IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Varshini B R
30Th January 2025

Outline

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

Executive Summary

- Summary of methodologies
- SpaceX Data Collection using SpaceX API
- SpaceX Data Collection with Web Scraping
- SpaceX Data Wrangling
- SpaceX Exploratory Data Analysis using SQL
- Space-X EDA DataViz Using Python Pandas and Matplotlib
- Space-X Launch Sites Analysis with Folium-Interactive Visual Analytics and Ploty Dash
- SpaceX Machine Learning Landing Prediction
- Summary of all results
- EDA results
- Interactive Visual Analytics and Dashboards
- Predictive Analysis(Classification)

Introduction

- Project background and context

Project background and context SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers

In this capstone, we will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

- Describe how data sets were collected.
  - Data was first collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API. This was done by first defining a series helper functions that would help in the use of the API to extract information using identification numbers in the launch data and then requesting rocket launch data from the SpaceX API url.
  - Finally to make the requested JSON results more consistent, the SpaceX launch data was requested and parsed using the GET request and then decoded the response content as a Json result which was then converted into a Pandas data frame.
  - Also performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches of the launch records are stored in a HTML. Using BeautifulSoup and request Libraries, I extract the Falcon 9 launch HTML table records from the Wikipedia page, Parsed the table and converted it into a Pandas data frame

Data Collection – SpaceX API

- Data collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API then requested and parsed the SpaceX launch data using the GET request and decoded the response content as a Json result which was then converted into a Pandas data frame

https://github.com/VarshiniBR/AIHackathon/blob/main/Complete%20the%20Data%20Collection%20API%20Lab.ipynb

c) · 531 KB Code 55% faster with GitHub Copilot

In [4]:
```
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

In [5]:
```
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

In [6]:
```
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

In [7]:
```
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

In [8]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [9]:
```
response = requests.get(spacex_url)
```

In [10]:
```
print(response.content)
```

Data Collection - Scraping

- Performed web scraping to collect Falcon 9 historical launch records from a Wikipedia using BeautifulSoup and request, to extract the Falcon 9 launch records from HTML table of the Wikipedia page, then created a data frame by parsing the launch HTML.

https://github.com/VarshiniBR/AIHackathon/blob/main/Complete%20the%20Data%20Collection%20with%20Web%20Scraping%20lab.ipynb

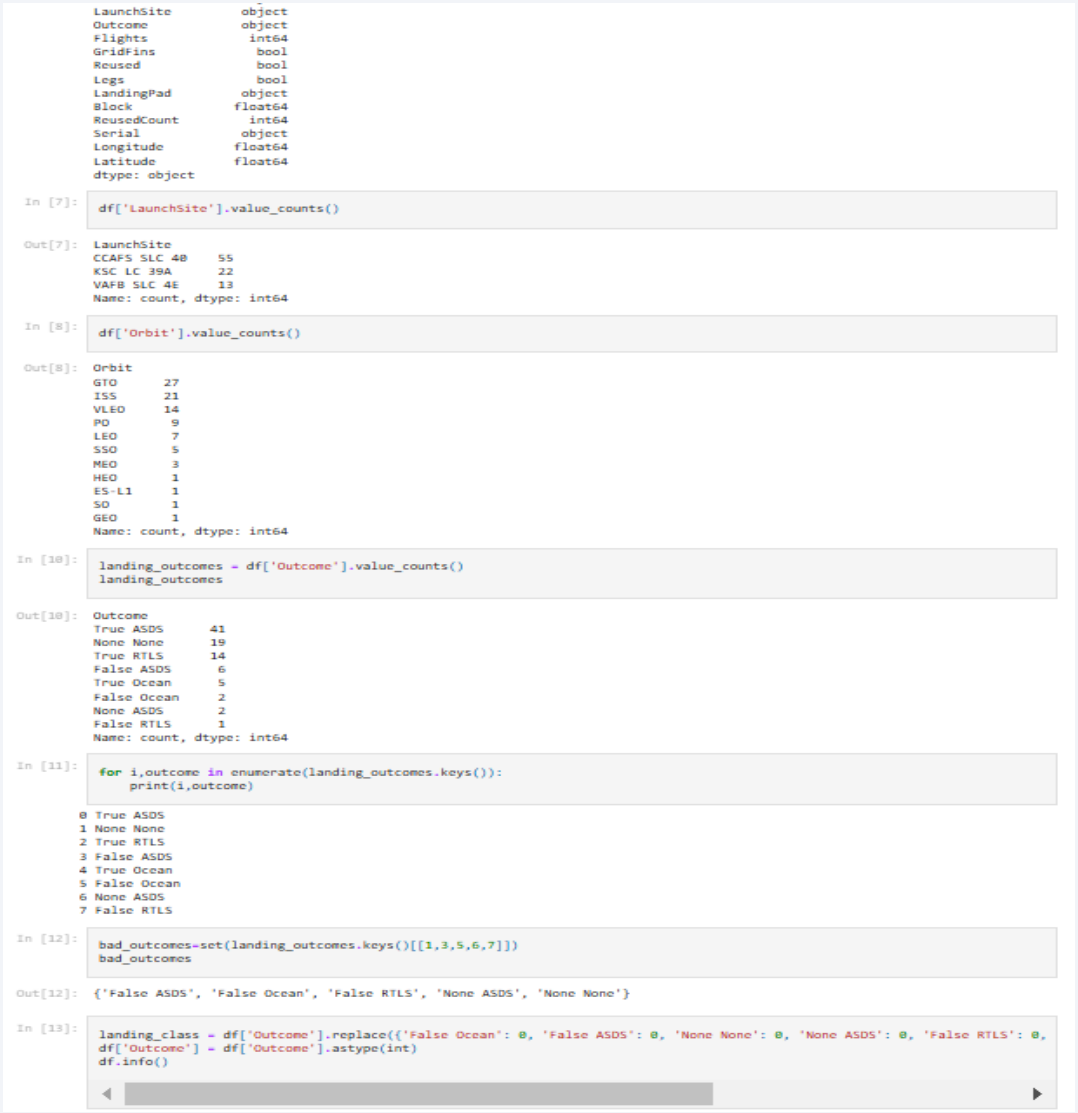
```
            new_mass=mass[0:mass.find("kg")+2]
        else:
            new_mass=0
        return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
        return colunm_name
```

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

]: response = requests.get(static_url).text

]: soup = BeautifulSoup(response, 'html.parser')

]: print(soup.title)

<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>

]: html_tables = soup.find_all("table")
   print(html_tables)

