Winning Space Race with Data Science

Daler Pulatov
12/17/2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection via API and web scraping
  - Data wrangling
  - Exploratory data analysis (EDA) via SQL and visualization
  - Interactive visual analytics and dashboard via Folium and Plotly Dash
  - Machine Learning Model
- Summary of all results
  - Data collection and wrangling insights with screenshots
  - EDA with visualization and SQL insights and screenshots
  - Interactive visual analytics and dashboard insights and screenshots
  - Machine Learning Prediction insights and screenshots

Introduction

- Project background and context

  SpaceX promotes Falcon 9 rocket launches on its website that costs 62 million dollars while competitors spend 165 million dollars. SpaceX's primary cost saving trick is reusing its rockets after the first landing stage. If the first stage landing is known, then the cost could be determined. Other companies may use this information to outbid SpaceX. Project goal is to build a machine learning pipeline that predicts the first stage landing.

Problems you want to find answers

- What factors contribute to the first stage landing success or failure?
- What is the relationship among multiple variables that predict the first stage landing success?
- What conditions must be put in place to increase the probability of the first stage landing success?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected via SpaceX API and web scraped from Wikipedia.
- Perform data wrangling
  - Data was wrangled by applying one-hot encoding to categorical values.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

Multiple methodologies were utilized to collect data:

- SpaceX launch data was requested and parsed by using GET request
- Response content was decoded through json() and turned into Pandas dataframe by employing json_normalize()
- As part of data cleanup process, PayloadMass mean() was calculated and then used to replace() np.nan i.e. missing values
- Falcon 9 launch records were web scraped from Wikipedia with BeautifulSoup; HTML table launch records were parsed to create Pandas data frame
- Number of launches were calculated through value_counts() method on the LaunchSite
- Number and occurrence of each orbit were calculated by employing value_counts() on the Orbit
- Number and occurrence of mission outcome per orbit type were calculated using value_counts() on the Outcome
- A list was created to classify Outcome values into 0s and 1s and assign it to a variable landing_class

Data Collection – SpaceX API

- Rocket launch data was requested from SpaceX API.

Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

- Response content decoded via .json() method and turned into a Pandas dataframe by employing .json.normalize()

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

- PayloadMass .mean() was calculated and used to .replace() np.nan i.e. missing values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [32]:
```
# Calculate the mean value of PayloadMass column
payloadmean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9.replace(np.nan, payloadmean)
```

[SpaceX Data Collection Jupyter Notebook on Github]

Data Collection - Scraping

- HTTP GET method was used to request the Falcon 9 Launch HTML page from Wikipedia.

```
In [4]:
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

In [11]:
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code

Out[11]: 200
```

- BeautifulSoup object was created from HTML response text content.

```
In [12]:
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')

In [14]:
# Use soup.title attribute
soup.title

Out[14]: List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

- Column names are extracted to create Pandas data frame by parsing launch HTML tables

[SpaceX Webscraping Jupyter Notebook on Github]

```
In [17]:
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

Data Wrangling

- Exploratory Data Analysis (EDA) was performed to find some patterns in data and determine training labels.
- Number of launches were calculated for each site using value_counts() method on LaunchSite
- Number and occurrence of each orbit were calculated by employing value_counts() on Orbit
- Number and occurrence of mission outcome per orbit type were calculated by utilizing value_counts() on Outcome to determine landing_outcomes
- A binary list to classify Outcome values was created and assigned to landing_class variable

In [5]:
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Out[5]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

In [6]:
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

Out[6]:
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

In [7]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()

landing_outcomes
```

Out[7]:
```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

In [10]:
```
#landing_class = 0 if bad_outcome
#landing_class = 1 otherwise
#landing_class = []
for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

SpaceX Data Wrangling Jupyter Notebook on Github

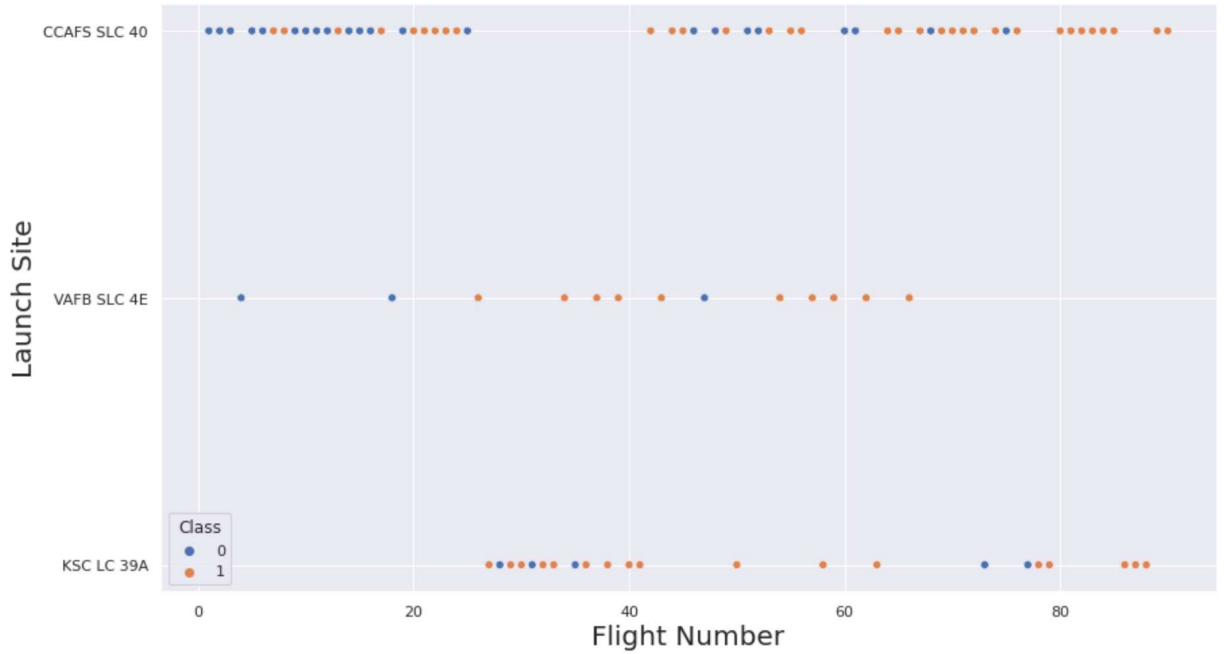
EDA with Data Visualization

- As part of EDA process, the relationship among multiple variables were plotted to identify those with high correlation that would be potentially removed from the predictive model

[SpaceX EDA with Visualization Jupyter Notebook on Github]

In [4]:

