IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Lourdes Devenney
19-FEB-2025

Outline

Executive Summary

- Summary of methodologies
  - Data was collected via SpaceX public REST API and processed via data wrangling techniques such as exploratory data analysis
  - Other items used were SQL, interactive maps with Folium, dashboards with Plotly Dash and predictive analytics

- Summary of all results
  - Interactive maps, dashboards, and predictive analytics

Introduction

- Project background and context
  - Space Y that would like to compete with SpaceX founded by Billionaire industrialist Allon Musk. The purpose is to determine the price of each SpaceX's Falcon 9 launch.
- Problems you want to find answers
  - Gather information about Space X and create dashboards.
  - Determine if SpaceX will reuse the first stage by training a machine learning model and use public information to predict if SpaceX will reuse the first stage.

Space Y

Section
1
Methodology

Methodology

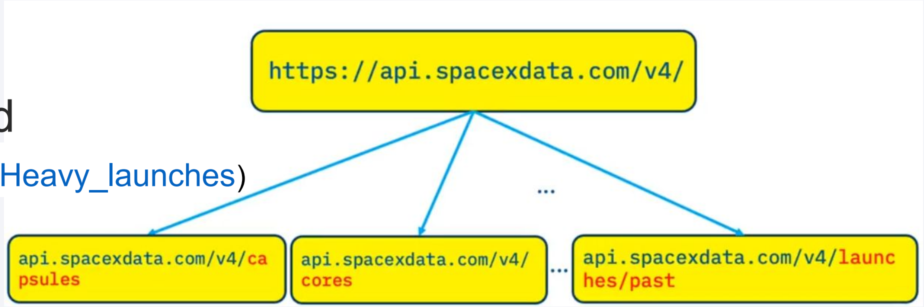
Executive Summary

- Data collection methodology:
  - Data was collected via public API and web scraping
- Perform data wrangling
  - Data was processed using exploratory data analysis data wrangling.
  - Dropped columns not used
- 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

- Datasets were collected using SpaceX REST API

Call SpaceX REST API → JSON File retured → Create Metaframe with JSON → Clean and export data



- Webscraping from Wikipedia was also used (https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches)

Request the Falcon9 Launch Wiki page from its URL → Extract all column/variable names from the HTML table header → Create a data frame by parsing the launch HTML tables

Data Collection – SpaceX API

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.js
```

We should see that the request was successfull with the 200 status response code

```
response=requests.get(static_json_url)
```

```
response.status_code
```

```
200
```

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

Now that we have removed some values we should reset the FlgihtNumber column

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

Data Wrangling

We can see below that some of the rows are missing values in our dataset.

```
[29]: data_falcon9.isnull().sum()
```

```
[29]: FlightNumber      0
      Date              0
      BoosterVersion    0
```

Task 3: Dealing with 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.

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

https://github.com/lfdevenney/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api%20(1).ipynb

Data Collection - Scraping

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all("table")
print(html_tables)
```

```
[<table class="col-begin" role="presentation">
<tbody><tr>
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Lat
into a Pandas dataframe

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.

https://github.com/lfdevenney/IBM-Applied-Data-Science-Capstone/blob/c69ab8bbbc5363042496207f9f918bb20d2da454/jupyter-labs-webscraping%20(1).ipynb

Data Wrangling

- Performed Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- There are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident.

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space La 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

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

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbi`

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

```
Orbit
GTO        27
```

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

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