[<table class="col-begin" role="presentation">
<tbody><tr>
<td class="col-break">
<div class="mw-heading mw-heading3"><h3 id="Rocket_configurations">Rocket configurations</h3></div>
<div class="chart noresize" style="padding-top:10px;margin-top:1em;max-width:420px;">
<div style="position:relative;min-height:320px;min-width:420px;max-width:420px;">
<div style="float:right;position:relative;min-height:240px;min-width:320px;max-width:320px;border-left:1px black solid;borde
-bottom:1px black solid;">
<div style="position:absolute;left:3px;top:224px;height:15px;min-width:18px;max-width:18px;background-color:LightSteelBlue;-
```

Data Wrangling

- After obtaining and creating a Pandas DF from the collected data, data was filtered using the ***BoosterVersion*** column to only keep the Falcon 9 launches, then dealt with the missing data values in the ***LandingPad*** and ***PayloadMass*** columns. For the ***PayloadMass*** , missing data values were replaced using mean value of column.
- Also performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models

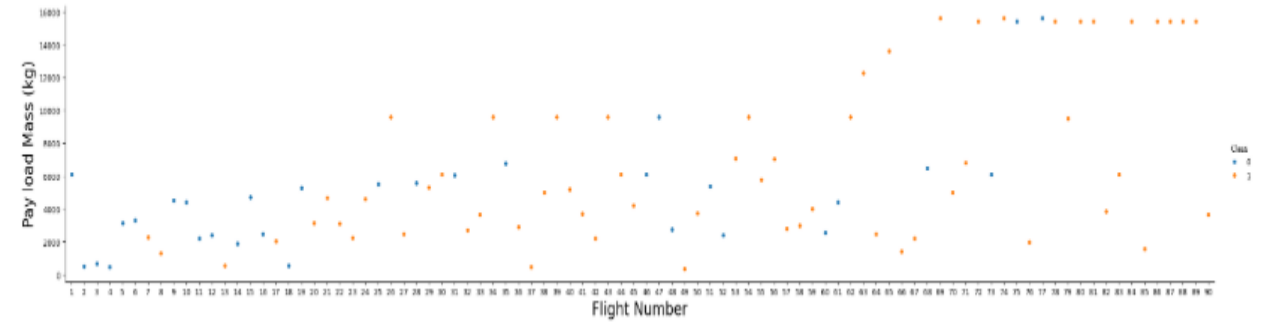
https://github.com/VarshiniBR/AIHackathon/blob/main/Data%20Wrangling.ipynb

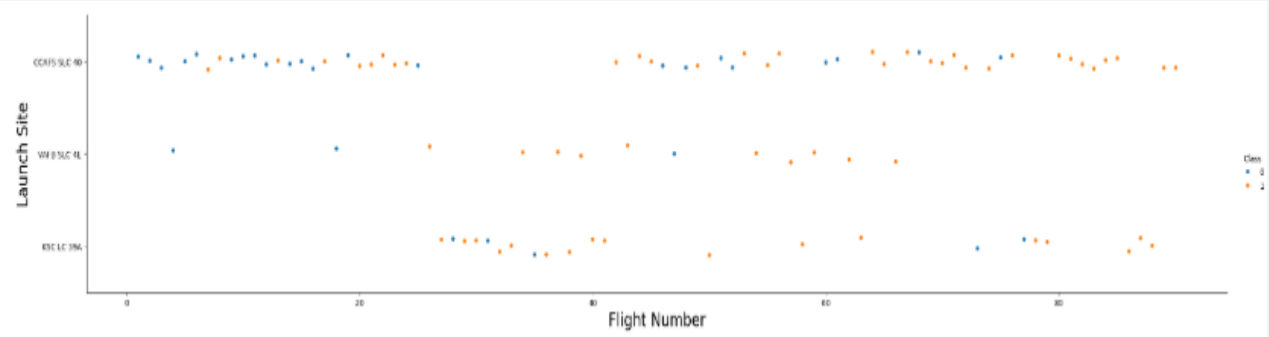


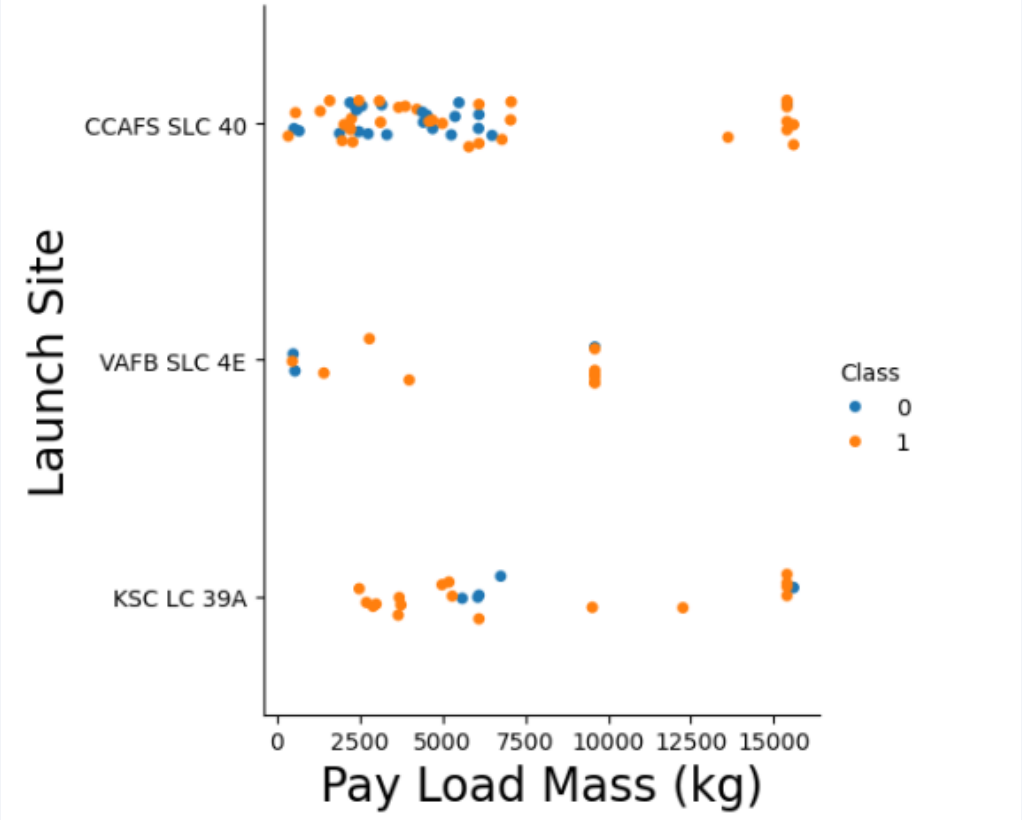
EDA with Data Visualization

- Performed data Analysis and Feature Engineering using Pandas and Matplotlib.i.e.
  - Exploratory Data Analysis
  - Preparing Data Feature Engineering
- Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, FlightNumber and Orbit type, Payload and Orbit type.
- Used Bar chart to Visualize the relationship between success rate of each orbit
- type
- Line plot to Visualize the launch success yearly trend.
- https://github.com/VarshiniBR/AIHackathon/blob/main/EDA%20with%20Visualization%20Lab.ipynb

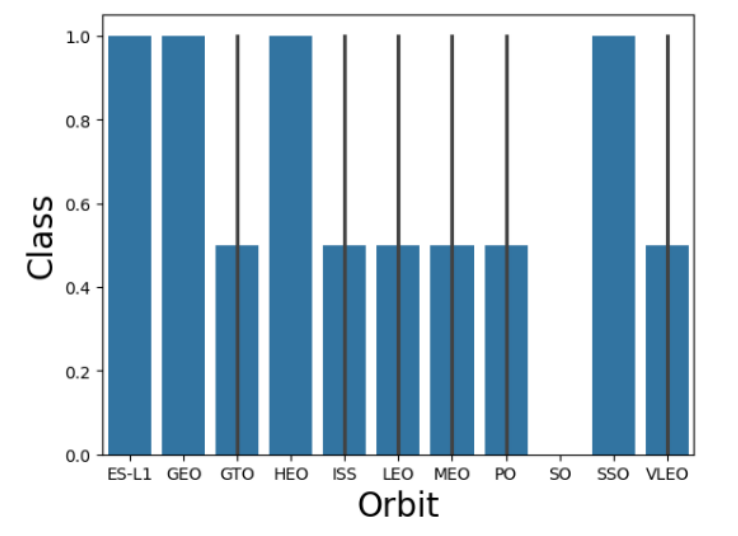
EDA with Data Visualization(Plots)

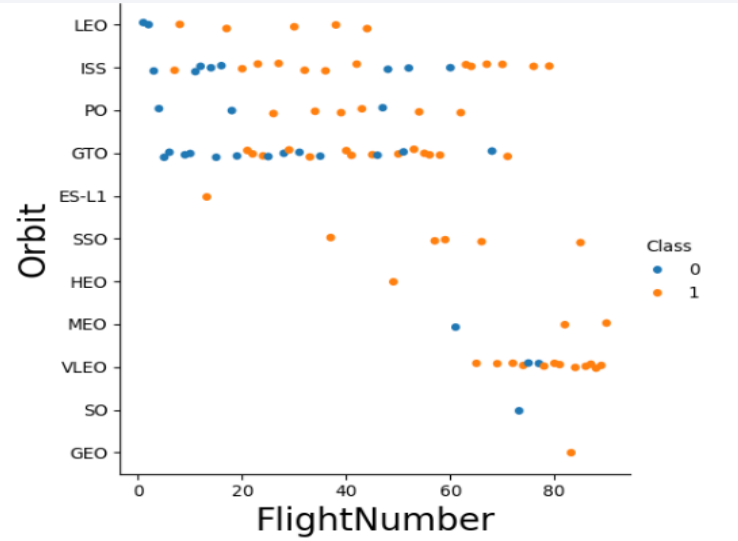


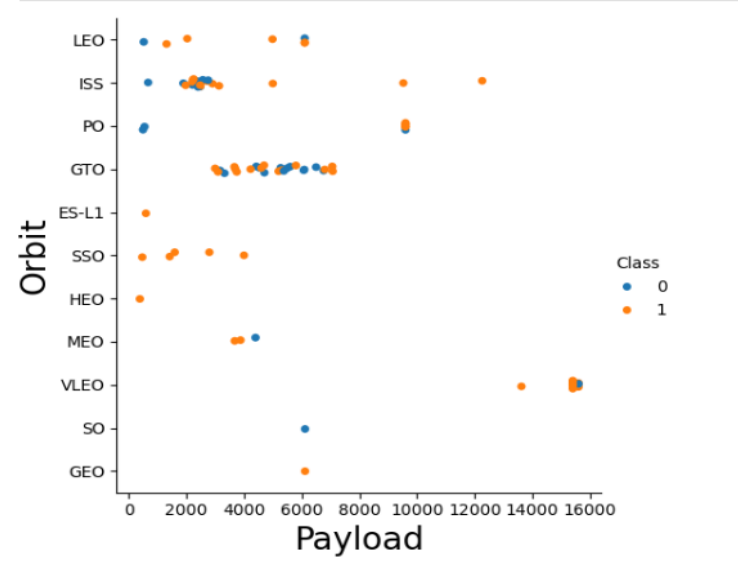


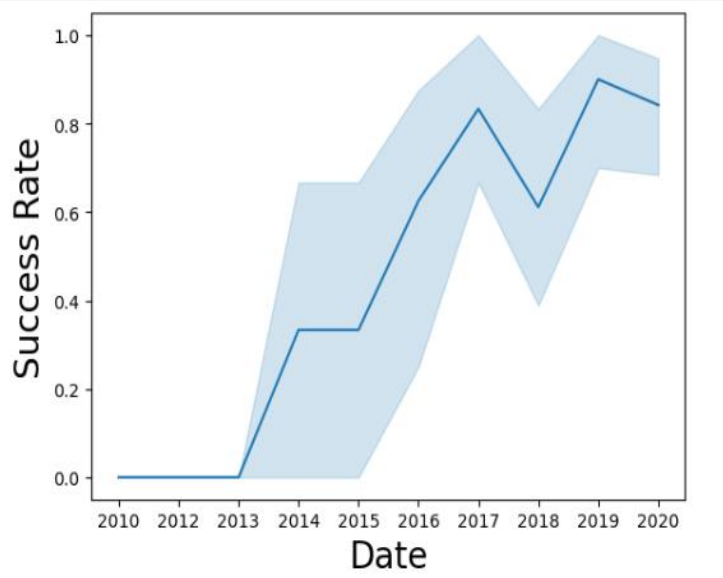


EDA with Data Visualization(Plots)









EDA with SQL

- The following SQL queries were performed for EDA
  - 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 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
  - https://github.com/VarshiniBR/AIHackathon/blob/main/Complete%20the%20EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Created folium map to marked all the launch sites, and created map objects such as markers, circles, lines to mark the success or failure of launches for each launch site.
- Created a launch set outcomes (failure=0 or success=1).
- https://github.com/VarshiniBR/AIHackathon/blob/main/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard application with Plotly dash by:
  - Adding a Launch Site Drop-down Input Component
  - Adding a callback function to render success-pie-chart based on selected site dropdown
