# IBM DATA SCIENCE CAPSTONE PROJECT

SPACEX LAUNCH
Case Study

By Matthew Walfish

# OUTLINE

SUMMARY

INTRODUCTION

METHODOLOGY

RESULTS

CONCLUSION

APPENDIX

Section 1:

# SUMMARY

- METHODOLOGIES USED:
  - *DATA COLLECTION*
  - *DATA WRANGLING*
  - *DATA VISUALIZATION*
  - *EXPLORATORY DATA ANALYSIS*
  - *FOLIUM MAP*
  - *DASH APP*
  - *CLASSIFICATION ANALYSIS*

- RESULTS SUMMARY:
  - *PRELIMINARY ANALYSIS*
  - *INTERACTIVE MAPS & DASHBOARDS*
  - *PREDICTIVE RESULTS*

Section 2:

# INTRODUCTION

~SpaceX advertises the Falcon 9 Rocket Launch to cost $62mil.

~Other competing providers have reported their costs per launch near $165mil.

~SpaceX reduces their mission costs by reusing the first stage of its launch; a propulsion system, in simple terms.

~My goal is to predict if the Falcon 9 Rocket Launch First Stage will successfully land back onto its ground base.

~Determining the success of this first stage helps in estimating and determining the final cost of each mission.

~This project takes a study at a variety of factors and variables that might affect the best results for each launch, and the overall success of each mission.

Section 3:

# METHODOLOGY

# Methodology:

Part 1: Data Collection Methods:
- SpaceX Rest API
- Web Scraping from Wikipedia

Part 2: Data Wrangling
- One Hot Encoding data fields for Machine Learning

Part 3: Data Analysis & Visualization
- Scatter Graphs & Bar Graphs show the relationships between different variables to show patterns of data

Part 4: Interactive Visual Analytics
- Folium
- Plotly Dash

Part 5: Predictive Analysis using Classification Models
- Building, Tuning, and Evaluating the models

# METHODOLOGY, PART 1: DATA COLLECTION METHODS

## Step 1:

Using the SpaceX Rest API, the following information was gathered:

- Launch
- Rockets used
- Payload delivered
- Launch specifications
- Landing outcomes.

## Step 2:

Obtaining Launch data through Wikipedia was conducted using BeautifulSoup.

# DATA COLLECTION METHOD, STEP 1: SPACEX REST API

- Get response from API
- Convert to .json file
- Clean Data
- Create dataframe

# DATA COLLECTION METHODS, STEP 2: WEB SCRAPING WITH BEAUTIFULSOUP

- Get response from HTML
- Create BeautifulSoup Object
- Find tables/column names
- Create dataframe
- Convert to CSV

*Click here for Lab 1: Data Collection*

# METHODOLOGY, PART 2: DATA WRANGLING

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

# Part 2: Data Wrangling, cont.

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

Perform Exploratory Data Analysis on Dataset

Calculate the number of launches at each site

Calculate the number of mission outcome per orbit type

Calculate the number and occurrence of each orbit

Create a landing outcome landing from Outcome column

Work out success rate for every landing in dataset

Export dataset to csv

Click here for Lab 2: Lab Wrangling

# METHODOLOGY, PART 3: EXPLORATORY DATA ANALYSIS

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

# DATA VISUALIZATION

Scatter Graphs

- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Orbit vs. Flight Number
- Payload vs. Orbit Type
- Orbit vs. Payload Mass

Bar Graphs:

- Mean vs. Orbit

Line Graphs:

Success Rate vs. Year

# EDA WITH SQL

Performed the following SQL Queries:

- Display the names of the unique launches in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the failed landing_outcomes in drone ship, their booster versions, and launch_site for months in 2017
- Rank the count of landing_outcomes (such as Failure (drone ship) or Success (ground pad) between June 4h, 2010-March 20th, 2017

Click here for Jupyter Notebook: Lab 4: EDA with SQL

# METHODOLOGY, PART 4: INTERACTIVE VISUAL ANALYTICS

Interactive Map w/Folium

Interactive Dashboard w/Flask & Dash

# FOLIUM

Click here for Lab 5: Visual Analytics

The following objects were added to Folium:

- Map Object – center location is NASA Johnson Space Center, Houston, TX
- Blue Circle indicates J.S.C., w/a popup label to indicate
- Each launch site has a circle based on (Lat, Long) coordinates
- Markers for all launch records
- Marker Clusters representing different grouped points with different information
- Distance between launch sites and various plots and line distances

# DASH APP & PLOTLY

Click here for SpaceX Dashboard

Plots and Graphs And Interactions were added to a Dashboard
via Plotly

Plot Charts:

- Total Launches by specific site, or all launch sites
- Displays launch information
- Pie Charts to represent the indivdual launch site success vs failures

Scatter Plots:

- Relationships between variables
- Slider for Payload Mass

# METHODOLOGY, PART 5: CLASSIFICATION ANALYSIS

## Building the Model:

- Load dataset into NumPy and Pandas
- Transform data
- Split our data into training and test sets
- Find how many test samples are present
- Pick ML Algorithms
- Use GridSearchCV to set parameters and algorithms
- Fit our dataset into objects and train dataset

## Evaluate Model:

- Check accuracy
- Fine-tune parameters and algorithms
- Plot confusion matrix

## Improving Model:

- Feature Engineering
- Algorithm Tuning

## Finding Classification Model:

- Best accuracy score === best performing model
- Jupyter Notebook has dictionary of algorithms

Click here for link to source code

Section 4:

# RESULTS

# EXPLORATORY DATA ANALYSIS: DATA VISUALIZATION

# Flight Number vs. Launch Site

jupyter jupyter-labs-eda-dataviz Last Checkpoint: 09/20/2021 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Logout

