IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Gaurav Dutta
02/20/2022

# Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

# Executive Summary

- Summary of methodologies
  - Data Collection
    - Using the SpaceX API and web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia
  - Data Wrangling
    - Exploring patterns in the data and determine the label for training supervised models
  - Exploratory Data Analysis and Feature Engineering
    - Using SQL, Pandas and Matplotlib to understand the relationship between success rate and different mission-related parameters
  - Interactive Visual Analytics
    - Using Folium to analyze existing launch site locations and Plotly Dash to analyze launch records interactively
  - Predictive Analysis (Classification)
    - Using different machine learning algorithms to predict the outcome of a launch.
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

# Introduction

- Project background and context
  - There are several companies interested in making space travel and satellite launches more affordable than now.
  - SpaceX is one of the most successful companies in the space transportation industry.
  - It’s flagship Falcon 9 rocket launches are almost 2.7 times cheaper than its competitors ($62m for SpaceX vs $165m for other companies).
  - SpaceX can ensure such savings by reusing the first stage of an orbital rocket
  - Predicting if the first stage will land can help in determining the cost of a rocket launch and this information can be used by other companies in their bids against SpaceX for a rocket launch.

- Problems we want to find answers
  - How are the different launch conditions/parameters related to the landing outcomes?
  - What are the optimal conditions required for future successful landings?

Section 1

# Methodology

# Methodology

- Summary of methodologies
  - Data Collection
    - Using the SpaceX API and web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia
  - Data Wrangling
  - Exploratory Data Analysis and Feature Engineering
  - Interactive Visual Analytics
  - Predictive Analysis (Classification)

# Data Collection – SpaceX API

Request rocket launch data from the API

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

Decode the response using .json() and .json_normalize()

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

Get specific information using the launch IDs

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

Get the Falcon 9 records and data cleaning

```
data_falcon9 = data_falcon[data_falcon['BoosterVersion']!='Falcon 1']
```

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

**Falcon 9 Data using API**

| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad | Block | ReusedCount | Serial | Longitude | Latitude |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 2010-06-04 | Falcon 9 | 6123.547647 | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B0003 | -80.577366 | 28.561857 |
| 5 | 2 | 2012-05-22 | Falcon 9 | 525.000000 | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B0005 | -80.577366 | 28.561857 |
| 6 | 3 | 2013-03-01 | Falcon 9 | 677.000000 | ISS | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B0007 | -80.577366 | 28.561857 |
| 7 | 4 | 2013-09-29 | Falcon 9 | 500.000000 | PO | VAFB SLC 4E | False Ocean | 1 | False | False | False | None | 1.0 | 0 | B1003 | -120.610829 | 34.632093 |
| 8 | 5 | 2013-12-03 | Falcon 9 | 3170.000000 | GTO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B1004 | -80.577366 | 28.561857 |

Github URL

# Data Collection – Web Scraping

Get HTTP response from Falcon 9 launch Wiki

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

```
http_response = requests.get(static_url)
```

Create a BeautifulSoup object from the HTML response

```
bsp = BeautifulSoup(http_response.text)
```

Extract all column/variable names from the HTML table header

```
html_tables = bsp.find_all('tr')
first_launch_table = html_tables[2]
```

```
for table in first_launch_table.find_all('th'):
    column_name = extract_column_from_header(table)
    if column_name is not None and len(column_name) > 0:
        column_names.append(column_name)
```

Create a data frame by parsing the HTML tables

```
df=pd.DataFrame(launch_dict)
df.head(6)
```

## Falcon 9 Data using Web Scraping

| | Flight No. | Launch site | Payload | Payload mass | Orbit | Customer | Launch outcome | Version Booster | Booster landing | Date | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CCAFS | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success\n | F9 v1.0B0003.1 | Failure | 4 June 2010 | 18:45 |
| 1 | 2 | CCAFS | Dragon | 0 | LEO | NASA (COTS)\nNRO | Success | F9 v1.0B0004.1 | Failure | 8 December 2010 | 15:43 |
| 2 | 3 | CCAFS | Dragon | 525 kg | LEO | NASA (COTS) | Success | F9 v1.0B0005.1 | No attempt\n | 22 May 2012 | 07:44 |
| 3 | 4 | CCAFS | SpaceX CRS-1 | 4,700 kg | LEO | NASA (CRS) | Success\n | F9 v1.0B0006.1 | No attempt | 8 October 2012 | 00:35 |
| 4 | 5 | CCAFS | SpaceX CRS-2 | 4,877 kg | LEO | NASA (CRS) | Success\n | F9 v1.0B0007.1 | No attempt\n | 1 March 2013 | 15:10 |
| 5 | 6 | VAFB | CASSIOPE | 500 kg | Polar orbit | MDA | Success | F9 v1.1B1003 | Uncontrolled | 29 September 2013 | 16:00 |

Github URL

# Methodology

- Summary of methodologies
  - Data Collection
  - Data Wrangling
    - Exploring patterns in the data and determine the label for training supervised models
  - Exploratory Data Analysis and Feature Engineering
  - Interactive Visual Analytics
  - Predictive Analysis (Classification)

# Data Wrangling

**Load the data**

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

**Preliminary data analysis**

```
df.isnull().sum()/df.count()*100
df.dtypes
```

1) Calculate the number of launches on each site

2) Calculate the number and occurrence of each orbit and the mission outcomes

```
df.value_counts('LaunchSite')
df.value_counts('Orbit')
landing_outcomes = df.value_counts('Outcome')
```

**Create a landing outcome label**

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

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

```
df['Class']=landing_class
```

## Data after wrangling

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

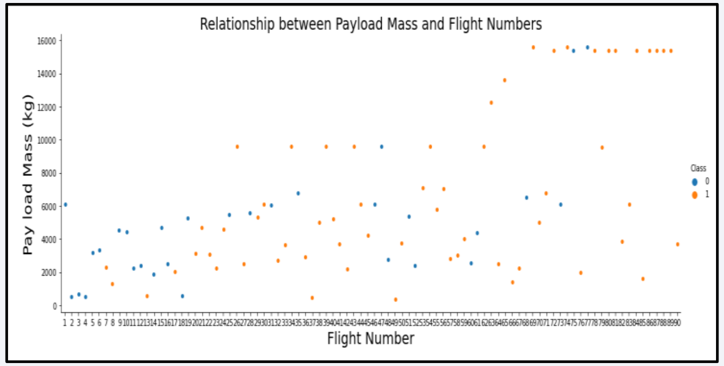
Github URL

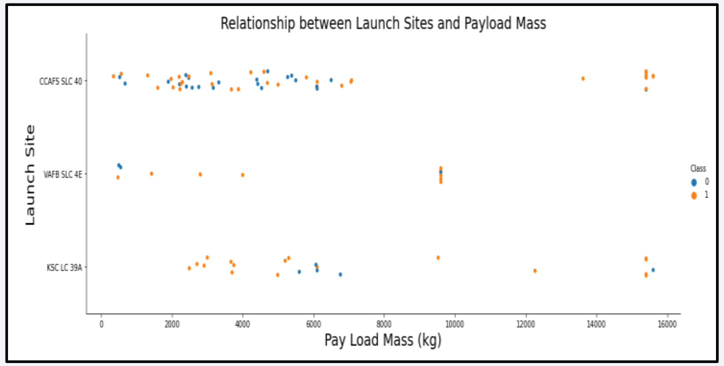
# Methodology

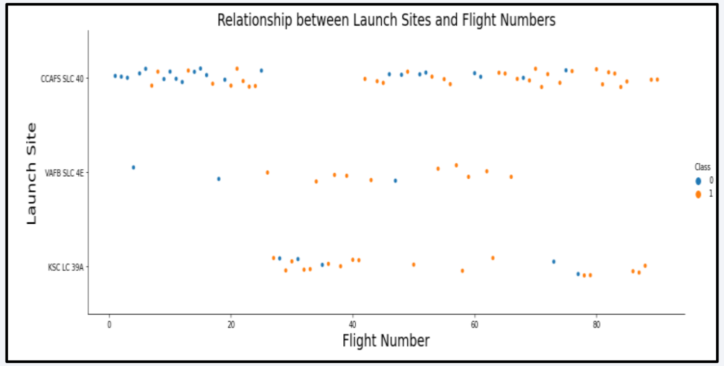
- Summary of methodologies
    - Data Collection
    - Data Wrangling
    - Exploratory Data Analysis (EDA) and Feature Engineering
        - Using SQL, Pandas and Matplotlib to understand the relationship between success rate and different mission-related parameters
    - Interactive Visual Analytics
    - Predictive Analysis (Classification)