  - Adding a Range Slider to Select Payload
  - Adding a callback function to render the success-payload-scatter-chart scatter plot

https://github.com/VarshiniBR/AIHackathon/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Summary of how I built, evaluated, improved, and found the best performing classification model
- After loading the data as a Pandas Dataframe, I set out to perform exploratory Data Analysis and determine Training Labels by;
  - creating a NumPy array from the column Class in data, by applying the method to_numpy() then assigned it to the variable Y as the outcome variable.
  - Then standardized the feature dataset (x) by transforming it using preprocessing.StandardScaler() function from Sklearn.
  - After which the data was split into training and testing sets using the function train_test_split from sklearn.model_selection with the test_size parameter set to 0.2 and random_state to 2.

Predictive Analysis (Classification)

- In order to find the best ML model/ method that would performs best using the test data between SVM, Classification Trees, k nearest neighbors and Logistic Regression;
  - First created an object for each of the algorithms then created a GridSearchCV object and assigned them a set of parameters for each model.
  - For each of the models under evaluation, the GridsearchCV object was created with cv=10, then fit the training data into the GridSearch object for each to Find best Hyperparameter.
  - After fitting the training set, we output GridSearchCV object for each of the models, then displayed the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.
  - Finally using the method score to calculate the accuracy on the test data for each
  - model and plotted a confussion matrix for each using the test and predicted outcomes.
  - https://github.com/VarshiniBR/AIHackathon/blob/main/Complete%20the%20Machine%20Learning%20Prediction%20lab.ipynb

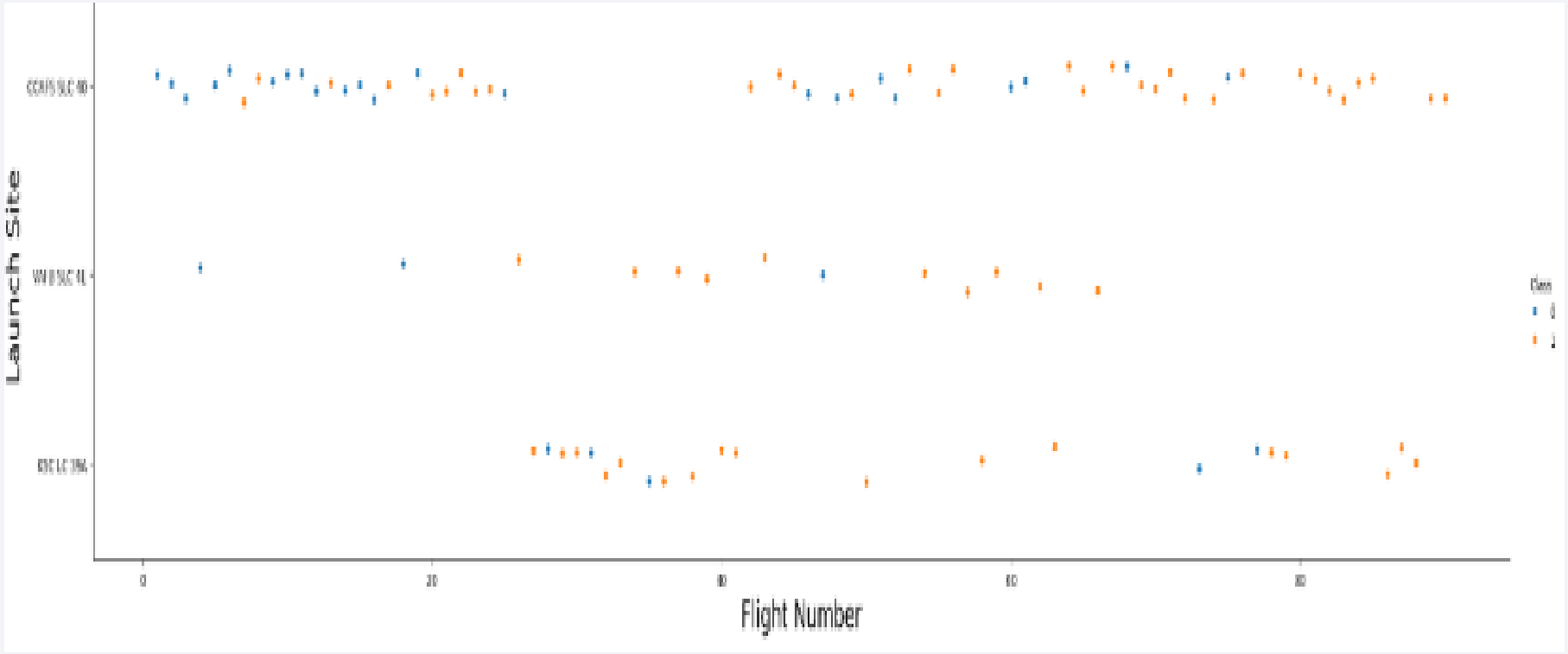
Results

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

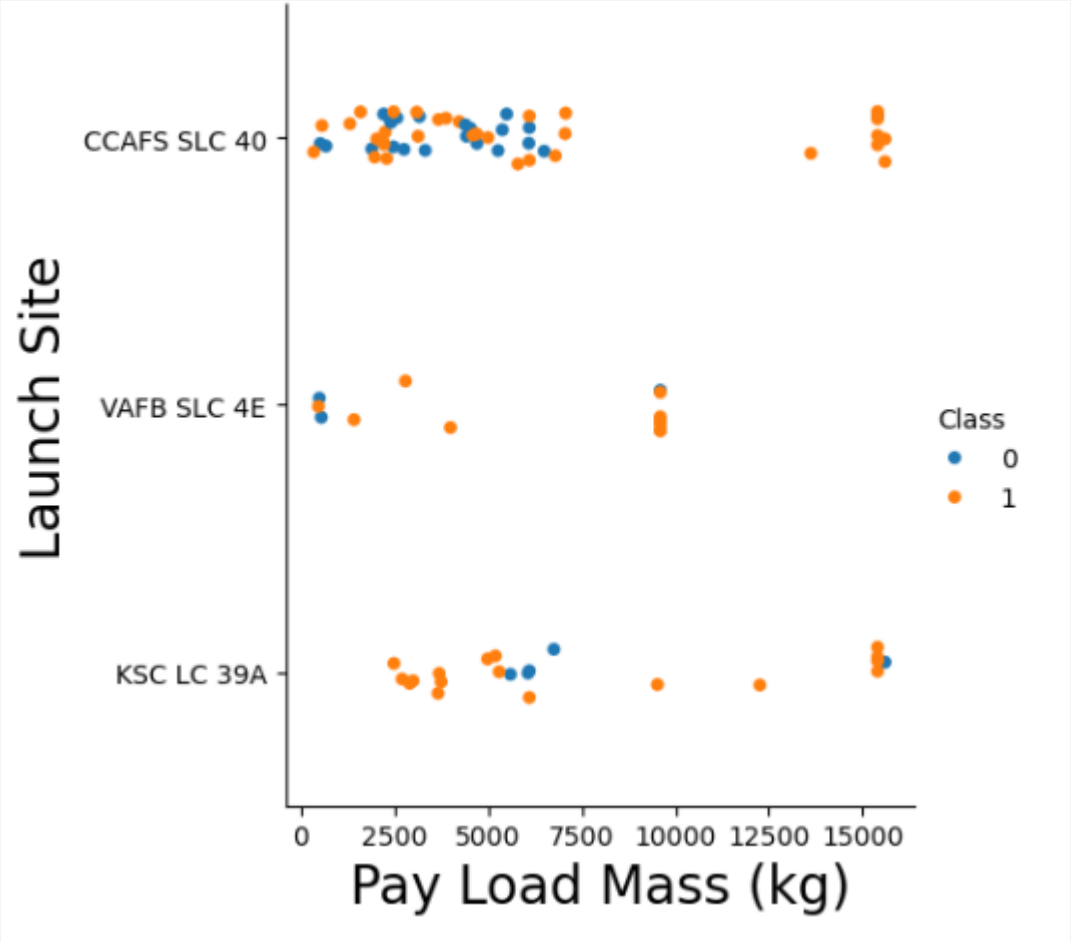
Section 2

Insights drawn from EDA

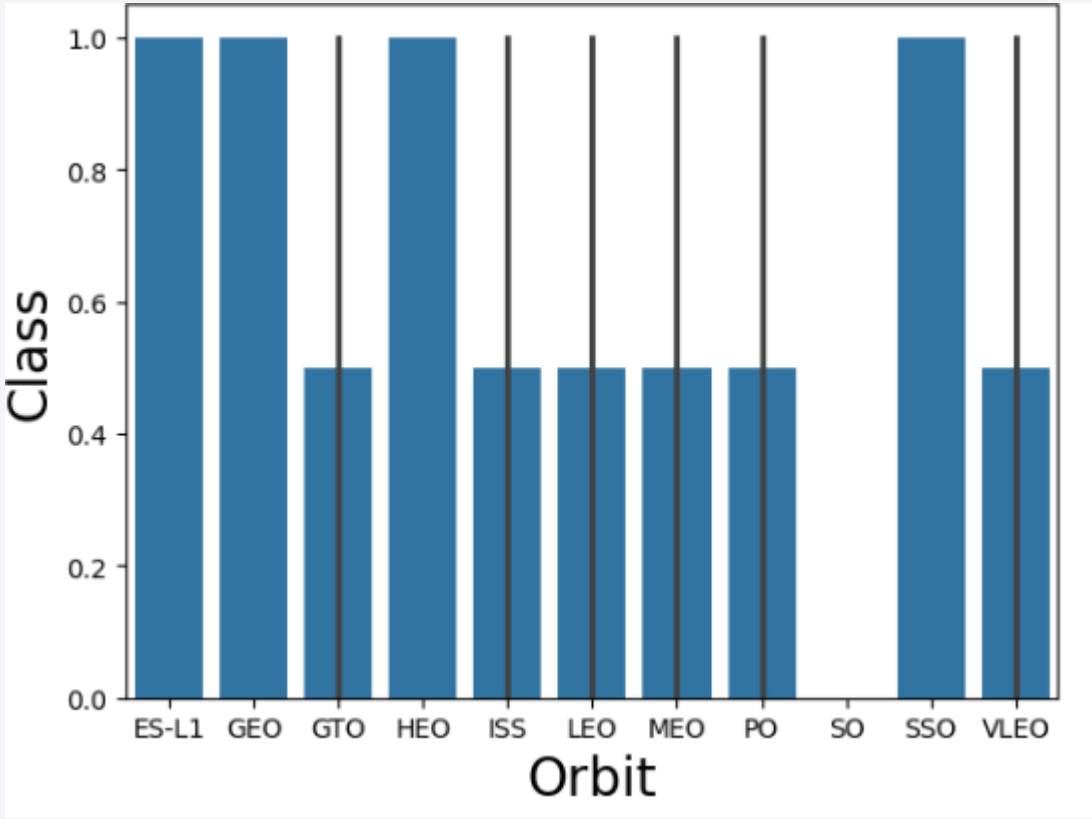
Flight Number vs. Launch Site



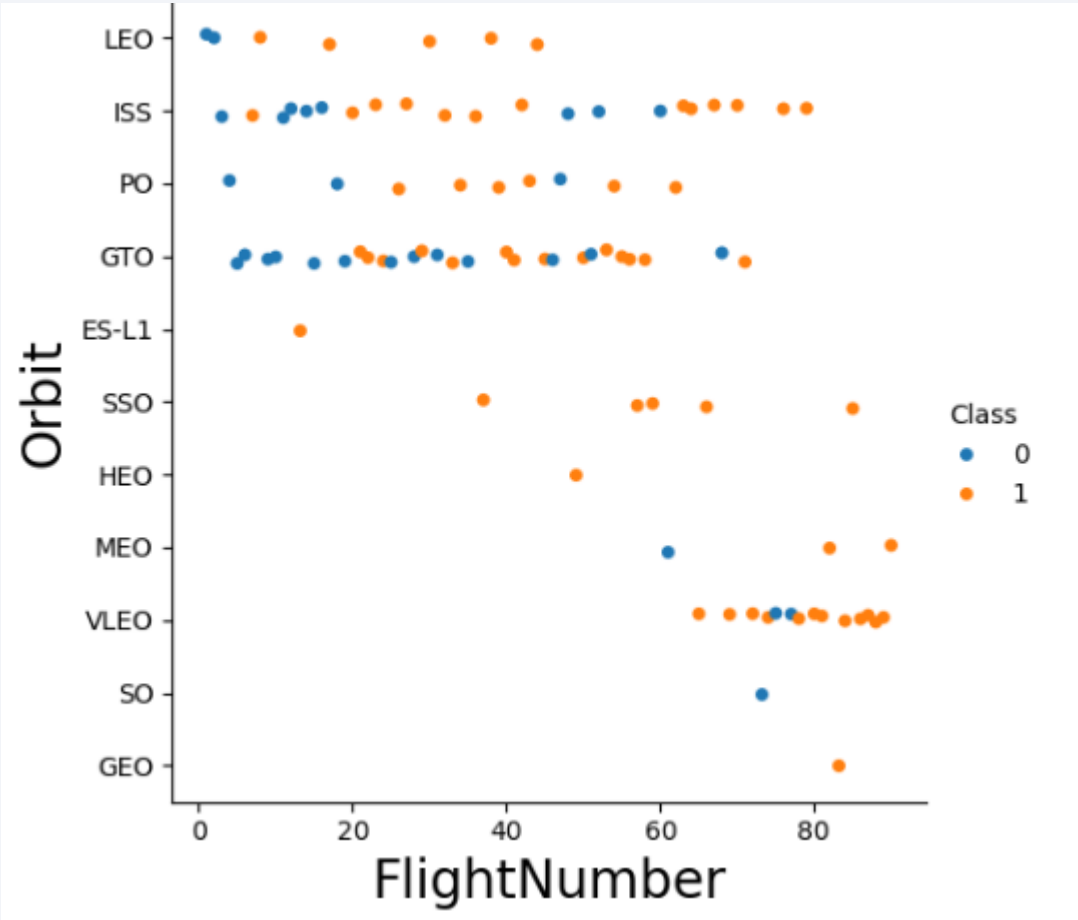
Payload vs. Launch Site



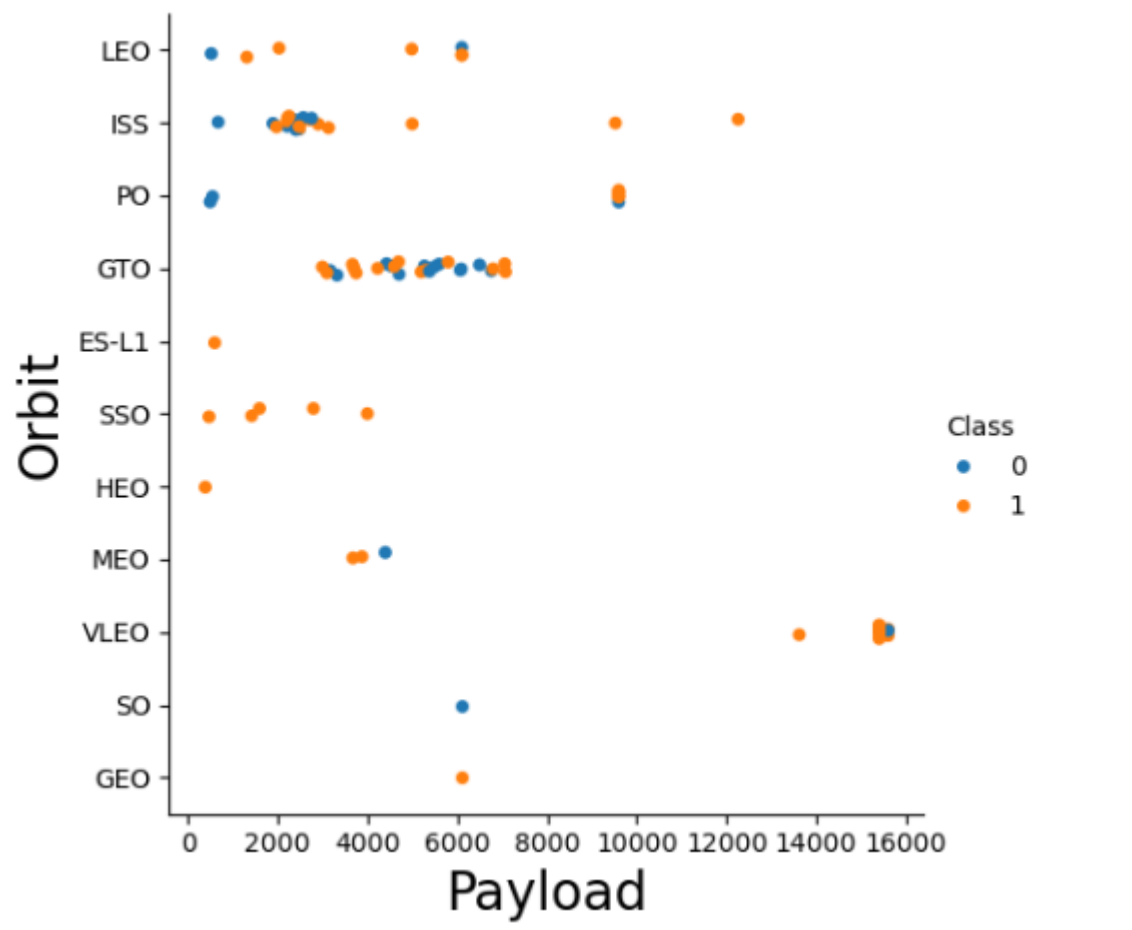
Success Rate vs. Orbit Type



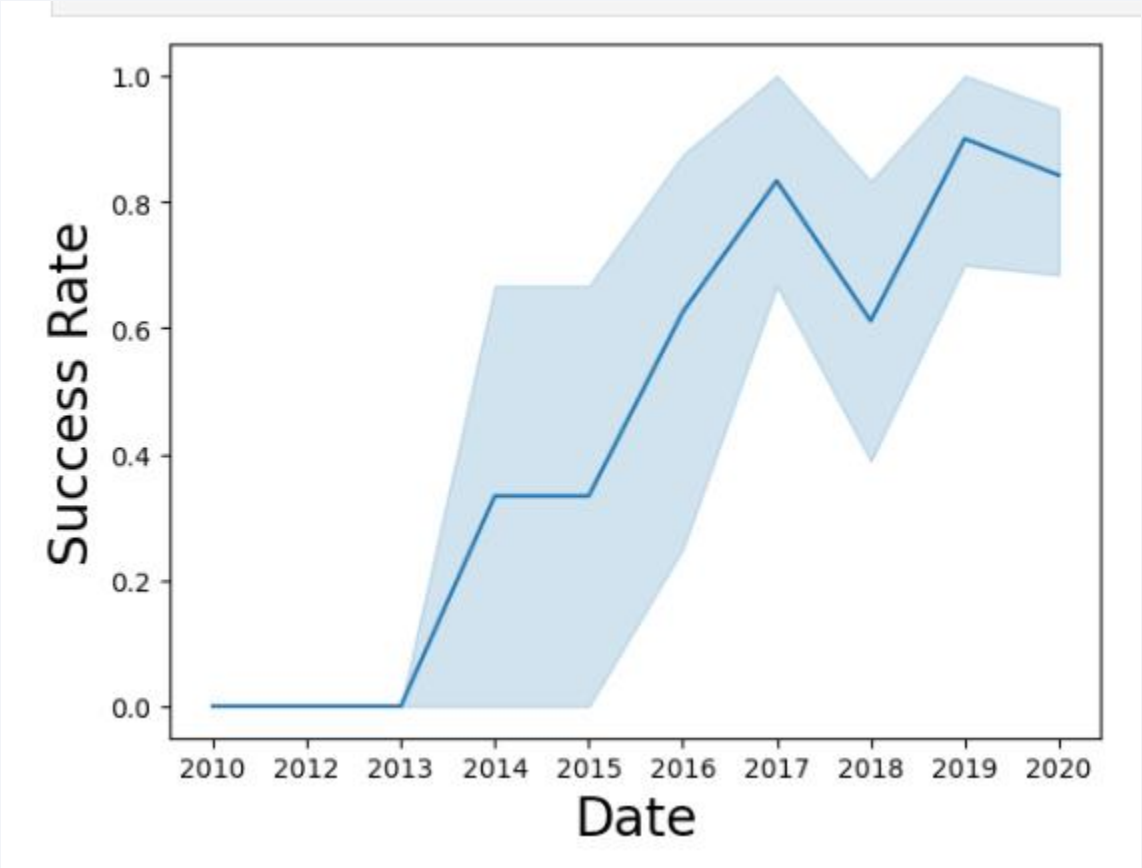
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

- Used ‘SELECT DISTINCT’ statement to return only the unique launch sites from the ‘LAUNCH_SITE’ column of the SPACEXTBL table

In [11]:
```
%sql SELECT Distinct LAUNCH_SITE FROM SPACEXTBL
```

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

Out[11]:

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

Launch Site Names Begin with 'CCA'

- Used ‘LIKE’ command with ‘%’ wildcard in ‘WHERE’ clause to select and display a table of all records where launch sites begin with the string 'CCA'

