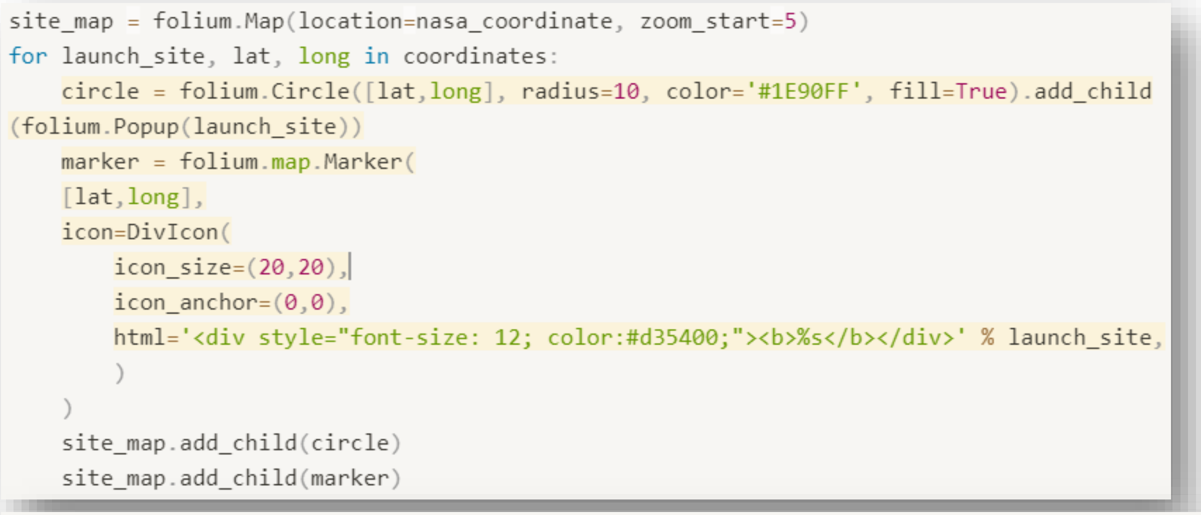
EDA with SQL

- **Query 1:** The names of the unique launch sites.
- **Query 2:** Five records where launch sites begin with the string 'CCA'.
  - CCA = Cape Canaveral Space Launch Complex
- **Query 3:** The total payload mass carried by boosters launched by NASA (CRS).
- **Query 4:** Average payload mass carried by booster version F9 v1.1.
- **Query 5:** The date when the first successful landing outcome in ground pad was achieved.
- **Query 6:** The names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
- **Query 7:** The total number of successful and failure mission outcomes
- **Query 8:** The names of the booster versions which have carried the maximum payload mass.
- **Query 9:** List of failed landing outcomes in drone ship, their booster versions, and launch site names for the in year 2015.
- **Query 10:** The occurrence of each landing outcomes (e.g., Failure (drone ship) or Success (ground pad) between June 4th, 2010, to March 20th, 2017, in descending order

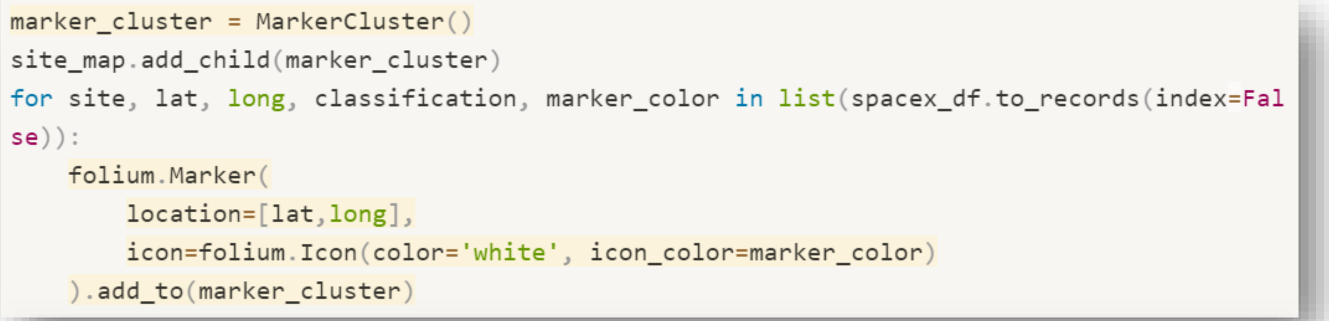
SpaceX EDA with SQL

Build an interactive map with Folium

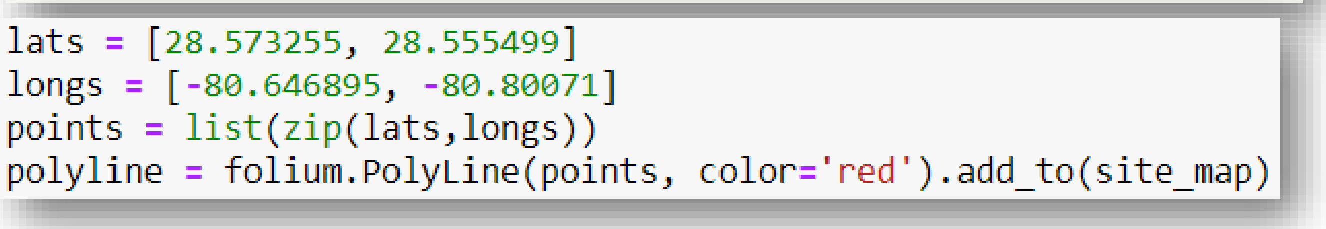
1. The first map objects we added was a circle for each launch site and a marker to label the launch site.
2. Next, we created a marker cluster object to illustrate the launch outcomes for each site, thus visualizing which sites have high success rates.
3. Finally, we added a Polyline object to show the distance between launch site KSC LC-39A and the Florida East Coast Railway



```
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
for launch_site, lat, long in coordinates:
    circle = folium.Circle([lat,long], radius=10, color='#1E90FF', fill=True).add_child
(folium.Popup(launch_site))
    marker = folium.map.Marker(
    [lat,long],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % launch_site,
        )
    )
    site_map.add_child(circle)
    site_map.add_child(marker)
```



```
marker_cluster = MarkerCluster()
site_map.add_child(marker_cluster)
for site, lat, long, classification, marker_color in list(spacex_df.to_records(index=Fal
se)):
    folium.Marker(
        location=[lat,long],
        icon=folium.Icon(color='white', icon_color=marker_color)
    ).add_to(marker_cluster)
```

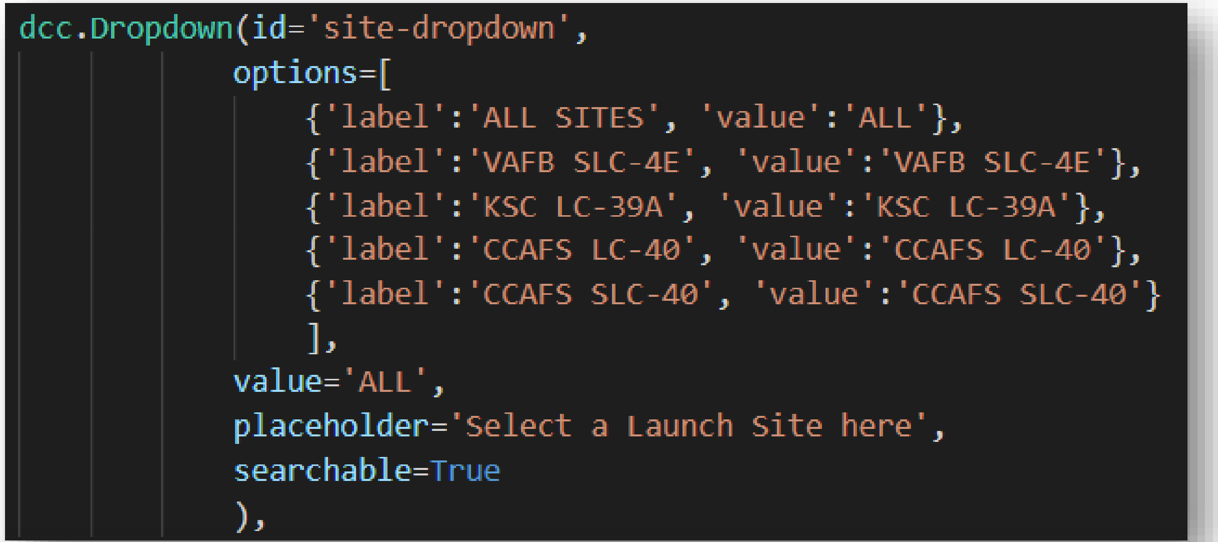


```
lats = [28.573255, 28.555499]
longs = [-80.646895, -80.80071]
points = list(zip(lats,longs))
polyline = folium.PolyLine(points, color='red').add_to(site_map)
```

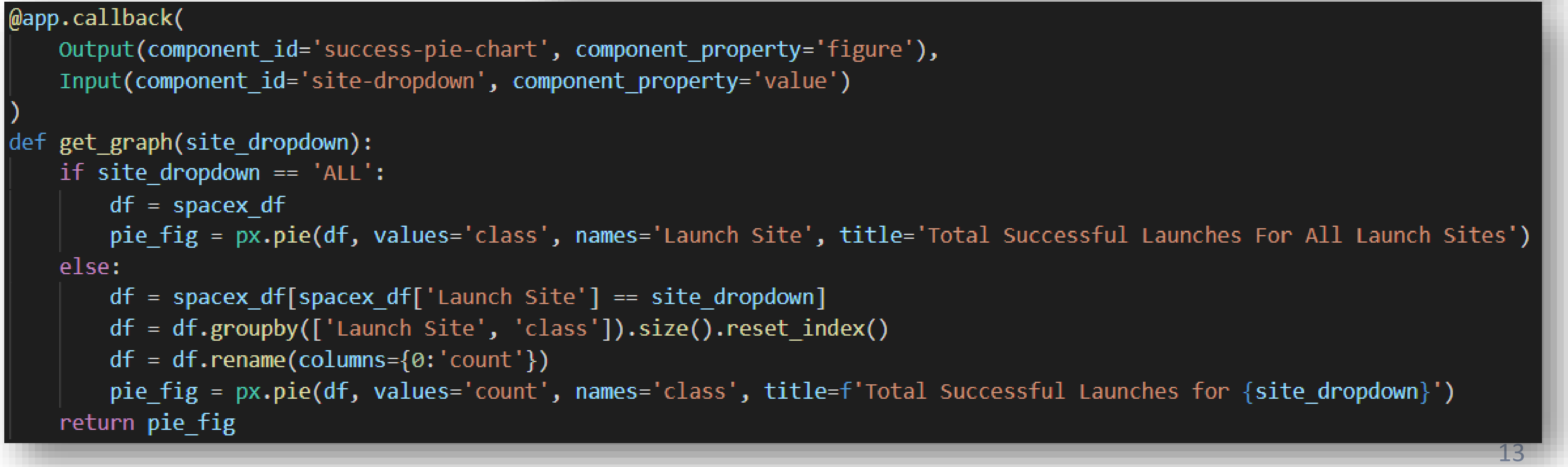
Interactive map with Folium Notebook URL

Build a Dashboard with Plotly Dash

- For our dashboard we created two types of graphs. First, a pie chart displaying the success/failures of each launch site including the successes for ALL launch sites. This type of chart provided an easy way of examining which site(s) had the best success ratio.



```
dcc.Dropdown(id='site-dropdown',
                options=[
                    {'label':'ALL SITES', 'value':'ALL'},
                    {'label':'VAFB SLC-4E', 'value':'VAFB SLC-4E'},
                    {'label':'KSC LC-39A', 'value':'KSC LC-39A'},
                    {'label':'CCAFS LC-40', 'value':'CCAFS LC-40'},
                    {'label':'CCAFS SLC-40', 'value':'CCAFS SLC-40'}
                    ],
                value='ALL',
                placeholder='Select a Launch Site here',
                searchable=True
                ),
```

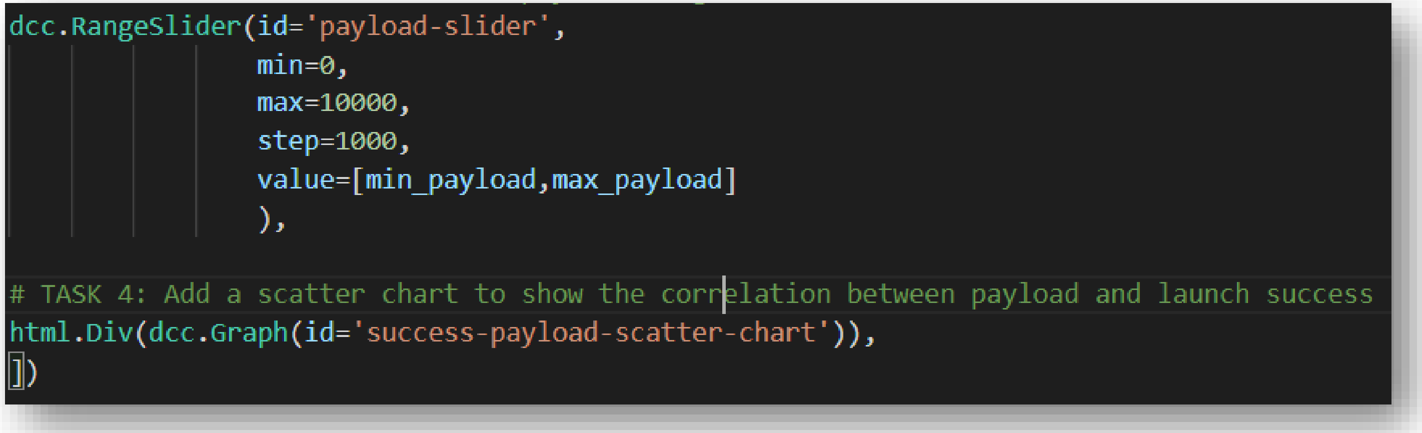


```
@app.callback(
    Output(component_id='success-pie-chart', component_property='figure'),
    Input(component_id='site-dropdown', component_property='value')
)
def get_graph(site_dropdown):
    if site_dropdown == 'ALL':
        df = spacex_df
        pie_fig = px.pie(df, values='class', names='Launch Site', title='Total Successful Launches For All Launch Sites')
    else:
        df = spacex_df[spacex_df['Launch Site'] == site_dropdown]
        df = df.groupby(['Launch Site', 'class']).size().reset_index()
        df = df.rename(columns={0:'count'})
        pie_fig = px.pie(df, values='count', names='class', title=f'Total Successful Launches for {site_dropdown}')
    return pie_fig
```

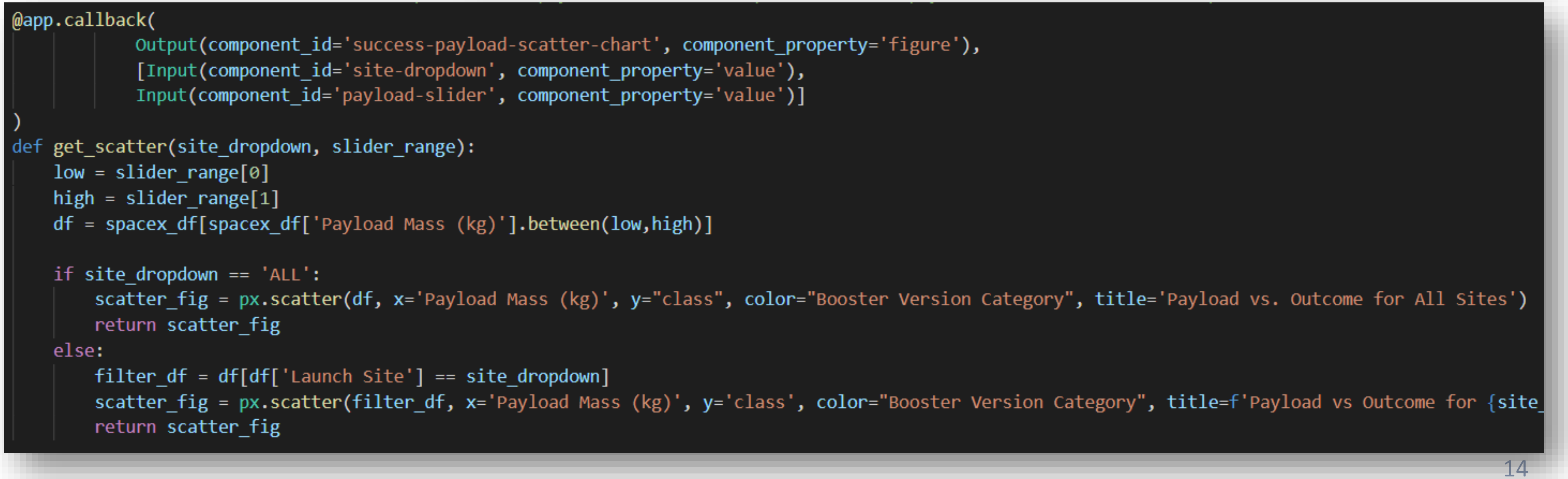
Plotly Dashboard URL

Build a Dashboard with Plotly Dash (continued..)