```
Outcome
True ASDS      41
None None      19
```

TASK 4: Create a landing outcome label from Outcome column

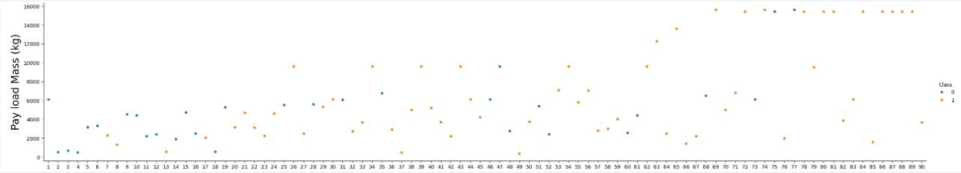
Using the `Outcome` , create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome` ; otherwise, it's one. Then assign it to the variable `landing_class` :

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

https://github.com/lfdevenney/IBM-Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Scatter Graphs
  - FlightNumber vs. PayloadMass
  - FlightNumber vs LaunchSite
  - Payload vs LaunchSite
  - Payload Mass vs. Orbit
  - Orbit vs FlightNumber
  - Orbit vs PayloadMass
- Success rate bar graph
- Success rate vs. year line graph



https://github.com/lfdevenney/IBM-Applied-Data-Science-Capstone/blob/main/edadataviz.ipynb

EDA with SQL

- Summary of the SQL queries performed
  - **Display the names of the unique launch sites in the space mission**
  - **Display 5 records where launch sites begin with the string 'CCA'**
  - **Display the 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 succesful landing outcome in ground pad was acheived.**
  - **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 records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.**
  - **Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.¶**

https://github.com/lfdevenney/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Initial center location to be NASA Johnson Space Center at Houston, Texas.
- Added a highlighted circle area with a text label on a specific coordinate
- Added blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
- Added a Circle object based on its coordinate (Lat, Long) values
- Marked the success/failed launches for each site on the map
- Used a green marker and if a launch was failed, we use a red marker (class=0)
- Calculated the distances between a launch site to its proximities
- Created a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
-