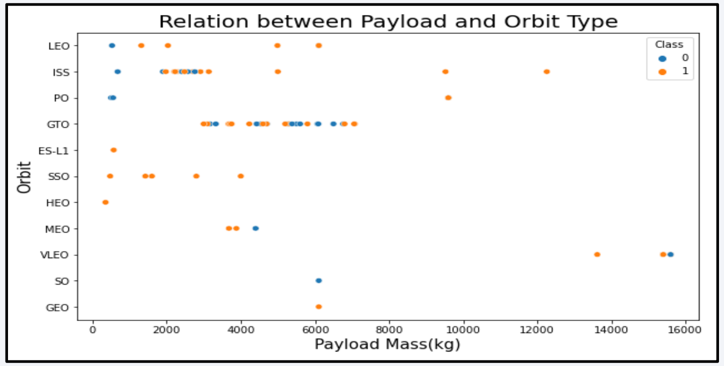
# EDA with Data Visualization

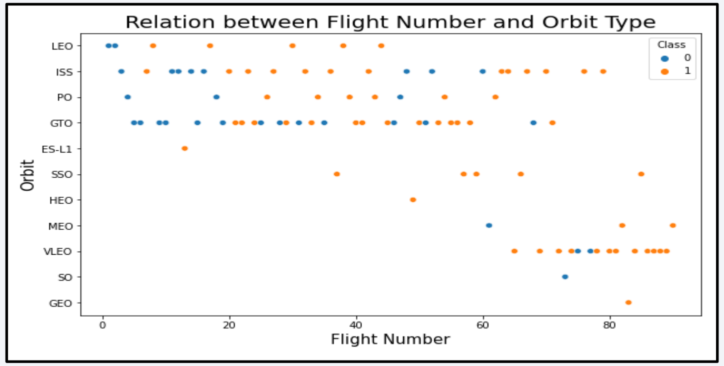
1) Using Scatter plots



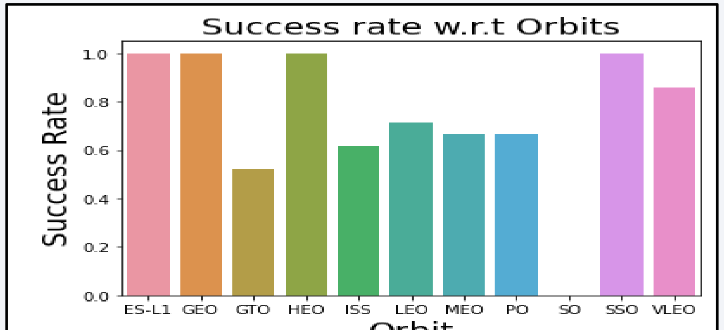



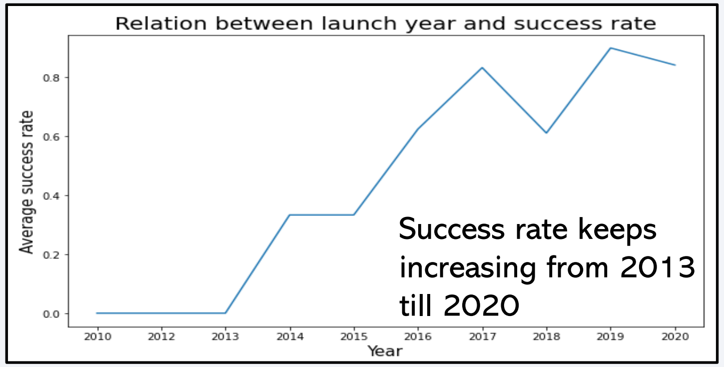




2) Using Bar plots




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

3) Using Line plots




Success rate keeps increasing from 2013 till 2020

[Github URL]

# EDA with SQL

SQL queries were used to retrieve the following information:

**Unique Launch Sites**

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

**Sites beginning with 'CCA'**

| DATE | time__utc_ | booster_version | launch_site | |
|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | |
| 2010-12-08 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Drag |
| 2012-05-22 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | |
| 2012-10-08 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | |

**Date of 1st successful ground pad landing**

| DATE |
|---|
| 2015-12-22 |

**Total no of successful and failed missions**

| mission_outcome | total |
|---|---|
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

**Failed landing outcomes in drone ship, their booster versions and launch sites in 2015**

| booster_version | launch_site | DATE | landing__outcome |
|---|---|---|---|
| F9 v1.1 B1012 | CCAFS LC-40 | 2015-01-10 | Failure (drone ship) |
| F9 v1.1 B1015 | CCAFS LC-40 | 2015-04-14 | Failure (drone ship) |

**Total payload mass by NASA (kgs)**

| total_payload_mass |
|---|
| 45596 |

**Avg payload mass carried by booster F9 v1.1**

| avg_payload_mass |
|---|
| 2928 |

**Boosters with success in drone ship and (Payload 4000-6000 kgs)**

| booster_version | payload_mass__kg_ | landing__outcome |
|---|---|---|
| F9 FT B1022 | 4696 | Success (drone ship) |
| F9 FT B1026 | 4600 | Success (drone ship) |
| F9 FT B1021.2 | 5300 | Success (drone ship) |
| F9 FT B1031.2 | 5200 | Success (drone ship) |

**Booster versions with the max payload mass**

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

**Count of landing outcomes between 2010-06-04 and 2017-03-20**

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

Github URL

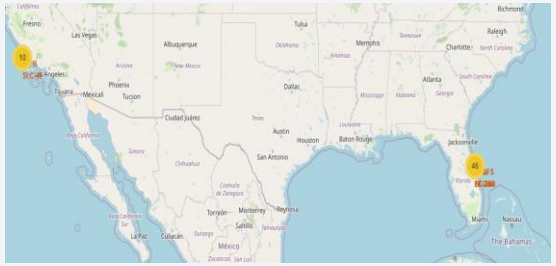
# Methodology

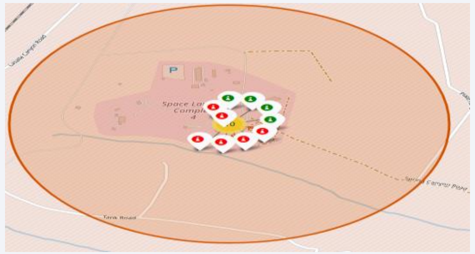
- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis and Feature Engineering
  - Interactive Visual Analytics
    - Using Folium to analyze existing launch site locations
    - Use Plotly Dash to analyze launch records interactively
  - Predictive Analysis (Classification)

# Build an Interactive Map with Folium

- Interactive visual analytics using Folium is used to analyze the existing launch locations.
- This is done by completing the following tasks.
  - Mark all launch sites on a map.


  - Mark the successful and failed launches for each site on the map.





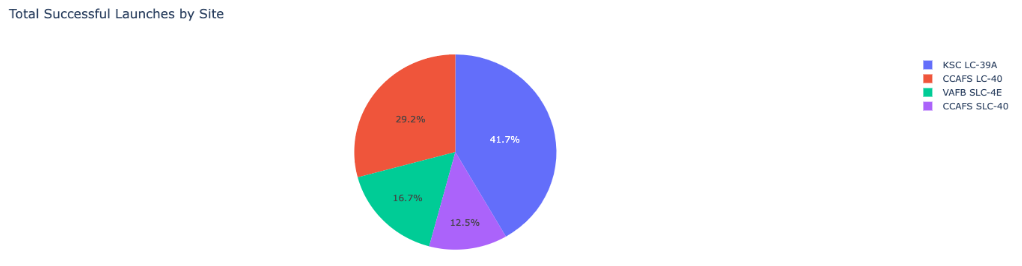
  - Calculate the distances between a launch site to its proximities like railways, highways, coastlines and nearby cities.



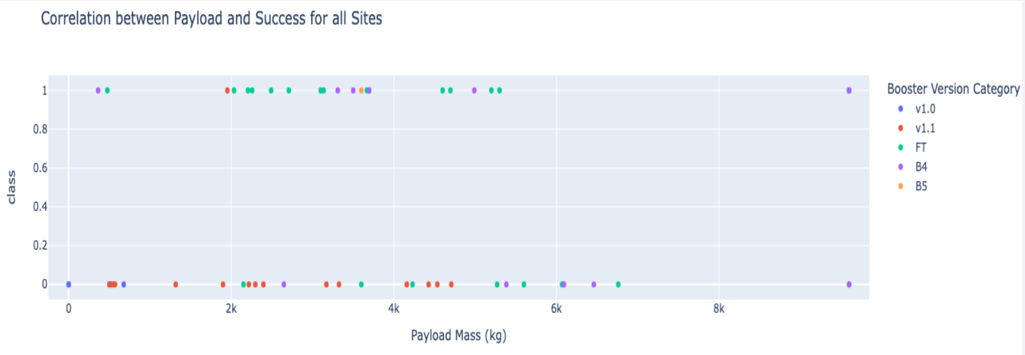
Github URL

# Build a Dashboard with Plotly Dash

- An interactive dashboard is built using Plotly and Dash to gain further insights into the data.
- The dashboard has two components.
  - A pie chart that can be used to visualize the successful launches using all the launch sites or a selected launch site.