```
In [12]: %sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
 * sqlite:///my_data1.db
Done.
```

Out[12]:

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

Total Payload Mass

- Used the ‘SUM()’ function to return and display the total sum of ‘PAYLOAD_MASS_KG’ column for Customer ‘NASA(CRS)’

```
In [13]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
 * sqlite:///my_data1.db
Done.
Out[13]: SUM(PAYLOAD_MASS__KG_)
                          45596
```

Average Payload Mass by F9 v1.1

- Used the ‘AVG()’ function to return and dispaly the average payload mass carried by booster version F9 v1.1

```
In [14]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION='F9 v1.1'
 * sqlite:///my_data1.db
Done.
Out[14]: AVG(PAYLOAD_MASS__KG_)
         2928.4
```

First Successful Ground Landing Date

- Used the ‘MIN()’ function to return and dispaly the first (oldest) date when first successful landing outcome on ground pad ‘Success (ground pad)’happened.

```
In [15]: %sql SELECT min(DATE) FROM SPACEXTBL WHERE Landing_Outcome='Success (ground pad)'
 * sqlite:///my_data1.db
Done.
Out[15]: min(DATE)
         2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- Used ‘Select Distinct’ statement to return and list the ‘unique’ names of boosters with
- operators >4000 and <6000 to only list booster with payloads btween 4000-6000 with
- landing outcome of ‘Success (drone ship)’.

```
In [16]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ between 4000 and 6000 AND Landing_Outcome='Success (dron
 * sqlite:///my_data1.db
Done.
```

Out[16]:

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

Total Number of Successful and Failure Mission Outcomes

- Used the ‘COUNT()’ together with the ‘GROUP BY’ statement to return total number of missions outcomes

```
In [17]: %sql SELECT COUNT(*) FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Success%' OR MISSION_OUTCOME LIKE '%Failure%'
 * sqlite:///my_data1.db
Done.
Out[17]: COUNT(*)
         101
```

Boosters Carried Maximum Payload

- Using a Subquery to return and pass the Max payload and used it list all the boosters that have carried the Max payload of 15600kgs

In [18]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Out[18]:

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

2015 Launch Records

- Used the ‘substr()’ in the select statement to get the month and year from the date column where substr(Date,7,4)='2015' for year and Landing_outcome was ‘Failure (drone ship)’ and return the records matching the filter.

```
In [68]: %sql SELECT CASE SUBSTR("Date", 6, 2)WHEN '01' THEN 'January'WHEN '02' THEN 'February'WHEN '03' THEN 'March'WHEN '04' THEN
```

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

Out[68]:

month	"Landing_Outcome"='Failure (drone ship)'	Booster_Version	Launch_Site
January	1	F9 v1.1 B1012	CCAFS LC-40
April	1	F9 v1.1 B1015	CCAFS LC-40