- Next, we produced a scatter plot in which we looked at the relationship between payload mass and class. In this plot we can easily identify different success rates based on payload mass AND how this might be related to the Booster Version



```
dcc.RangeSlider(id='payload-slider',
                min=0,
                max=10000,
                step=1000,
                value=[min_payload,max_payload]
                ),

# TASK 4: Add a scatter chart to show the correlation between payload and launch success
html.Div(dcc.Graph(id='success-payload-scatter-chart')),
])
```



```
@app.callback(
            Output(component_id='success-payload-scatter-chart', component_property='figure'),
            [Input(component_id='site-dropdown', component_property='value'),
            Input(component_id='payload-slider', component_property='value')]
)
def get_scatter(site_dropdown, slider_range):
    low = slider_range[0]
    high = slider_range[1]
    df = spacex_df[spacex_df['Payload Mass (kg)'].between(low,high)]

    if site_dropdown == 'ALL':
        scatter_fig = px.scatter(df, x='Payload Mass (kg)', y="class", color="Booster Version Category", title='Payload vs. Outcome for All Sites')
        return scatter_fig
    else:
        filter_df = df[df['Launch Site'] == site_dropdown]
        scatter_fig = px.scatter(filter_df, x='Payload Mass (kg)', y='class', color="Booster Version Category", title=f'Payload vs Outcome for {site_
        return scatter_fig
```

Plotly Dashboard URL

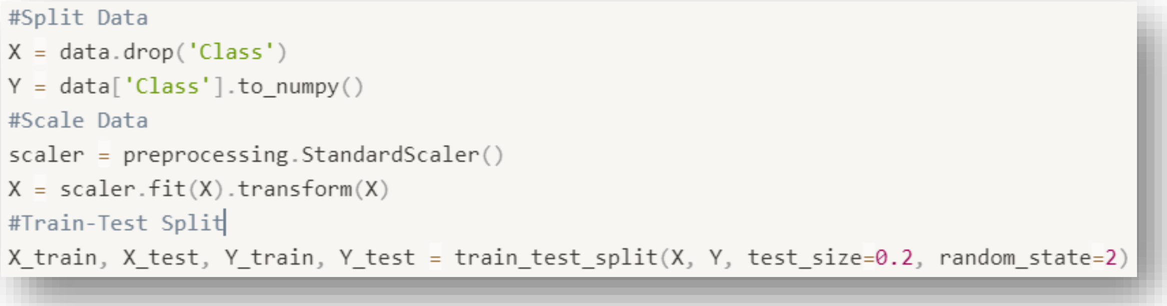
Predictive analysis (Classification)

- The cleaned data was fitted onto four classification models:
  1. **Logistic Regression:** Logistic regression is analogous to linear regression but tries to predict a categorical or discrete target field instead of a numeric one. The goal of logistic regression is to build a model to predict the class of each launch and the probability of each sample belonging to a class.
  2. **Support Vector Machine (SVM):** SVM is a supervised algorithm that can classify cases by finding a separator the SVM algorithm outputs an optimal hyperplane that categorizes new examples. SVM has several advantages such as good accuracy in high dimensional spaces.
  3. **Decision Tree Classifier:** Decision trees create a tree-like model by testing an attribute and branching the cases based on the result of the test. The underlying principle in creating decision tree's is determining the **information gain**—the information that can increase the level of certainty after splitting. In other words, when creating decision trees, we are looking for trees that have the lowest entropy (randomness) in their nodes.
  4. **K-Nearest Neighbors (KNN):** KNN is an algorithm that classifies cases data points based on their similarity to other cases. data points that are near each other are said to be neighbors. In this model, K is defined as the number of neighbors the model must consider when classifying data points.
- To find the best hyper-parameters for each model we used GridSearchCV. GridSearchCV is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. Thus, we can easily select the optimal parameters for your model
- Finally, we calculated the accuracy score for each model to determine which model performed best.

SpaceX Classification Analysis Notebook

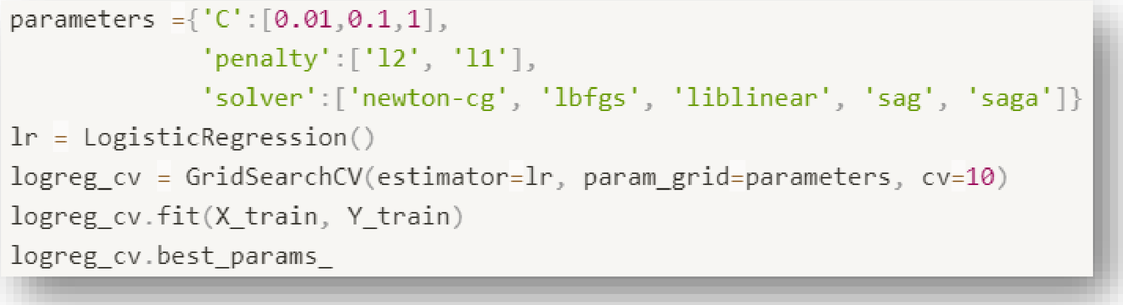
Predictive analysis (continued..)

1. Split DataFrame, then scale data and perform a train-test-split



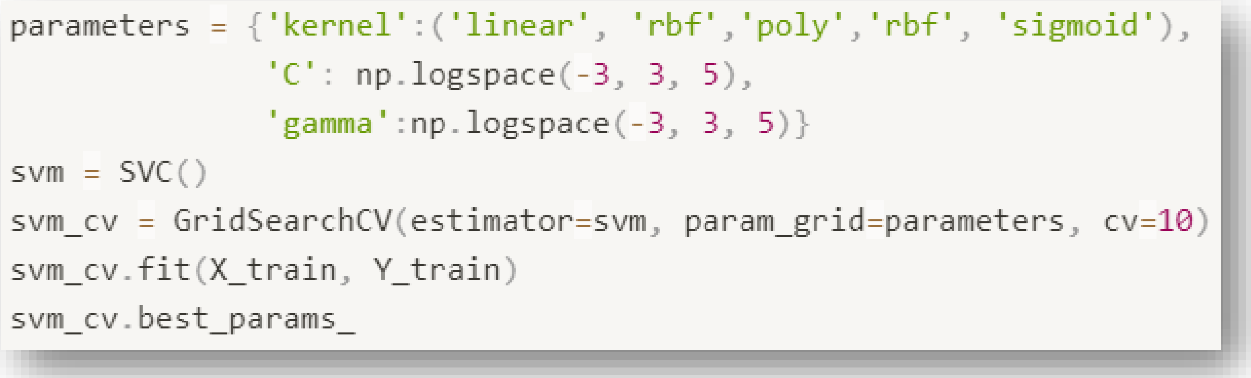
```
#Split Data
X = data.drop('Class')
Y = data['Class'].to_numpy()
#Scale Data
scaler = preprocessing.StandardScaler()
X = scaler.fit(X).transform(X)
#Train-Test Split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

2. Create a logistic regression object then create a GridSearchCV object. Fit the object to find the best parameters from the dictionary parameters



```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2', 'l1'],
             'solver':['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']}
lr = LogisticRegression()
logreg_cv = GridSearchCV(estimator=lr, param_grid=parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
logreg_cv.best_params_
```

3. Create SVM object then create a GridSearchCV object svm_cv with cv - 10. Fit the object to find the best parameters from the dictionary parameters.

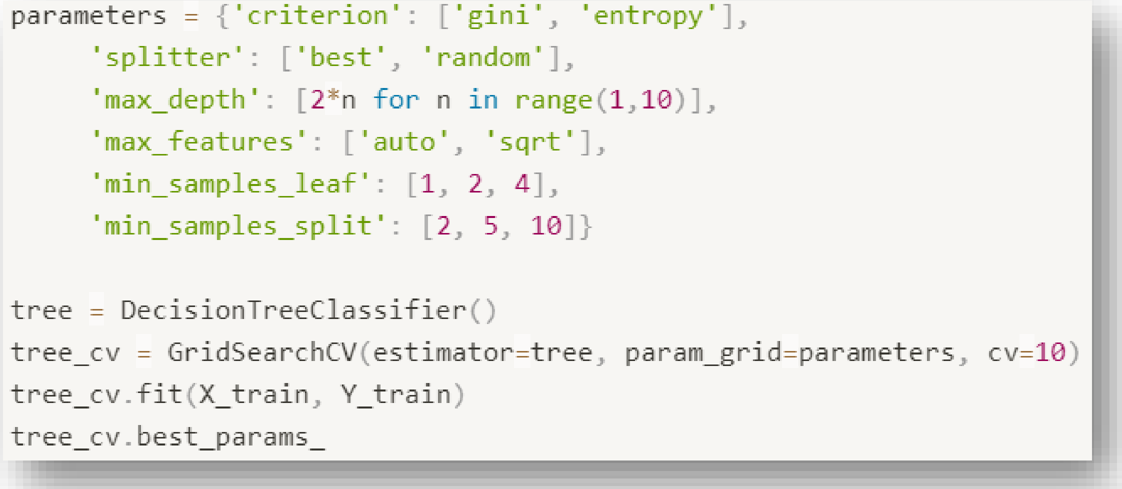


```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
svm_cv = GridSearchCV(estimator=svm, param_grid=parameters, cv=10)
svm_cv.fit(X_train, Y_train)
svm_cv.best_params_
```

SpaceX Classification Analysis Notebook

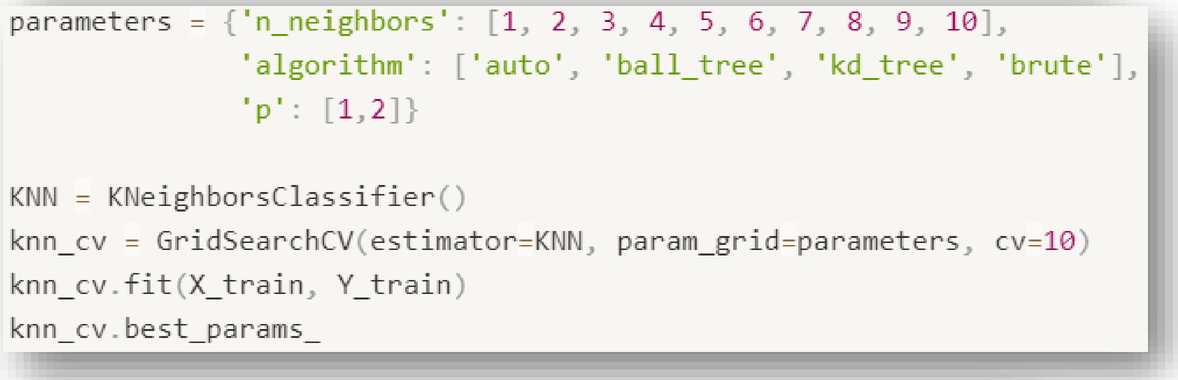
Predictive analysis (continued..)

4. Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters



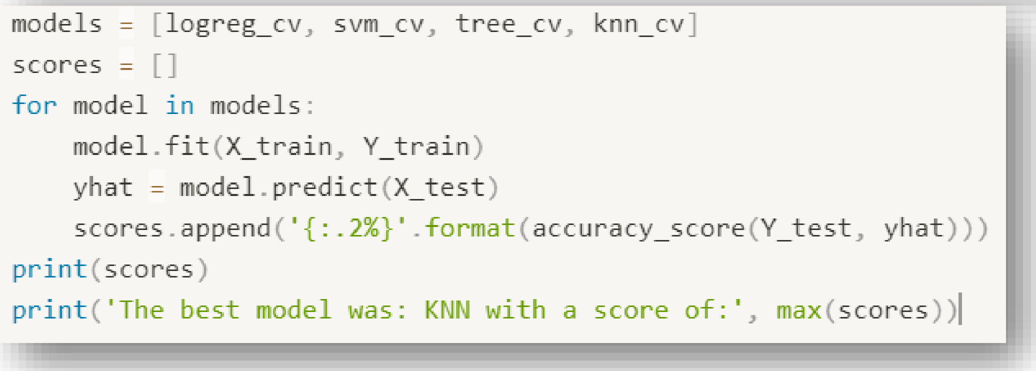
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
tree_cv = GridSearchCV(estimator=tree, param_grid=parameters, cv=10)
tree_cv.fit(X_train, Y_train)
tree_cv.best_params_
```

5. Create KNN object then create a GridSearchCV object knn_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters



```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
knn_cv = GridSearchCV(estimator=KNN, param_grid=parameters, cv=10)
knn_cv.fit(X_train, Y_train)
knn_cv.best_params_
```

6. Find best performing model



```
models = [logreg_cv, svm_cv, tree_cv, knn_cv]
scores = []
for model in models:
    model.fit(X_train, Y_train)
    yhat = model.predict(X_test)
    scores.append('{:.2%}'.format(accuracy_score(Y_test, yhat)))
print(scores)
print('The best model was: KNN with a score of:', max(scores))
```

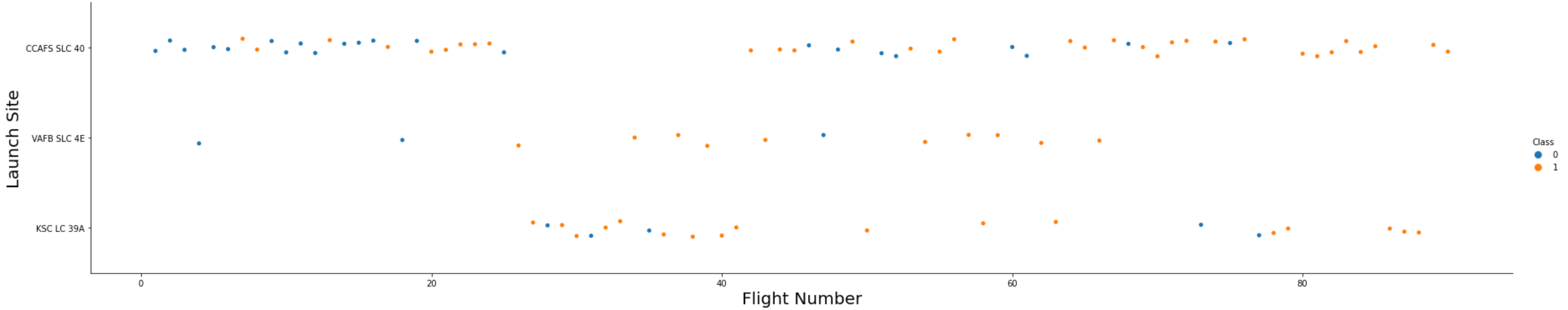
SpaceX Classification Analysis Notebook

PRIME
+12.0
COMPETITIVE ANALYSIS
80%
75%
POLL TITLE
INFOGRAPHIC

Results

EDA with Visualization

Flight Number vs. Launch Site



This scatter point plots displays the relationship between flight number and launch site. Here, we see that the launch site is our dependent variable while the points on the graph are color-coded according to a launches class. Class 0 represents unsuccessful launches while Class 1 are successful launches.

The launch sites are as follows:

- CCAFS SLC-40: Cape Canaveral Space Launch Complex 40
- VAFB SLC-4E: Vandenberg AFB Space Launch Complex 4
- KSC LC 39A: Kennedy Space Center Launch Complex 39A

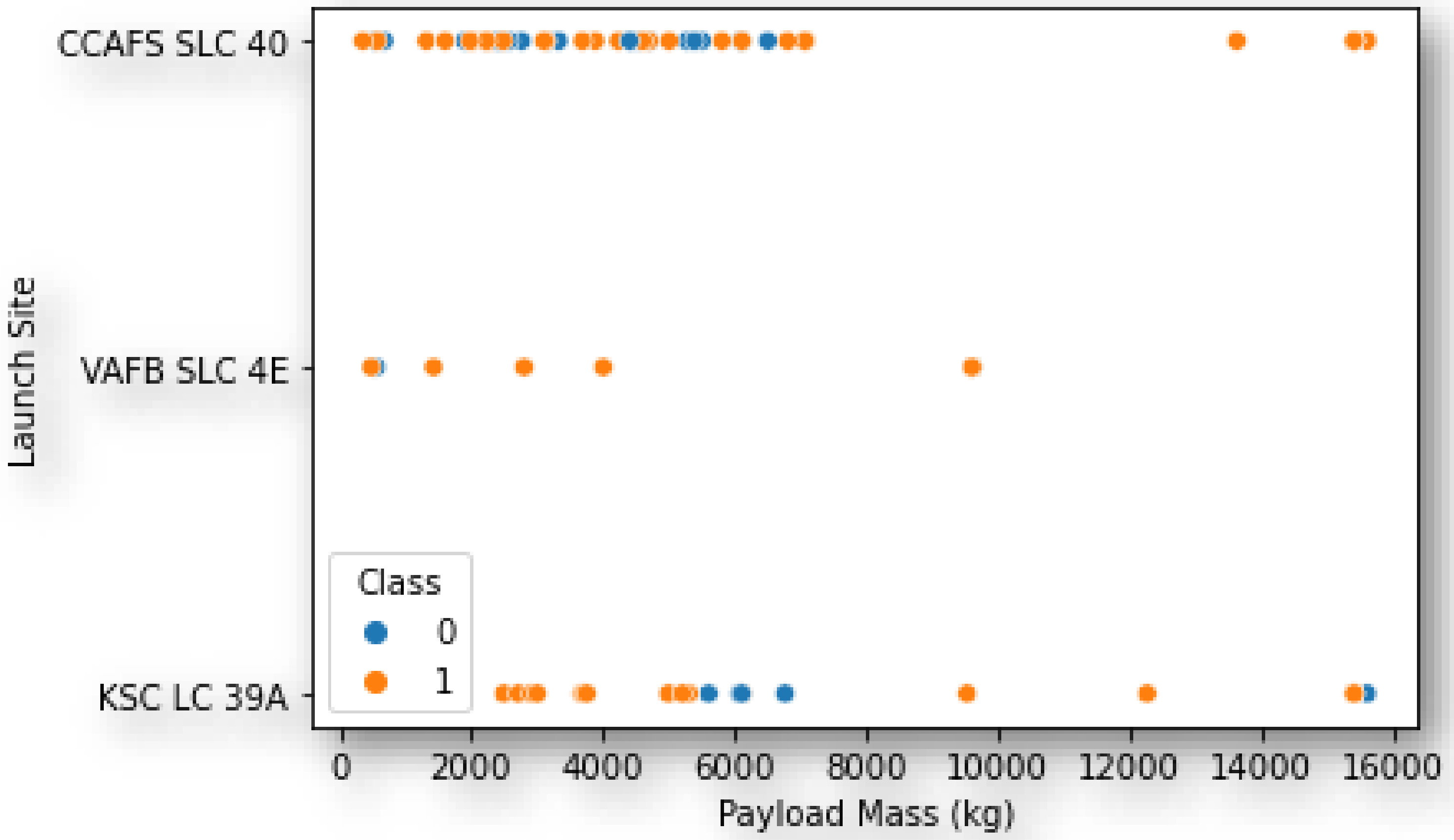
Payload vs. Launch Site

In this plot we can see the relationship between Payload Mass and Launch Site. Additionally, the points are the scatter plot are colored according to their class 0 are unsuccessful launches and class 1 are successful launches.