```
# 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.set_theme(color_codes = True)
plt.figure(figsize = (14, 8))
sns.scatterplot(x = "FlightNumber", y = "LaunchSite", hue = "Class", data = df)
plt.xlabel("Flight Number", fontsize = 20)
plt.ylabel("Launch Site", fontsize = 20)
plt.show()
```



EDA with SQL

- SQL queries were written to explore SpaceX Falcon 9 dataset by completing the following tasks:
  - ❖ Display the names of unique launch sites
  - ❖ Show the total payload mass carried by NASA launch boosters
  - ❖ Find out average payload mass carried by booster version F9 v1.1
  - ❖ List the date when the first landing outcome in ground pad was successful
  - ❖ Figure out the names of boosters which have successfully landed in a drone ship within specified payload mass
  - ❖ Ascertain the total number of successful and failure mission outcomes.
  - ❖ Determine the names of booster versions which have carried the maximum payload mass
  - ❖ Uncover the failed landing outcomes in a drone ship, their booster versions, and launch site names
  - ❖ Rank landing outcomes count between specific dates in a certain order

SpaceX EDA with SQL Jupyter Notebook on Github

Build an Interactive Map with Folium

- Folium was utilized to build interactive visualization that marks all launch sites on the map centered around NASA Johnson Space Center in Houston

- Since launch success rates depend on many factors like payload mass, orbit type, location, etc., Folium was essential to mark launch accomplishments/failures for each site

- Folium was useful tool for calculating distances between a launch site and its proximal objects: railways, highways and coastlines

SpaceX Interactive Visualization with Folium Jupyter Notebook on Github

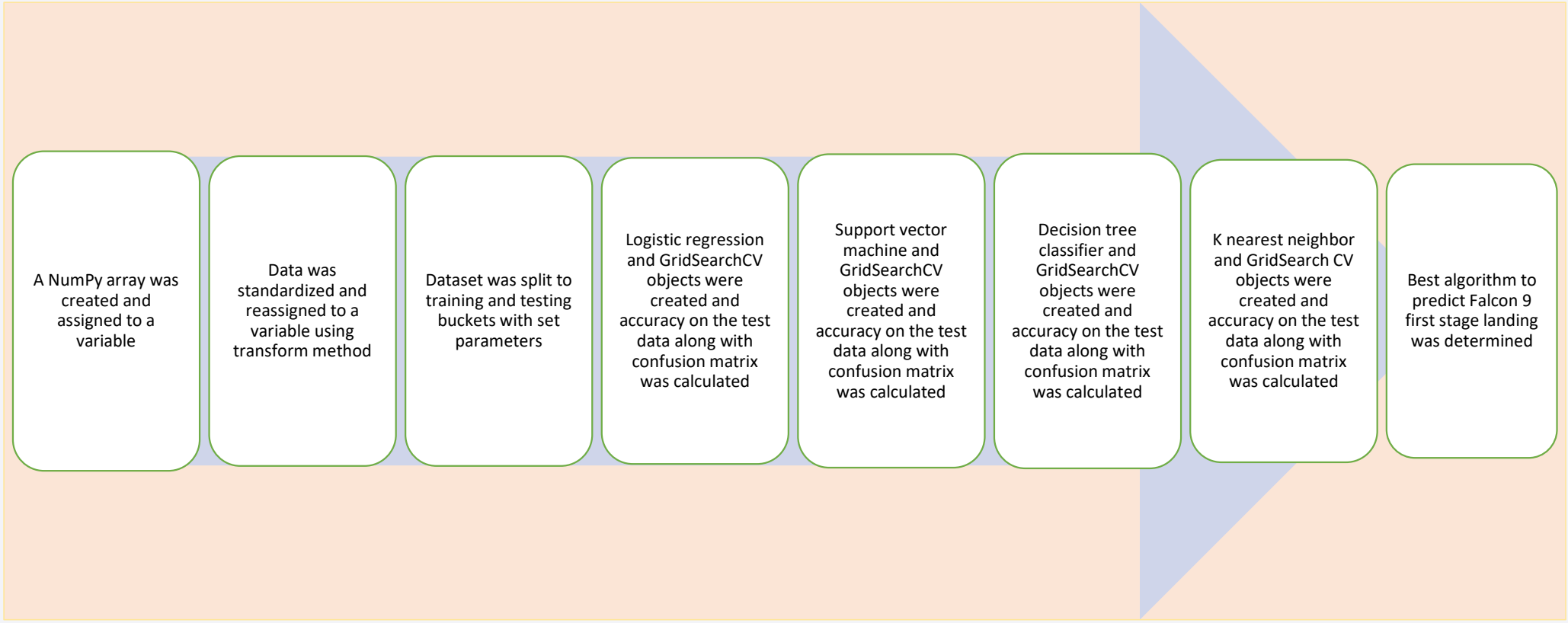
Build a Dashboard with Plotly Dash

- Plotly Dash was an essential tool to build an interactive dashboard that plotted pie charts showing total number of launches by different sites

- Scatter graph was drawn with Plotly Dash to show the relationship between Outcome and PayloadMass that helps determine highly correlated variables that would potentially be removed from the predictive model