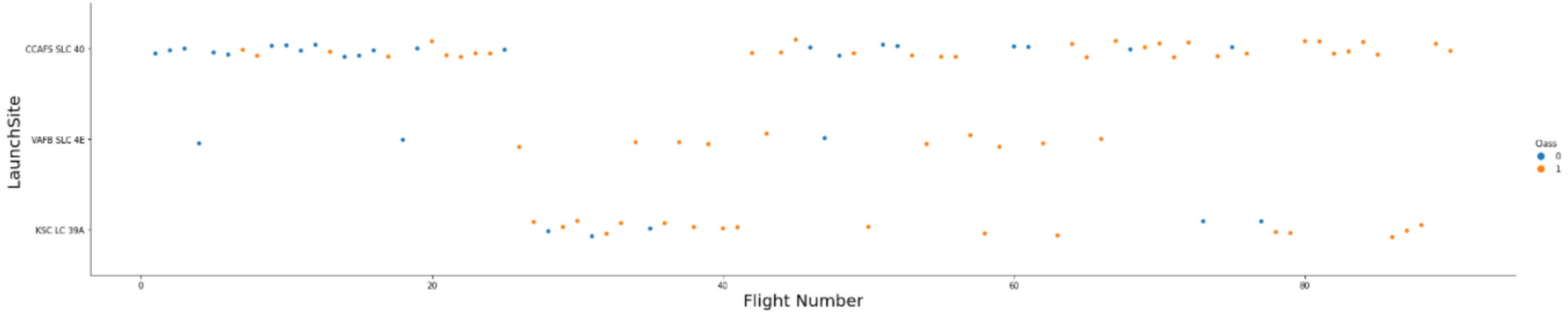
Run Markdown

### TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

In [14]:

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```




```
Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

THE OUTCOME IMPROVES, AS THE AMOUNT OF LAUNCHES GROWS.
```

# Payload vs. Launch Site

jupyter jupyter-labs-eda-dataviz Last Checkpoint: 09/20/2021 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

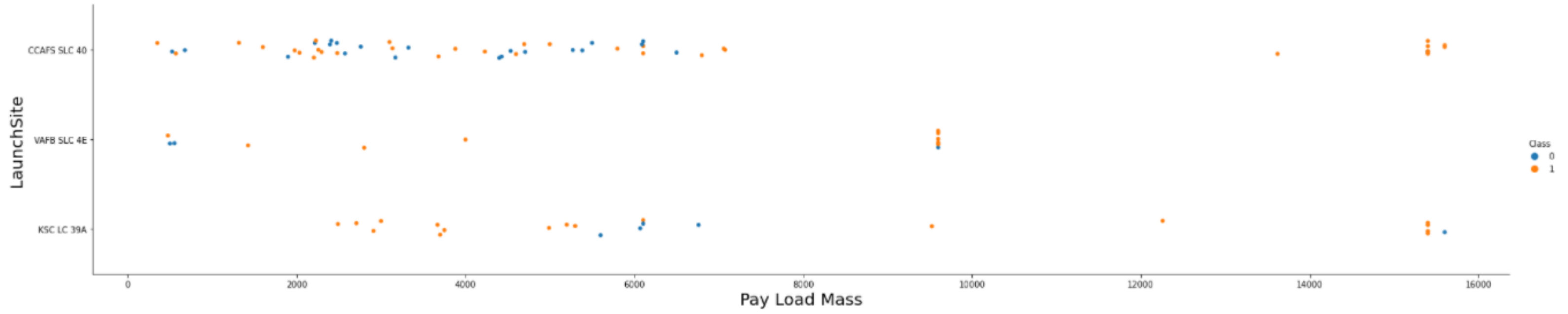
Run Markdown

### TASK 2: Visualize the relationship between Payload and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

In [5]:

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay Load Mass",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```




```
Now try to explain any patterns you found in the Payload Vs. Launch Site scatter point chart.

SUCCESS RATE IMPROVES WITH THE INCREASE OF PAYLOAD, LIKE AT SITE SLC40
```

In [ ]:

# Success Rate vs. Orbit Type

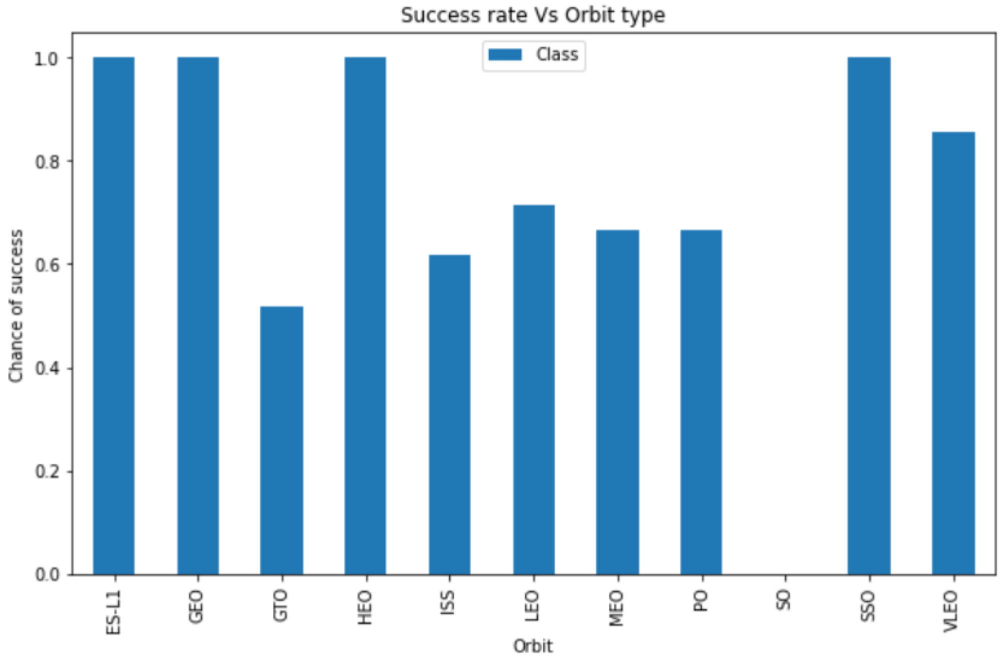
jupyter jupyter-labs-eda-dataviz Last Checkpoint: 09/20/2021 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Markdown




```
Analyze the ploted bar chart try to find which orbits have high sucess rate.

ES-L1, GEO, HEO, SSO have the best success rates
```