The launch sites are as follows:

- CCAFS SLC-40 is Cape Canaveral Space Launch Complex 40
- VAFB SLC-4E is Vandenberg AFB Space Launch Complex 4
- KSC LC 39A is
- Kennedy Space Center Launch Complex 39A

- The payload is the load carried by an aircraft or spacecraft consisting of things (such as passengers or instruments) necessary to the purpose of the flight

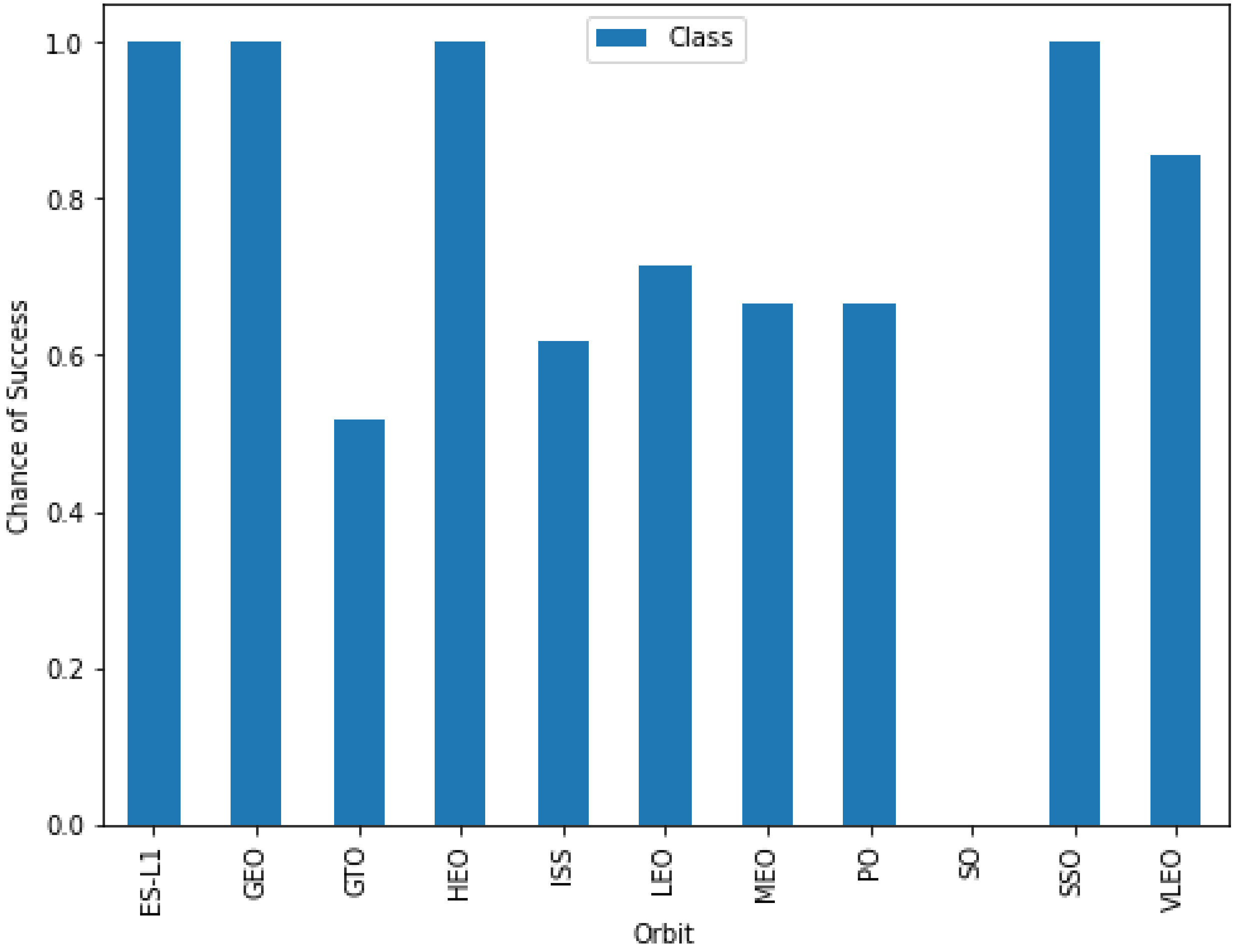


Success rate vs. Orbit type

This bar chart looks at the rate of success for each orbit type.

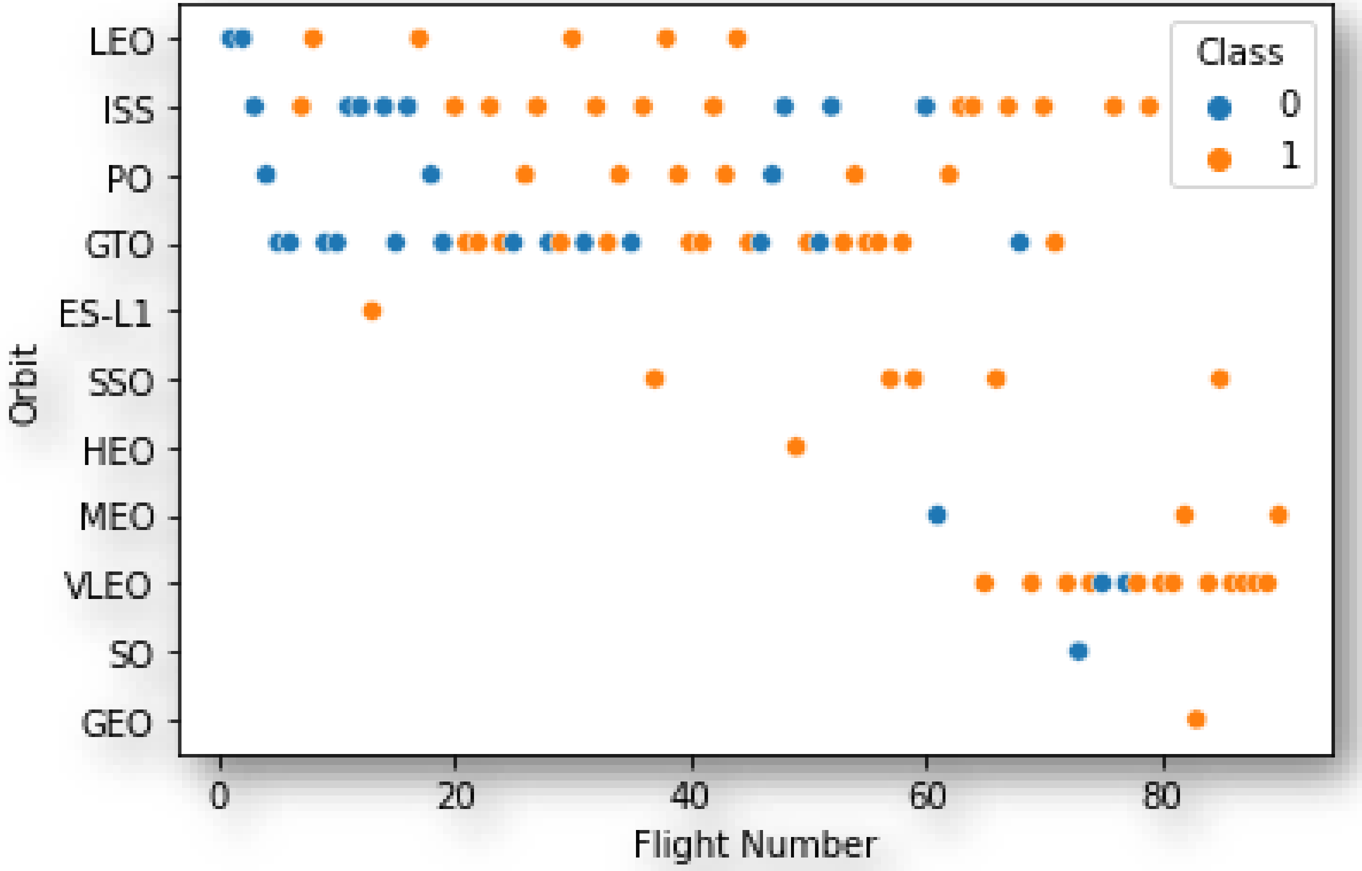
We can see 11 different orbit types:

1. LEO: Low Earth orbit (LEO)is an Earth-centered orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth.
2. VLEO: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km
3. GTO A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator.
4. SSO (or SO): It is a Sun-synchronous orbit also called a Heli synchronous orbit is a nearly polar orbit around a planet..
5. ES-L1 :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth \[5] .
6. HEO A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth.
7. ISS A modular space station (habitable artificial satellite) in low Earth orbit.
8. MEO Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi).
9. HEO Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi) \[9]
10. GEO It is a circular geosynchronous orbit 35,786 kilometers (22,236 miles) above Earth's equator and following the direction of Earth's rotation.
11. PO It is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited.



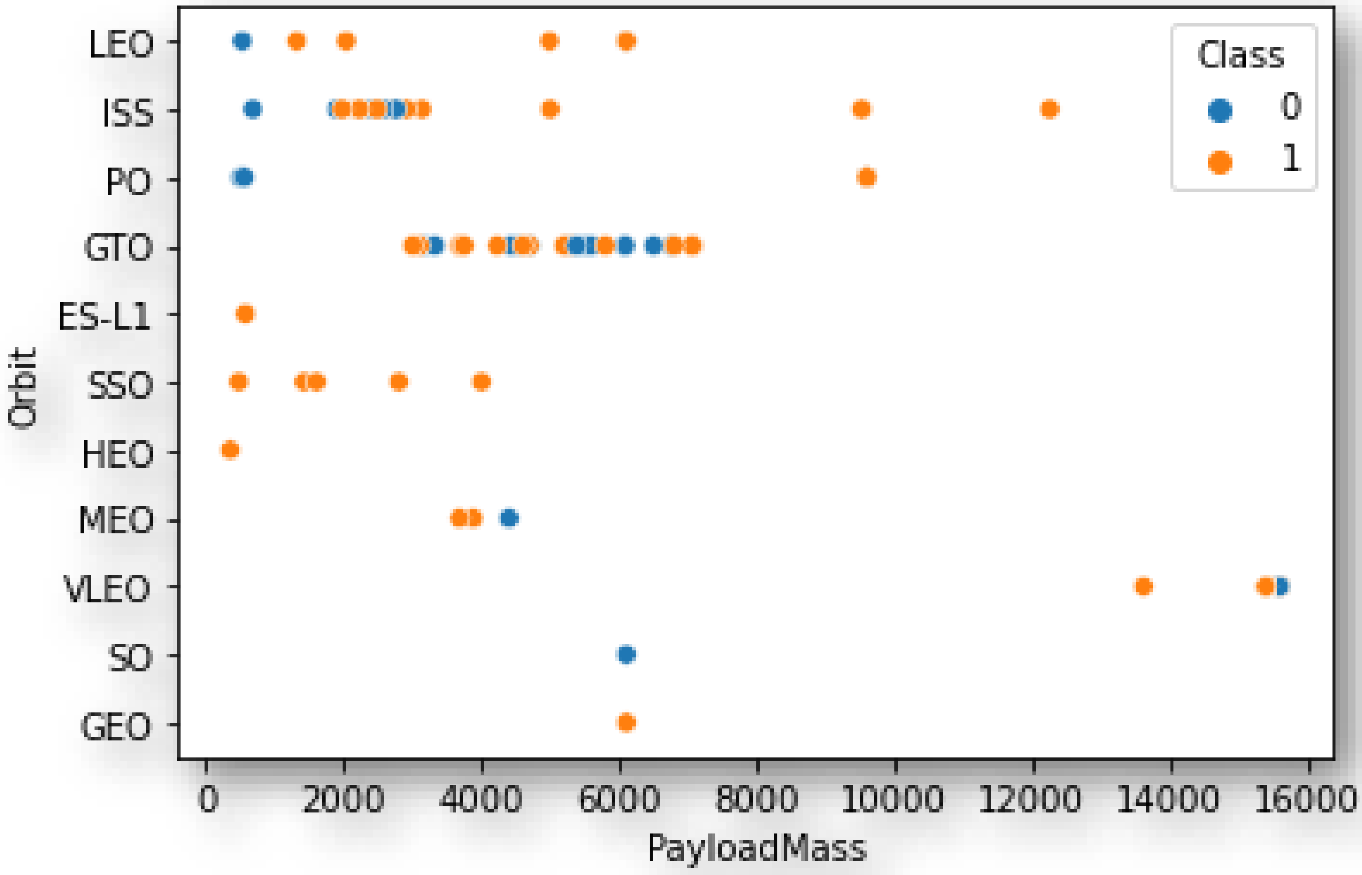
Flight Number vs. Orbit type

Rather than looking at success rate for each orbit type in this graph we plot the points of the scatter plot according to their class. Similarly, flight number is plotted on the x-axis as the independent variable. This graph also shows an alternate way of looking at which orbit types are most successful.