SpaceX Plotly Dashboard Jupyter Notebook on Github

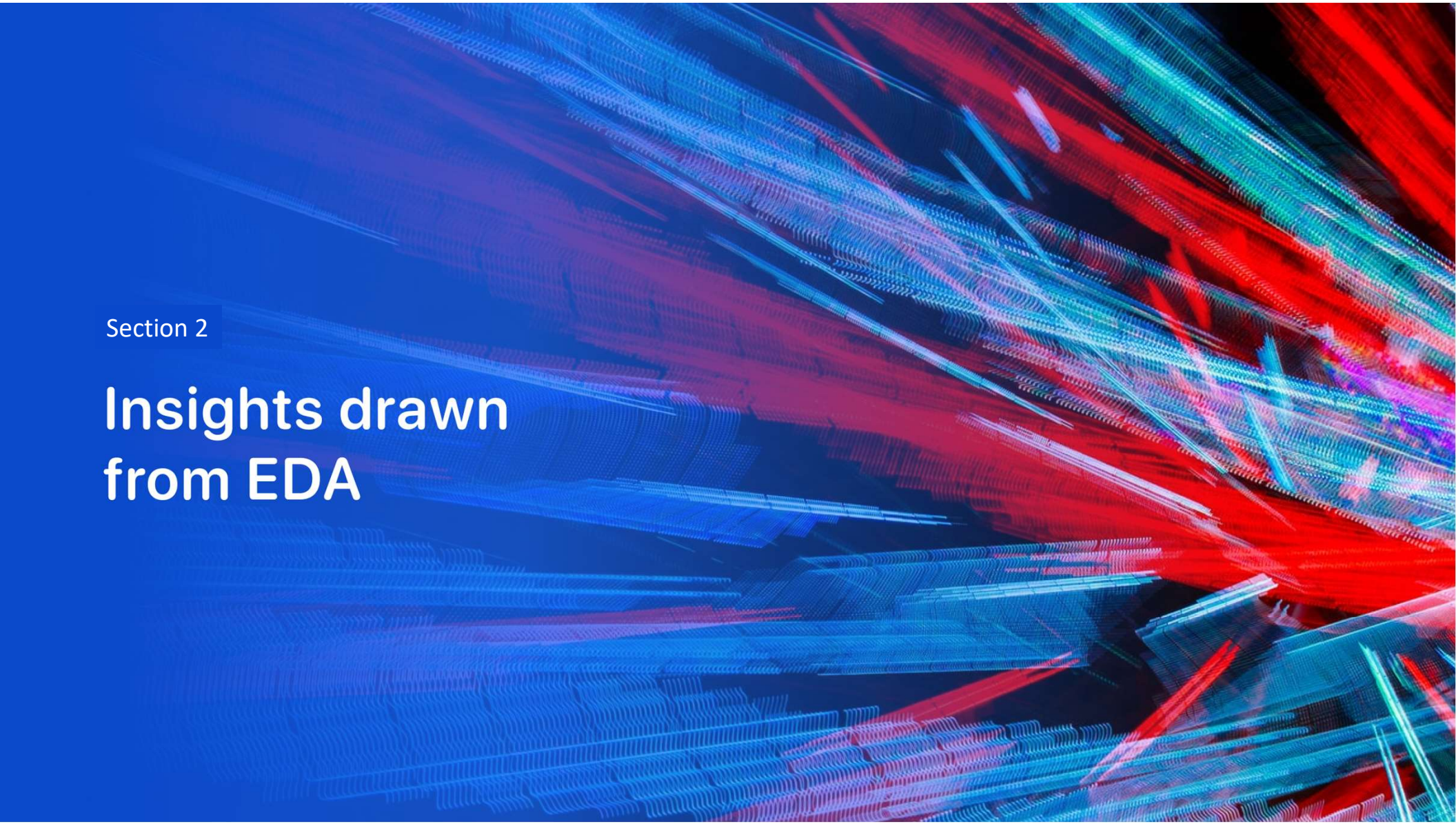
Predictive Analysis (Classification)



SpaceX Machine Learning Model Jupyter Notebook on Github

Results

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

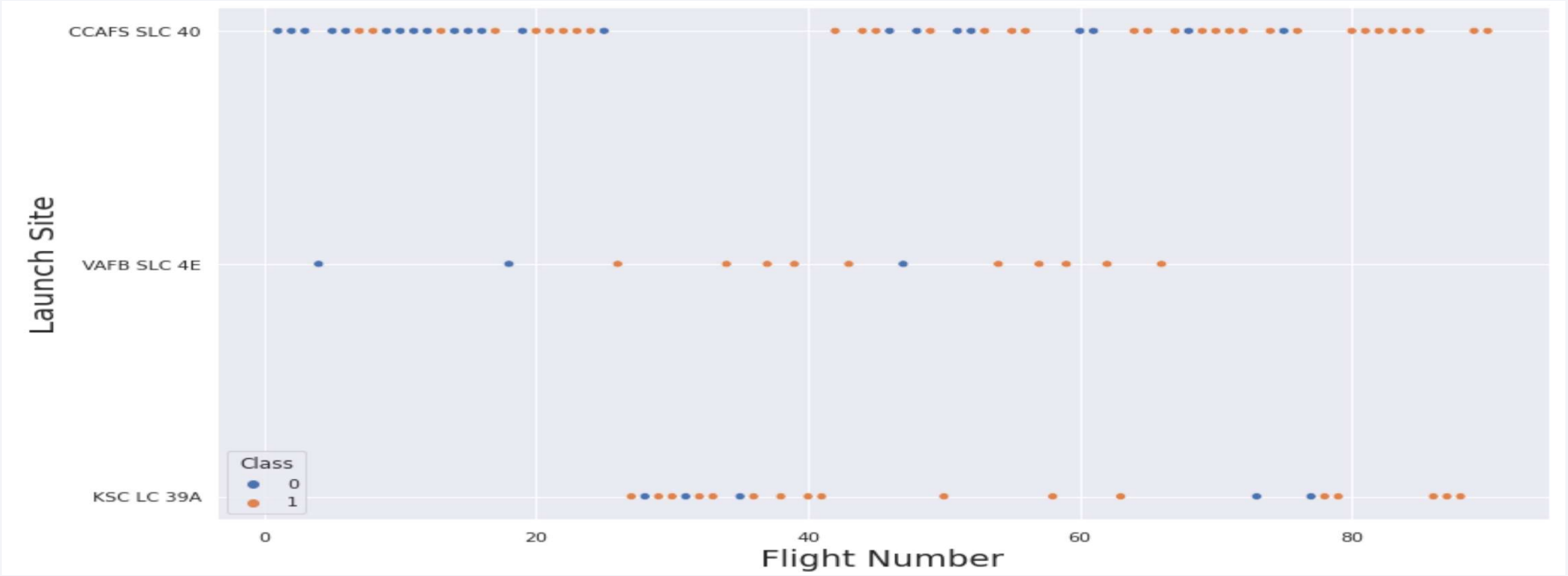


Section 2

Insights drawn from EDA

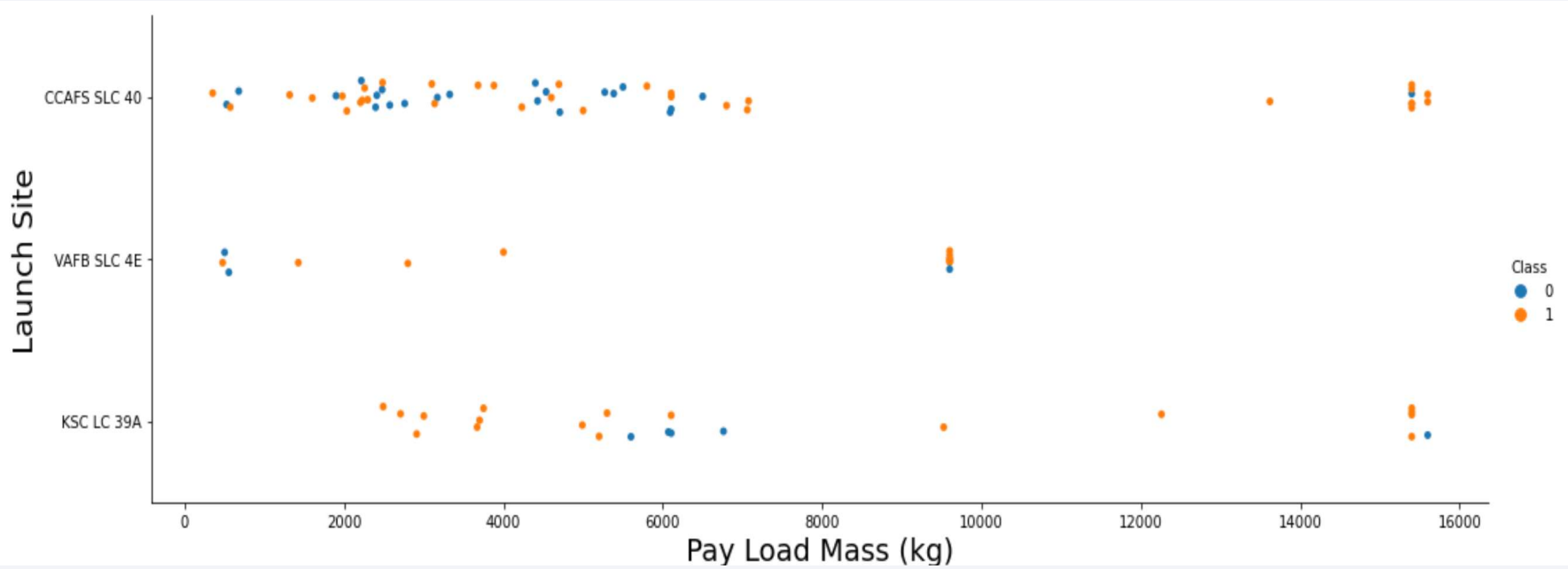
Flight Number vs. Launch Site

- As the flight number increases at VAFB SLC 4E and KSC LC 39A launch sites, the more successful these flights become
- However, CCAFS SLC 40 launch site has no correlation because of its inconsistent pattern



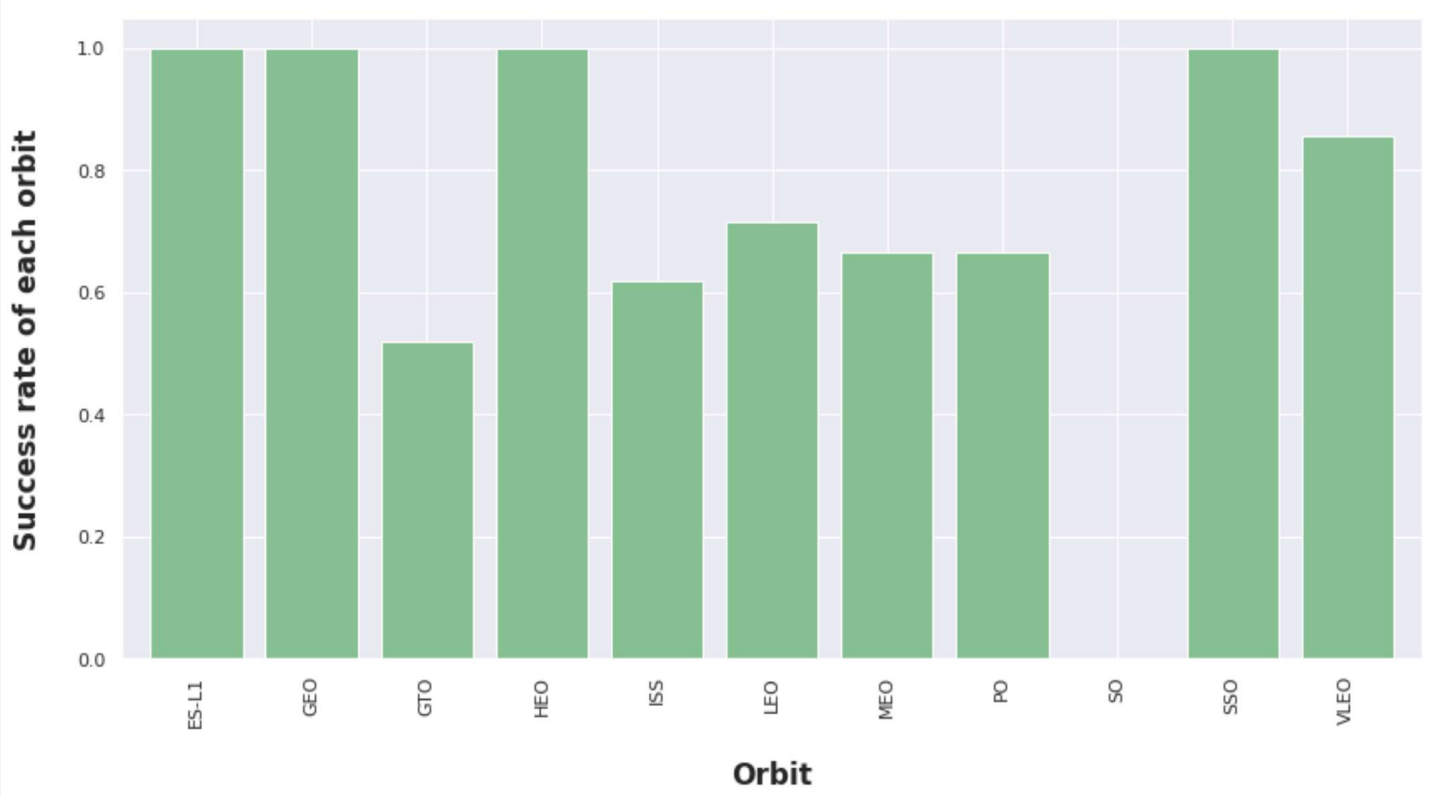
Payload vs. Launch Site

- Payload mass (kg) and launch success rate are positively correlated for all sites
- Payload with a mass greater than 8,000 kg has the highest launch success ratio



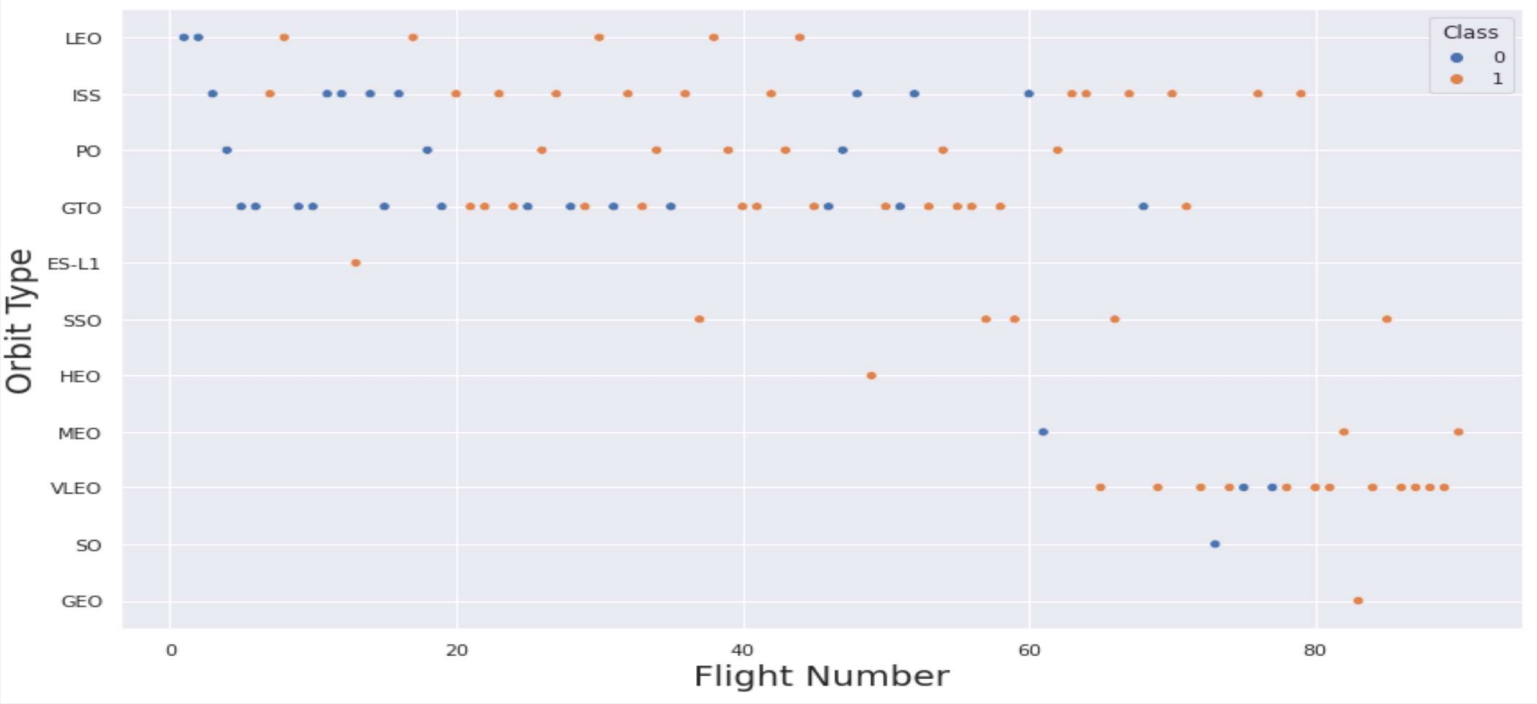
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO orbit types have the equally highest success rate as opposed to others