  - Scatter plots showing the correlation between payload mass and the success rates for all the launch sites or for a selected launch site.






Github URL

# Methodology

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis and Feature Engineering
  - Interactive Visual Analytics
  - Predictive Analysis (Classification)
    - Using different machine learning algorithms to predict the outcome of a launch.

# Predictive Analysis (Classification)

**Create a Pandas series from the column Class in the data that would be the labels**

```
Y = data['Class'].to_numpy()
print(type(Y))
print(Y.shape)
```

**Standardize the data**

```
X = preprocessing.StandardScaler().fit(X).transform(X)
print(X.shape)
#X[0:5]
```

**Split the data into training and test datasets**

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

**Find the best hyperparameters for logistic regression, SVM, Decision Treesand K-nearest Neighbors**

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`.

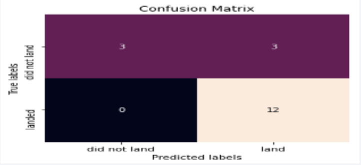
```
In [18]: parameters ={'C':[0.01,0.1,1],
                      'penalty':['l2'],
                      'solver':['lbfgs']}
```

```
In [23]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
         lr=LogisticRegression()
         logreg_cv = GridSearchCV(lr, parameters)
         logreg_cv.fit(X_train, Y_train)

Out[23]: GridSearchCV(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 [24]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
         print("accuracy :",logreg_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
         accuracy : 0.8342857142857143
```