# Flight Number vs. Orbit Type

jupyter jupyter-labs-eda-dataviz Last Checkpoint: 09/20/2021 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

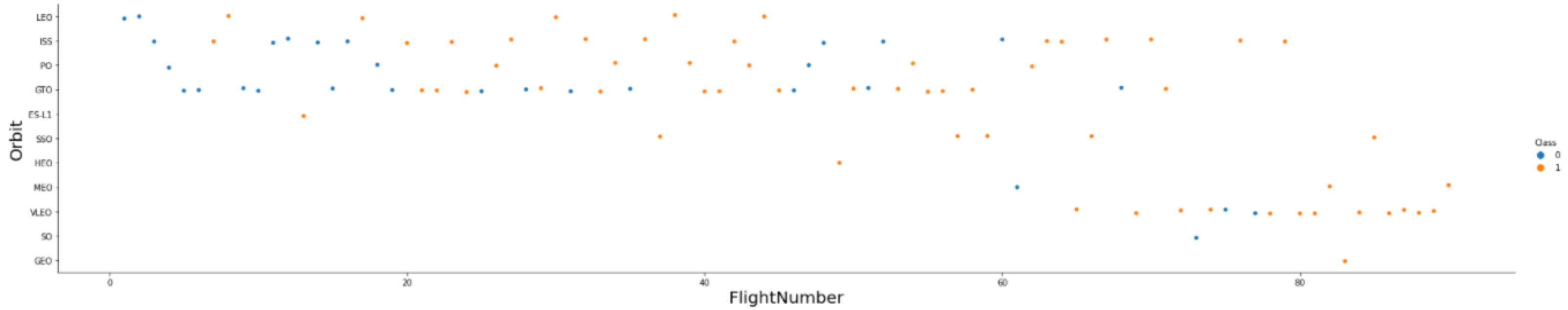
Run Markdown

### TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [7]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```




```
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be
no relationship between flight number when in GTO orbit.

SUCCESS RATE IMPROVES WTIH EACH MISSION.
```

# Payload vs. Orbit Type

jupyter jupyter-labs-eda-dataviz Last Checkpoint: 09/20/2021 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

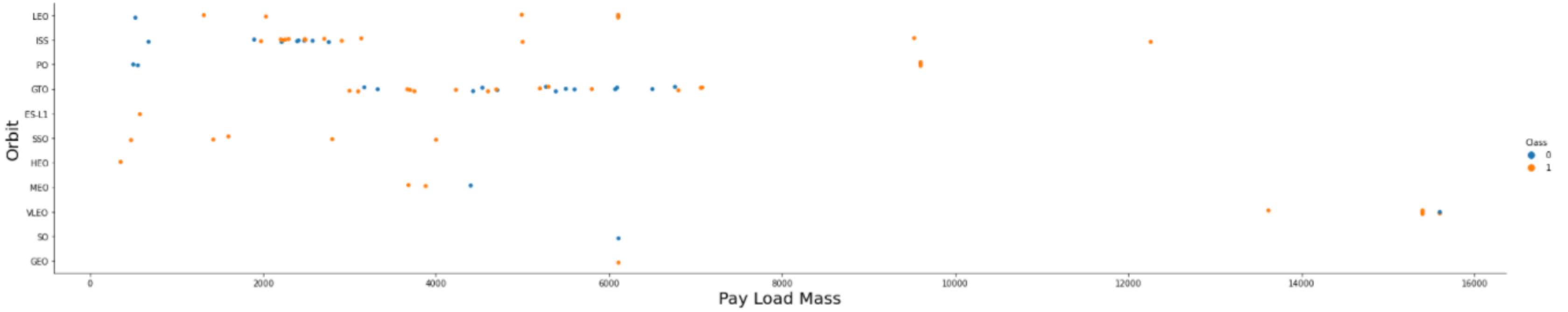
```
SUCCESS RATE IMPROVES WTIH EACH MISSION.
```

### TASK 5: Visualize the relationship between Payload and Orbit type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

In [8]:

```python
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay Load Mass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```




You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

# Launch Success Yearly Trend

jupyter jupyter-labs-eda-dataviz Last Checkpoint: 09/20/2021 (autosaved)

Logout

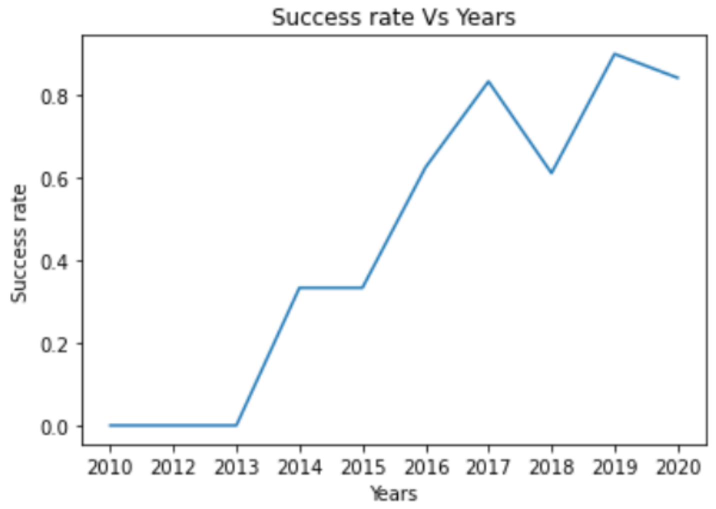
File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Markdown

In [10]:

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
plt.plot(df_y['Year'],df_y['Class'])
plt.title('Success rate Vs Years')
plt.xlabel('Years')
plt.ylabel('Success rate')
plt.show()
```