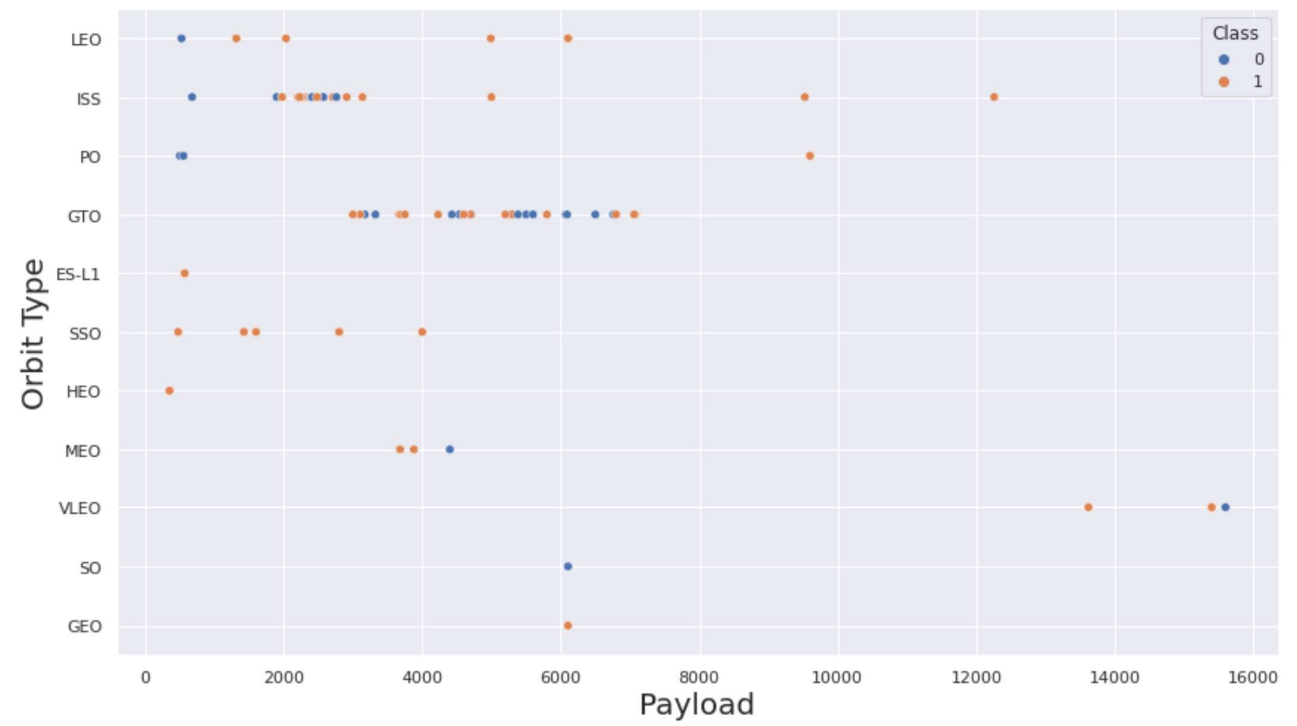
Flight Number vs. Orbit Type

- Launch success rate is positively correlated with a flight number for LEO and PO orbits; as the flight number increases, so does the success rate
- No correlation between flight number and launch success rate for GTO and ISS orbits could be observed
- Other orbits do not have sufficient data to extract a meaningful relationship



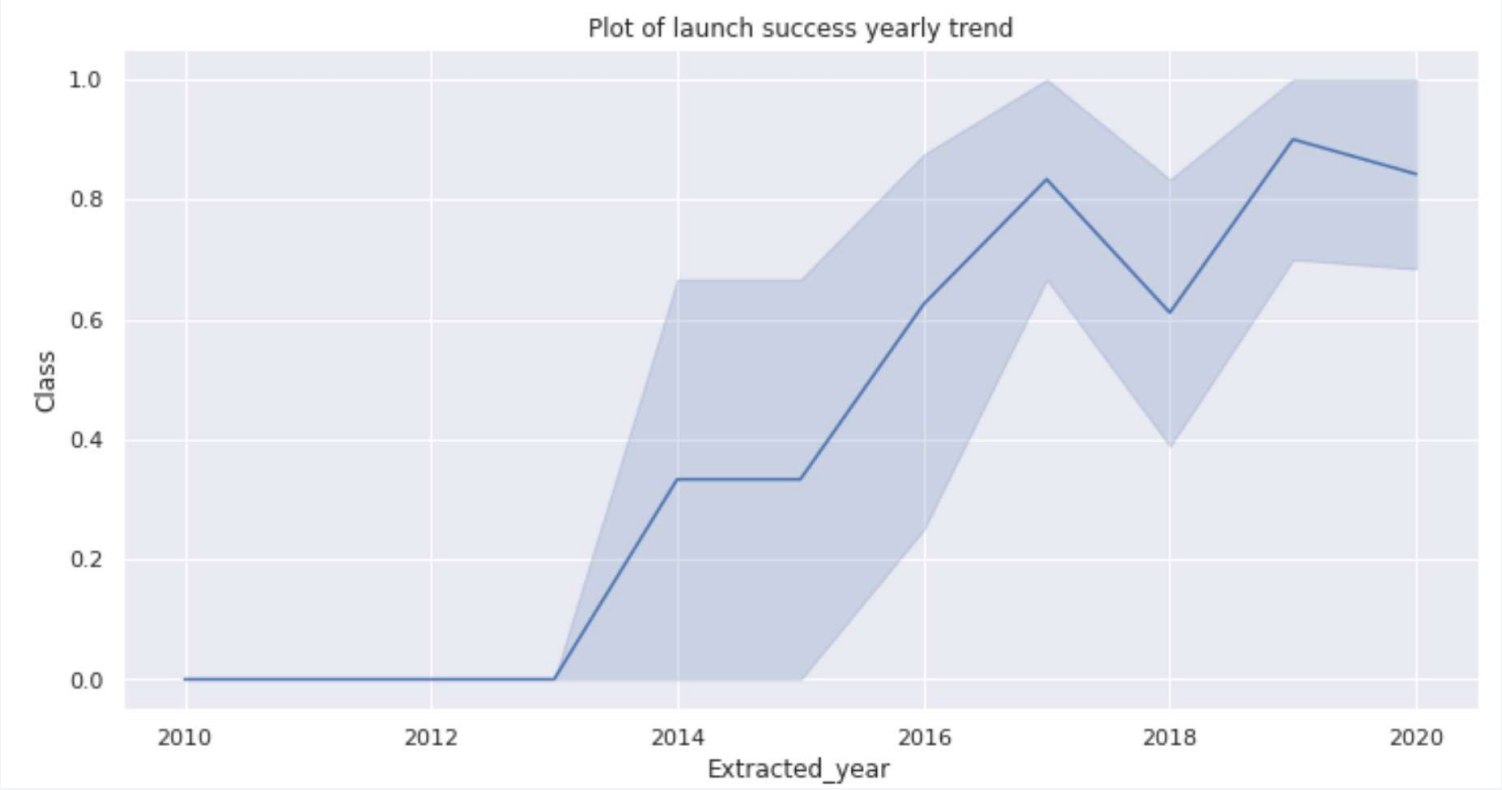
Payload vs. Orbit Type

- LEO, ISS and PO orbit types have positive relationship between payload mass and launch success rate. As payload increases, so does launch success rates
- GTO orbit shows no correlation between payload mass and launch success rate



Launch Success Yearly Trend

- Yearly launch success has been on an upward trend since 2013



All Launch Site Names

- To pull unique launch sites, SELECT Distinct syntax was written with a %sql magic on Python

In [17]:
```
%sql SELECT Distinct LAUNCH_SITE FROM SPACEX_DATA
```