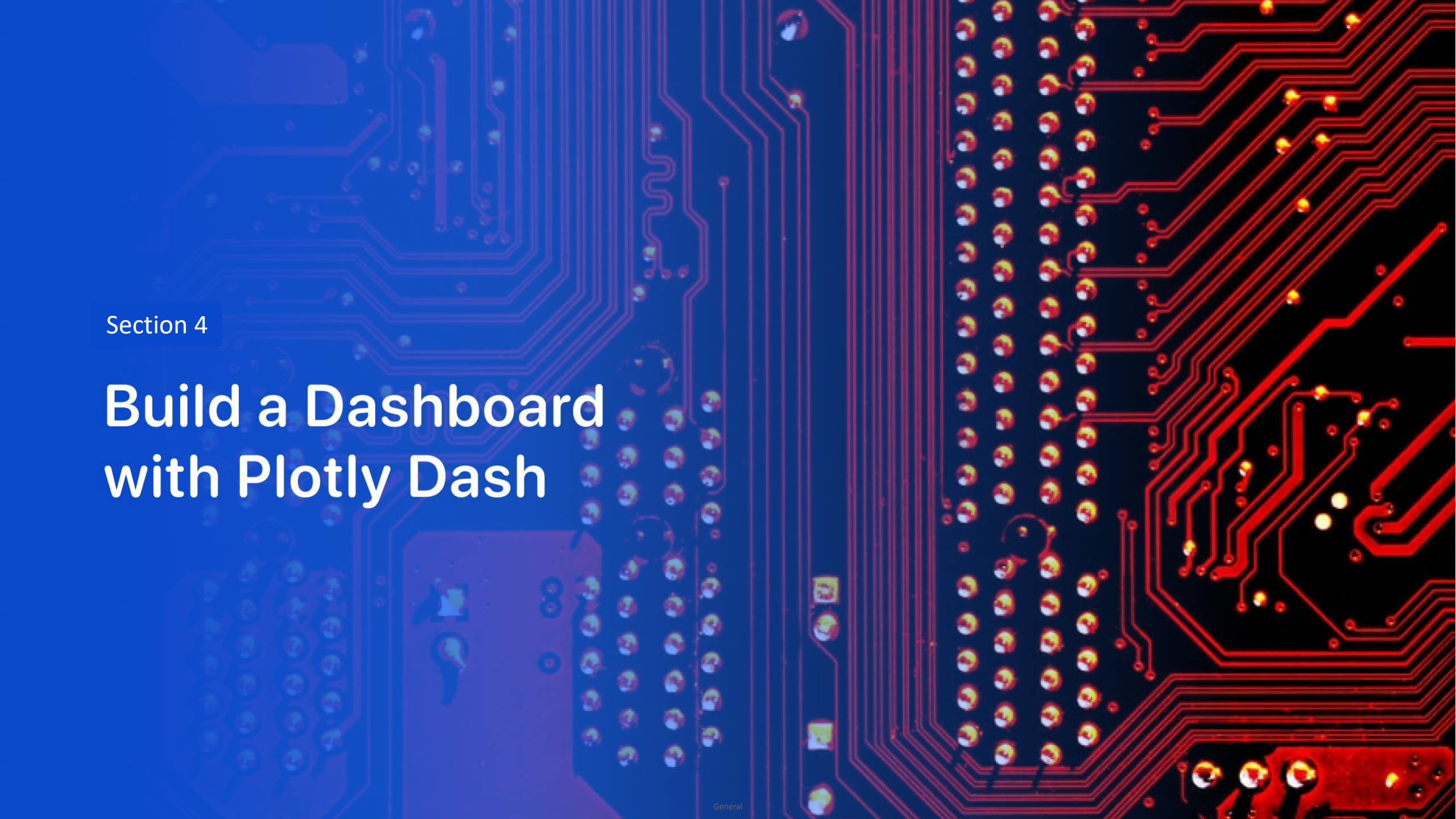
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- 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

```
In [69]: %sql select "Landing_Outcome",count("Landing_Outcome") from SPACEXTABLE where "Date" between 20100604 and 20170320 group by
 * sqlite:///my_data1.db
Done.
```

Out[69]:

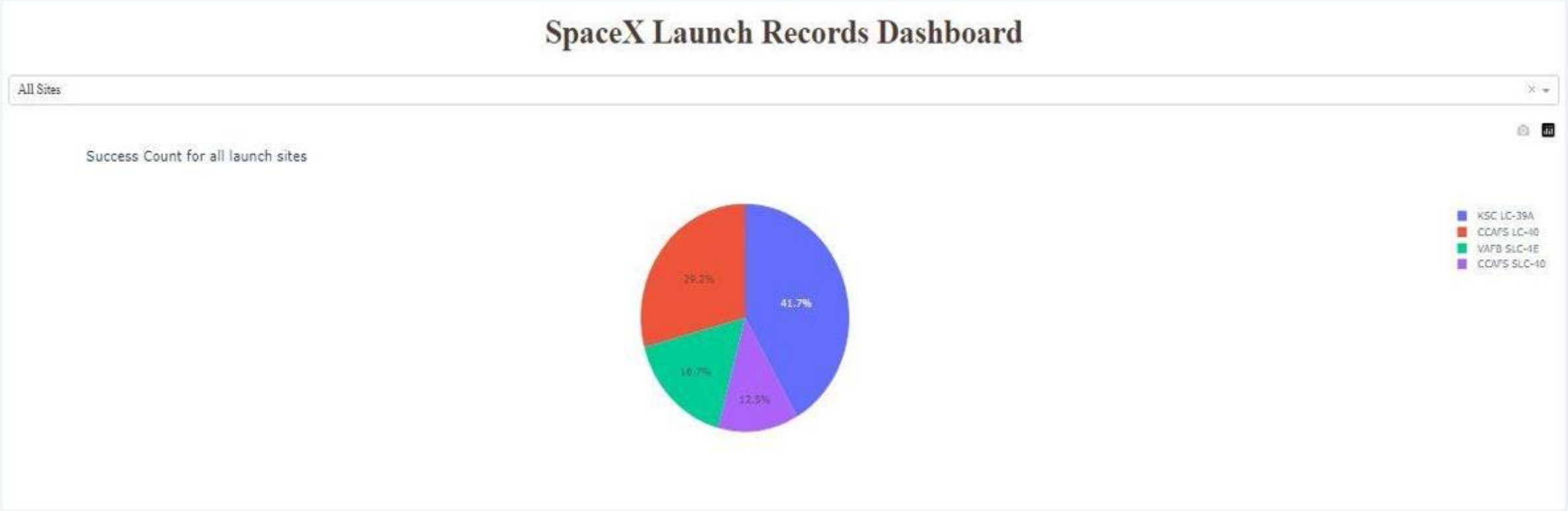
Landing_Outcome	count("Landing_Outcome")
Success (drone ship)	12
No attempt	12
Success (ground pad)	8
Failure (drone ship)	5
Controlled (ocean)	4
Uncontrolled (ocean)	2
Precluded (drone ship)	1