https://github.com/lfdevenney/IBM-Applied-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Built a Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time.
- This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart. You will be guided to build this dashboard application via the following tasks:
- Added a Launch Site Drop-down Input Component
- Added a callback function to render success-pie-chart based on selected site dropdown
- Added a Range Slider to Select Payload
- Added a callback function to render the success-payload-scatter-chart scatter plot

Predictive Analysis (Classification)

- Summary of how you built, evaluated, improved, and found the best performing classification model
    - Performed exploratory Data Analysis and determined Training Labels
        - created a column for the class
        - Standardized the data
        - Split into training data and test data
    - Identified best Hyperparameter for SVM, Classification Trees and Logistic Regression
        - Identified the method performs best using test data

https://github.com/lfdevenney/IBM-Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

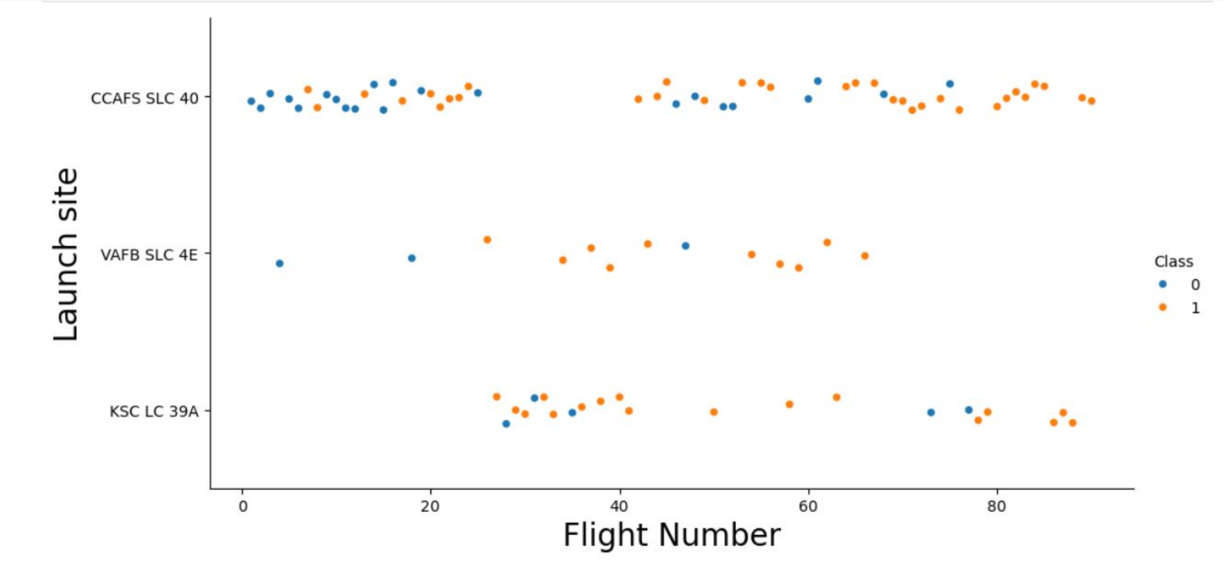
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section
2

Insights drawn from EDA

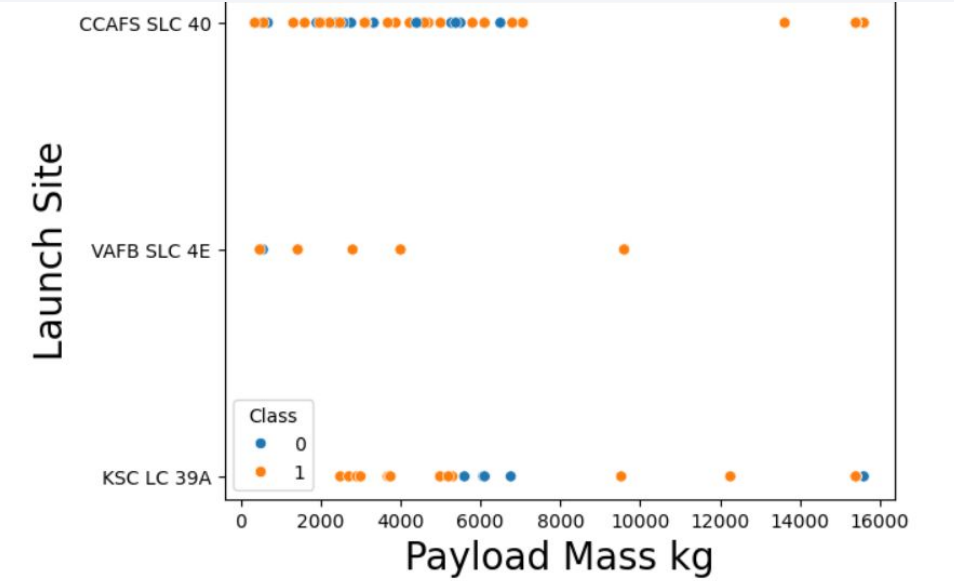
Flight Number vs. Launch Site

- For each Launch Site, the success rate is increasing



Payload vs. Launch Site

- Depending on the Launch Site for the VAFB-SLC launchsite there are no rockets launched for heavy payload mass(greater than 10000)



Success Rate vs. Orbit Type

- Success rate for different orbit types: ES-L1, GEO, HEO, SSO have the best success rates

TASK 3: Visualize the relationship between success rate of each orbit type

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

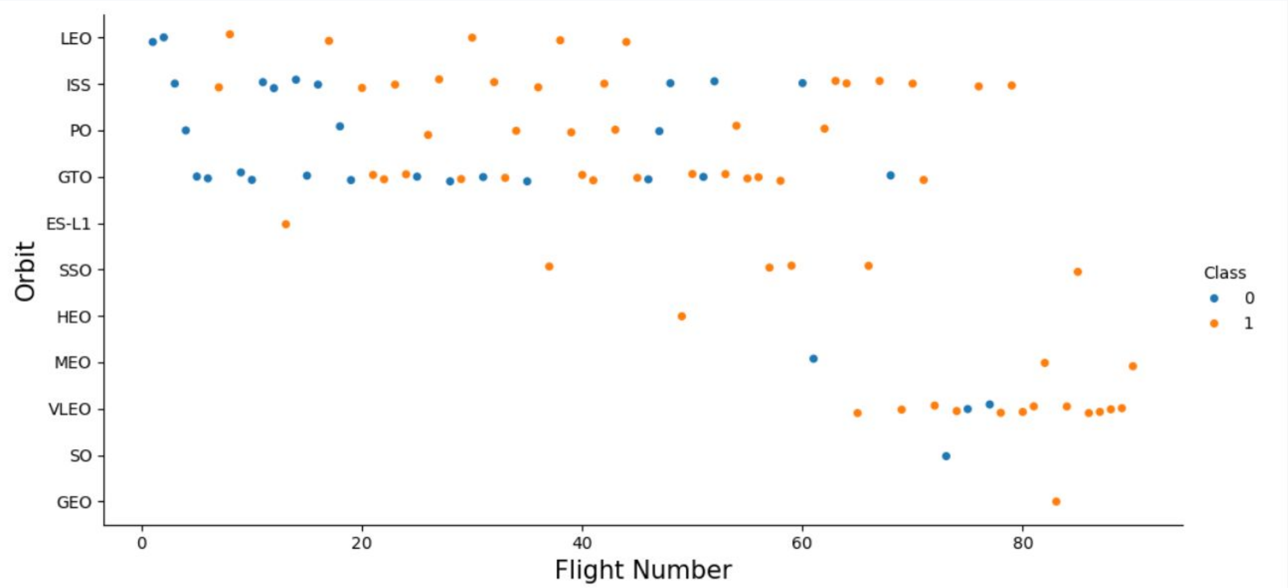
```
]: # HINT use groupby method on Orbit column and get the mean of Class column
bar1=df.groupby("Orbit")["Class"].mean()
bar1
```