you can observe that the sucess rate since 2013 kept increasing till 2020

# All Launch Site Names

### Task 1

***Display the names of the unique launch sites in the space mission***

In [4]:
```
%sql SELECT DISTINCT launch_site from SPACEX
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Out[4]:

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

# Launch Site Names Begin with 'KSC'

### Task 2

***Display 5 records where launch sites begin with the string 'KSC'***

In [18]:
```
%sql SELECT * from SPACEX WHERE launch_site LIKE 'KSC%' LIMIT 5
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Out[18]:

| DATE | time__utc_ | booster_version | launch_site | payload | payload_mass__kg_ | orbit | customer | mission_outcome | landing__outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2017-02-19 | 14:39:00 | F9 FT B1031.1 | KSC LC-39A | SpaceX CRS-10 | 2490 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 2017-03-16 | 06:00:00 | F9 FT B1030 | KSC LC-39A | EchoStar 23 | 5600 | GTO | EchoStar | Success | No attempt |
| 2017-03-30 | 22:27:00 | F9 FT B1021.2 | KSC LC-39A | SES-10 | 5300 | GTO | SES | Success | Success (drone ship) |
| 2017-05-01 | 11:15:00 | F9 FT B1032.1 | KSC LC-39A | NROL-76 | 5300 | LEO | NRO | Success | Success (ground pad) |
| 2017-05-15 | 23:21:00 | F9 FT B1034 | KSC LC-39A | Inmarsat-5 F4 | 6070 | GTO | Inmarsat | Success | No attempt |

# Total Payload Mass

### Task 3

***Display the total payload mass carried by boosters launched by NASA (CRS)***

In [10]:
```
%sql SELECT SUM(payload_mass__kg_) FROM SPACEX WHERE customer LIKE 'NASA%'
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Out[10]:

| 1 |
|---|
| 99980 |

# Average Payload Mass by F9 v1.1

## Task 4

***Display average payload mass carried by booster version F9 v1.1***

In [11]:
```
%sql SELECT AVG(payload_mass__kg_) FROM SPACEX WHERE booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Out[11]:

| 1 |
|---|
| 2928 |

# First Successful Ground Landing Date

## Task 5

***List the date where the succesful landing outcome in drone ship was acheived.***

*Hint:Use min function*

In [12]:
```
%sql SELECT MIN(DATE) FROM SPACEX where LANDING__OUTCOME = 'Success (ground pad)'
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Out[12]:

| 1 |
|---|
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

## Task 6

***List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000***

In [13]:
```
%sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEX where PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000 and LANDING__OUTCON
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Out[13]:

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

# Total Number of Successful and Failure Mission Outcomes

## Task 7

***List the total number of successful and failure mission outcomes***

In [14]:

```
%sql SELECT count(*),LANDING__OUTCOME from SPACEX group by "LANDING__OUTCOME" having landing__outcome like 'Success%' or landing_
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Out[14]:

| 1 | landing__outcome |
|---|---|
| 3 | Failure |
| 5 | Failure (drone ship) |
| 2 | Failure (parachute) |
| 38 | Success |
| 14 | Success (drone ship) |
| 9 | Success (ground pad) |

# Boosters Carried Maximum Payload

### Task 8

***List the names of the booster_versions which have carried the maximum payload mass. Use a subquery***

In [15]:
```
%sql SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEX WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX)
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Out[15]:

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

# 2017 Launch Records

### Task 9

***List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017***

In [20]:
```
`_version,launch_site,date from SPACEX where Landing__Outcome like 'Success (drone ship)' AND DATE BETWEEN '2017-01-01' and '2017-
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

Out[20]:

| landing__outcome | booster_version | launch_site | DATE |
|---|---|---|---|
| Success (drone ship) | F9 FT B1029.1 | VAFB SLC-4E | 2017-01-14 |
| Success (drone ship) | F9 FT B1021.2 | KSC LC-39A | 2017-03-30 |
| Success (drone ship) | F9 FT B1029.2 | KSC LC-39A | 2017-06-23 |
| Success (drone ship) | F9 FT B1036.1 | VAFB SLC-4E | 2017-06-25 |
| Success (drone ship) | F9 FT B1038.1 | VAFB SLC-4E | 2017-08-24 |
| Success (drone ship) | F9 B4 B1041.1 | VAFB SLC-4E | 2017-10-09 |
| Success (drone ship) | F9 FT B1031.2 | KSC LC-39A | 2017-10-11 |
| Success (drone ship) | F9 B4 B1042.1 | KSC LC-39A | 2017-10-30 |

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

### Task 10

***Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.***

In [17]:

```
%sql SELECT count(*) as Counter ,LANDING__OUTCOME from SPACEX where date BETWEEN '2010-06-04' and '2017-03-20' group by "LANDING_
```

```
 * ibm_db_sa://jnw65006:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