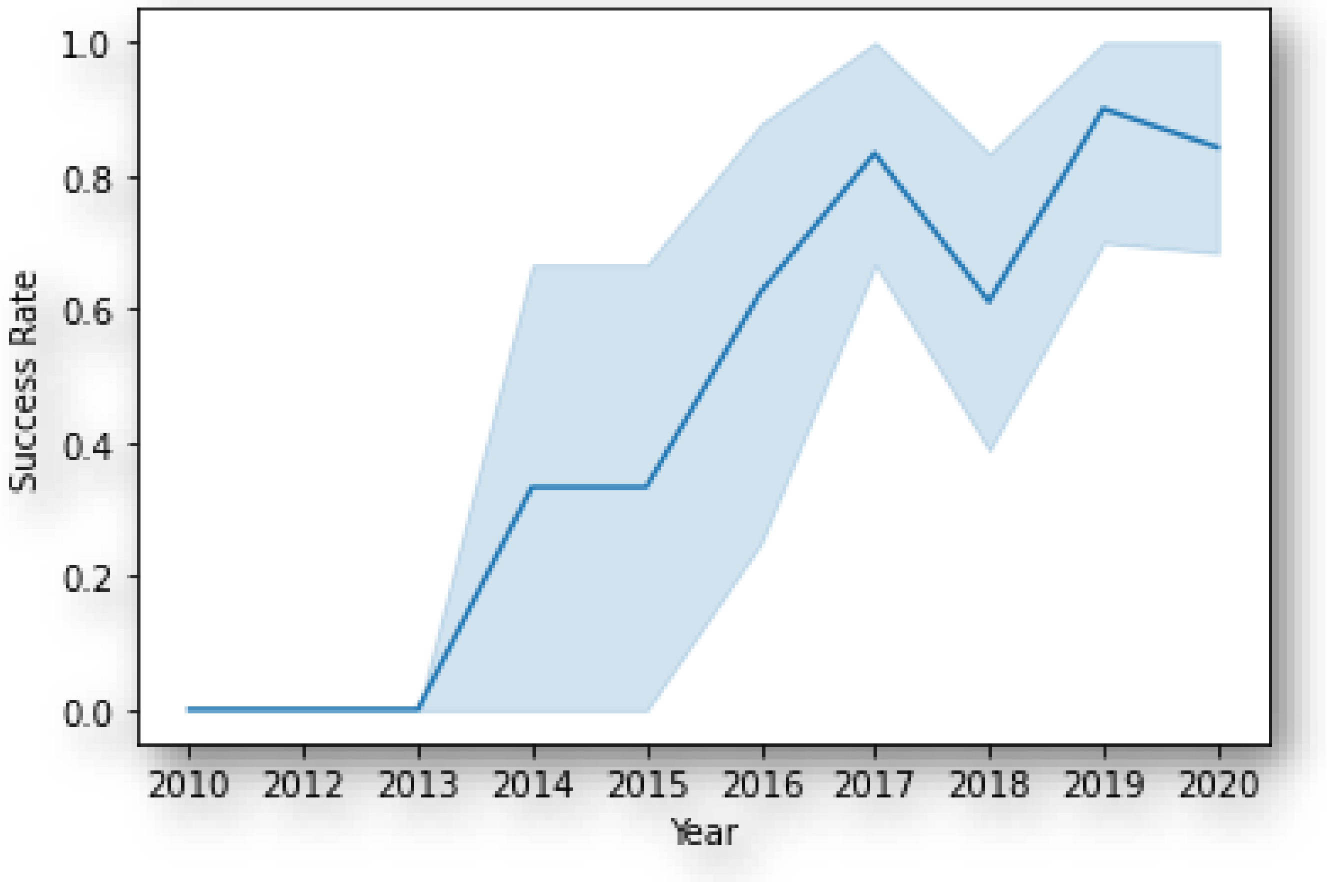
Payload vs. Orbit type

- Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type. Since our points are labelled according to their class, we can also see which orbit types tend to have successful launches and at what payload mass.
- Payload mass is shown in kilograms



Launch success yearly trend

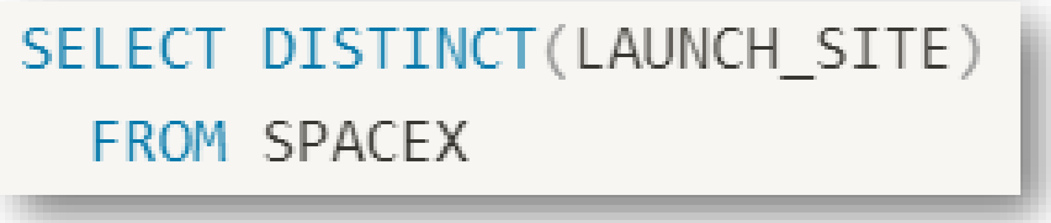
We can also display launch success across several years. Rather than drilling down on individual launch sites this graph displays the average success rate of launches across all sites per year. In addition, this line chart also contains shaded areas that repersent confidence intervals—simply put, a confidence interval is the mean of your estimate plus and minus the variation in that estimate



EDA with SQL

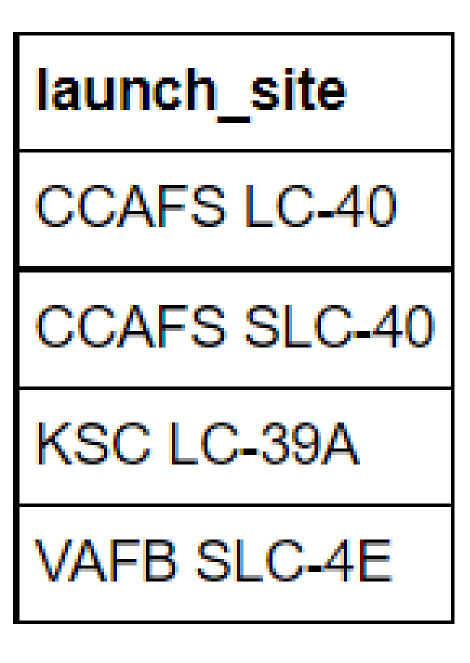
All launch site names

- Query execution:



```
SELECT DISTINCT(LAUNCH_SITE)
  FROM SPACEX
```

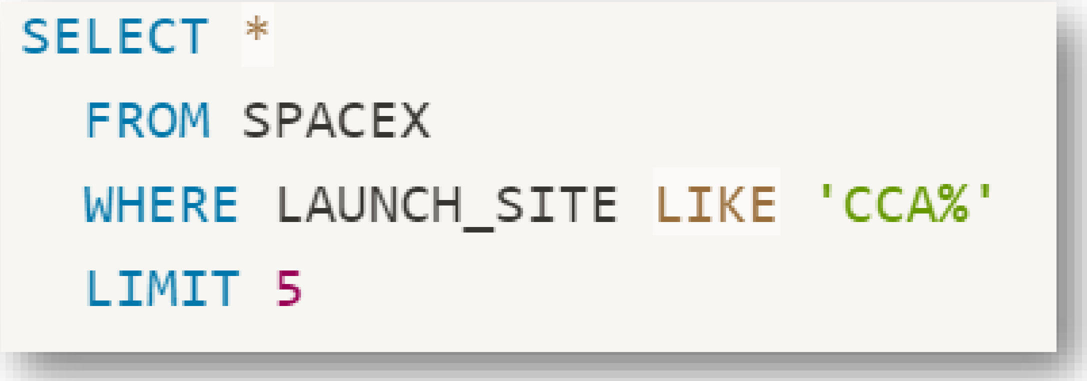
- Results: names of all four launch sites. There are only three launch sites in this dataset as Space Launch Complex 40 (SLC-40), was previously Launch Complex 40 (LC-40)



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

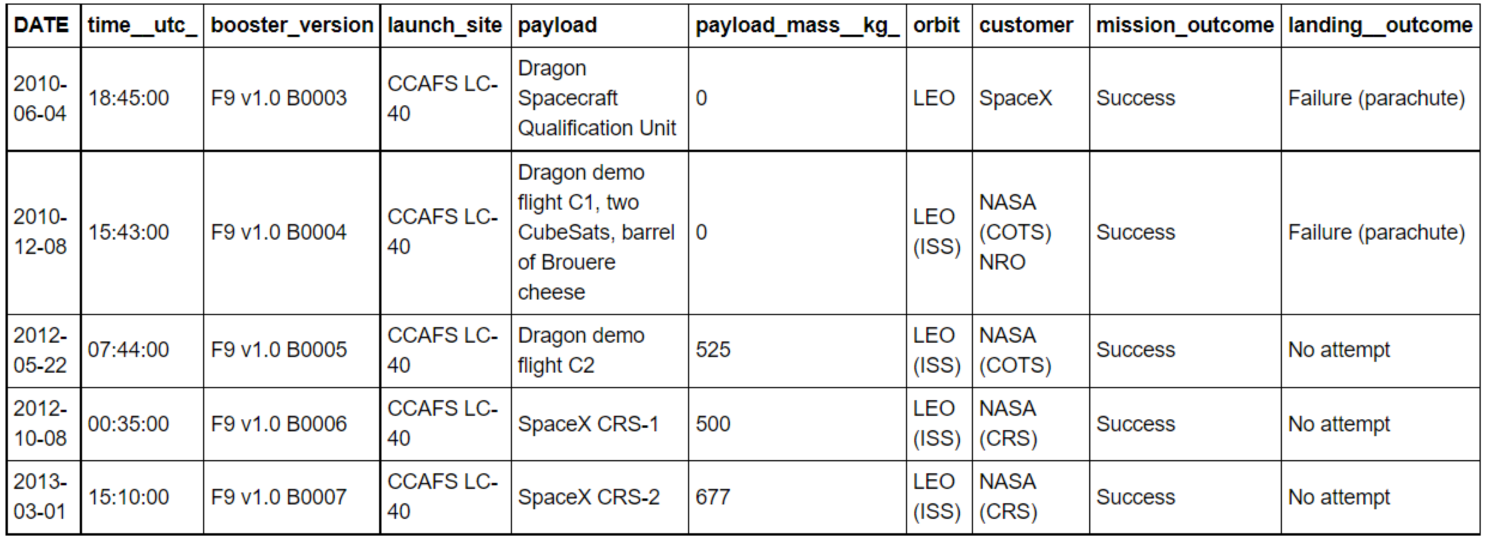
Launch site names begin with `CCA`

- Query Execution:



```
SELECT *
  FROM SPACEX
  WHERE LAUNCH_SITE LIKE 'CCA%'
  LIMIT 5
```

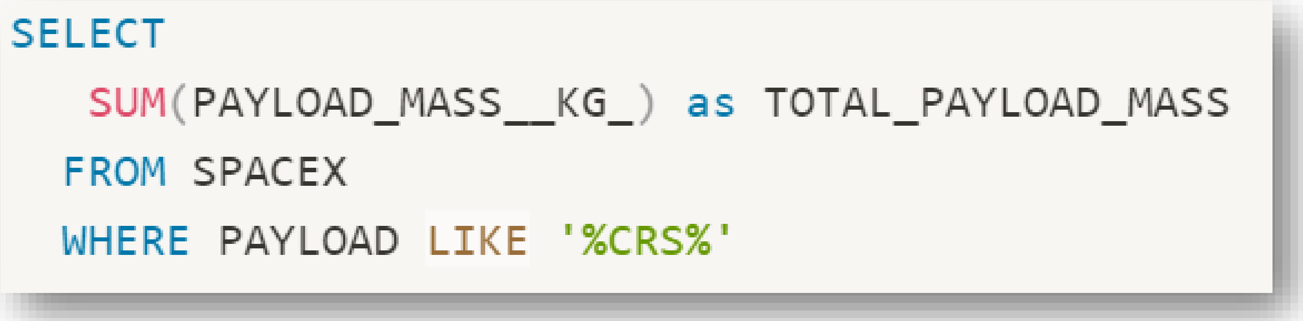
- Result: list of five launch records that occurred at Space Launch Complex 40/Launch Complex 40



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

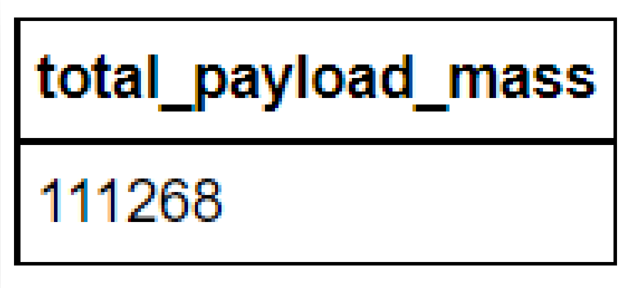
Total payload mass

- Query Execution:



```
SELECT
   SUM(PAYLOAD_MASS__KG_) as TOTAL_PAYLOAD_MASS
  FROM SPACEX
  WHERE PAYLOAD LIKE '%CRS%'
```

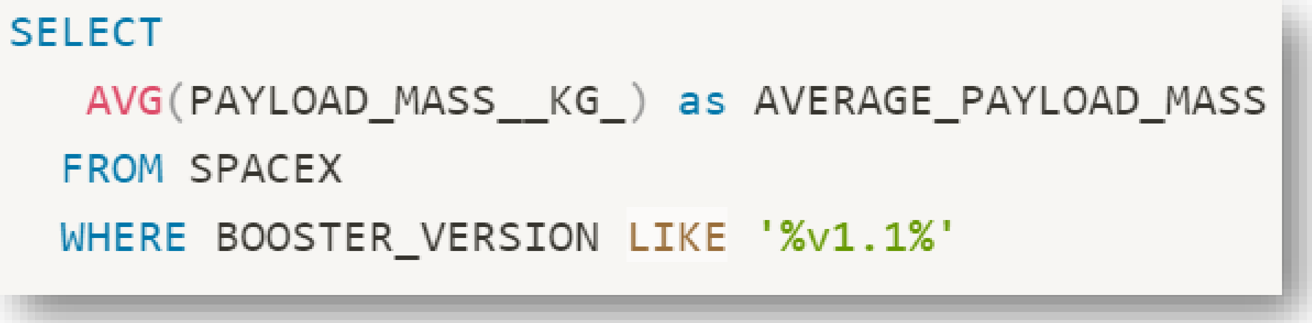
- Result: The total mass in kilograms for all payloads carried by NASA boosters (i.e., ‘CRS’)



total_payload_mass
111268

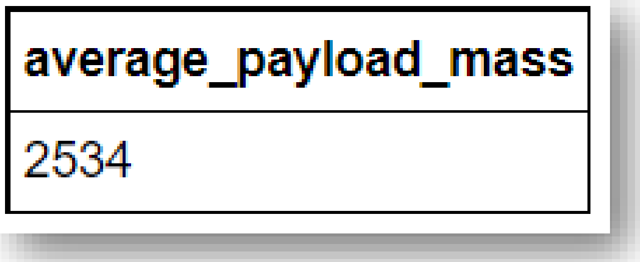
Average payload mass by F9 v1.1

- Query execution:



```
SELECT
   AVG(PAYLOAD_MASS__KG_) as AVERAGE_PAYLOAD_MASS
  FROM SPACEX
  WHERE BOOSTER_VERSION LIKE '%v1.1%'
```

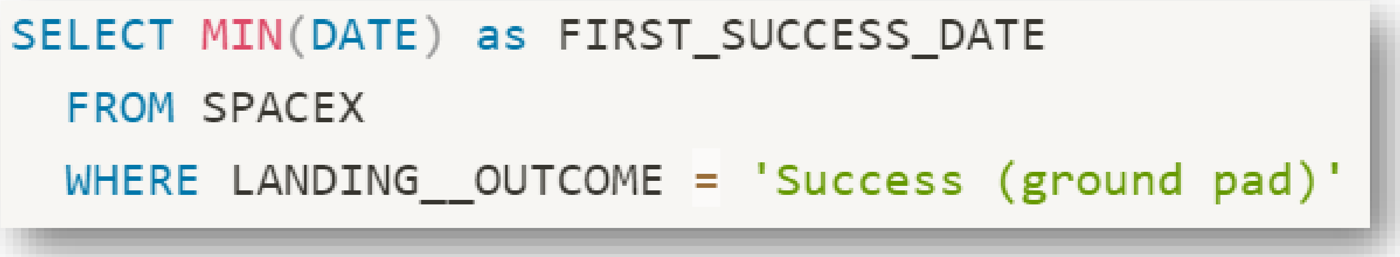
- Result: The average payload mass in kilograms carried by the Falcon 9 booster version 1.1



average_payload_mass
2534

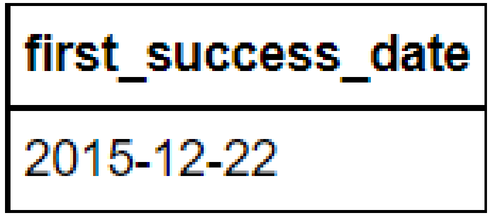
First successful ground landing date

- Query execution:



```
SELECT MIN(DATE) as FIRST_SUCCESS_DATE
  FROM SPACEX
  WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

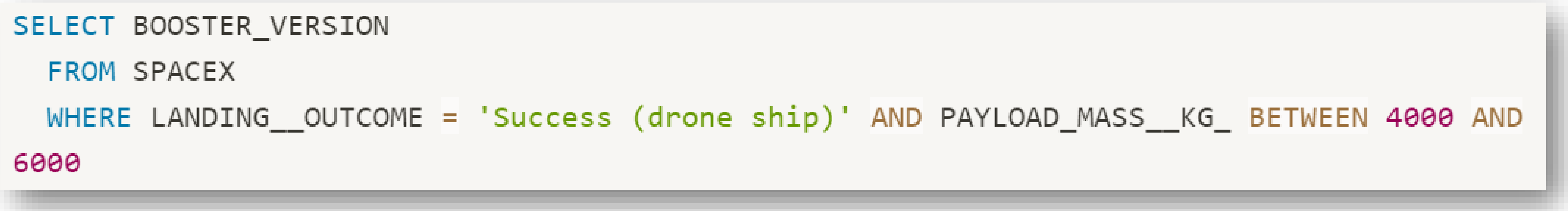
- Result: The date of the first successful landing outcome in ground pad on December 22nd, 2015



first_success_date
2015-12-22

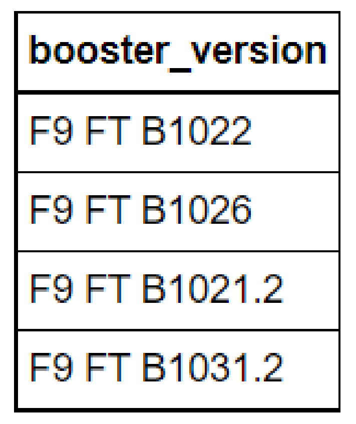
Successful drone ship landing with payload between 4000 and 6000

- Query execution:



```
SELECT BOOSTER_VERSION
  FROM SPACEX
  WHERE LANDING__OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND
6000
```

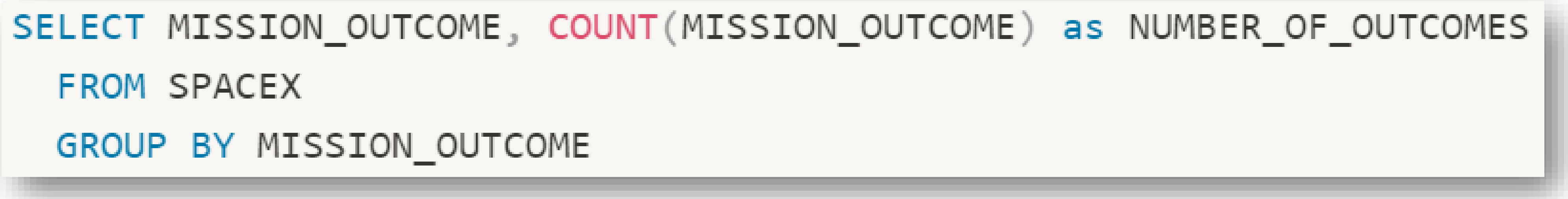
- Result: List of all boosters which have a payload mass between 4000-6000kg's which have landed successfully in drone ship



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

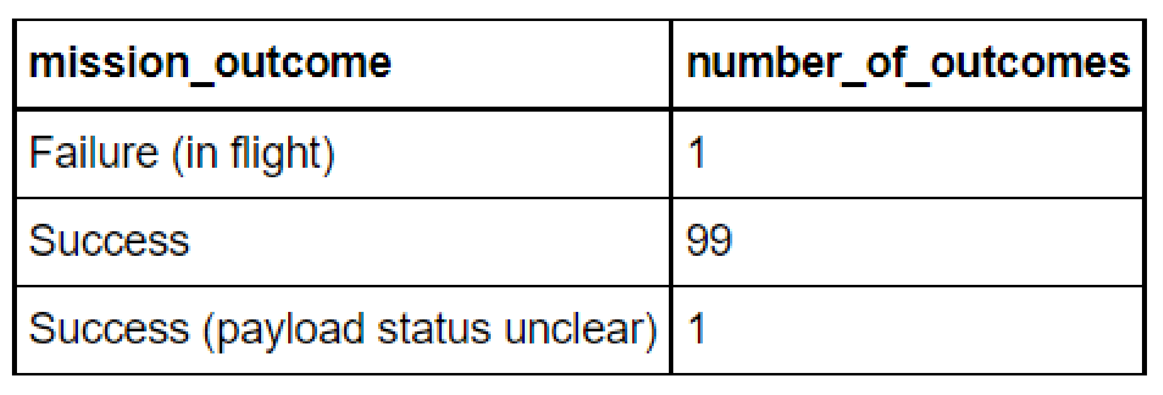
Total number of successful and failure mission outcomes

- Query execution:



```
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) as NUMBER_OF_OUTCOMES
  FROM SPACEX
  GROUP BY MISSION_OUTCOME
```

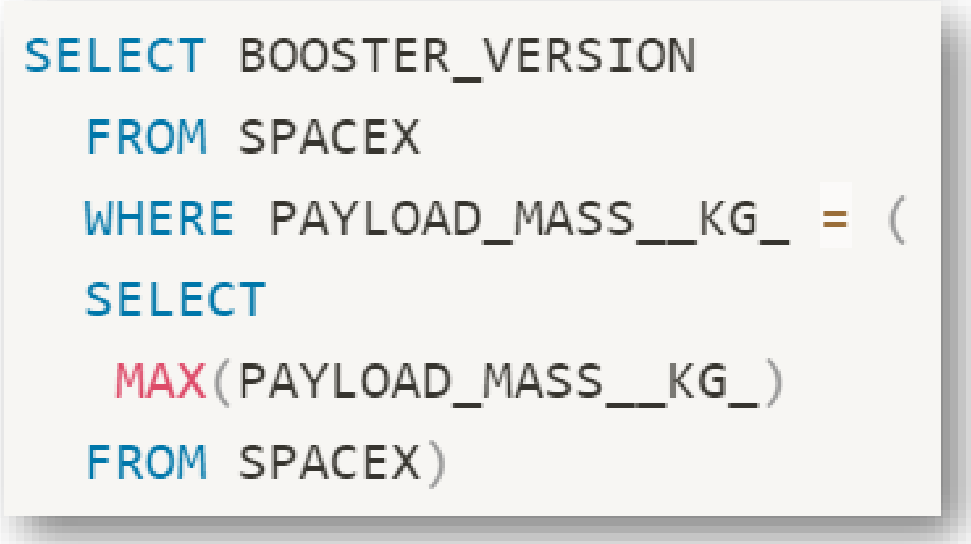
- Result: The total number of occurrences for each mission outcome



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

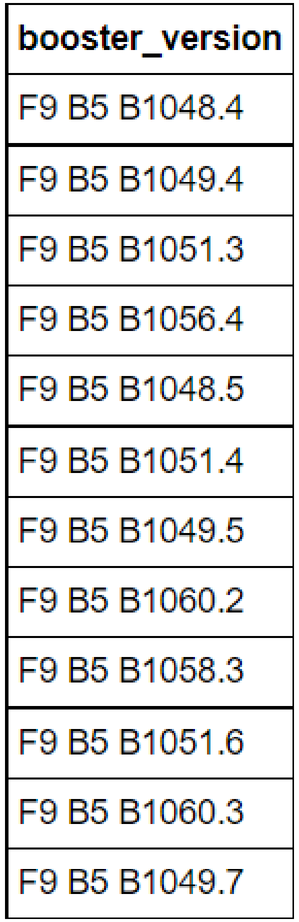
Boosters carried maximum payload

- Query execution:



```
SELECT BOOSTER_VERSION
  FROM SPACEX
  WHERE PAYLOAD_MASS__KG_ = (
  SELECT
   MAX(PAYLOAD_MASS__KG_)
  FROM SPACEX)
```

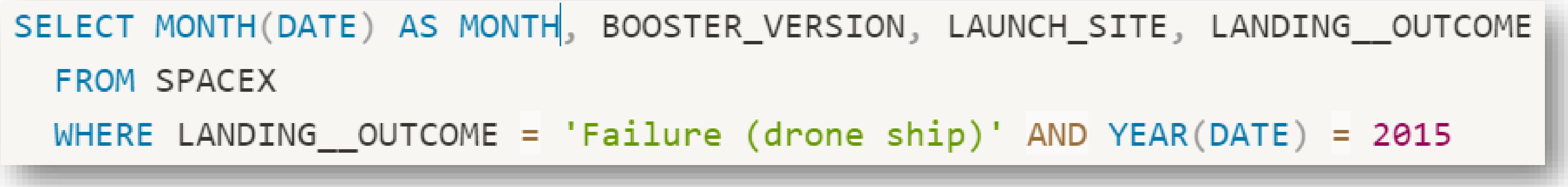
Result: A list of boosters that have carried payloads with maximum payload mass



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

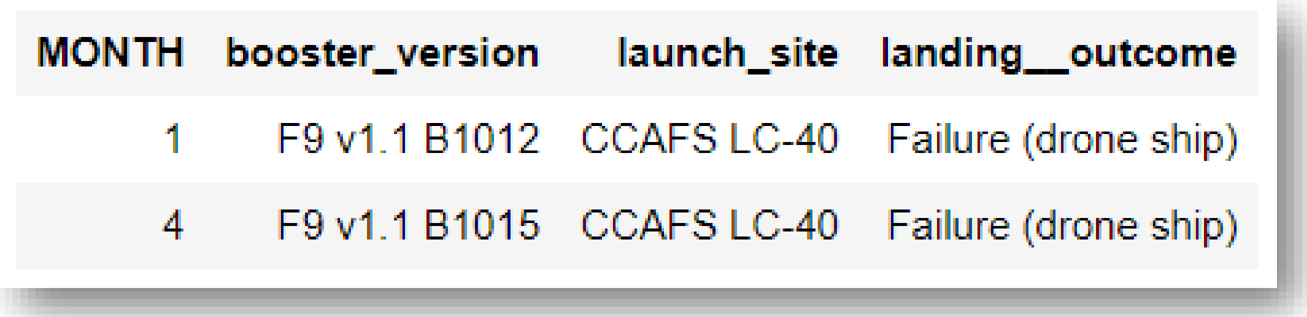
2015 launch records

- Query execution:



```
SELECT MONTH(DATE) AS MONTH, BOOSTER_VERSION, LAUNCH_SITE, LANDING__OUTCOME
  FROM SPACEX
  WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND YEAR(DATE) = 2015
```

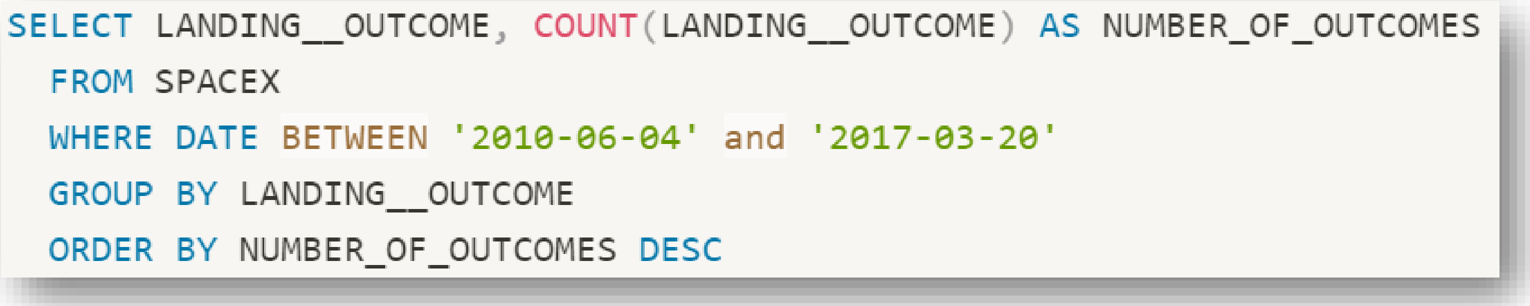
- Result: The records of failed landing outcomes in drone ship along with their, booster versions, and launch site for the months in year 2015



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

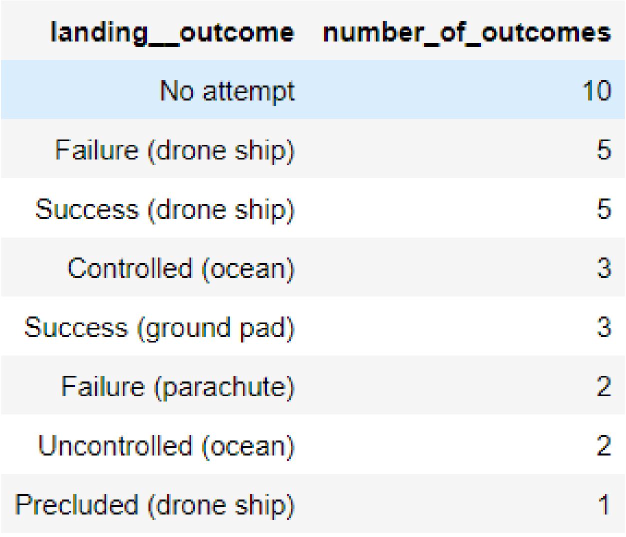
Rank success count between 2010-06-04 and 2017-03-20

- Query execution:



```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS NUMBER_OF_OUTCOMES
  FROM SPACEX
  WHERE DATE BETWEEN '2010-06-04' and '2017-03-20'
  GROUP BY LANDING__OUTCOME
  ORDER BY NUMBER_OF_OUTCOMES DESC
```

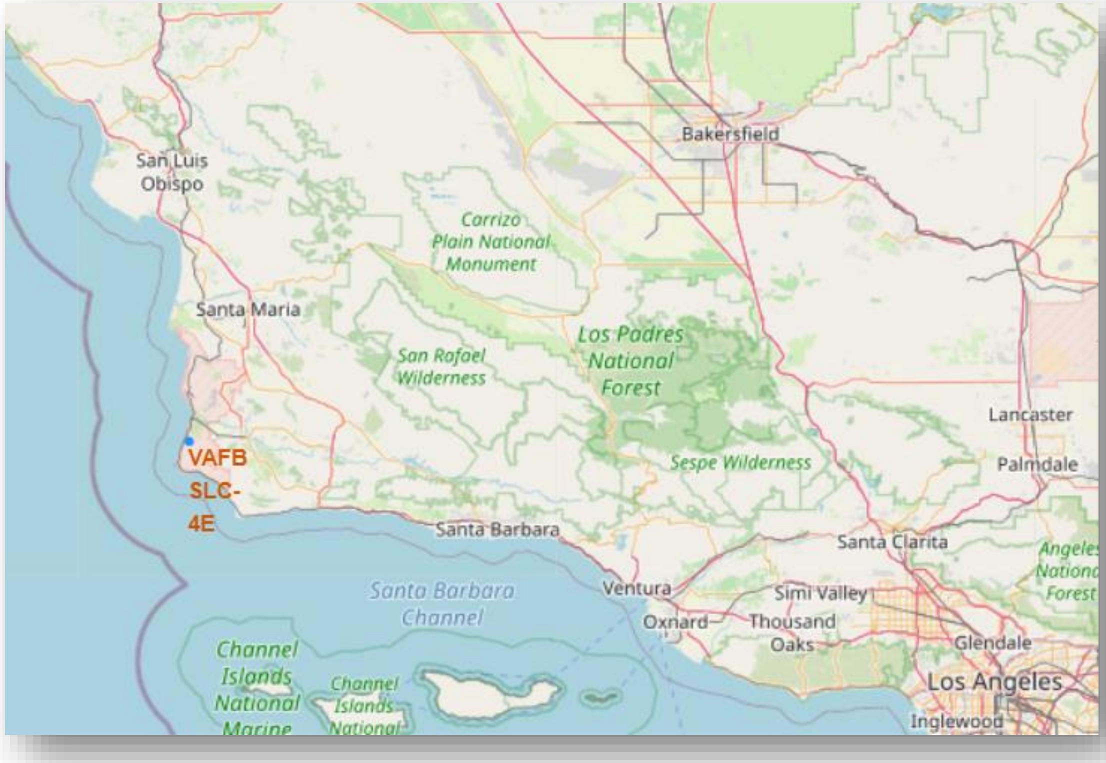
- Result: The number of successful landing outcomes between June 4th, 2010, to March 20th, 2017, in descending order

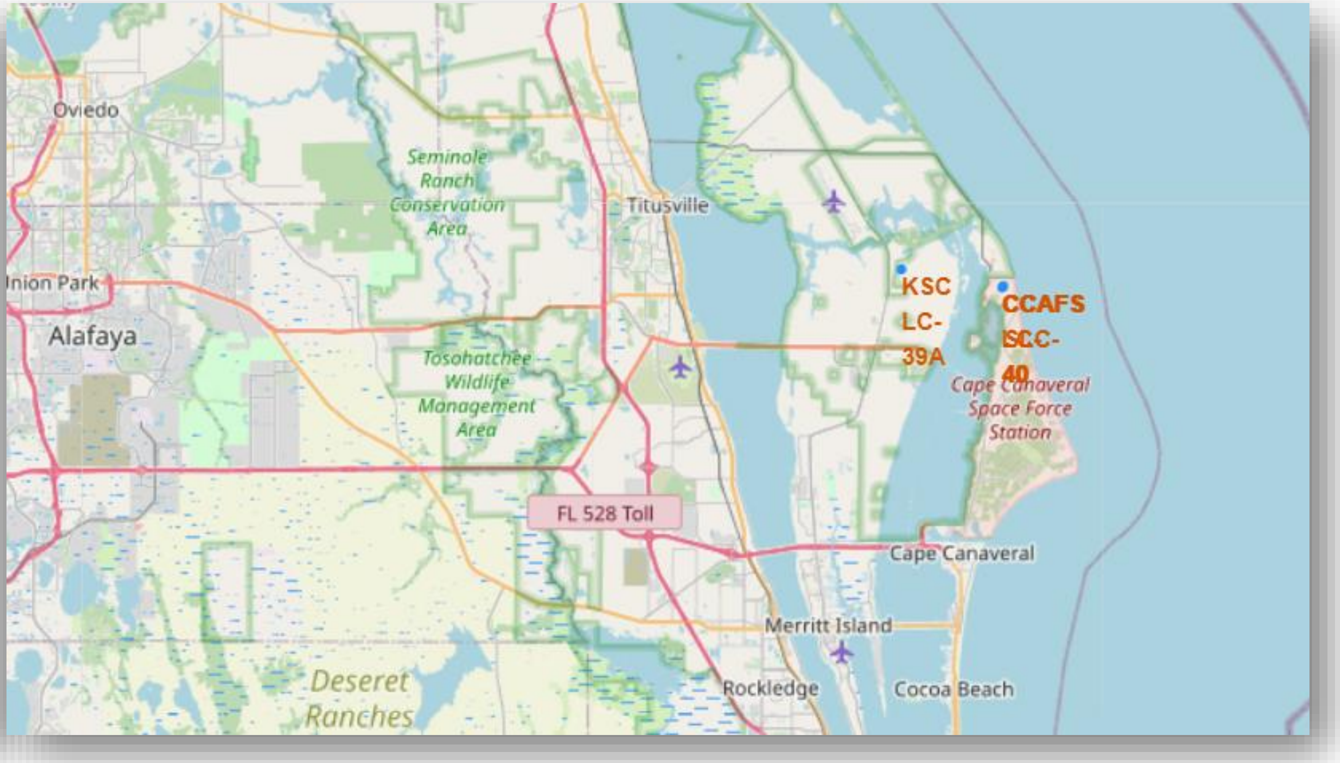


landing__outcome	number_of_outcomes
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Interactive map with Folium

Folium map with all launch sites marked

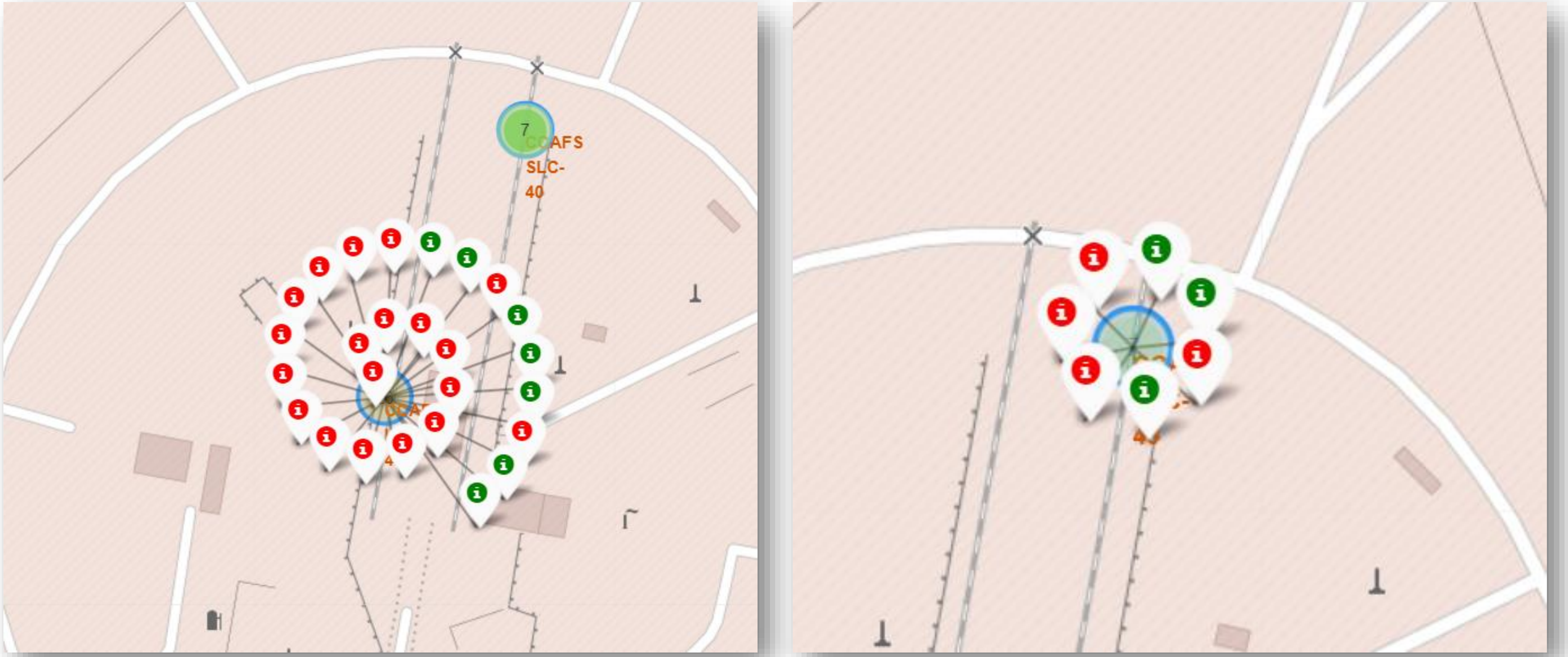




Note that CCFAS SLC-40 and CCFAS LC-40 are the same launch sites

Folium map with the success/failed launches for each site

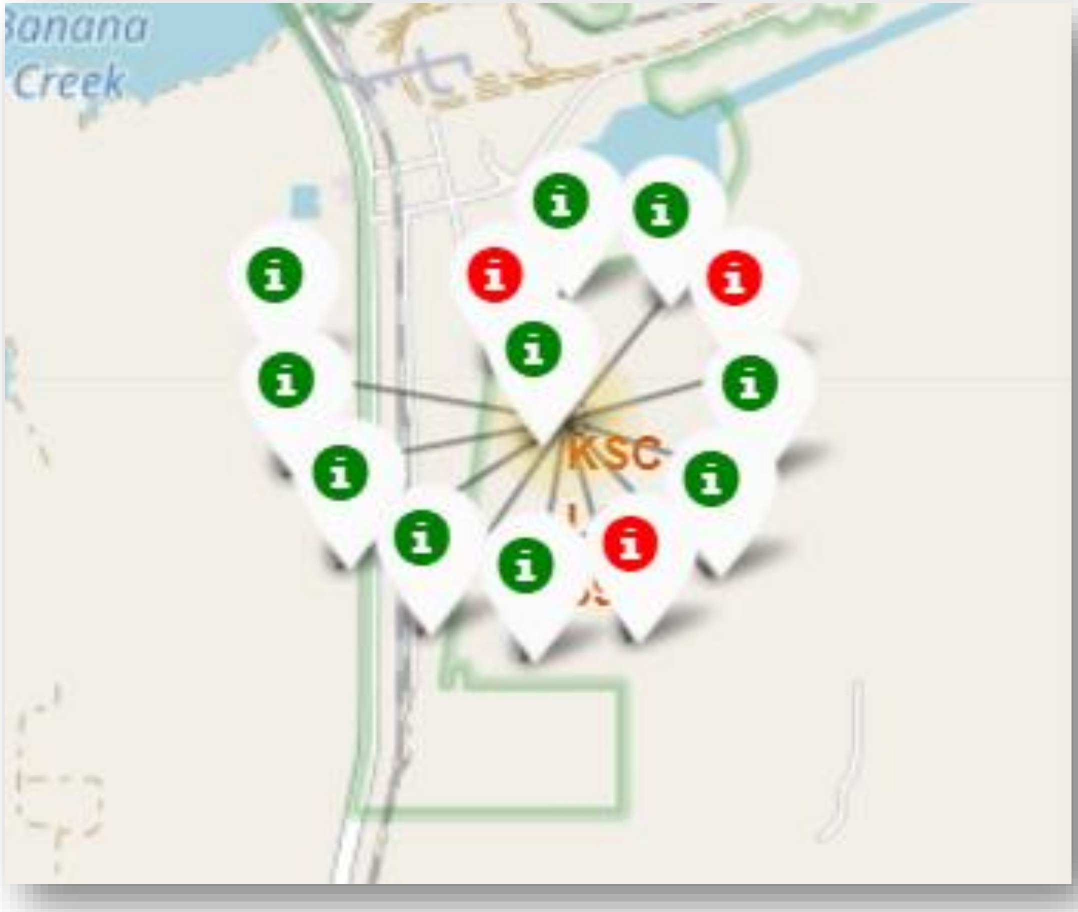
CCAFS LC-40/CCAFS SLC-40



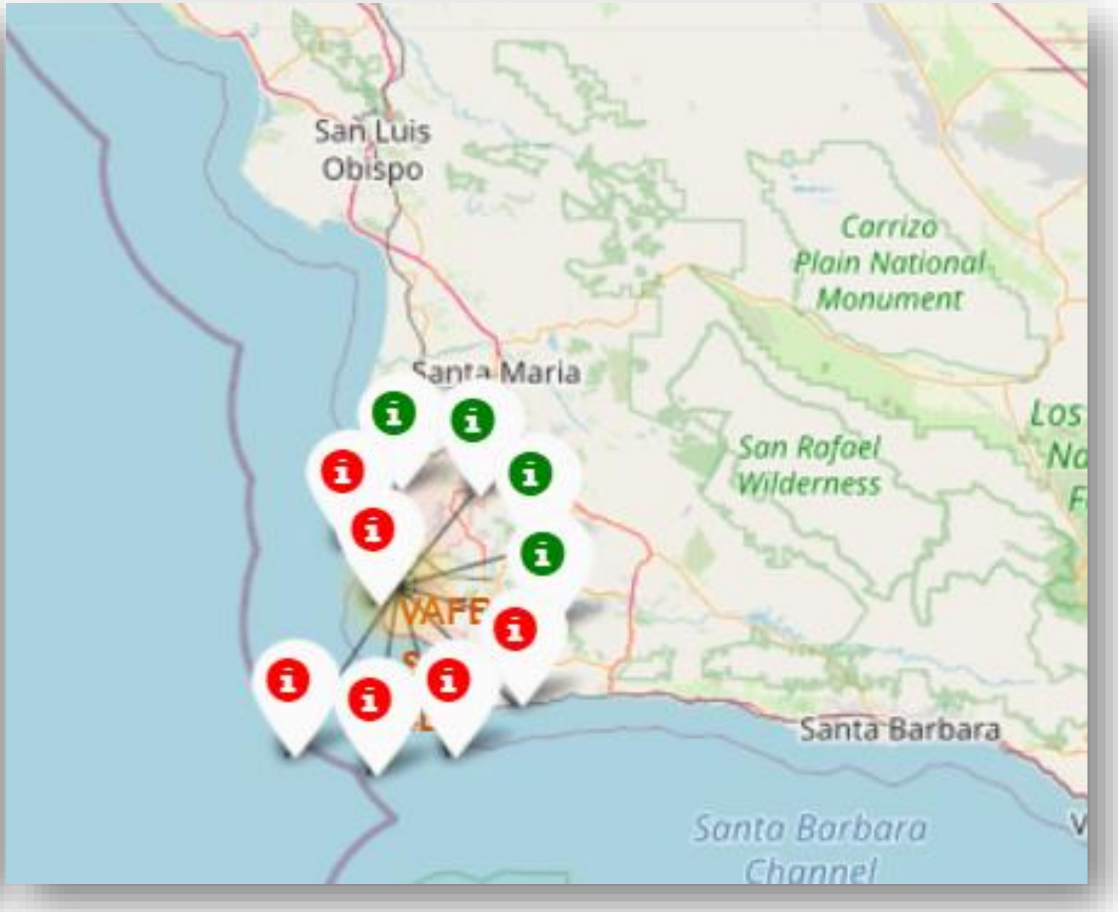
- Green = successful launch
- Red = unsuccessful launch

Folium map with the success/failed launches for each site (continued…)

KSC LC-39A

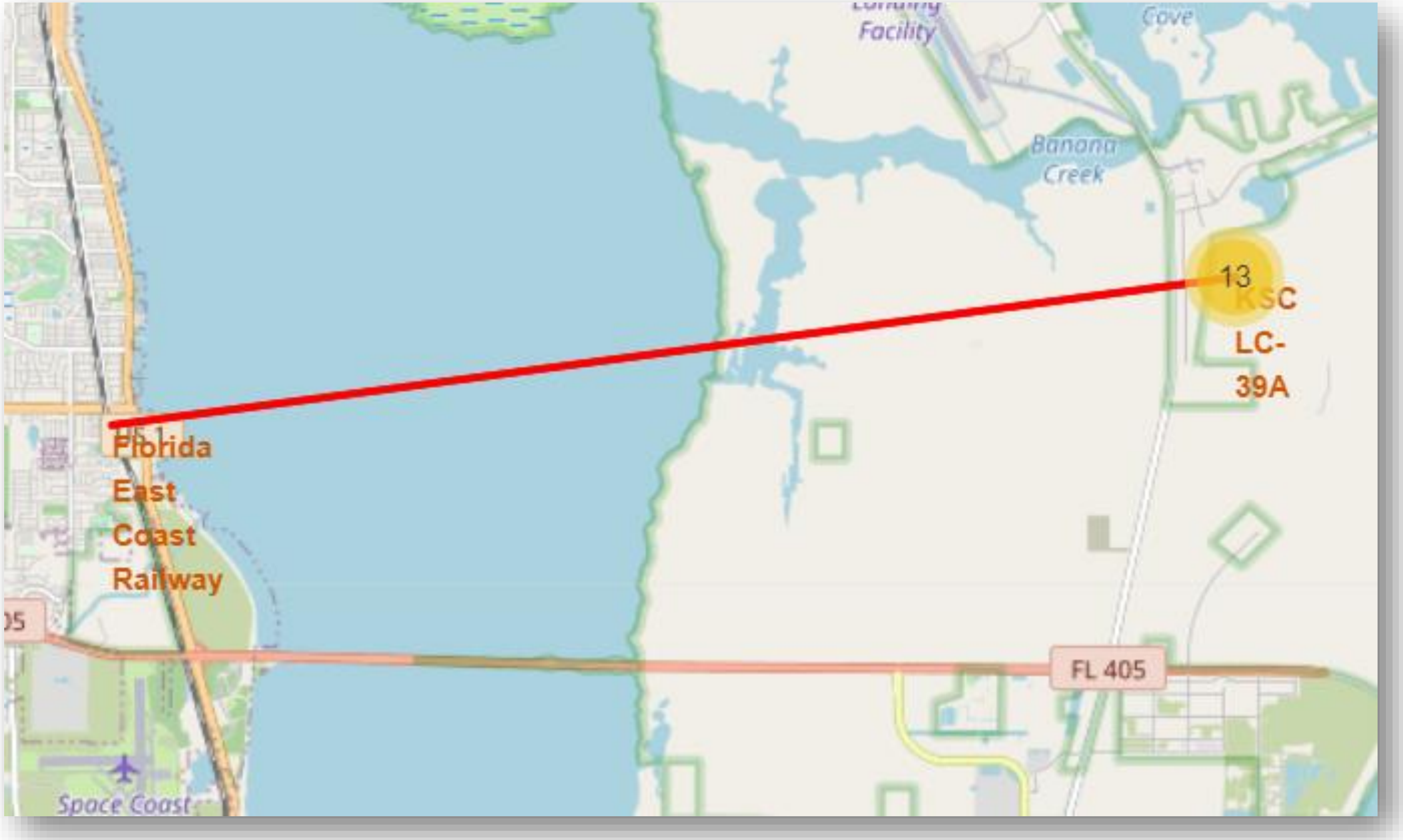


VAFB SLC-4E



- Green = successful launch
- Red = unsuccessful launch

Map illustrating the distances between KSC LC-39A and the Florida East Coast Railway

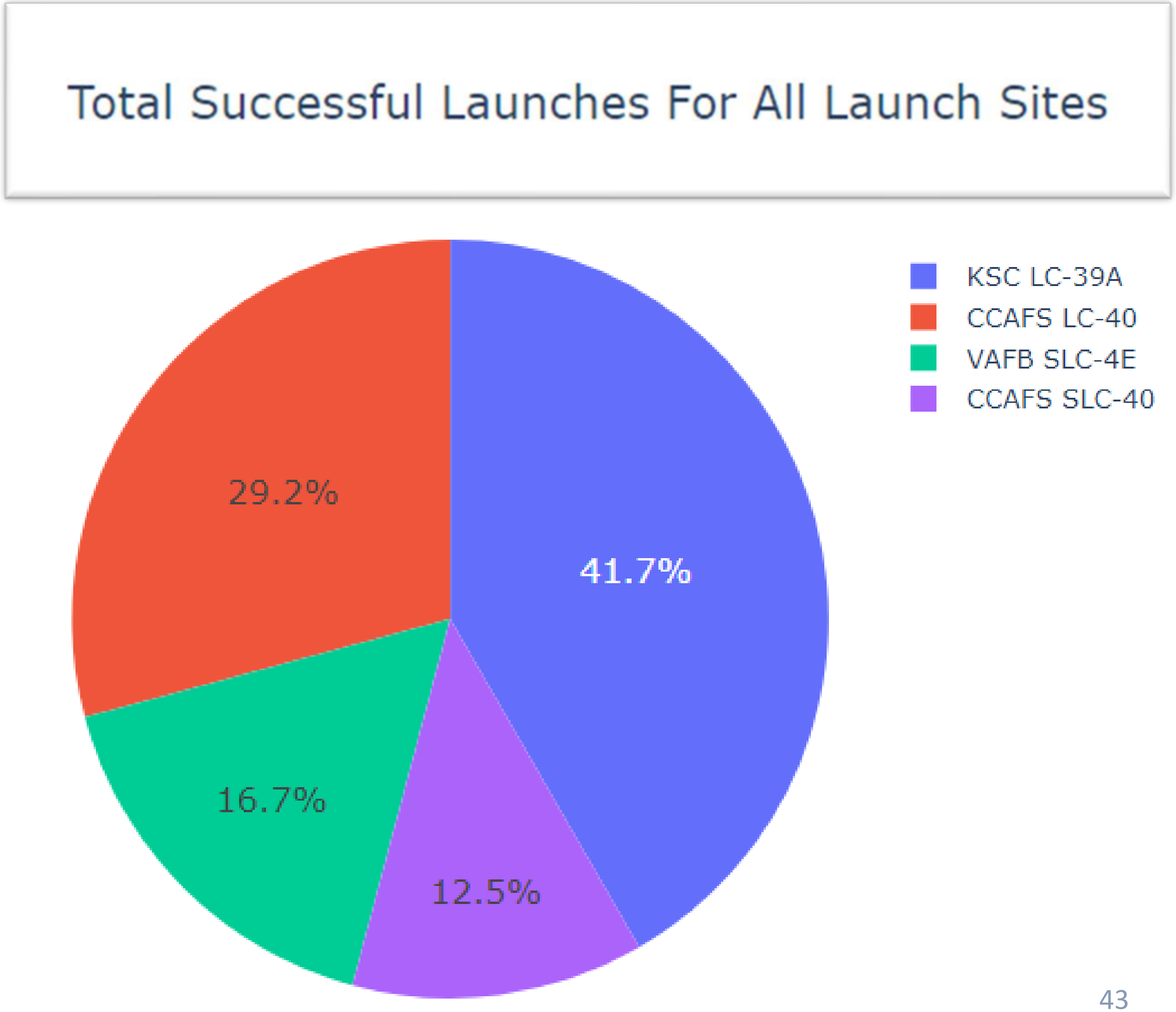


The distance between FECR and KSC LC-39A was calculated to be 15.15km

Build a Dashboard with Plotly Dash

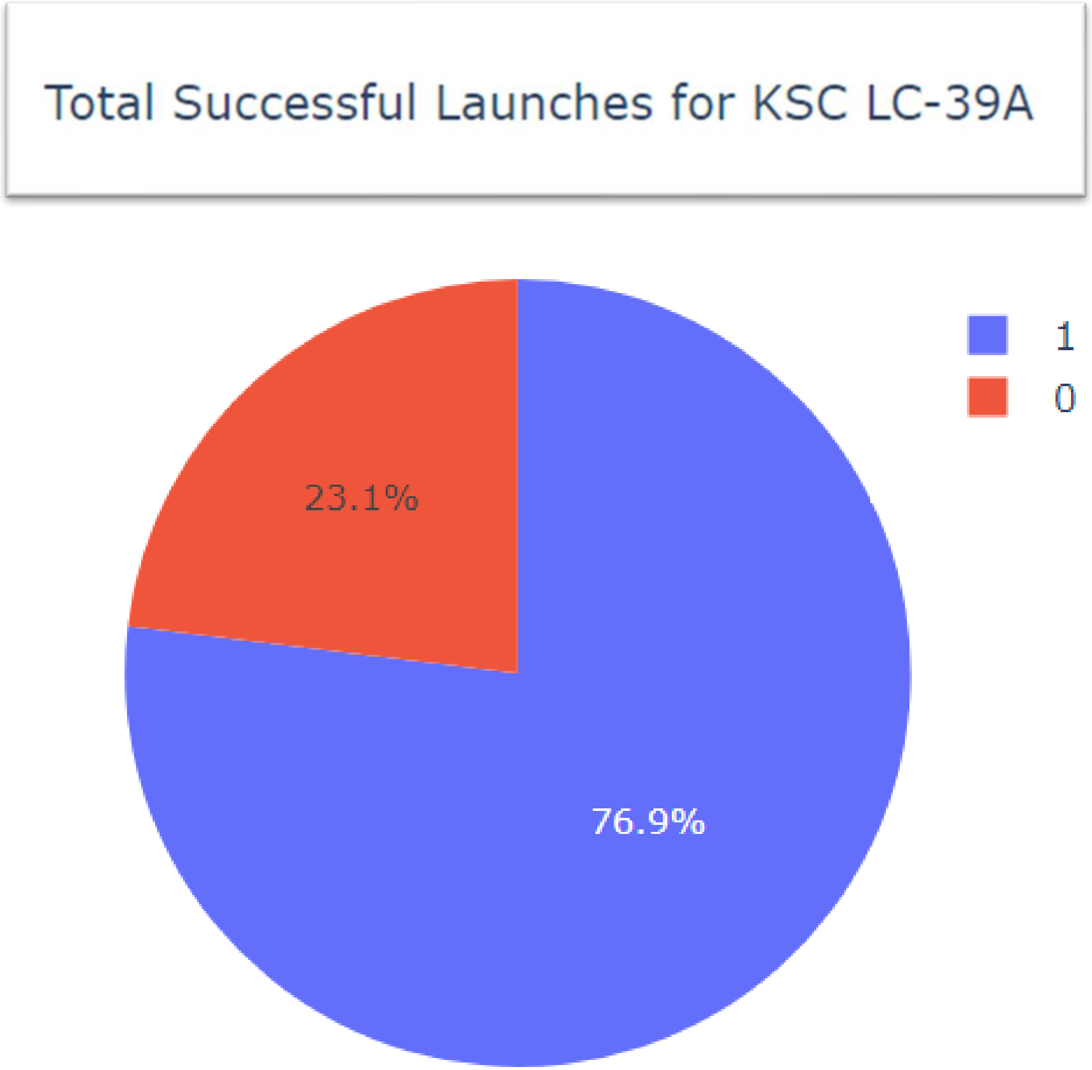
Pie Chart of Launches for All Sites

This pie chart displays the number of successful launches for each site. Such that KSC LC-39A is shown to have the most successful while CCAFS SLC-40 has had the least success.



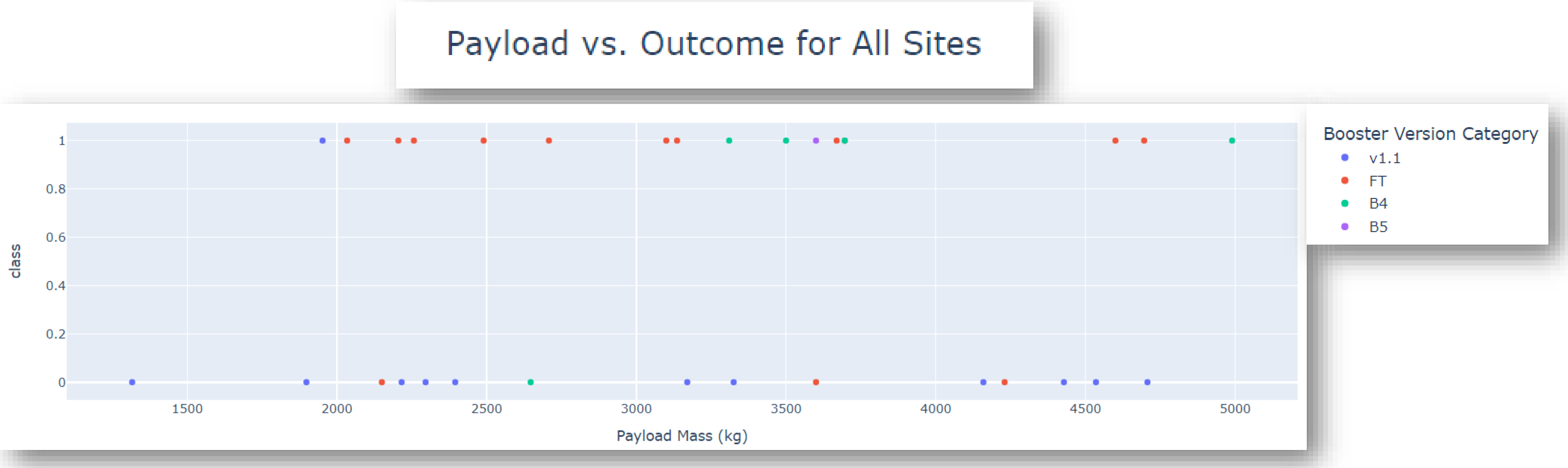
Pie Chart of KSC LC-39A Launches

In this pie chart we focus on launch site KSC LC-39A since this site had the highest ratio of successes. The blue are repersents the succesful class (i.e. class 1) while the red area is the unsuccessful class (i.e. class 0)



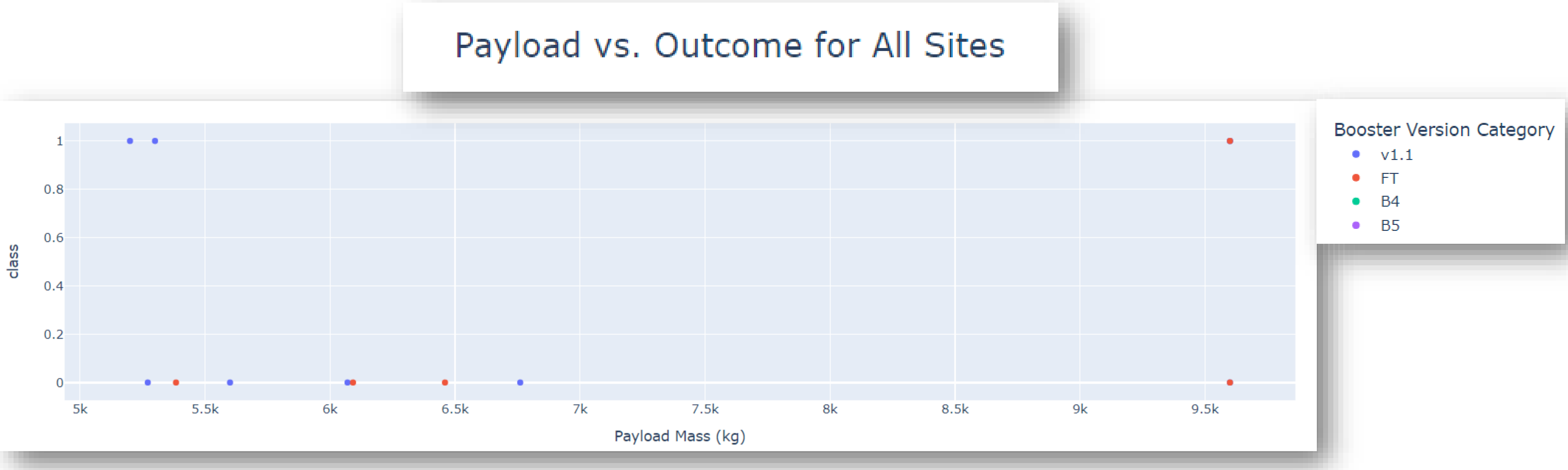