```
]: Orbit
ES-L1    1.000000
GEO      1.000000
GTO      0.518519
HEO      1.000000
ISS      0.619048
LEO      0.714286
MEO      0.666667
PO       0.666667
SO       0.000000
SSO      1.000000
VLEO     0.857143
Name: Class, dtype: float64
```

Analyze the plotted bar chart to identify which orbits have the highest success rates.

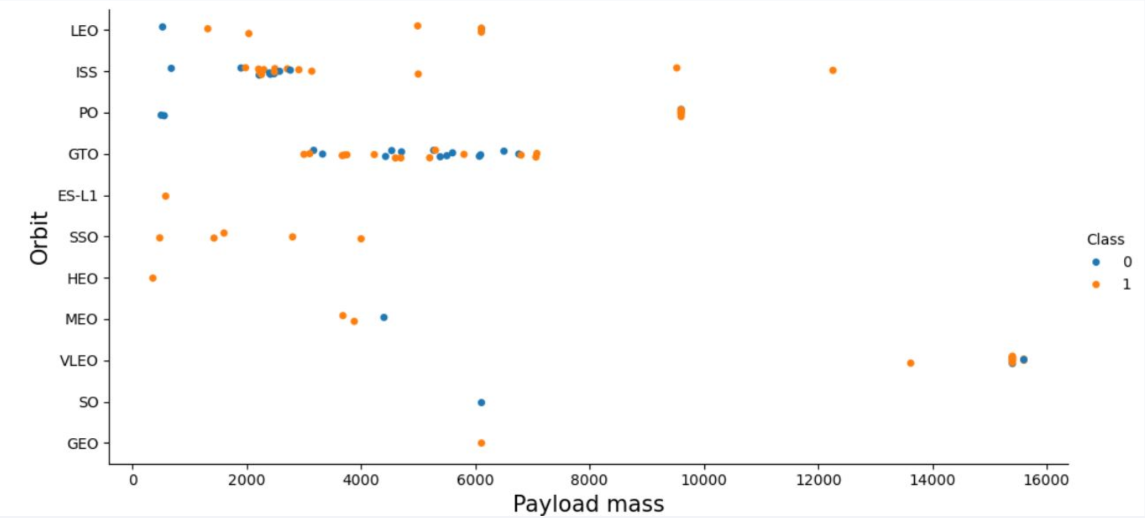
Flight Number vs. Orbit Type

- Observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.



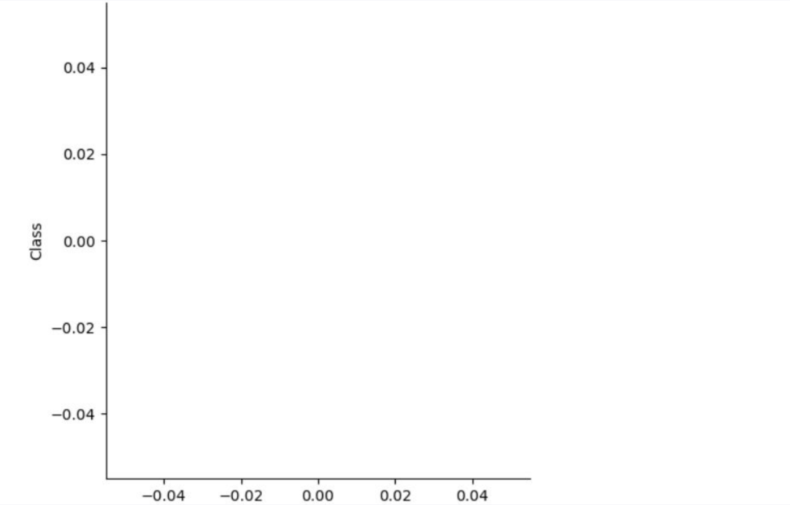
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.



Launch Success Yearly Trend

- Observe that the success rate since 2013 kept increasing till 2020



All Launch Site Names

- Using the key word DISTINCT, results in unique launch sites results

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

[12]:

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

[12]:

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

Results

Launch Site Names Begin with 'CCA'

- The WHERE key word filters the result set with CCA and limits the number of rows returned by 5 with key word LIMIT

Display 5 records where launch sites begin with the string 'CCA'

