IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ming Yan
Dec/02/2023

Outline

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

Executive Summary (1)

- **Summary of methodologies**

✓ **Data Collection** was performed using the **SpaceX REST API** and by **Web Scraping** the SpaceX Wikipedia page.

✓ **Data Wrangling** enabled the handling of missing values and formatting the data

✓ **Exploratory Data Analysis (EDA)** was performed with **SQL** queries and **Python** visualization libraries.

✓ **Data Visualization** was carried out with **Folium maps** and using **Plotly Dash** for comprehensive dashboarding.

✓ **Predictive Analysis** (Classification) was implemented with the Python **scikit-learn** package

Executive Summary (2)

- **Summary of results**

✓ **Data Collection** and **Data Wrangling** allowed for the obtainment of a robust dataset.

✓ **EDA** identified the **features** to be used for the predictive analysis stage.

✓ **Interactive Analytics Visualizations** are presented in screenshots.

✓ **Predictive Analysis** using **Machine Learning** techniques indicated the best **classification model** to predict successful launches amid a number of constructed and evaluated classifiers.

Introduction

➢ Project background and context

SpaceX is an American spacecraft manufacturer, making commercial space travel possible at relatively affordable prices compared to its competitors, boasting a cost approximately 37.58% that of competing market prices for their Falcon 9 rockets, at 62 million USD. That incredible price difference is due to the reuse of the SpaceX first launch stage. Determining whether the first stage will land successfully and is able to be reused can help us predict the launch cost for the company.

SpaceY is a new company exploring the viability of competing with SpaceX for a share of the space tourism market.

➢ Key Questions

1. What parameters affect the successful landing of the SpaceX first stage?
2. How do feature interactions affect the success or failure of a landing?
3. Does the rate of successful landings increase over time? Why?
4. What is an appropriate binary classification model to predict whether a first stage retrieval will be successful or not?

Section 1

Methodology

Methodology

- **Data collection methodology**
  - ✓ SpaceX REST API (https://api.spacexdata.com/v4/rockets/)
  - ✓ Web Scrapping from SpaceX Wikipedia page (https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches)
- **Performed data wrangling**
  - ✓ Data Filtering (dropping unnecessary features)
  - ✓ Missing values handling
  - ✓ Data preparation using One Hot Encoding for categorical features
- **Performed exploratory data analysis (EDA) using SQL and visualization**
- **Performed interactive visual analytics using Folium and Plotly Dash**
- **Perform predictive analysis using classification models**
  - ✓ Building, tuning, evaluating classification models (Logistic Regression, Support Vector Machines, Decision Tree Classifier, K Nearest Neighbours Classifier)
  - ✓ Selecting the best performing classification model to predict success or failure of first stage landings

Data Collection

We used **two data collection** methods to ensure as much information as possible was collected, namely:

- ✓ SpaceX REST API (https://api.spacexdata.com/v4/rockets/)
- ✓ Web Scrapping from SpaceX Wikipedia page (https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches)

Flow charts detailing the data collection processes are given in the next two slides

• The data obtained from the SpaceX REST API came in the form of columns, whose names are given in the following list:
FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude

• The data obtained by Web Scrapping the SpaceX Wikipedia page came in the form of columns, whose names are given in the following list:
Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time

Data Collection – SpaceX API

GITHUB LINK 1: Data Collection API

Request response from API

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

Convert response to .json file and parse into pandas data frame

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

Request Extra Data using given functions

Transform Data, Create new data frame from response lists dictionary

```
[21]: launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
[23]: # Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
df.head()
```

Filter new data frame

```
[31]: # Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

Export dataset to .csv file

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection – Web Scraping

GITHUB LINK 2: Data Collection Web Scraping

Request HTML response from Wikipedia

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

Create BeautifulSoup object from HTML response

```
[9]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, "html.parser")
```

Find tables

```
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Extract column names

```
[28]: column_names = []

# Apply find_all() function with `th` element on first_lau
# Iterate each th element and apply the provided extract_c
# Append the Non-empty column name (`if name is not None ar
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if(name != None and len(name) > 0):
        column_names.append(name)
```

Create dictionary

```
[30]: launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each valu
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']=[]
```

Parse HTML table data

```
[31]: extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikit
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corres
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
```