**Evaluate the performance of the different methods**






Github URL

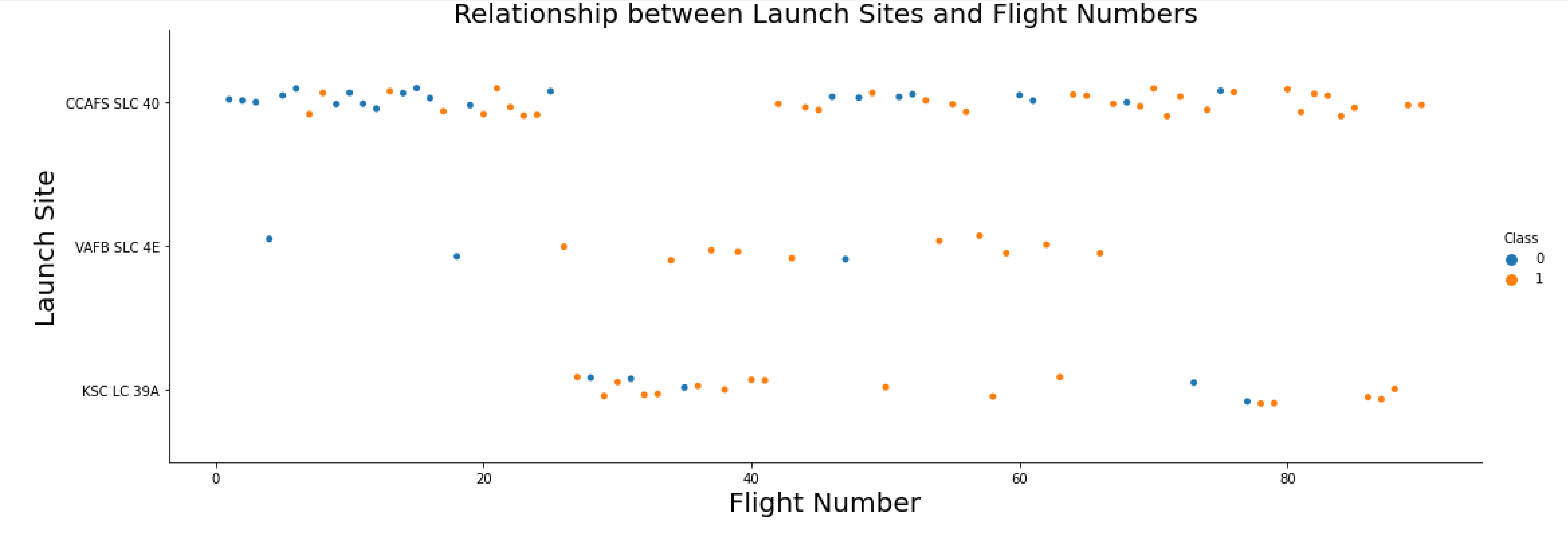
# Results

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

Section 2

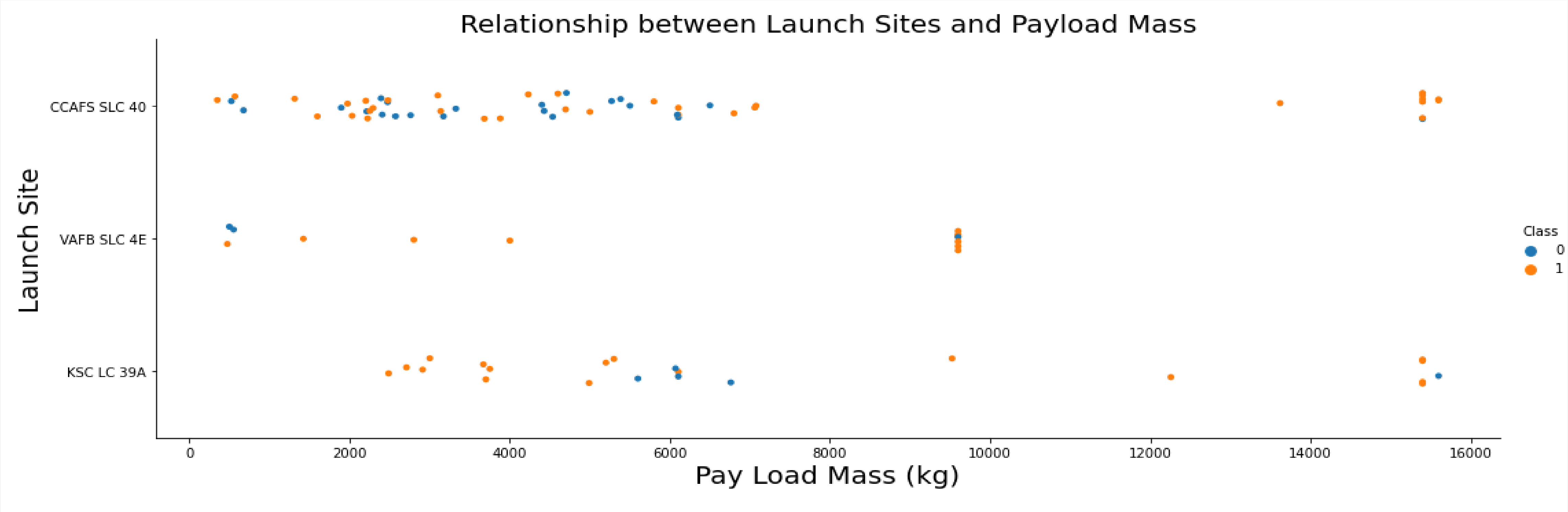
# Insights drawn from EDA

# Flight Number vs. Launch Site



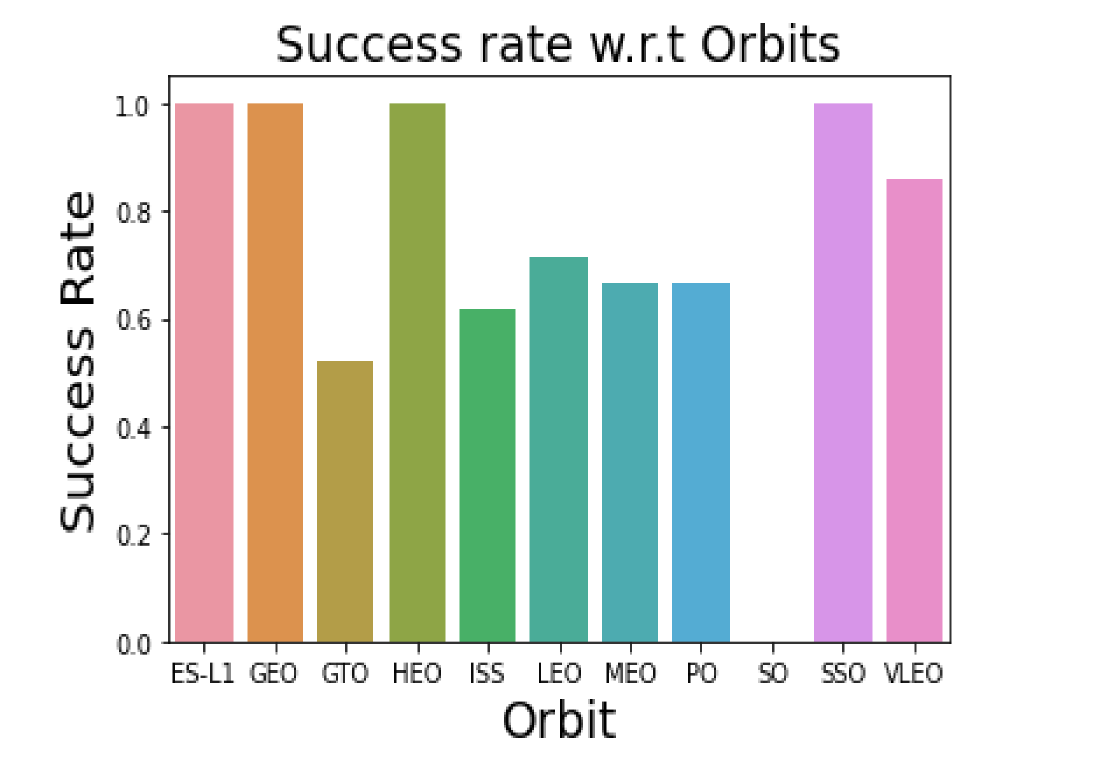

- Here class 0 (blue) represents an unsuccessful launch while class 1 (orange) represents a successful launch.
- As the flight numbers increased from the different launch sites, the success rate of a launch have also increased from each of these sites.

# Payload vs. Launch Site



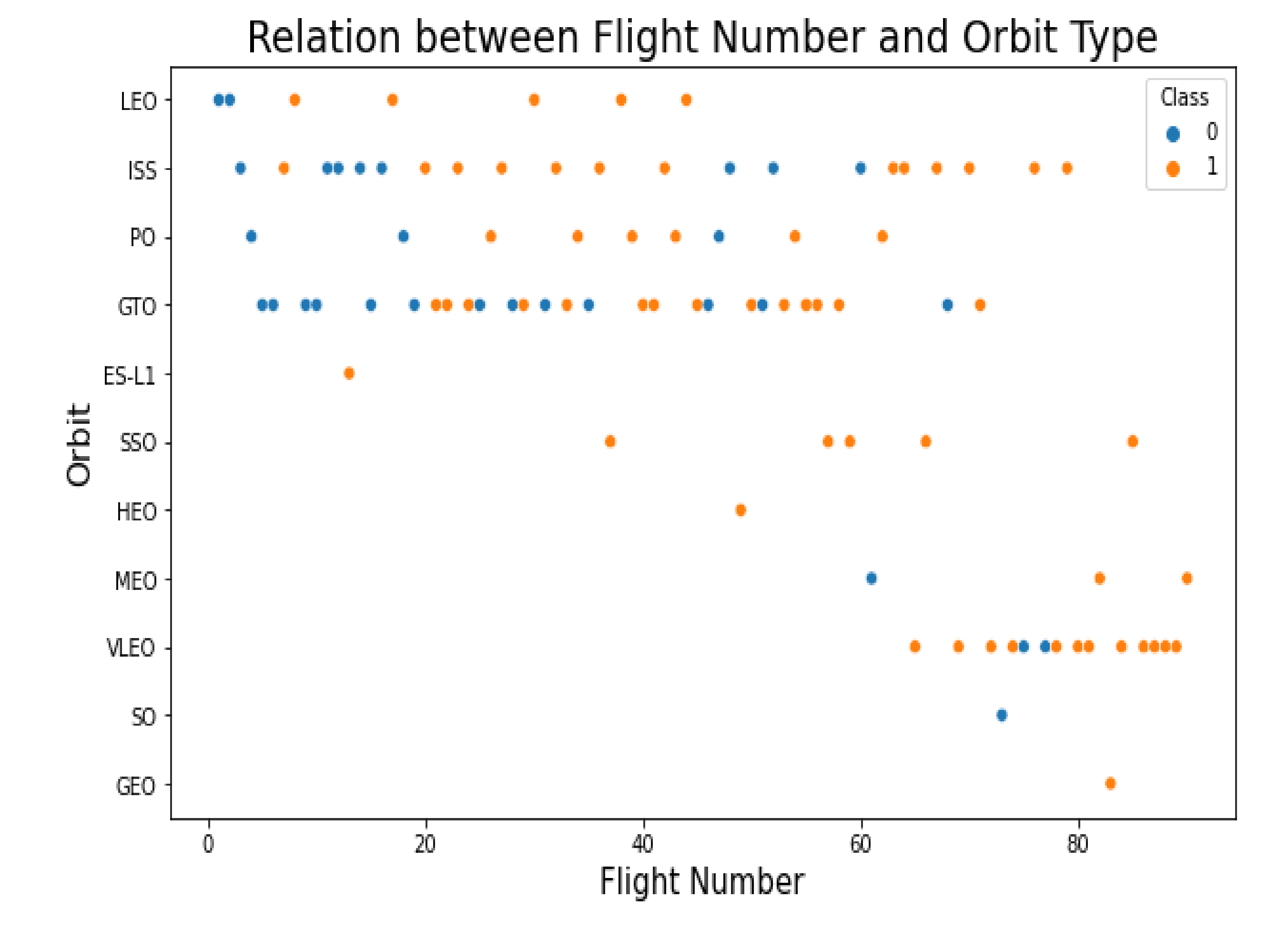

- Here class 0 (blue) represents an unsuccessful launch while class 1 (orange) represents a successful launch.
- For the VAFB SLC 4E launch site, there have been no rockets launched with a payload mass > 10,000 kg.
- There seems to be no clear correlation between the success of a launch and the payload mass being carried by a rocket for the 3 launch sites.