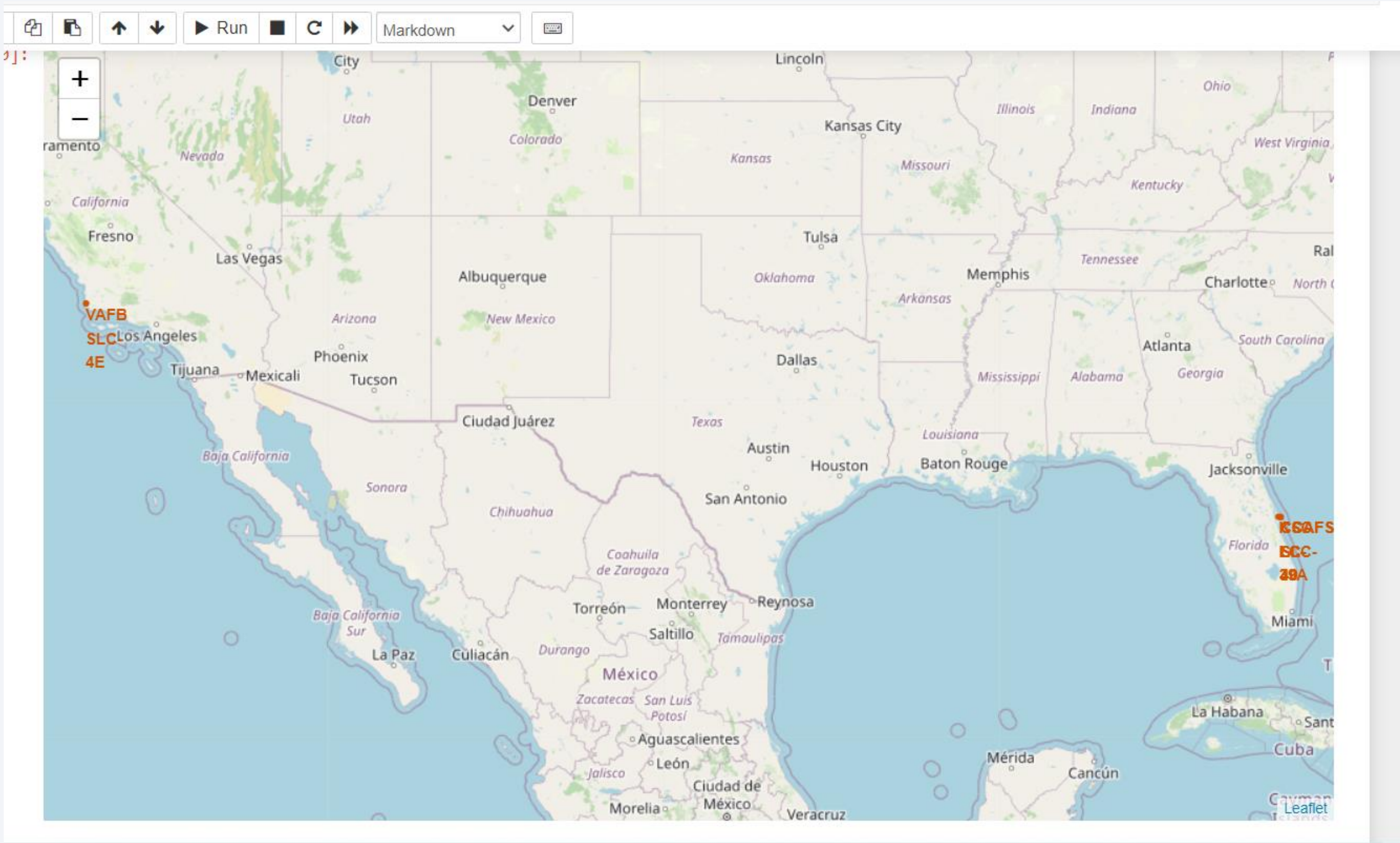
Out[17]:

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

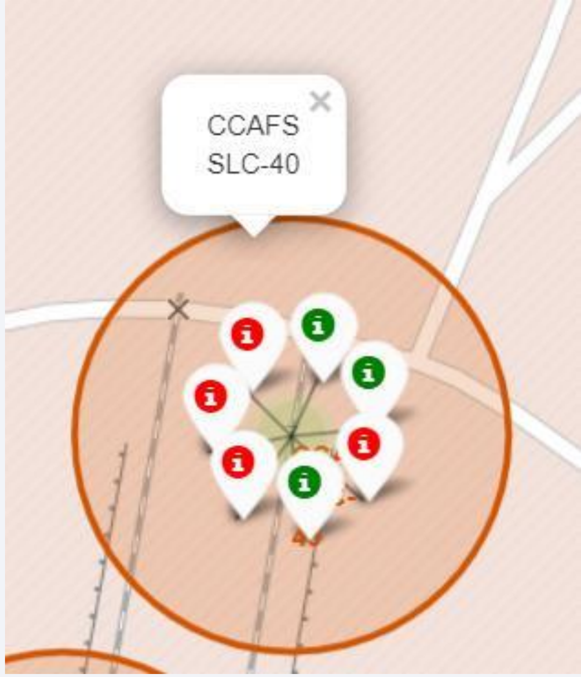
Results, Part 2:

# LAUNCH SITES PROXIMITIES ANALYSIS

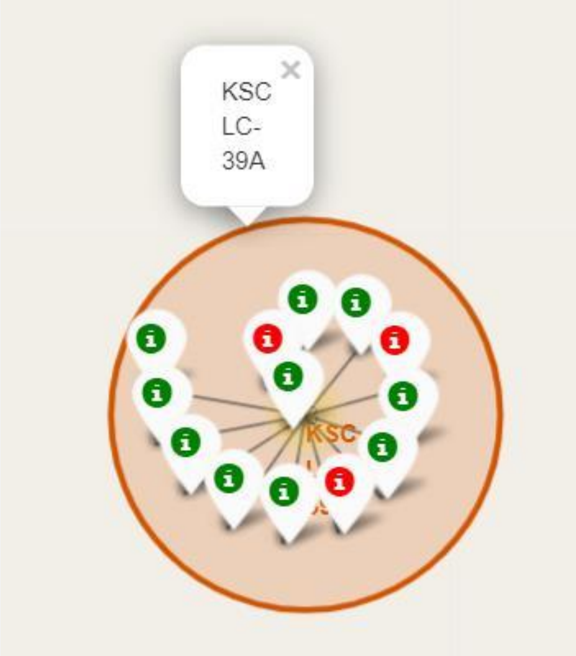
# Folium 1: All Launch Sites on Map



# Folium 2: Success/Failures For Each Site On Map