```
 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[17]:

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

Launch Site Names Begin with 'CCA'

- To retrieve the first 5 records where launch sites begin with `CCA`, the following syntax was entered with %sql magic for Python

In [18]:
```
%sql SELECT * FROM SPACEX_DATA WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[18]:

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-12	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt

Total Payload Mass

- The total payload carried by boosters from NASA is 22,007 kg as calculated using the following query

In [19]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEX_DATA WHERE CUSTOMER='NASA (CRS)'
```

```
 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[19]:

1
22007

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is 3,676 kg as calculated by the query below

```
In [20]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEX_DATA WHERE BOOSTER_VERSION='F9 v1.1'
 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
Out[20]:
1
3676
```

First Successful Ground Landing Date

- The first successful landing outcome in ground pad was achieved on 1/5/2017 as determined by the %sql magic syntax below

```
In [21]: %sql SELECT min(DATE) FROM SPACEX_DATA WHERE LANDING__OUTCOME='Success (ground pad)'

 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.

Out[21]:        1

        2017-01-05
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- Below are the names of boosters which have successfully landed on a drone ship and had a payload mass greater than 4000 but less than 6000
- A %sql magic with WHERE | BETWEEN | AND clauses was run and written on Python through Jupyter notebook to obtain the desired outcome

In [22]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEX_DATA WHERE PAYLOAD_MASS__KG_ between 4000 and 6000 AND LANDING__OUTCOME='Success (drone ship)'
```

```
 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[22]:

booster_version
F9 FT B1022
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- 45 successful and failure mission outcomes were calculated by using '**%**' wildcard to count all records that contain **Success** or **Failure** keywords

```
In [23]: %sql SELECT COUNT(*) FROM SPACEX_DATA WHERE MISSION_OUTCOME LIKE '%Success%' OR MISSION_OUTCOME LIKE '%Failure%'
 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
Out[23]: 1

45
```

Boosters Carried Maximum Payload

- Here are the names of the booster which have carried the maximum payload mass obtained by utilizing a %sql magic subquery

In [24]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEX_DATA WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX_DATA)
```

```
 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[24]:

booster_version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3

2015 Launch Records

- A failed landing outcome in a drone ship with booster version and launch site name in 2015 could be seen below
- The query result was achieved by employing WHERE | AND | LIKE clauses and "%" wildcard

In [25]:
```
%sql SELECT TO_CHAR(TO_DATE(MONTH("DATE"), 'MM'), 'MONTH') AS MONTH_NAME, \
    LANDING__OUTCOME AS LANDING__OUTCOME, \
    BOOSTER_VERSION AS BOOSTER_VERSION, \
    LAUNCH_SITE AS LAUNCH_SITE \
    FROM SPACEX_DATA WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND "DATE" LIKE '%2015%'
```

```
 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[25]:

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

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

- Below is the count of landing outcomes (such as Failure on a drone ship or Success on a ground pad) between 2010-06-04 and 2017-03-20 in a descending order
- The query result was attained by applying WHERE | BETWEEN | AND | GROUP BY | ORDER BY clauses and "%" wildcard

In [26]:
```
%sql SELECT "DATE", COUNT(LANDING__OUTCOME) as COUNT FROM SPACEX_DATA \
    WHERE "DATE" BETWEEN '2010-06-04' and '2017-03-20' AND LANDING__OUTCOME LIKE '%Success%' \
    GROUP BY "DATE" \
    ORDER BY COUNT(LANDING__OUTCOME) DESC
```

```
 * ibm_db_sa://klc32942:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[26]:

DATE	COUNT
2016-06-05	1
2016-08-04	1
2017-01-05	1
2017-03-06	1

Section 3

Launch Sites Proximities Analysis

SpaceX Launch Site Locations

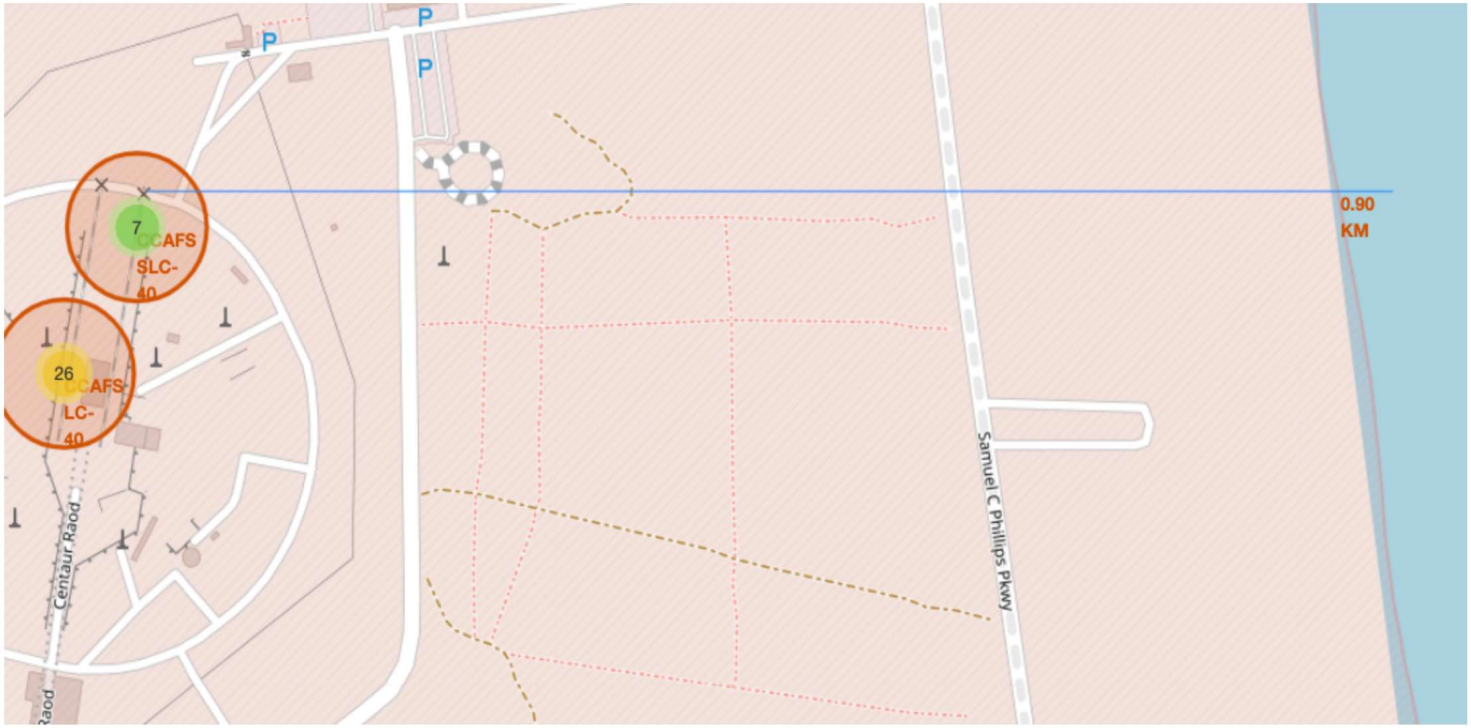
- SpaceX launch sites are in the Western and Eastern US coastal areas north of Los Angeles and Miami