# Success Rate vs. Orbit Type



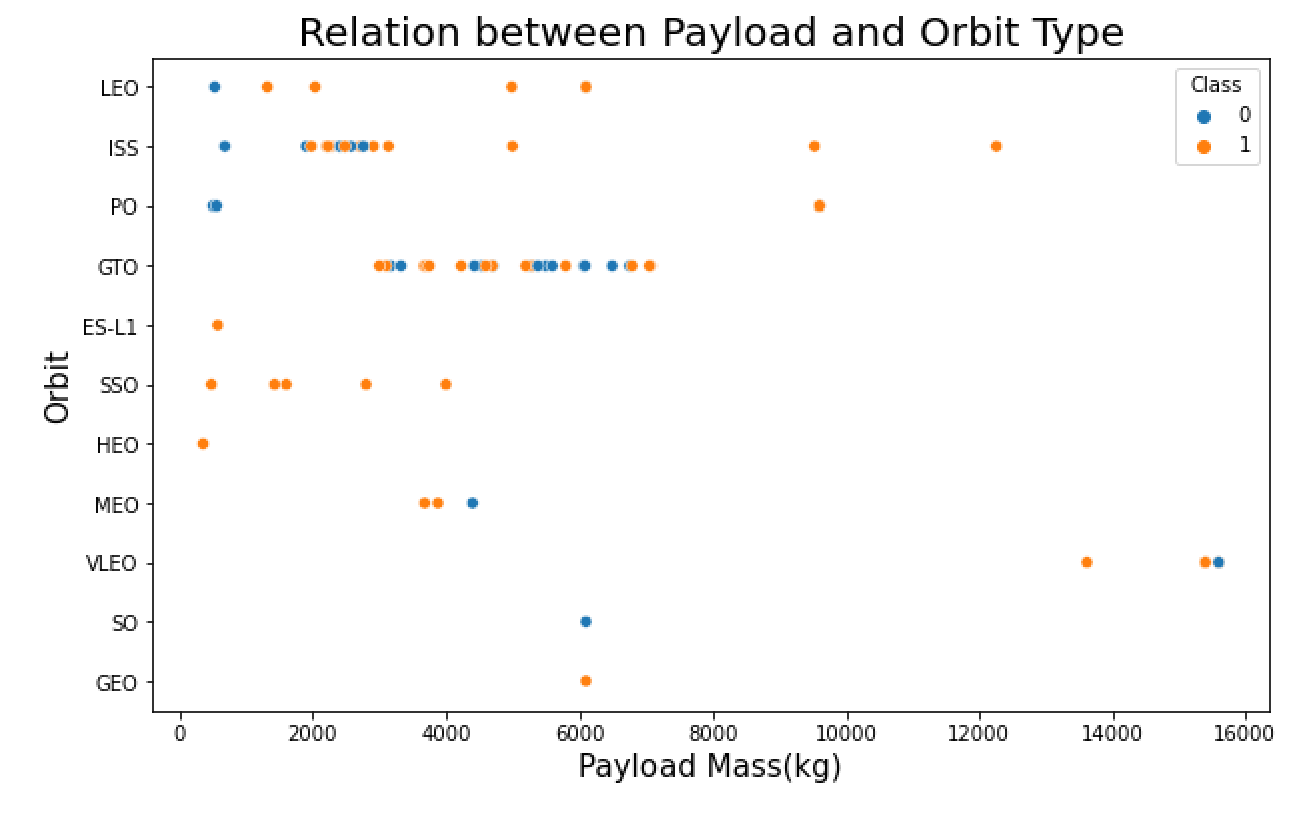

- Rockets launched into the orbits ES-L1, GEO, HEO and SSO have 100% success rates.
- For more than 1 launch, rockets launched into the orbit have a success rate of around 50% which is the lowest amongst all the orbits.
- There was only a single launch into the orbit SO which was not a success.

# Flight Number vs. Orbit Type



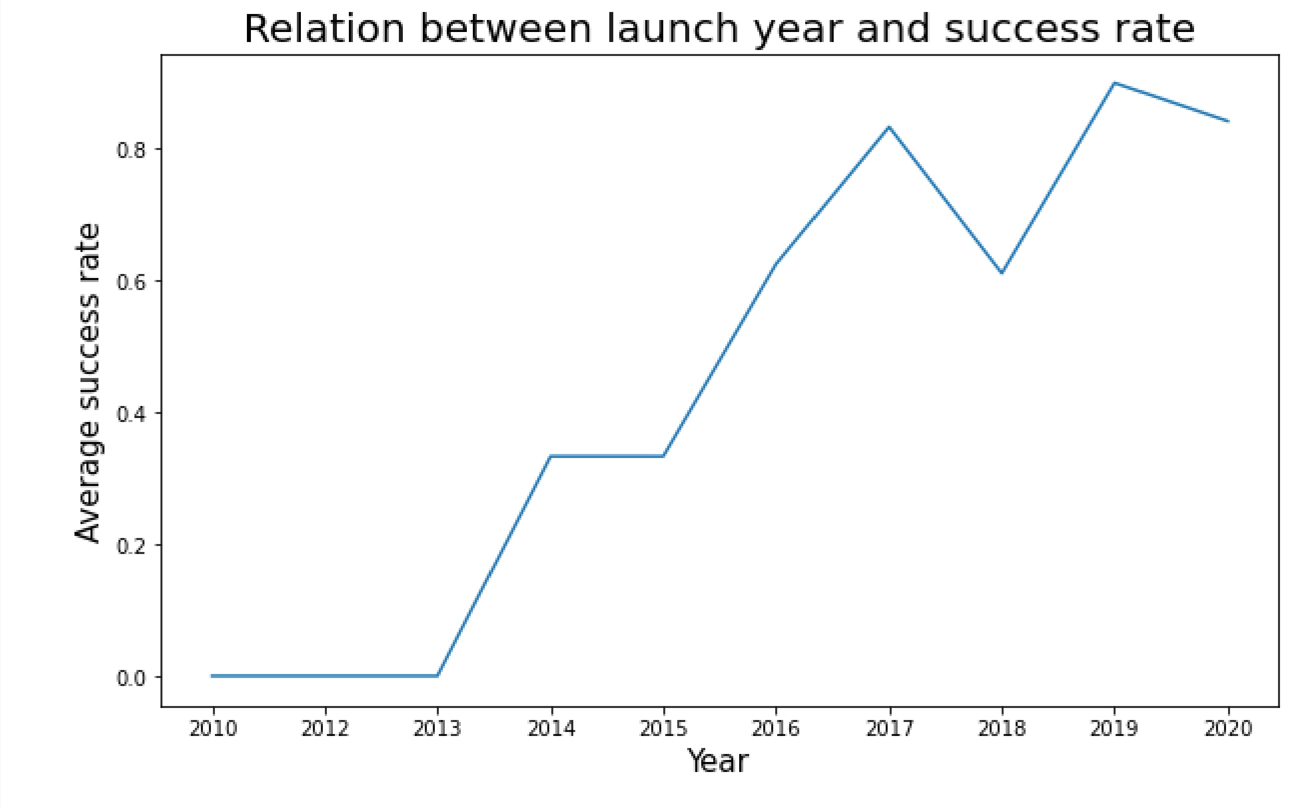

- Here class 0 (blue) represents an unsuccessful launch while class 1 (orange) represents a successful launch.
- For rockets launched into the orbit LEO, the success rate increased with flight numbers.
- There seems to be no clear relation between the success rate and the flight number for the orbits ISS, PO, GTO and VLEO.

# Payload vs. Orbit Type




- Here class 0 (blue) represents an unsuccessful launch while class 1 (orange) represents a successful launch.
- With heavy payloads the successful landing or positive landing rate are more for the orbits Polar, LEO and ISS than the others.
- However, for the remaining orbits like GTO, the relationship cannot be clearly identified.

# Launch Success Yearly Trend




- Since 2013, the success rate of the rocket launches have kept on steadily increasing until 2018 where there was a small dip in the rate.
- In recent years, the success rate has been close to 80%.

# All Launch Site Names

```
%sql select distinct(launch_site) from SPACEXTBL
```

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

]:

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

- The SQL DISTINCT clause is used to retrieve the unique values (i.e., the launch site names in this case) from a table (SPACEXTBL).

# Launch Site Names Begin with 'CCA'

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

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

| DATE | time__utc_ | booster_version | launch_site | payload | payload_mass__kg_ | orbit | customer | mission_outcome | landing__outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 2010-12-08 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of Brouere cheese | 0 | LEO (ISS) | NASA (COTS) NRO | Success | Failure (parachute) |
| 2012-05-22 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

- The SQL LIKE clause along with the % sign is used to retrieve the site numbers beginning with 'CCA' from the table (SPACEXTBL).

- The LIMIT clause is used to display only 5 records from the query.

# Total Payload Mass

```
%sql select sum(PAYLOAD_MASS__KG_) as total_payload_mass from SPACEXTBL where CUSTOMER = 'NASA (CRS)'
```

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

| total_payload_mass |
|---|
| 45596 |

- The SQL WHERE clause is used to retrieve the results for CUSTOMER (i.e., NASA) from a table (SPACEXTBL).
- The SUM function is then used to calculate the sum of columns (PAYLOAD_MASS__KG_).

# Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as avg_payload_mass from SPACEXTBL where booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

| avg_payload_mass |
|---|
| 2928 |

- The SQL WHERE clause is used to retrieve the results for booster_version (i.e., F9 v1.1) from a table (SPACEXTBL).
- The AVERAGE function is then used to calculate the average of the column (PAYLOAD_MASS__KG_).

# First Successful Ground Landing Date

```
%sql select min(DATE) as date from SPACEXTBL where landing__outcome ='Success (ground pad)'
```

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

| DATE |
|---|
| 2015-12-22 |

- The SQL WHERE clause is used to retrieve the results for landing__outcome (i.e., Success (ground pad)) from a table (SPACEXTBL).

- The MIN function is then used to calculate the first successful landing date

# Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_, LANDING__OUTCOME FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)'  \
AND (PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000)
```

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

2]:

| booster_version | payload_mass__kg_ | landing__outcome |
|---|---|---|
| F9 FT B1022 | 4696 | Success (drone ship) |
| F9 FT B1026 | 4600 | Success (drone ship) |
| F9 FT B1021.2 | 5300 | Success (drone ship) |
| F9 FT B1031.2 | 5200 | Success (drone ship) |

- The SQL WHERE clause is used to retrieve the results for landing__outcome (i.e., Success (drone ship)) from the table SPACEXTBL.
- The AND operator is then used to impose an additional condition that the PAYLOAD_MASS__KG_ is between 4000 and 6000 kgs.

# Total Number of Successful and Failure Mission Outcomes

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

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total FROM SPACEXTBL GROUP BY MISSION_OUTCOME
```

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

]:

| mission_outcome | total |
|---|---|
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

- The SQL COUNT( ) function is used to calculate the total number of columns.
- The GROUP BY statement groups rows that have the same values (i.e., failure and success).
- There is almost 99% success rate for all the missions.