```
13]: %sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

```
 * sqlite:///my_data1.db
Done.
```

13]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Results

Total Payload Mass

- The key word SUM adds payload mass rows for “NASA (CRS)”

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

```
[14]: %sql SELECT SUM(PAYLOAD_MASS__KG_) as PM_KG_TOTAL, Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)'
```

```
 * sqlite:///my_data1.db
Done.
```

[14]:

PM_KG_TOTAL	Customer
45596	NASA (CRS)

Result

Average Payload Mass by F9 v1.1

- The key word AVG provides the average of payload mass for “F9 v1.1”

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

```
[15]: %sql SELECT AVG(PAYLOAD_MASS__KG_) as PM_KG_AVG FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%'
```

```
 * sqlite:///my_data1.db
Done.
```

[15]:

PM_KG_AVG
2534.6666666666665

Result

First Successful Ground Landing Date

- The key word MIN provides the first successful landing date

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
6]: cur.execute('''SELECT MIN(Date), "Landing _Outcome" FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (ground pad)"''')
    cur.fetchall()
```

```
6]: [(None, 'Landing _Outcome')]
```

Result

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

[17]:
```
%sql SELECT DISTINCT Booster_Version FROM SPACEXTBL WHERE Mission_Outcome = 'Success' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 60
```

```
 * sqlite:///my_data1.db
Done.
```

[17]:

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 FT B1032.2
F9 B4 B1040.2
F9 B4 B1040.2
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

Results

- Using key word DISTINCT returns successful drone ship landing with payload between 4000 and 6000

Total Number of Successful and Failure Mission Outcomes

- Key word COUNT returns number of mission outcomes

List the total number of successful and failure mission outcomes

```
[18]: %sql SELECT Mission_Outcome, COUNT(Mission_Outcome) as Total FROM SPACEXTBL GROUP BY Mission_Outcome;
```

```
 * sqlite:///my_data1.db
Done.
```

[18]:

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Results

Boosters Carried Maximum Payload

- Key word MAX returns maximum payload results using a sub-query

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

```
[19]: %sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

```
 * sqlite:///my_data1.db
Done.
```

[19]:

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

Results

2015 Launch Records

- Using the YEAR key word to select from the DATE field returns landing outcome of for “Failure (drone ship)”

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
[20]: %sql SELECT DATE, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE YEAR(DATE) = '2015' AND LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such function: YEAR
[SQL: SELECT DATE, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE YEAR(DATE) = '2015' AND LANDING__OUTCOME = 'Failure (drone ship)' ;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