Create data frame from dictionary and export dataset to .csv file

```
[33]: launch_dict
df=pd.DataFrame(launch_dict)
df
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling (1)

Main Task:

- ✓ Create a label with landing outcomes where **1=SUCCESS** and **0=FAILURE**.
- ✓ Successful Outcomes: **True ASDS, True RTLS, True Ocean** -> **Class 1**
- ✓ Unsuccessful Outcomes: **False ASDS, False Ocean, False RTLS** -> **Class 0**
- ✓ Uknown Outcomes: **None None, None ASDS** -> **Class 0**

Tasks Performed:

- ✓ Initial **EDA** to get familiar with dataset
- ✓ **Calculations** of launches statistics per launch and orbit type
- ✓ Categorical **training label 'Class'** was created

GITHUB LINK 3: Data Wrangling

Data Wrangling (2)

Calculate number of launches for each site

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

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

Calculate number of launches per orbit type

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

[13]: 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
```

Mission Outcome per orbit type

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

[14]: 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
```

Create landing outcome 'Class' column

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

[16]: {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

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

```
[18]: df['Class']=landing_class
      df[['Class']].head(8)
```

Export dataset to .csv file

```
df.to_csv("dataset_part_2.csv", index=False)
```

GITHUB LINK 3: Data Wrangling

EDA with Data Visualization

GITHUB LINK 4: EDA with Data Visualization

Scatter Plots:

- What is it? A chart showing the relationship between the observations of two variables

✓ Flight Number VS Payload Mass

✓ Flight Number VS Launch Site

✓ Payload VS Launch Site

✓ Orbit VS Flight Number

✓ Payload VS Orbit Type

Bar Plot:

- What is it? A plot that presents categorical data with rectangular bars with lengths proportional to the values that they represent

✓ Success Rate VS Orbit Type

Line Plot:

- What is it? A type of chart which displays information as a series of data points called 'markers' connected by straight line segments

✓ Success Rate VS Year

EDA with SQL

GITHUB LINK 5: EDA with SQL

The following queries were performed using SQL:

1. Displaying the names of the unique launch sites in the space mission
2. Displaying 5 records where launch sites begin with the string 'CCA'
3. Displaying the total payload mass carried by boosters launched by NASA (CRS)
4. Displaying average payload mass carried by booster version F9 v1.1
5. Listing the date when the first successful landing outcome in ground pad was achieved
6. Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. Listing the total number of successful and failure mission outcomes
8. Listing the names of the booster versions which have carried the maximum payload mass
9. Listing the failed landing outcomes in drone ship, their booster versions and launch site names for the months in year 2015
10. Ranking 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

Build an Interactive Map with Folium

The Folium map object created is a map centered on NASA Johnson Space Center in Houston, Texas

- ✓ Red circle at NASA Johnson Space Center's coordinate with label showing its name
- ✓ Red circles at each launch site coordinates with label showing launch site name.
- ✓ Marker clusters for each site with markers showing successful and unsuccessful landings.
- ✓ Markers and corresponding lines showing distance between a launch site and key proximities (railway, highway, coastline, city)

The above elements serve the purpose of better exploring the relationship between launch sites and the success rate of launches occurring there.

GITHUB LINK 6: Interactive Folium Map

Build a Dashboard with Plotly Dash