# Boosters Carried Maximum Payload

```
%sql select booster_version,  PAYLOAD_MASS__KG_ from spacextbl where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextbl)
```

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

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

- Using a sub-query with MAX( ) function to find the maximum payload pass.
- Using the WHERE clause to retrieve the booster-versions that have carried the maximum payload mass.

# 2015 Launch Records

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

```
%sql select booster_version, launch_site, date, landing__outcome from spacextbl where date like '2015%' and landing__outcome = 'Failure (drone ship)'
```

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

]:

| booster_version | launch_site | DATE | landing__outcome |
|---|---|---|---|
| F9 v1.1 B1012 | CCAFS LC-40 | 2015-01-10 | Failure (drone ship) |
| F9 v1.1 B1015 | CCAFS LC-40 | 2015-04-14 | Failure (drone ship) |

- The SQL WHERE clause is used to retrieve the results for the YEAR 2015.
- The AND operator is then used to impose an additional condition on the landing outcome (i.e., Failure (drone ship)).
- The ‘2015%’ operator is used to retrieve all the rows starting with date 2015. Using ‘YEAR(DATE)=2015‘ is another alternative.

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

```
%sql select landing__outcome, count(*) as  count_landing_outcomes from spacextbl \
where (date > '2010-06-04' and date < '2017-03-20') \
group by landing__outcome \
order by count_landing_outcomes desc
```

```
 * ibm_db_sa://ghs79289:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

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

- The WHERE clause is used to filter the rows for a date between 2010-06-04 and 2017-03-20.
- The GROUP BY statement groups rows based on the landing outcome and the COUNT(*) function counts the total number of landings for each outcome.