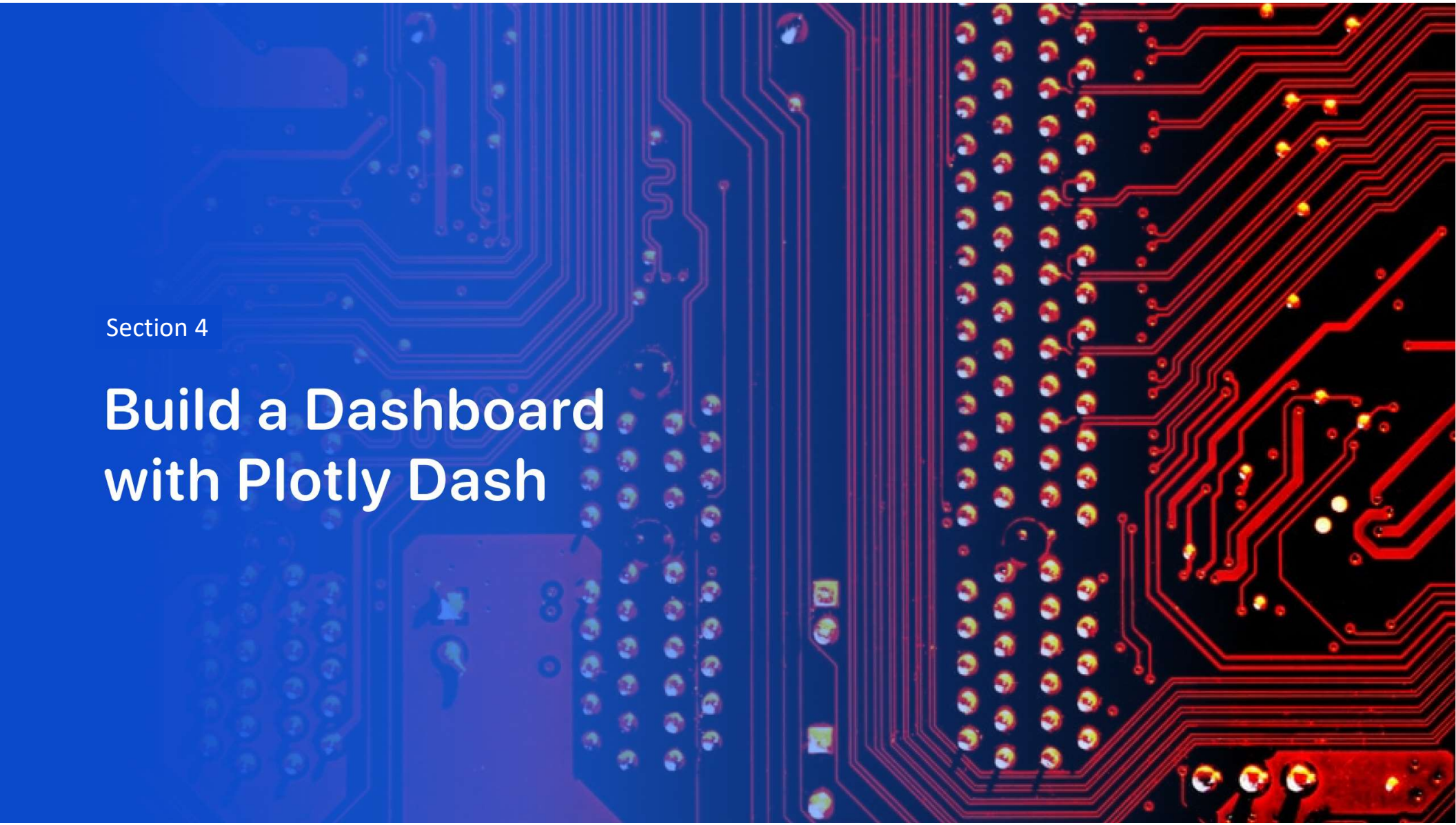
Launch Success and Failure

- CCAFS SLC-40 launch site in Florida has an equal number of successful, (marked in green) and failed (marked in red) launches

Launch Site Proximity to Existing Infrastructure

- CCAFS SLC-40 launch site in Florida is located 0.9 kilometers away from the coastline and is not close to any major highway, city or railway infrastructures