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

- Query uses key word BETWEEN to rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
[ ]: %sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS Landing_Count FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP B
```

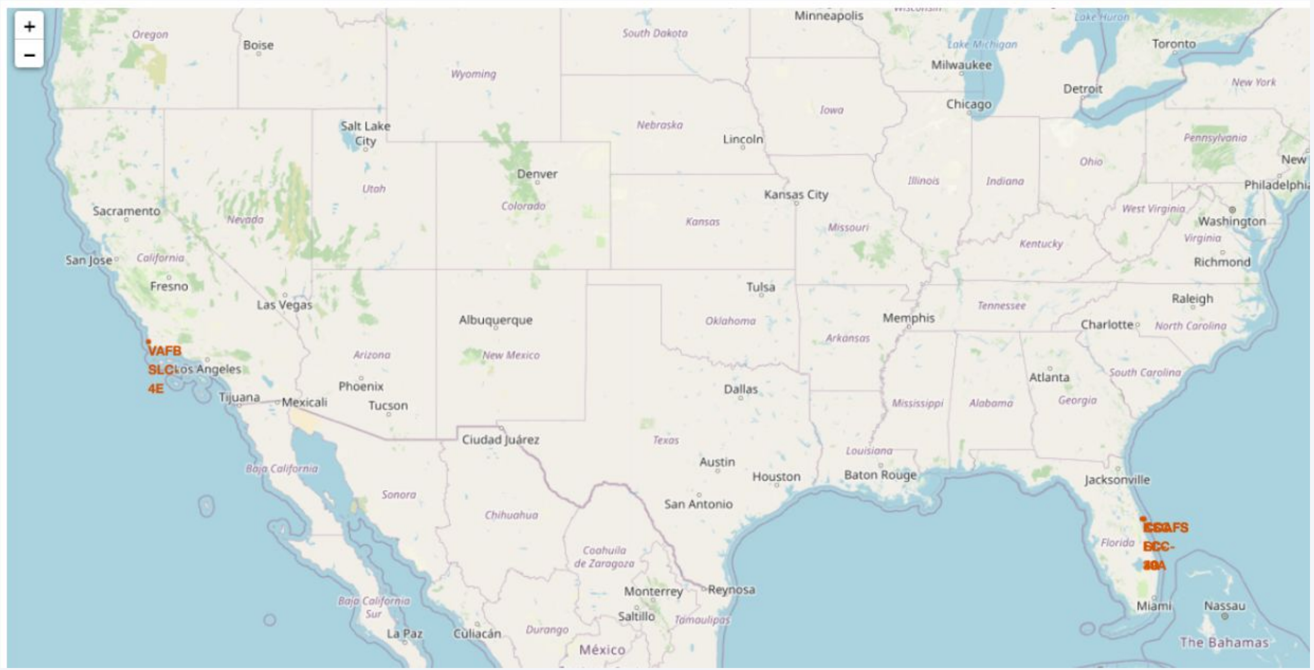
```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS Landing_Count FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING__OUTCOME ORDER BY COUNT(LANDING__OUTCOME) DESC;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