- The ORDER BY keyword is then used to sort the records by total number of landings. The DESC keyword sorts the rows in descending order.

Section 3

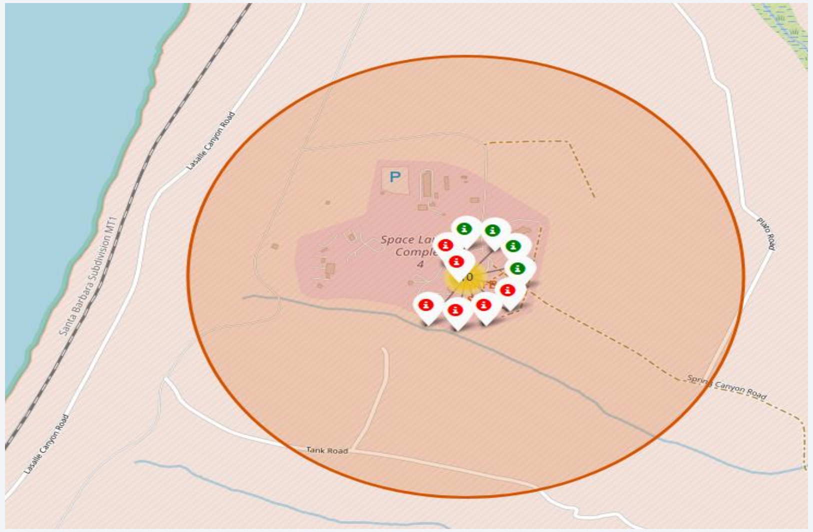
# Launch Sites Proximities Analysis

# Locations of all Launch Sites

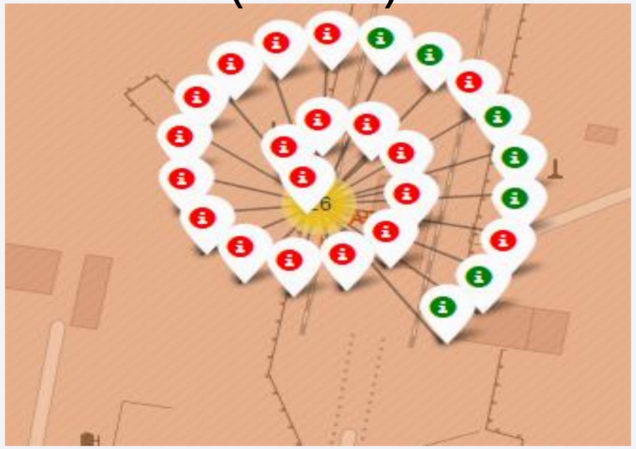

- All the launch sites are near the coasts in the US states of Florida and California

# Color-labeled Launch Outcomes for each Site

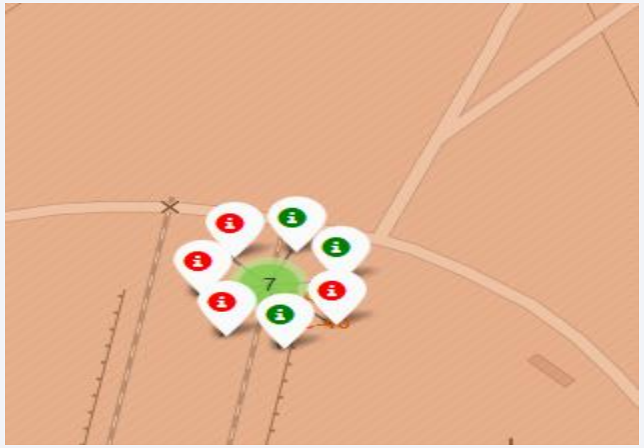
VAFB SCL-4E
(California)



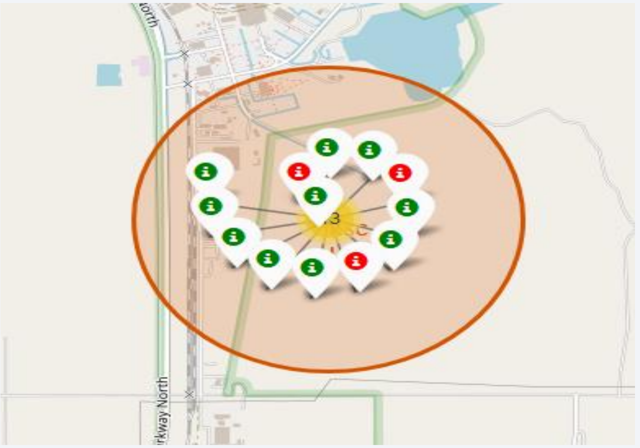
CCAFS LC-40
(Florida)



CCAFS SLC-40
(Florida)



KSC LC-39A
(Florida)



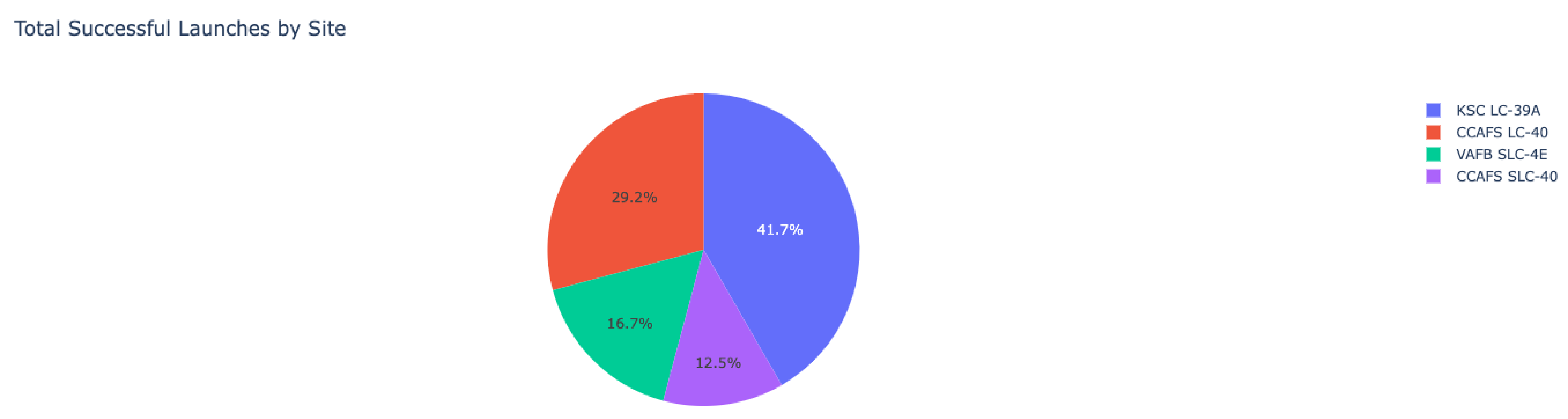
**Successful landings are shown in green markers while the unsuccessful ones are in red.*

**Launches from the KSC LC-39A launch site have the maximum probability of success**

Section 4

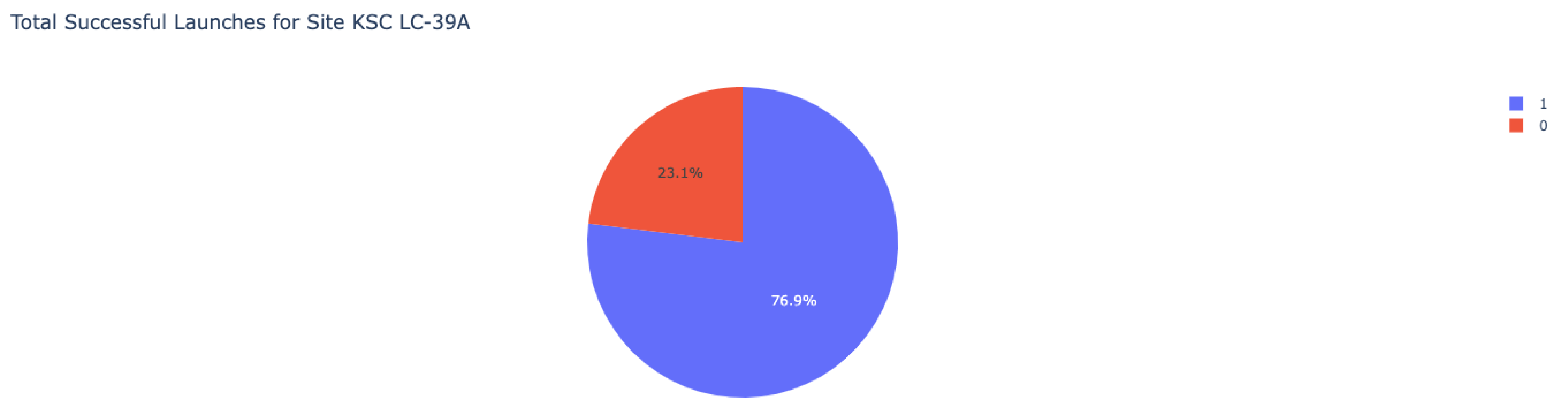
# Build a Dashboard with Plotly Dash

# Launch success count for all sites



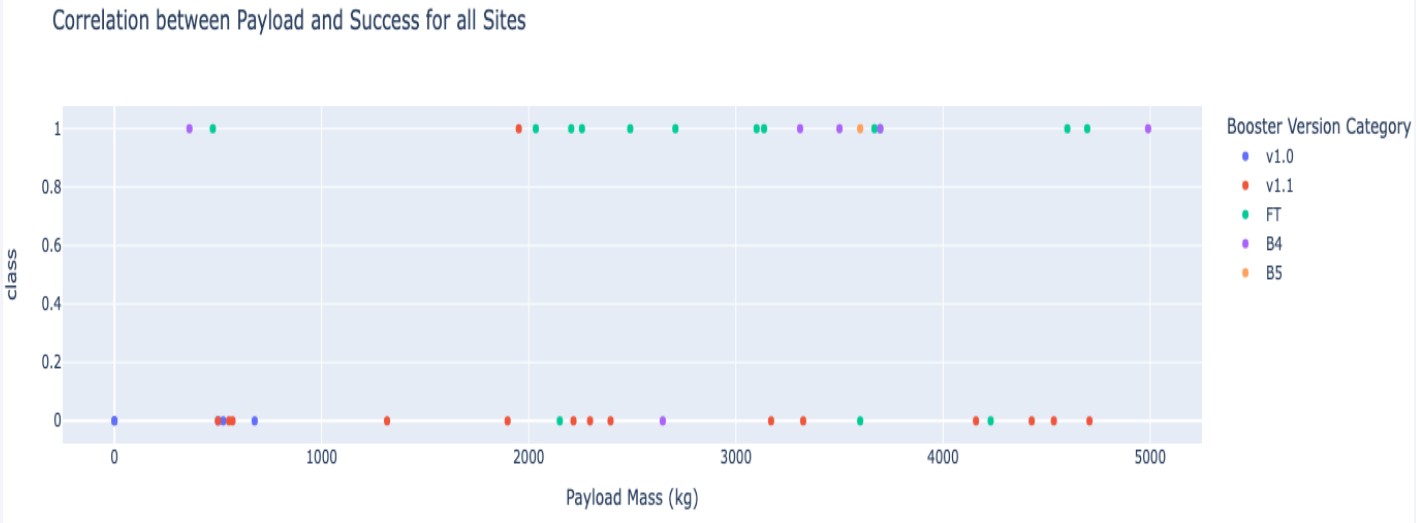

- The launch site KSC LC-39A has the most successful launches amongst all the launch sites.
- The launch sites CCAFS SLC-40 and VAFB SLC-4E have the lowest launch success amongst all the sites.

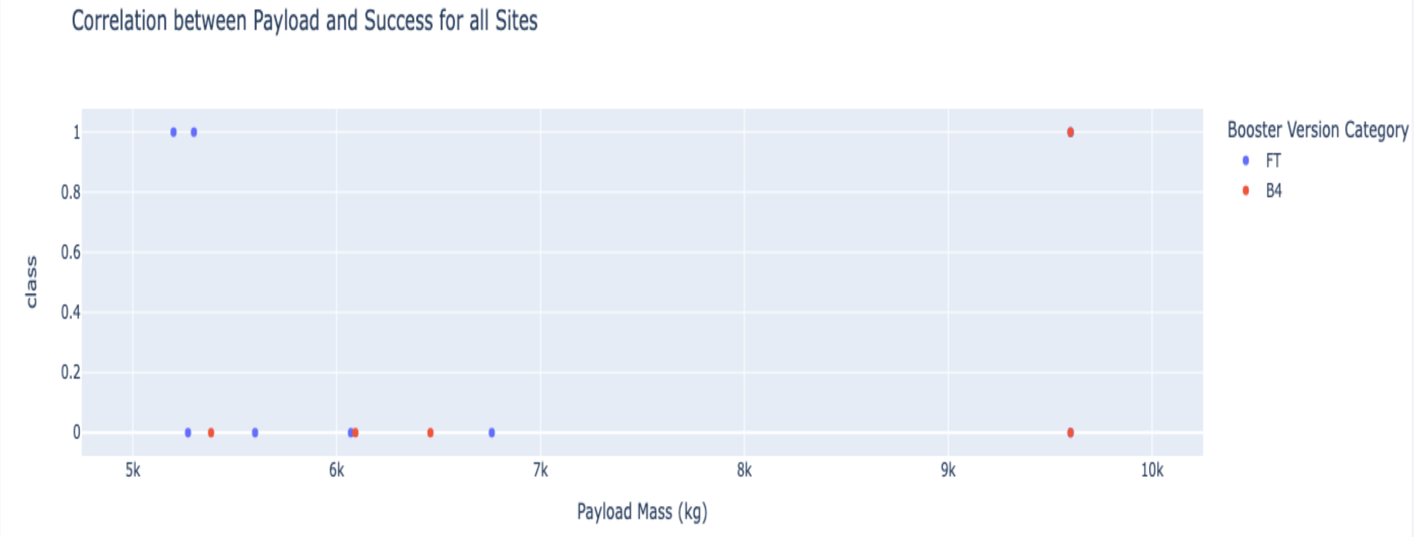
# Launch site with highest launch success ratio




- The launch site KSC LC-39A has the most successful launches amongst all the launch sites.
- It has 10 successful landings and 3 failed landings. Here blue represents the successful landings while red represents the failed ones.

# Payload vs Launch Outcome



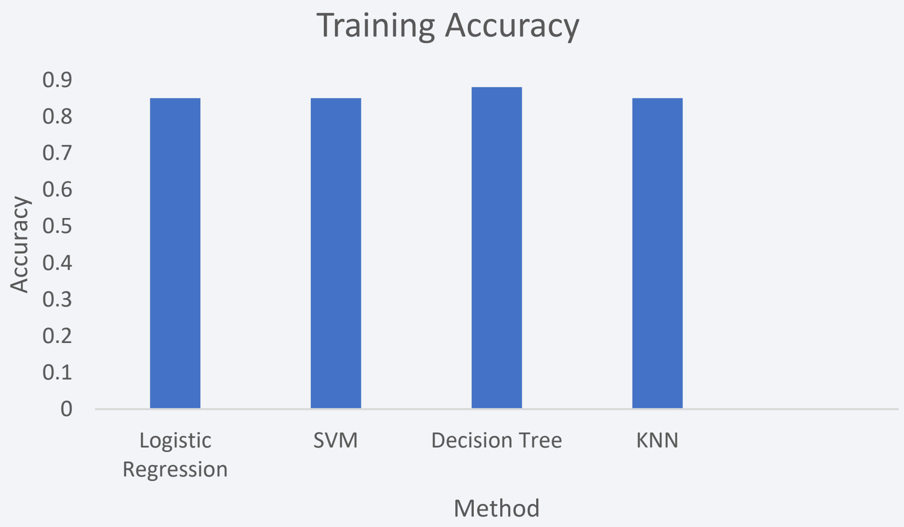

- The heavy payloads are usually launched using the booster versions FT and B4.
- The launch outcomes seem to be more favorable when the payload mass is less than 5000 kgs than for heavier payload masses (5000 – 10,000 kgs).