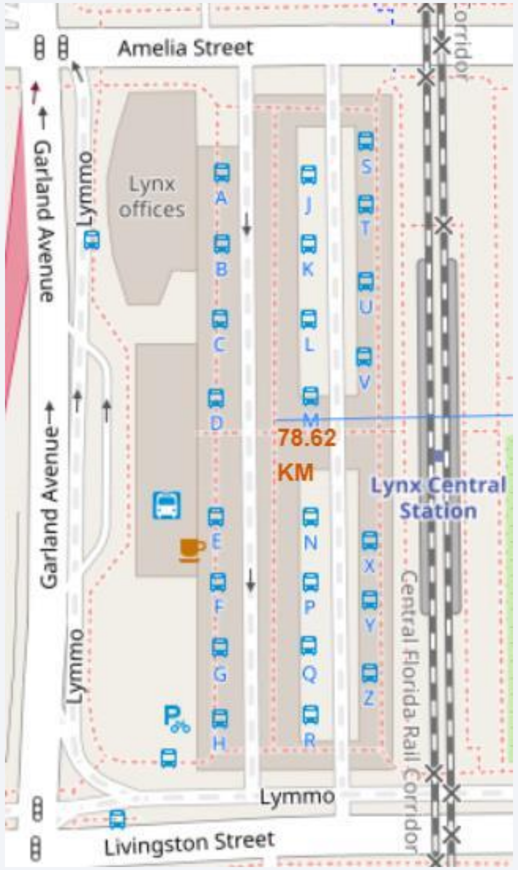







# Folium 3: Calculate Distance



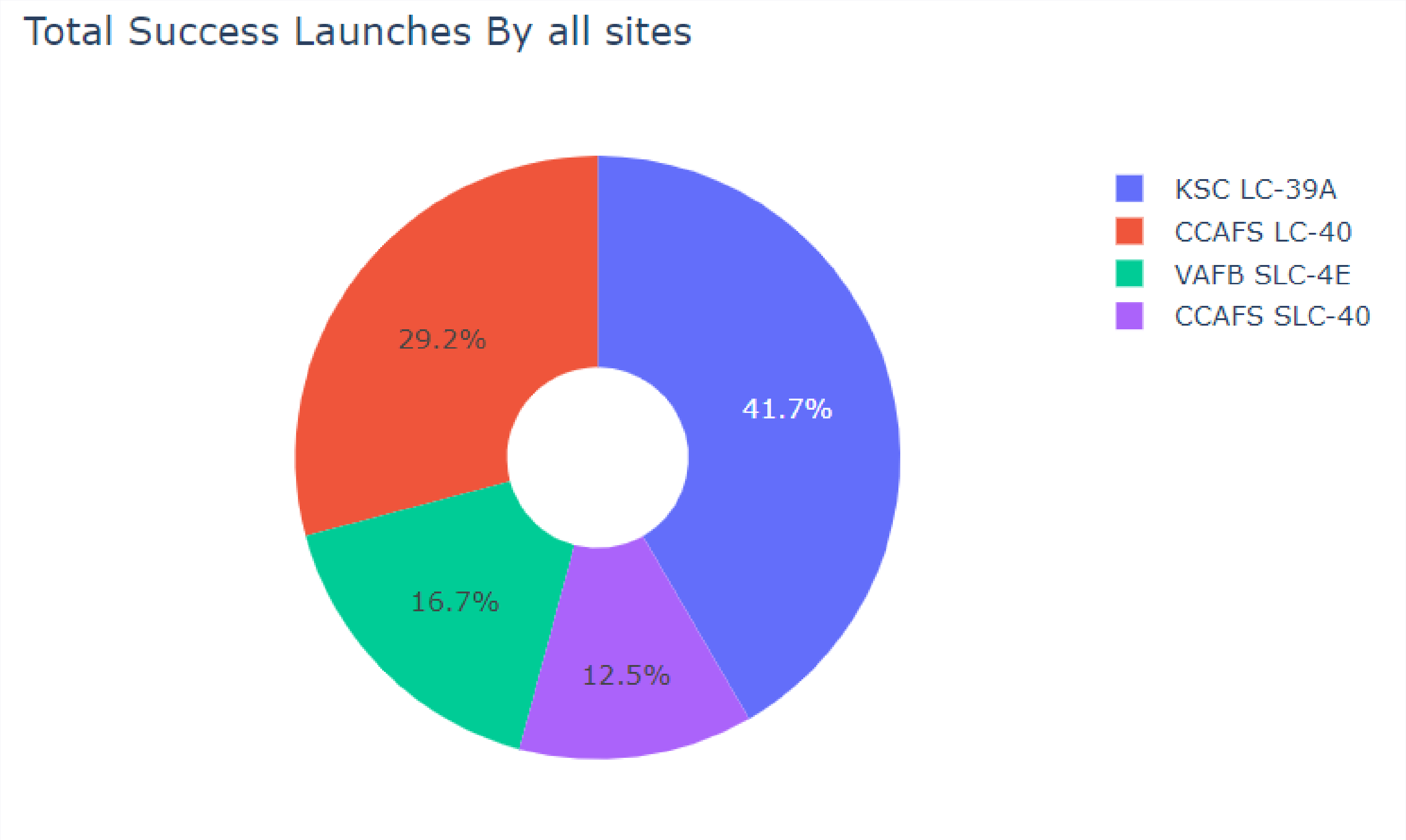




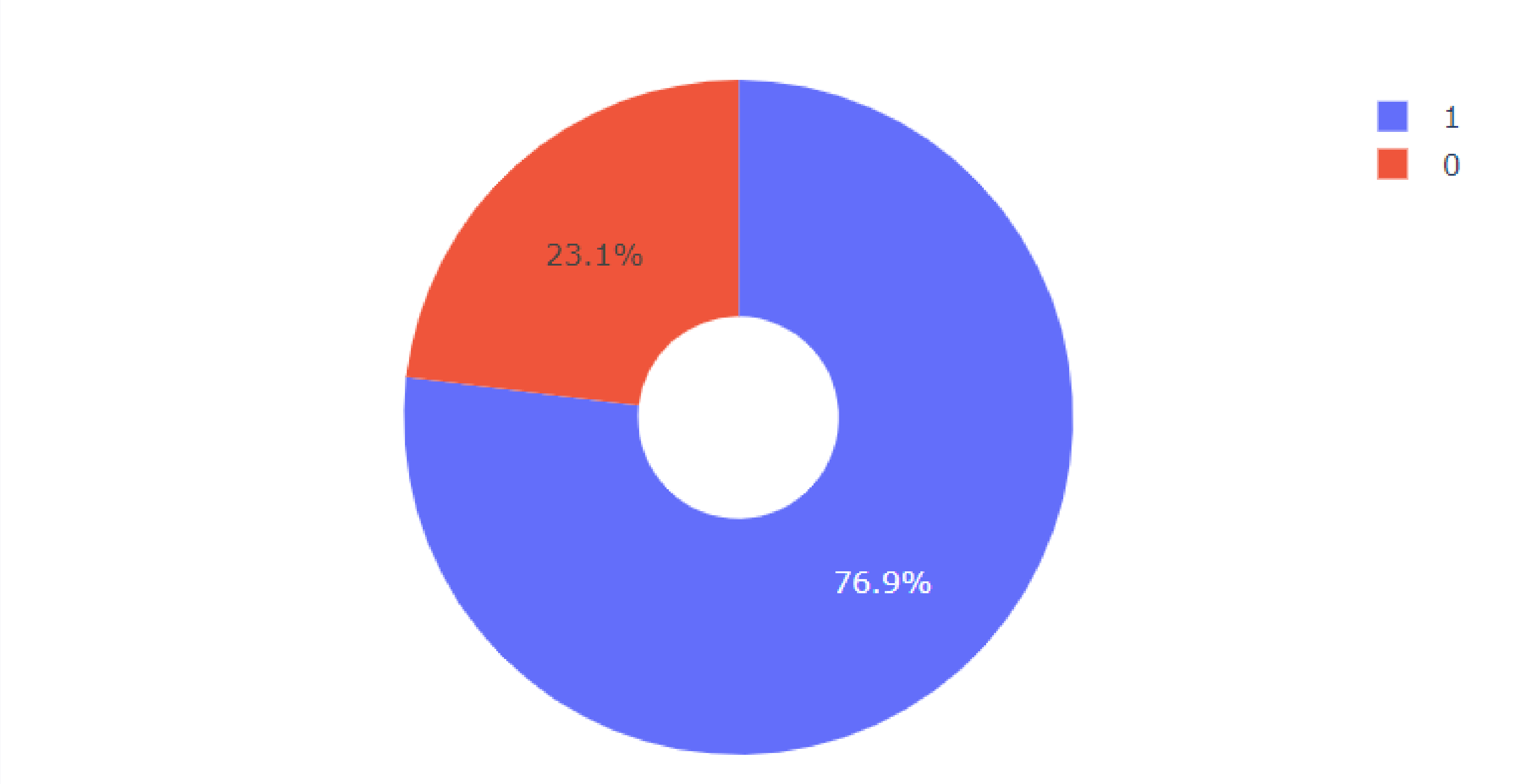
Results, Part 3:

# BUILD A DASHBOARD WITH PLOTLY DASH

# Success Pie Chart

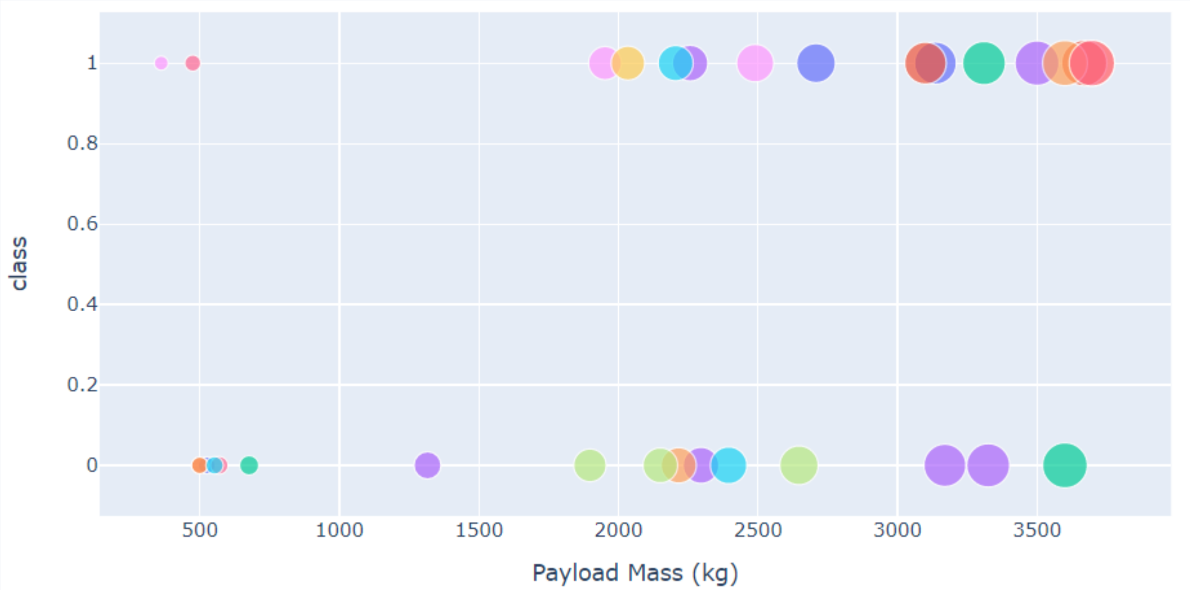


# KSC LC-39A is the launch site with highest success ratio

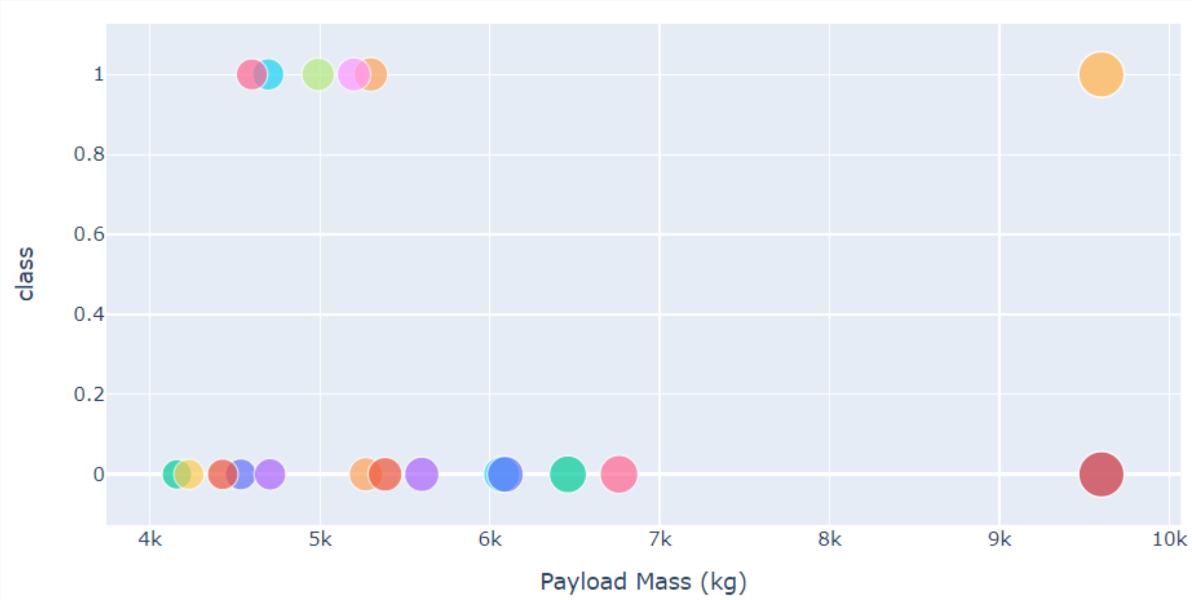


# <Dashboard Screenshot 3>

Low Weight Payload




High Weight Payload




There's more success with low weight payloads rather than high weight payloads, and shouldn't probably go above 6k.

Results, Part 4:

# ML MODELS

# Classification Accuracy

### TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
In [14]: parameters ={'C':[0.01,0.1,1],
                      'penalty':['l2'],
                      'solver':['lbfgs']}
```

```
In [18]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
         lr=LogisticRegression()
         logreg_cv = GridSearchCV(lr, parameters, cv=10)

         logreg_cv.fit(X_train, Y_train)

Out[18]: GridSearchCV(cv=10, estimator=LogisticRegression(),
                      param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                                  'solver': ['lbfgs']})
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
In [19]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
         print("accuracy :",logreg_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
         accuracy : 0.8464285714285713
```

### TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv - 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
In [22]: parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
                       'C': np.logspace(-3, 3, 5),
                       'gamma':np.logspace(-3, 3, 5)}
         svm = SVC()
```

```
In [23]: svm_cv = GridSearchCV(svm, parameters, cv=10)
         svm_cv.fit(X_train, Y_train)

Out[23]: GridSearchCV(cv=10, estimator=SVC(),
                      param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
                1.00000000e+03]),
                                  'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
                1.00000000e+03]),
                                  'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

```
In [24]: print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
         print("accuracy :",svm_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
         accuracy : 0.8482142857142856
```

LogReg = 0.8464

SVM = 0.8482

**Tree = 0.8892**

KNN = 0.8482

### TASK 8

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`.

```
In [27]: 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()
```

```
In [28]: tree_cv = GridSearchCV(tree, parameters, cv=10)
         tree_cv.fit(X_train, Y_train)

Out[28]: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
                      param_grid={'criterion': ['gini', 'entropy'],
                                  'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                                  'max_features': ['auto', 'sqrt'],
                                  'min_samples_leaf': [1, 2, 4],
                                  'min_samples_split': [2, 5, 10],
                                  'splitter': ['best', 'random']})
```

```
In [29]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
         print("accuracy :",tree_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 4,
         'min_samples_split': 10, 'splitter': 'random'}
         accuracy : 0.8892857142857142
```

### TASK 10

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
In [32]: 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()
```

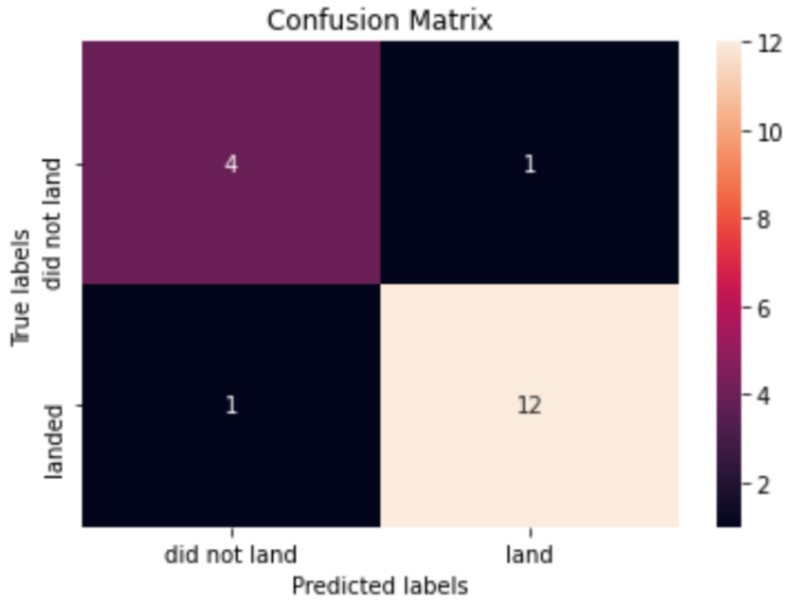
```
In [33]: knn_cv = GridSearchCV(KNN, parameters, cv=10)
         knn_cv.fit(X_train, Y_train)

Out[33]: GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
                      param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                                  'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                                  'p': [1, 2]})
```

```
In [34]: print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
         print("accuracy :",knn_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
         accuracy : 0.8482142857142858
```

# Confusion Matrix




Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.

# Conclusions

- The Tree Classifier Algorithm is the best Model for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional to time as they succeed the more they learn, and the more time goes on
- We can see that KSC LC 39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES L1 has the best Success Rate

# THANK YOU!