Section
3

Launch Sites Proximities Analysis

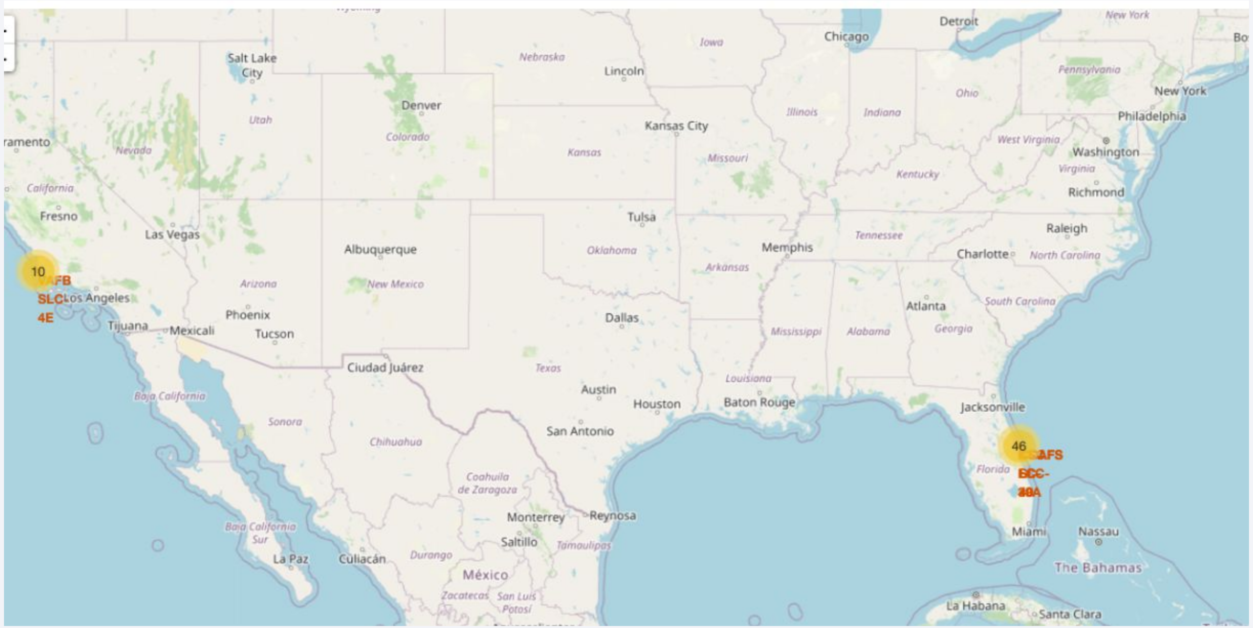
<Folium Map Launch Sites>

- # For each launch site, added a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label



<Folium Map Marker Success/Failure>

- for each row in spacex_df data frame, # create a Marker object with its coordinate, # and customize the Marker's icon property to indicate if this launch was successed or failed, # e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']



<Folium Map PolyLine>

Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate

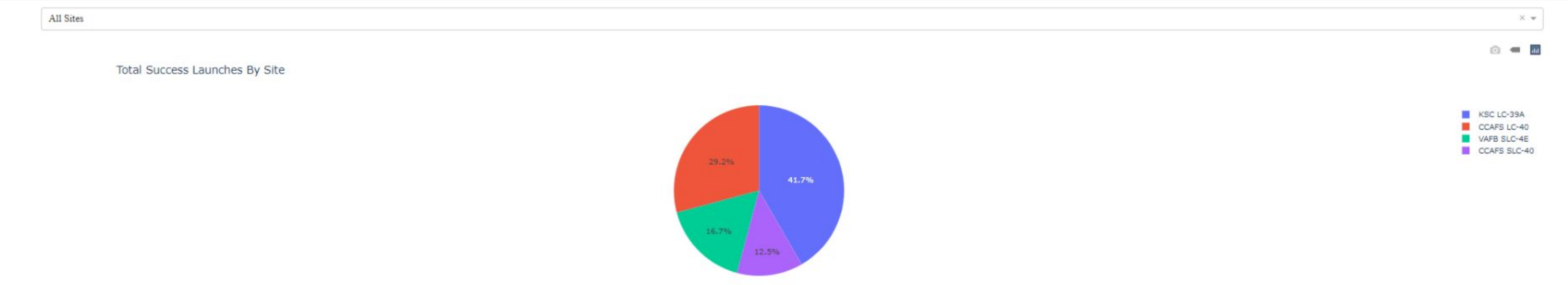
lines=folium.PolyLine(locations=coordinates, weight=1)

Section
4

Build a Dashboard with Plotly Dash

<Dashboard All Launch Sites>

The callback function gets the selected launch site from `site-dropdown` and render pie chart visualizing launch success counts.