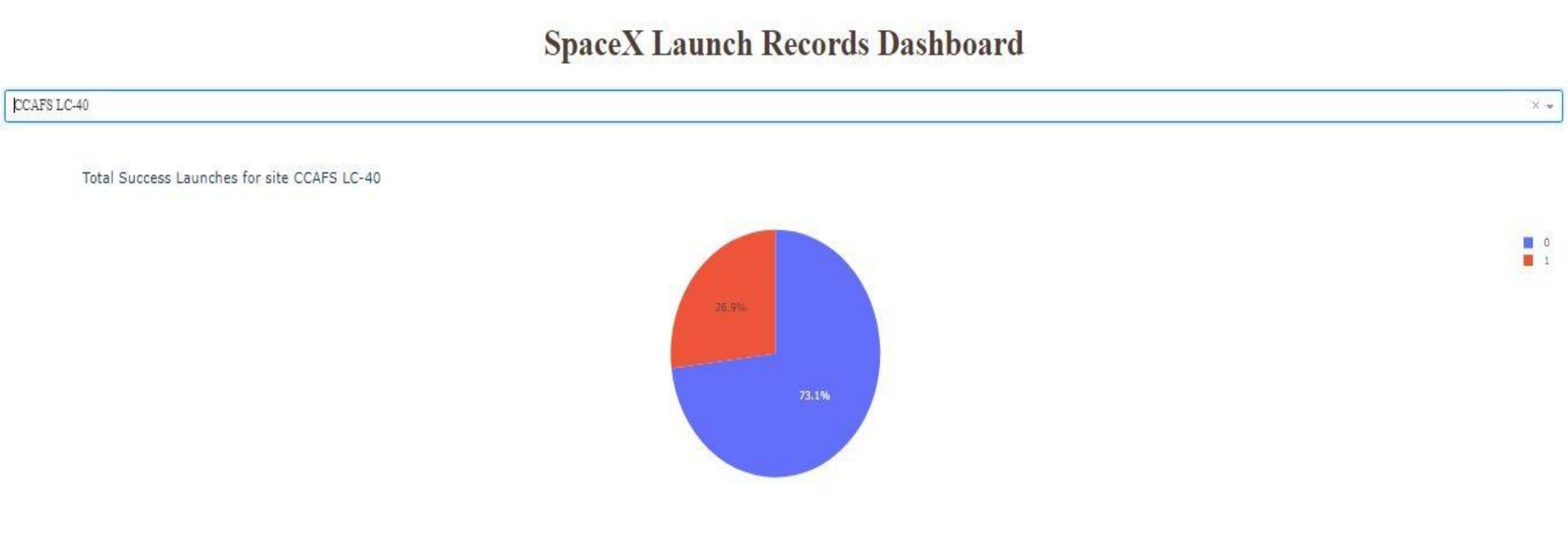
Section 4

Build a Dashboard with Plotly Dash

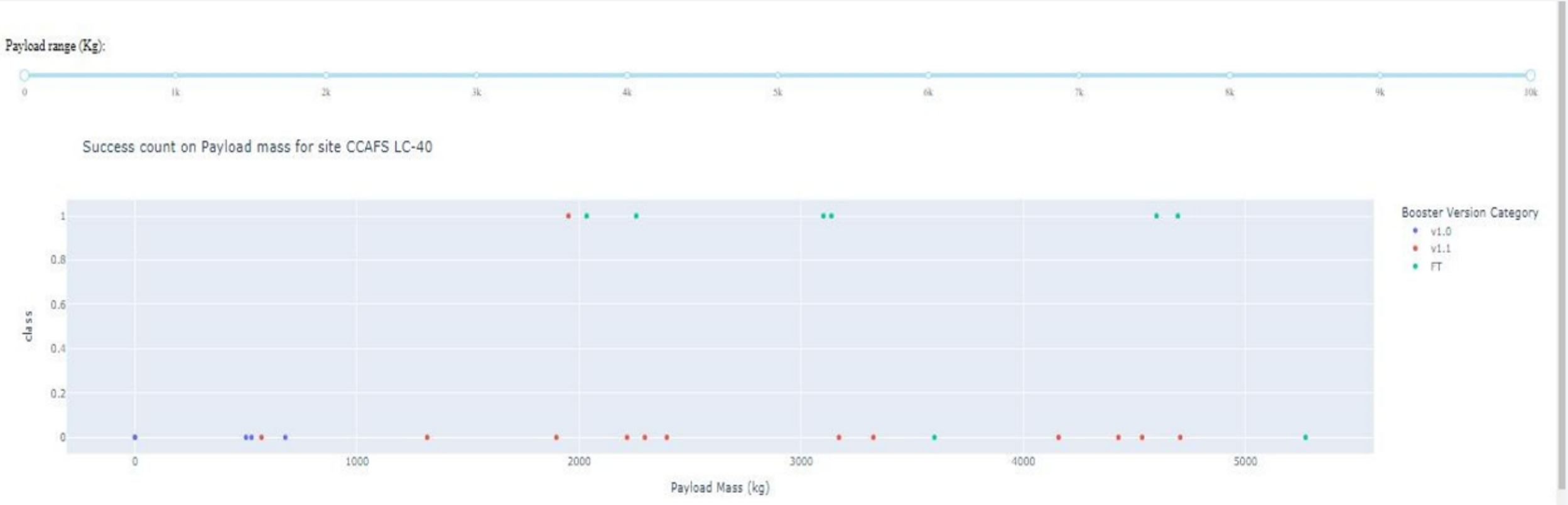
<Dashboard Screenshot 1>



<Dashboard Screenshot 2>



<Dashboard Screenshot 3>



Section 5

Predictive Analysis (Classification)

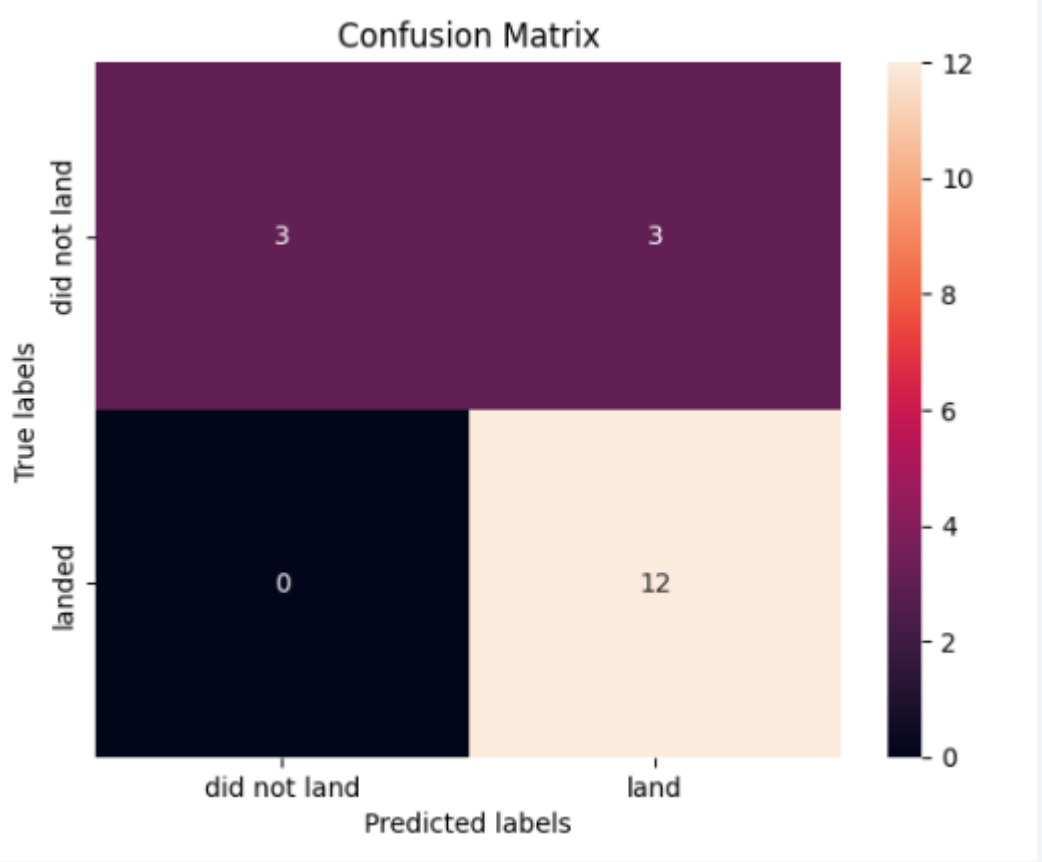
Classification Accuracy

Out[68]:

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

Confusion Matrix

- All the 4 classification model had the same confusion matrixes and were able equally distinguish between the different classes. The major problem is false positives for all the models.



Conclusions

- Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
- We can deduce that, as the flight number increases in each of the 3 launch sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80th flight
- If you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC Launch site there are no rockets launched for heavy payload mass(greater than 10000).
- Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.
- 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.

Conclusions

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS. However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here
- And finally the success rate since 2013 kept increasing till 2020.

Thank you!