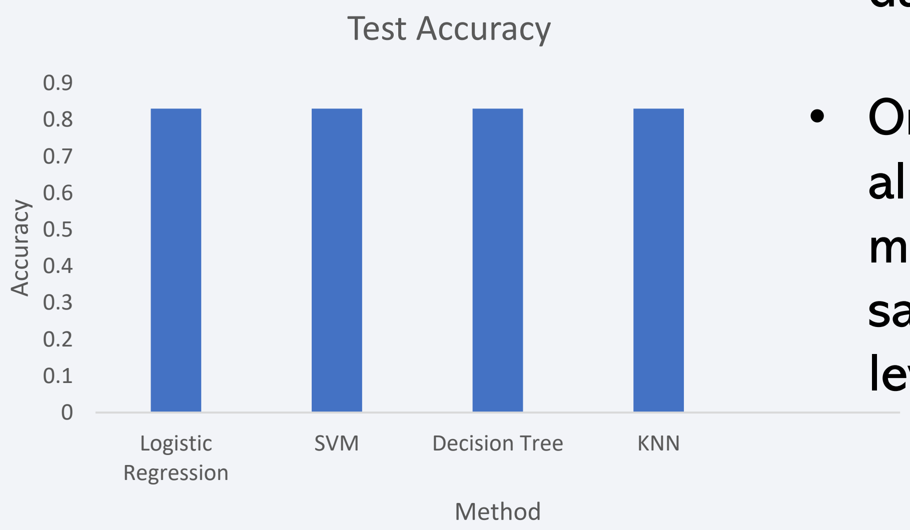
Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

| Classifier | Training Accuracy | Test Accuracy |
|---|---|---|
| Logistic Regression | 0.85 | 0.83 |
| SVM | 0.85 | 0.83 |
| Decision Tree | 0.88 | 0.83 |
| KNN | 0.85 | 0.83 |



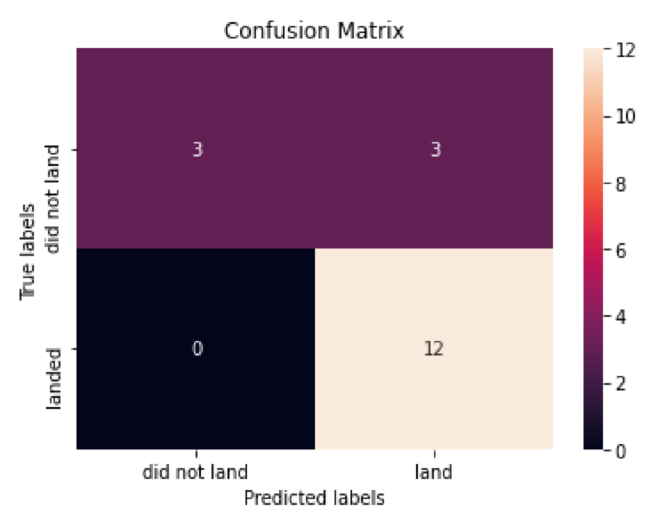




- Decision Tree classifier had a slightly better performance than the other classifiers on the training data.
- On the test data, all the 4 tested methods had the same accuracy level.

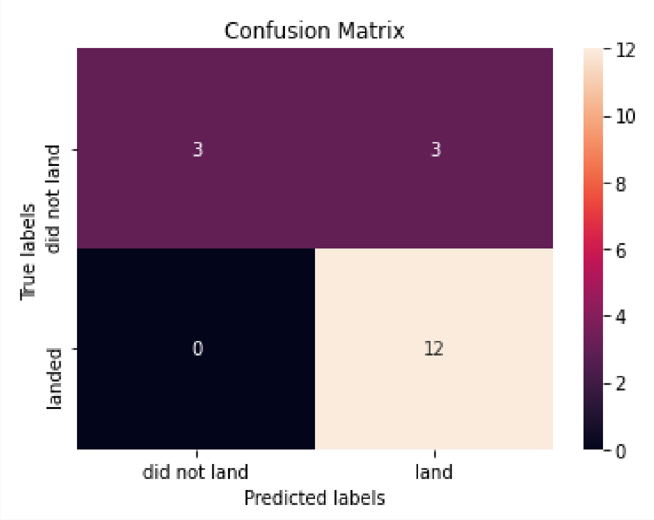
# Confusion Matrix

- All the 4 classification methods have the same classification matrix.
- The problem here is the number of false positives, i.e., unsuccessful landings incorrectly classified as successful landings.
- Overall, all the used 4 classification methods can reasonably distinguish between the different classes.

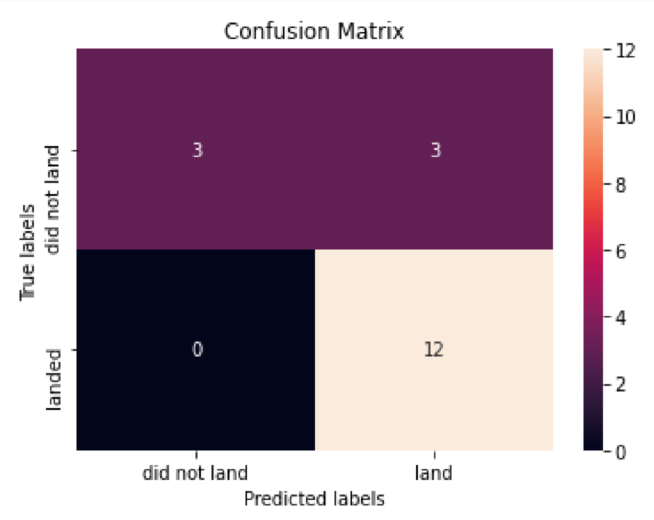
Logistic Regression



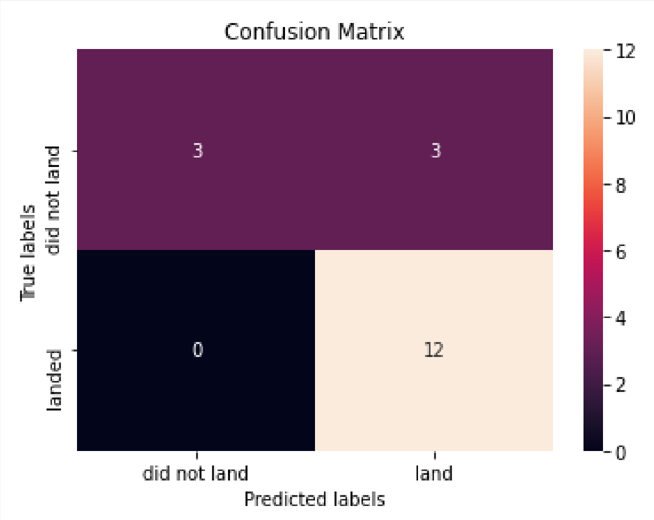

SVM




Decision Tree




KNN



# Conclusions

- The success rate of the SpaceX Falcon 9 launches have steadily increased from 2013 till 2020 and is close to 80% in recent years.
- The success rate has also increased with the number of flights.
- Rockets launched into the orbits ES-L1, GEO, HEO and SSO have 100% success rates.
- Amongst the 4 launch sites used by SpaceX, the launch site KSC LC-39A has the best success rate.
- Rockets launched with a low payload ( < 5000 kgs) have a higher success rate than the ones with heavy payloads (5000 – 10000 kgs).
- All the 4 classifier models (Logistic Regression, SVM, Decision Trees and KNN) used had the same accuracy (≈ 83.33%) on the test dataset.
- Any of these 4 machine learning models along with the insights above can be used to determine whether a future Falcon 9 launch will be successful.

# Thank you!