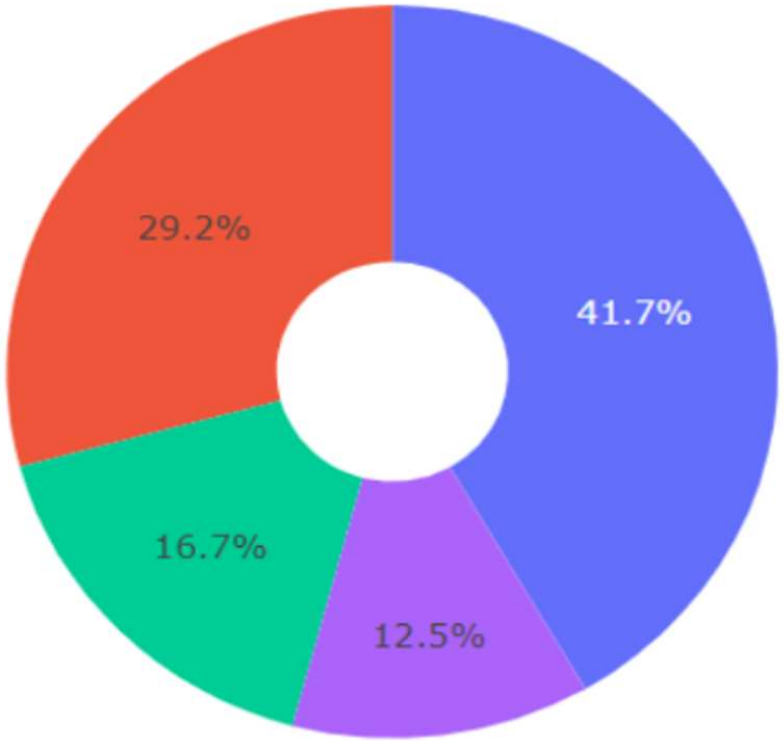




Section 4

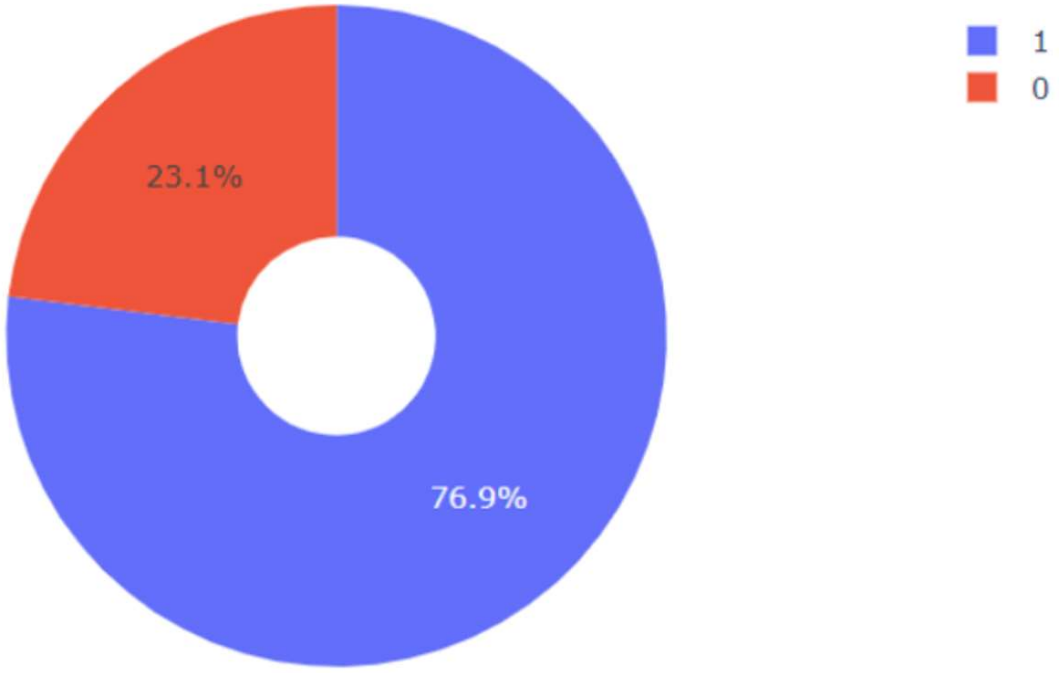
Build a Dashboard with Plotly Dash

Total Successful Launch Ratio by Sites | Pie Chart



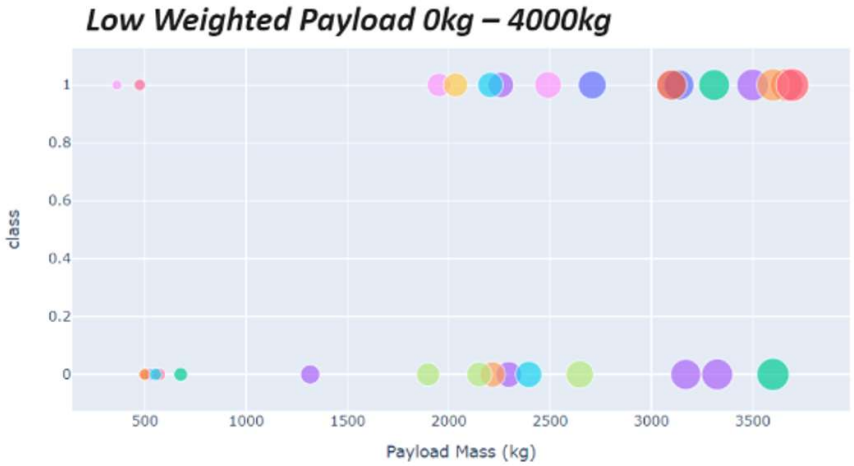
- KSC LC-39A is the most successful launch site at 42%
- Roughly 71% of all successful launches were from KSC LC39-A and CCAFS LC-40

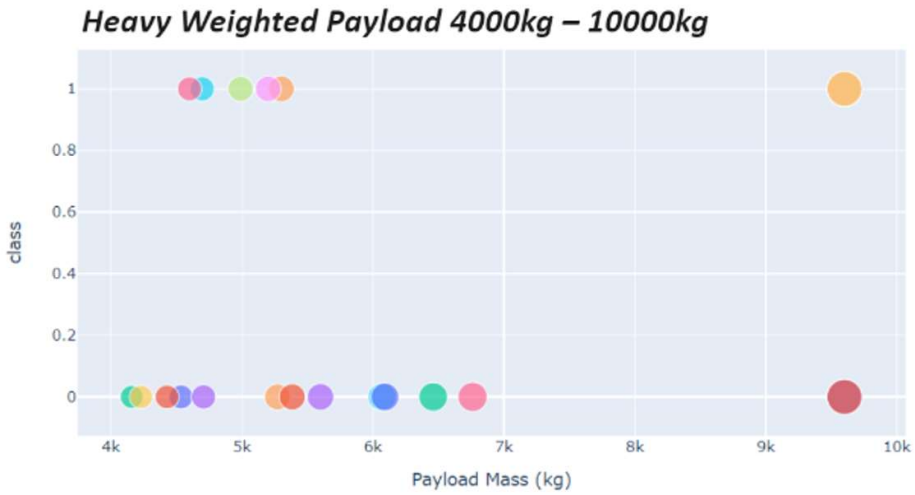
Highest Launch Site Success Ratio | Pie Chart



- KSC LC-39A success ratio is 77% - highest among all launch sites

Payload vs Launch Outcome | Scatter Plot





- Success rate for *Low Weighted Payload* is higher than *Heavy Weighted Payload*

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Decision Tree is the best method to predict first stage landing success/failure rate as it has the highest classification accuracy among all machine learning techniques

In [42]:

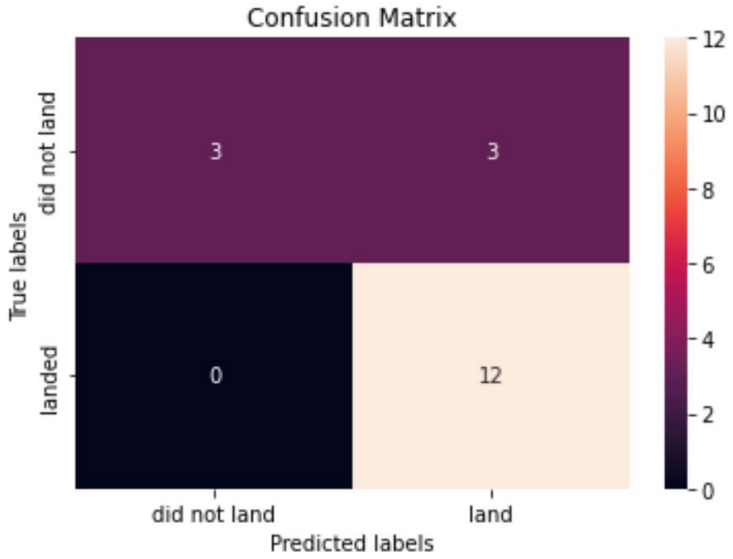
```
algorithms = {'KNN':knn_cv.best_score_,'Decision Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Decision Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Decision Tree with a score of 0.8767857142857143
Best Params is : {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'rando
m'}
```

Confusion Matrix

- As outlined by the confusion matrix, the predictive model did an excellent job at accurately predicting landing successes on 12 occasions i.e. 100% of the time
- However, the Decision Tree model did a poor job at incorrectly predicting landing failures on 3 occasions i.e. 50% of the time
- Lastly, the model correctly predicted landing failures on 3 occasions i.e. 50% of the time

```
In [31]: yhat = svm_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- VAFB SLC 4E & KSC LC 39A sites and LEO and PO orbits have the greatest chance of successful launches when the flight frequency is increased
- Heavy payloads i.e. mass > 8,000 kg and ES-L1, GEO, HEO and SSO orbit types have the highest launch success rates
- Annual successful launches have been rising steadily since 2013
- Total payload carried by NASA boosters: 22,007 kg
- Average payload carried by SpaceX booster version F9 v1.1: 3,676 kg
- F9 FT B1022 and F9 FT B1031.2 booster versions have successfully landed carrying payload mass between 4,000 and 6,000 kg
- Total number of successful/failure missions: 45
- F9 B5 B1048.4, F9 B5 B1049.4, F9 B5 B1049.5, F9 B5 B1060.2 and F9 B5 B1058.3 booster versions carried the maximum payload mass
- SpaceX has two launch sites: one close to Los Angeles and the other is close to Miami
- SpaceX launch site near Miami is less than one kilometer away from the coastline
- KSC LC-39A is most successful launch site and has the highest success ratio
- Low weighted payloads (0 – 4,000 kg) are more successful at landing than heavy weighted payloads (4,000 – 10,000 kg)
- Decision Tree is the best predictive modelling technique with the highest classification accuracy; it is excellent at predicting successful landings but not so much on failed landings

Appendix

- My SpaceX Falcon 9 first stage landing assignment labs can be found in the Github repository below: https://github.com/MarvelPlato/IBM-Data-Science-Professional-Certificate
- To the right of this statement are code snippets that turns categorical values into dummy variables and casts the data frame to a variable type: float64

In [27]:

```
# HINT: Use get_dummies() function on the categorical columns
features_one_hot = features

features_one_hot = pd.concat([features_one_hot,pd.get_dummies(df['Orbit'])], axis = 1)
features_one_hot.drop(['Orbit'], axis = 1, inplace = True)

features_one_hot = pd.concat([features_one_hot,pd.get_dummies(df['LaunchSite'])], axis = 1)
features_one_hot.drop(['LaunchSite'], axis = 1, inplace = True)

features_one_hot = pd.concat([features_one_hot,pd.get_dummies(df['LandingPad'])], axis = 1)
features_one_hot.drop(['LandingPad'], axis = 1, inplace = True)

features_one_hot = pd.concat([features_one_hot,pd.get_dummies(df['Serial'])], axis = 1)
features_one_hot.drop(['Serial'], axis = 1, inplace = True)

features_one_hot.head()
```

In [28]:

```
# HINT: use astype function
features_one_hot = features_one_hot.astype(float)
features_one_hot
```

Thank you!