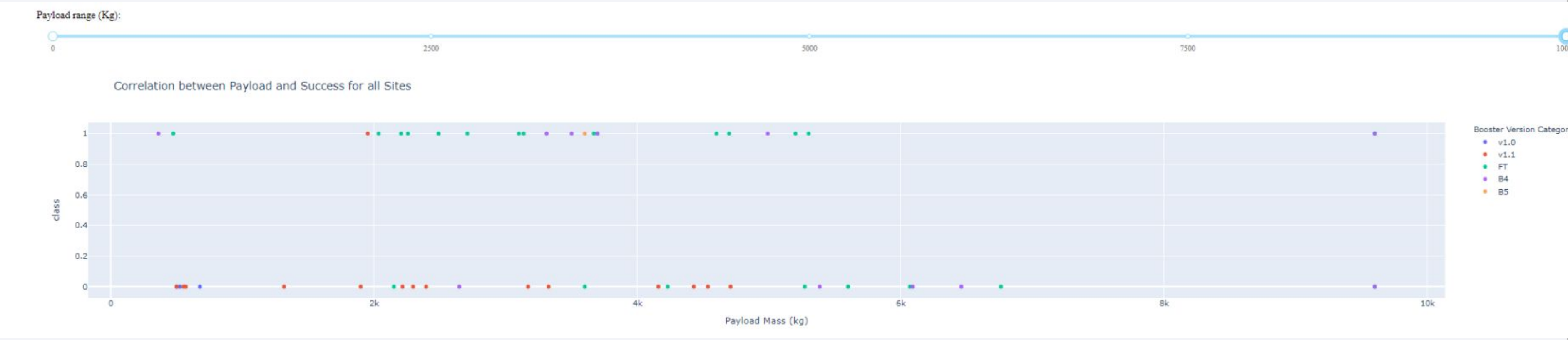


<Dashboard Payload to Mission Outcome>

Find if variable payload is correlated to mission outcome. From a dashboard point of view, we want to be able to easily select different payload range and see if we can identify some visual patterns.

<Dashboard Launch Outcome>

- Plot a scatter plot with the x axis to be the payload and the y axis to be the launch outcome (i.e., class column). As such, we can visually observe how payload may be correlated with mission outcomes for selected site(s).
- In addition, we want to color-label the Booster version on each scatter point so that we may observe mission outcomes with different boosters.



Section
5

Predictive Analysis (Classification)

Classification Accuracy

- Visualizing the built model accuracy for all built classification models, is 83%

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`.

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

```
[13]: 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)
```

```
[13]: ▸ GridSearchCV
      ▸ estimator: LogisticRegression
        ▸ LogisticRegression
```

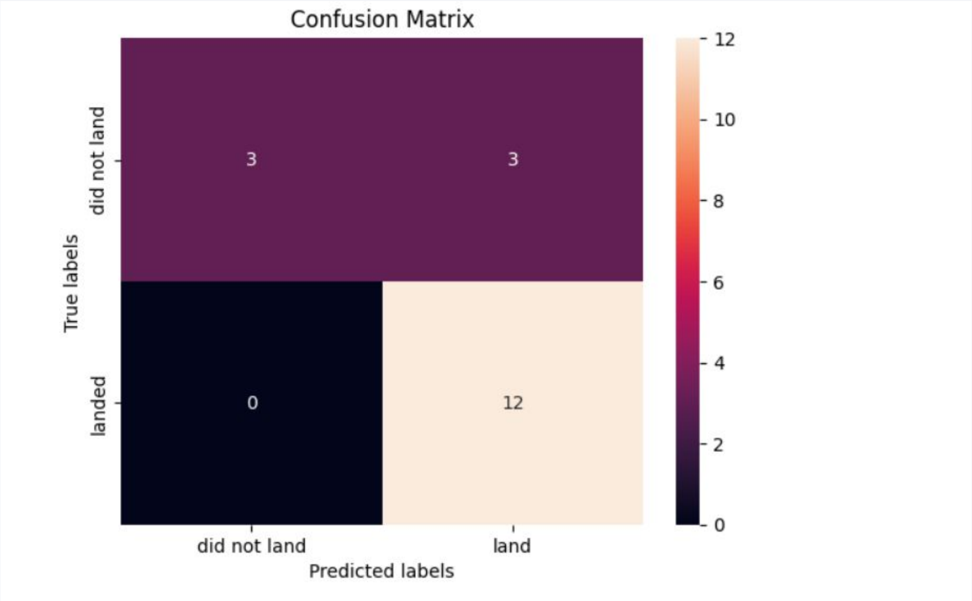
We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
[14]: 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
```

Confusion Matrix

- Confusion matrix of the best performing model of 83%



Conclusions

- As the number of launch sites increases, the rate of success increases, with most early flights not successful
- Orbit types ES-L1, SSO, HEO, GEO have the highest success rate
- The Decision Tree classifier is the best machine learning algorithm for this analysis
- KSC LC-39 A had the most successful launches
- Launch success rate started to increase in 2013 to 2020

Thank you!