Scatter Plot of Payload vs Launch Outcome

- The first chart displays the launch outcomes for payloads weigh between 1000 – 5000kg's. In addition, the launch outcomes are further cateogrized by their booster version

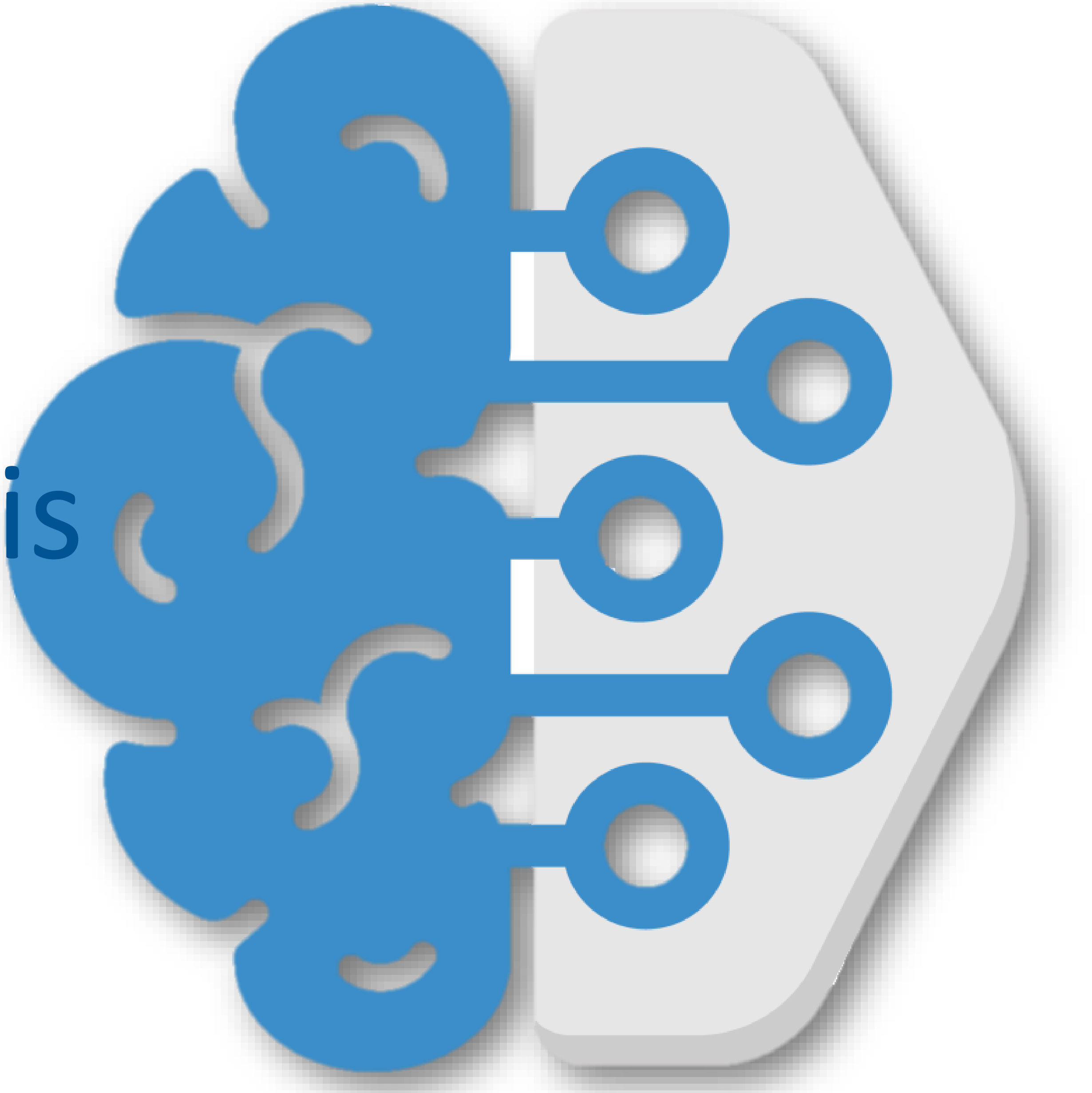


Scatter Plot of Payload vs Launch Outcome (cont..)

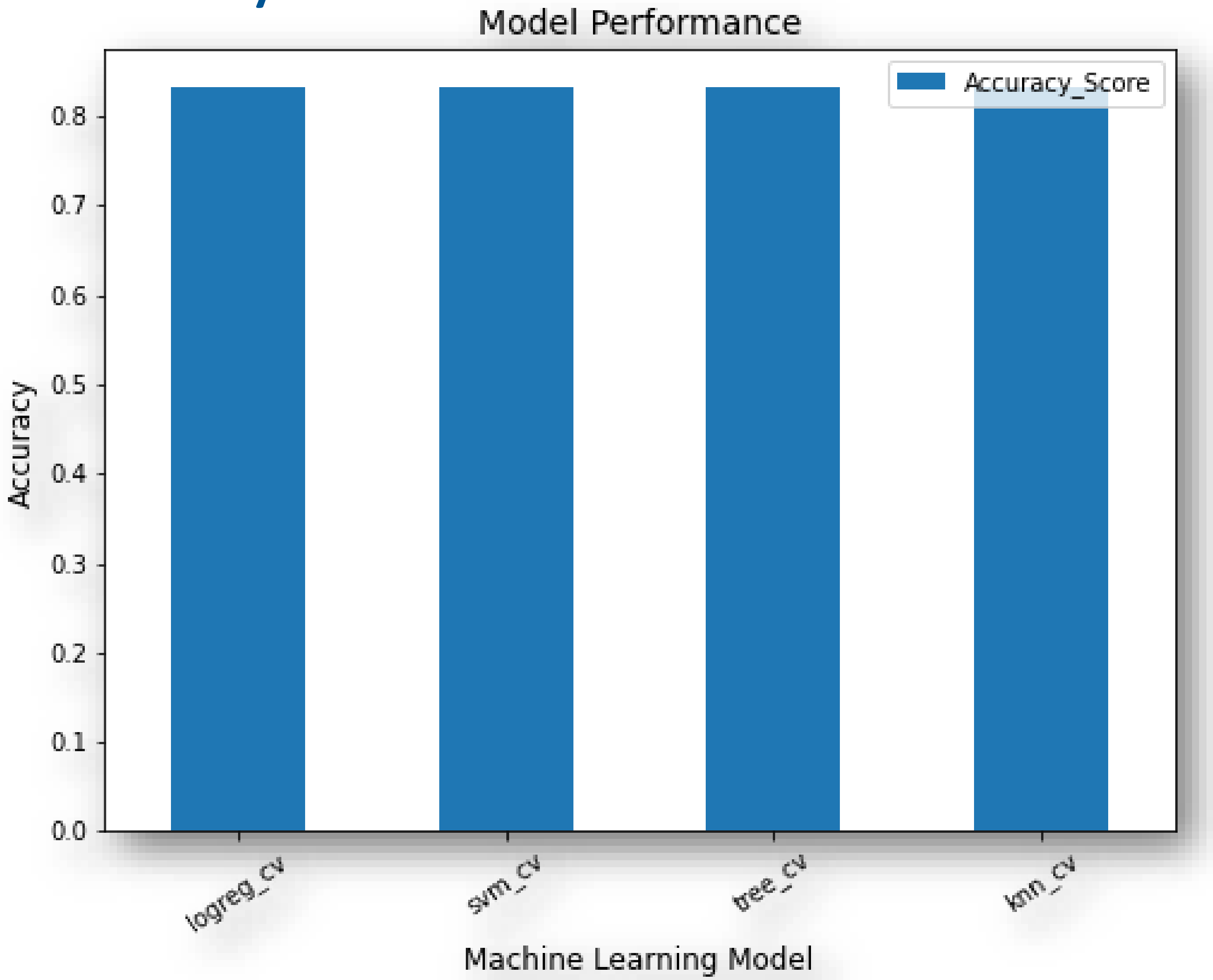
- In this chart we display the same information but for payloads weighing between 5000 – 10,000 kg’s



Predictive Analysis (Classification)



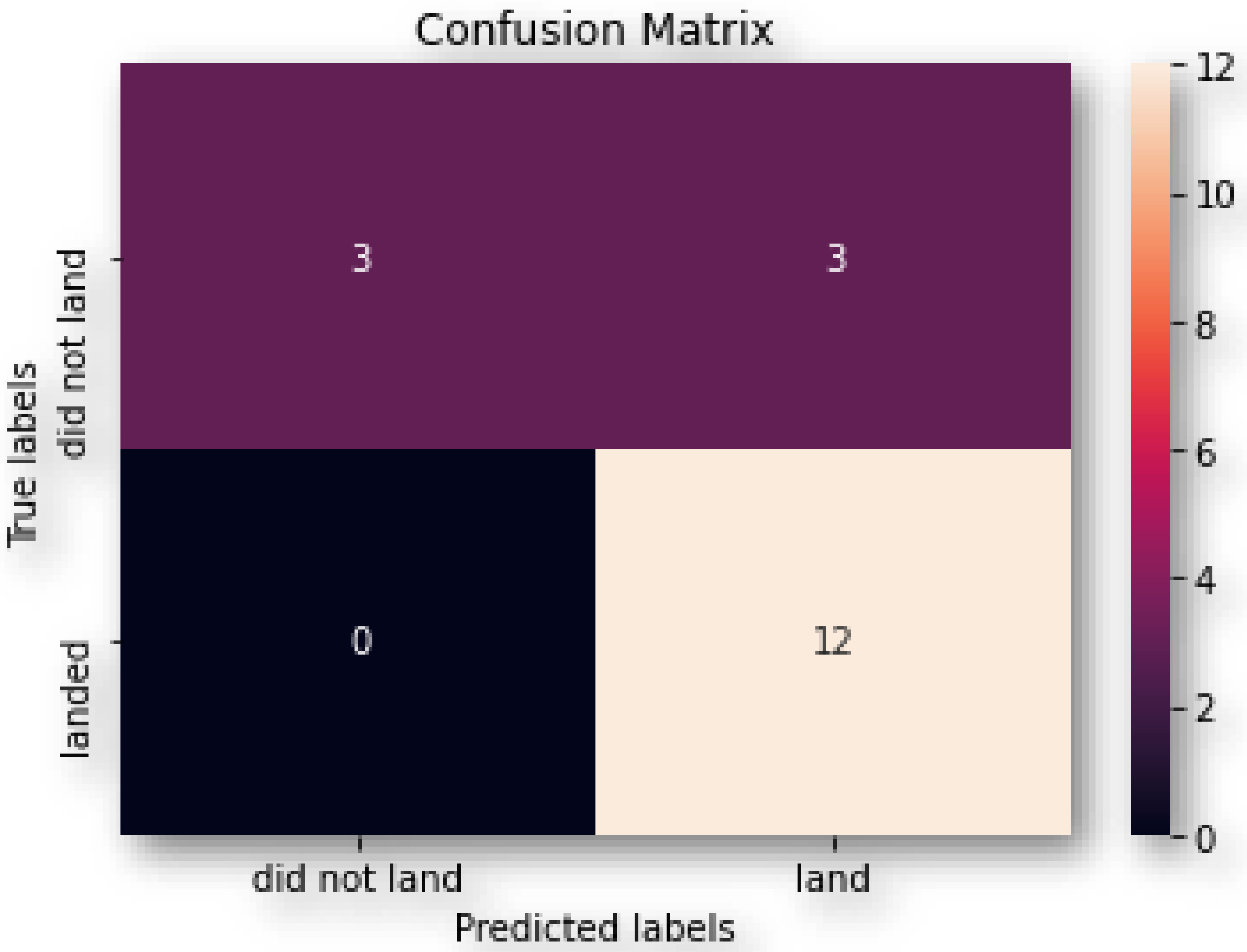
Classification Accuracy



All models performed equally with an accuracy score of 0.8333 or 83.33%

Confusion Matrix

When examining the confusion matrix, we can see that the True labels are on the y-axis while predicted labels are on the x-axis. True labels represent the actual labels in the dataset (i.e., y-test) and as the name suggests predicted labels are those predicted from our model (determined from X_test). Furthermore, if we look closer, we can see that our model performed very well ,however, it shows some difficulty when trying to predict when the Falcon 9 landed when it in fact did not land. This type of error is called a false positive.



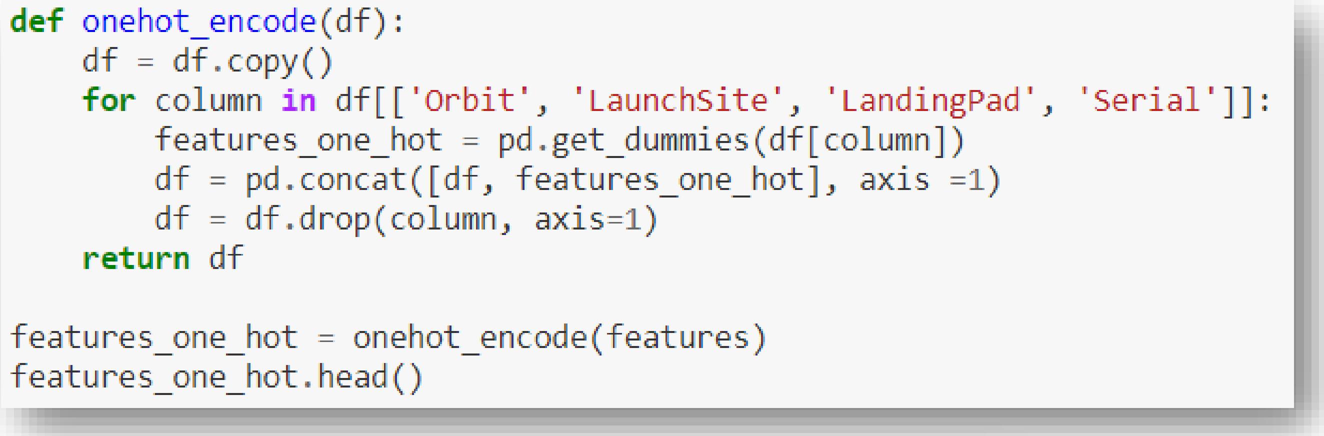
CONCLUSION

- The purpose of this project was to examine the relationship between several rocket launch features and how they potentially impact the landing outcome of Falcon 9 first stage. Using various ML model's, we were able to determine the landing outcome with 83% accuracy.
- The results from this project can help those who are interested in increasing their chances of producing landing outcomes with the best success.
- Finally, by determining whether the first stage will land successfully, findings from this project will help relevant stakeholders and companies to compete with SpaceX.

Appendix

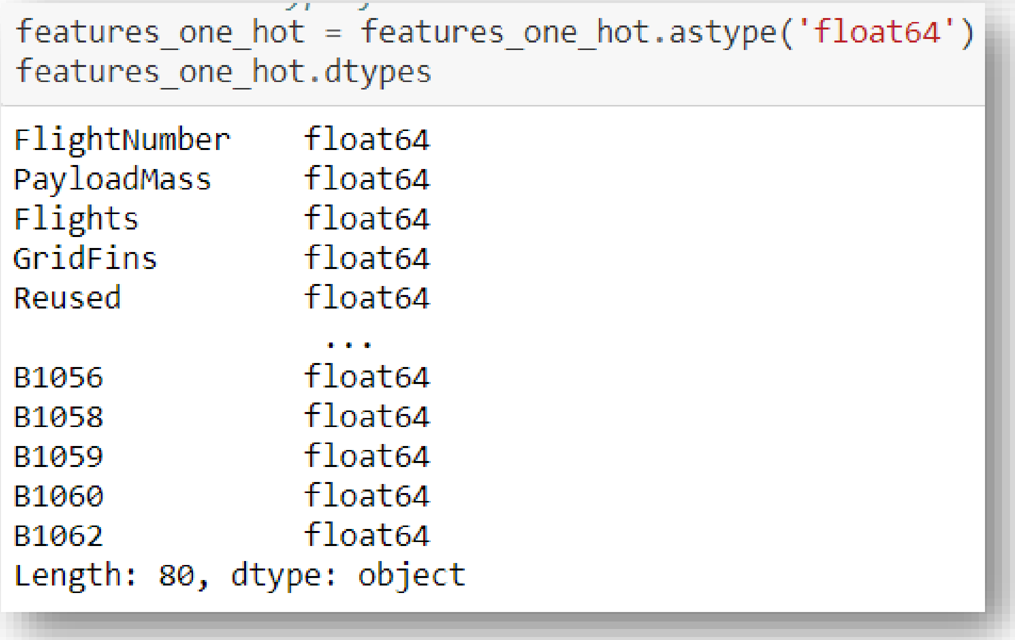
Feature Engineering

1. Create dummy variables for categorical columns



```
def onehot_encode(df):
    df = df.copy()
    for column in df[['Orbit', 'LaunchSite', 'LandingPad', 'Serial']]:
        features_one_hot = pd.get_dummies(df[column])
        df = pd.concat([df, features_one_hot], axis =1)
        df = df.drop(column, axis=1)
    return df

features_one_hot = onehot_encode(features)
features_one_hot.head()
```

2. Cast all numeric columns to ‘float64’

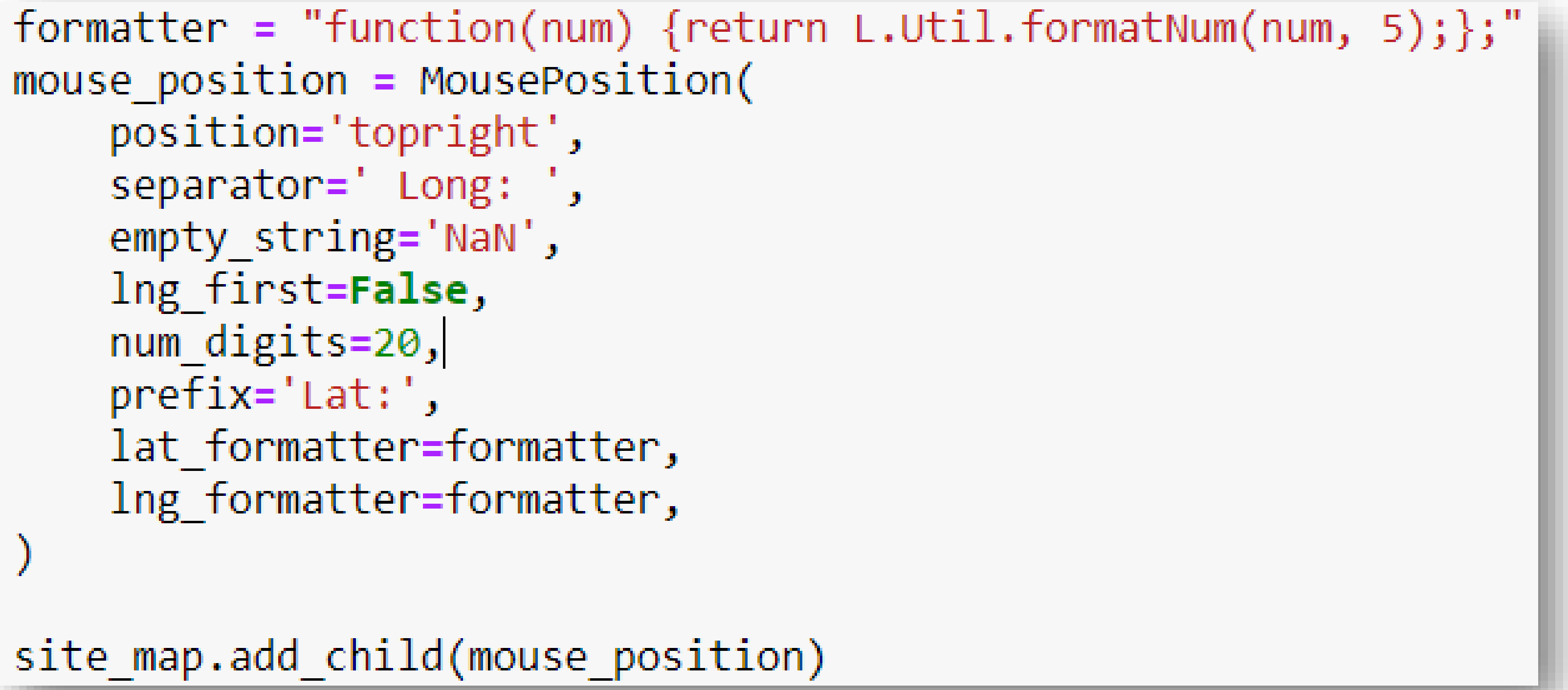


```
features_one_hot = features_one_hot.astype('float64')
features_one_hot.dtypes
```

```
FlightNumber    float64
PayloadMass     float64
Flights         float64
GridFins        float64
Reused          float64
                  ...
B1056           float64
B1058           float64
B1059           float64
B1060           float64
B1062           float64
Length: 80, dtype: object
```

Folium Maps

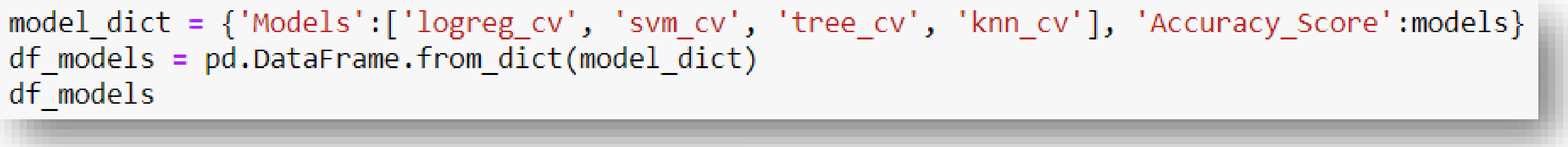
Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map



```
formatter = "function(num) {return L.Util.formatNum(num, 5);};"
mouse_position = MousePosition(
    position='topright',
    separator=' Long: ',
    empty_string='NaN',
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)

site_map.add_child(mouse_position)
```

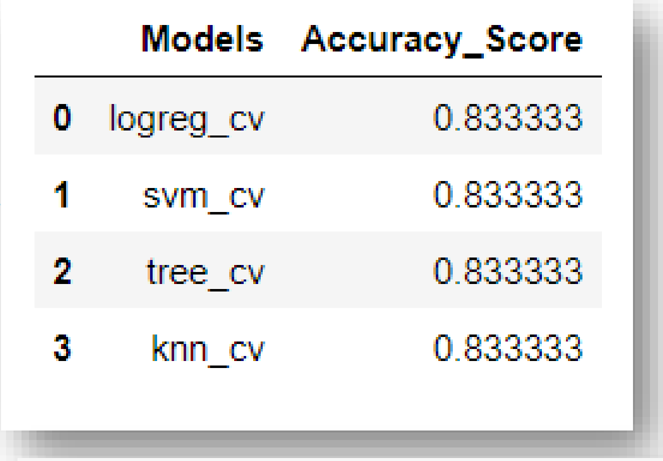
Finding the Best Model

1. Create a dictionary containing model names and their accuracy score



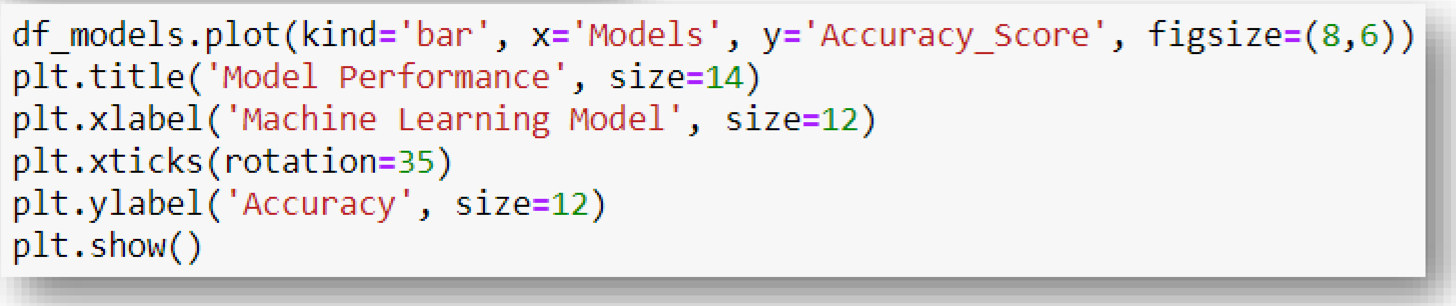
```
model_dict = {'Models':['logreg_cv', 'svm_cv', 'tree_cv', 'knn_cv'], 'Accuracy_Score':models}
df_models = pd.DataFrame.from_dict(model_dict)
df_models
```

2. Convert the dictionary to a DataFrame to easily plot



	Models	Accuracy_Score
0	logreg_cv	0.833333
1	svm_cv	0.833333
2	tree_cv	0.833333
3	knn_cv	0.833333

3. Plot model names on the x-axis and accuracy score on the y-axis



```
df_models.plot(kind='bar', x='Models', y='Accuracy_Score', figsize=(8,6))
plt.title('Model Performance', size=14)
plt.xlabel('Machine Learning Model', size=12)
plt.xticks(rotation=35)
plt.ylabel('Accuracy', size=12)
plt.show()
```