The Dashboard created is comprised of 4 main elements:

1. **Dropdown Menu**: Select specific Launch Site or All Launch Sites
2. **Pie Chart**: Shows the Success and Failure Rates of the Launch Site selected. In the case of All Launch Sites being selected the pie chart shows the percentage of total successful launches per launch site.
3. **Range Slider**: Select Range for attribute Payload Mass
4. **Scatter Plot**: Shows the relationship between the Payload Mass feature and Success Rate for the selected Launch Site

GITHUB LINK 7: Interactive Dashboard

Predictive Analysis (Classification)

- **Data Preparation:**

1. Data Normalization
2. Training and Test Set Data Split

- **Model Preparation:**

1. Selection of Machine Learning Algorithms (Logistic Regression, Support Vector Machines, Decision Trees, K-Nearest Neighbours Classifiers)
2. Perform grid search for each algorithm and train grid search models with training set

[GITHUB LINK 8: Machine Learning Prediction]

- **Model evaluation and Comparison**

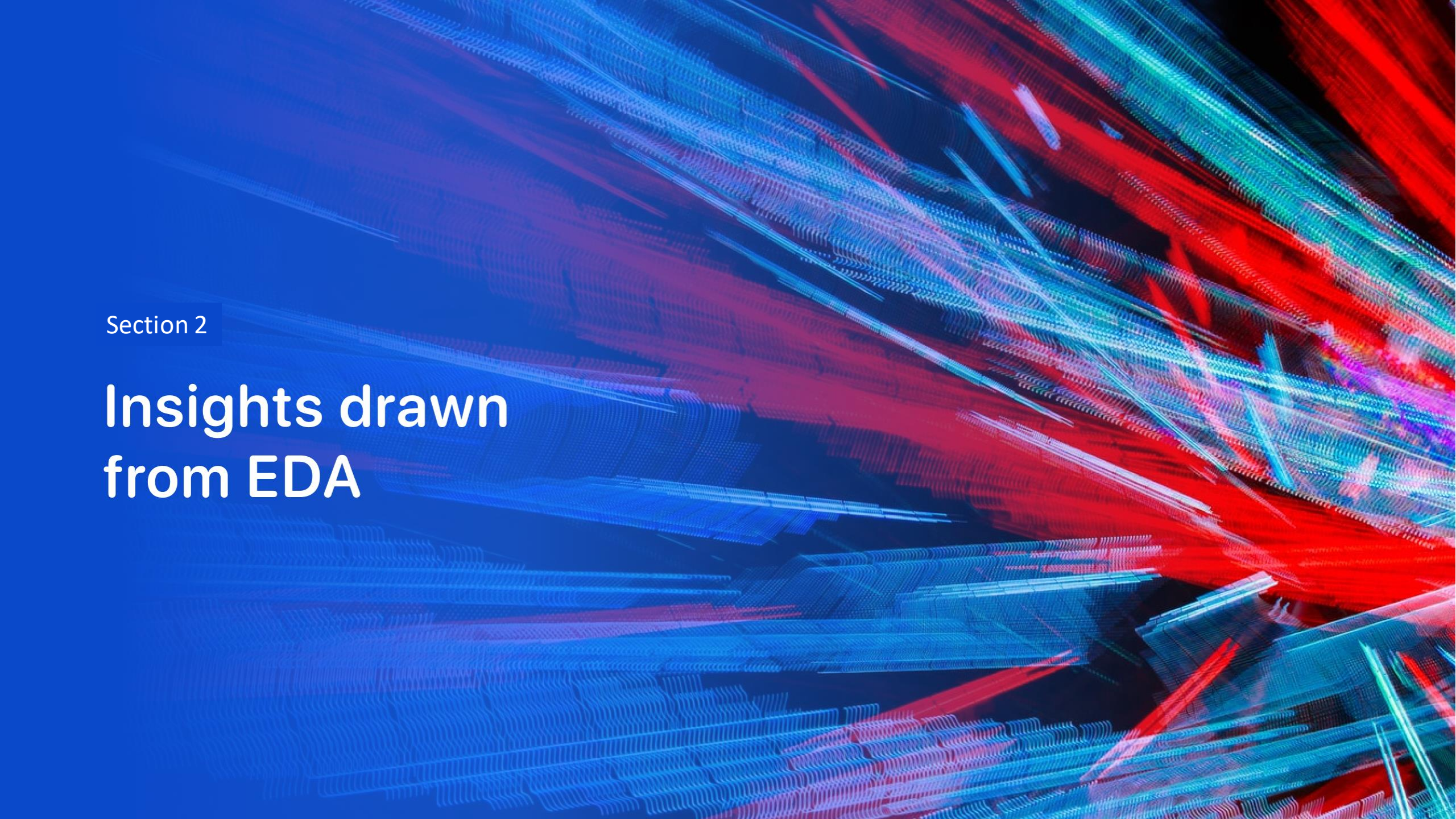
1. Get best hyperparameters for each type of model
2. Compute accuracy for each model with test dataset and plot Confusion Matrix
3. Comparison of models based on their accuracy

Results

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

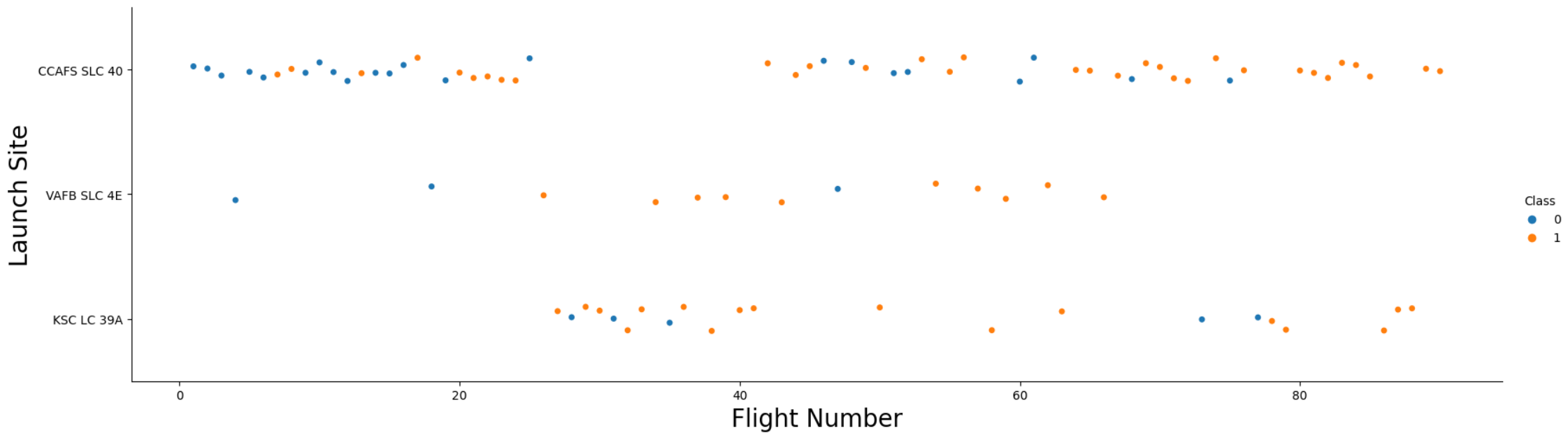
Section 2

Insights drawn from EDA



Exploratory Data Analysis with Data Visualization Results

Flight Number VS Launch Site

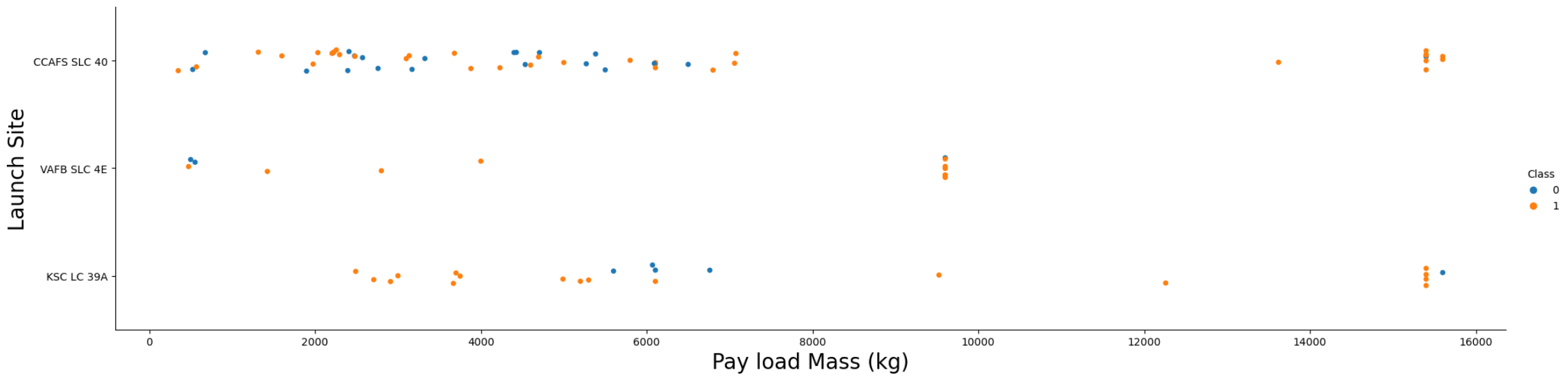


Using cat plot instead of scatter plot because we are working with categorical attributes.

Insights:

- The success rate for each launch site increases over time. Flight 20 seems to be a turning point
- Majority of launches have been carried out at the CCAFS SLC 40 launch site.
- VAFB SLC 4E and KSC LC 39A appear to have better success rates (77%) than CCAFS SLC 40 (60%) as they have undertaken more recent flights which were successful.

Payload VS Launch Site

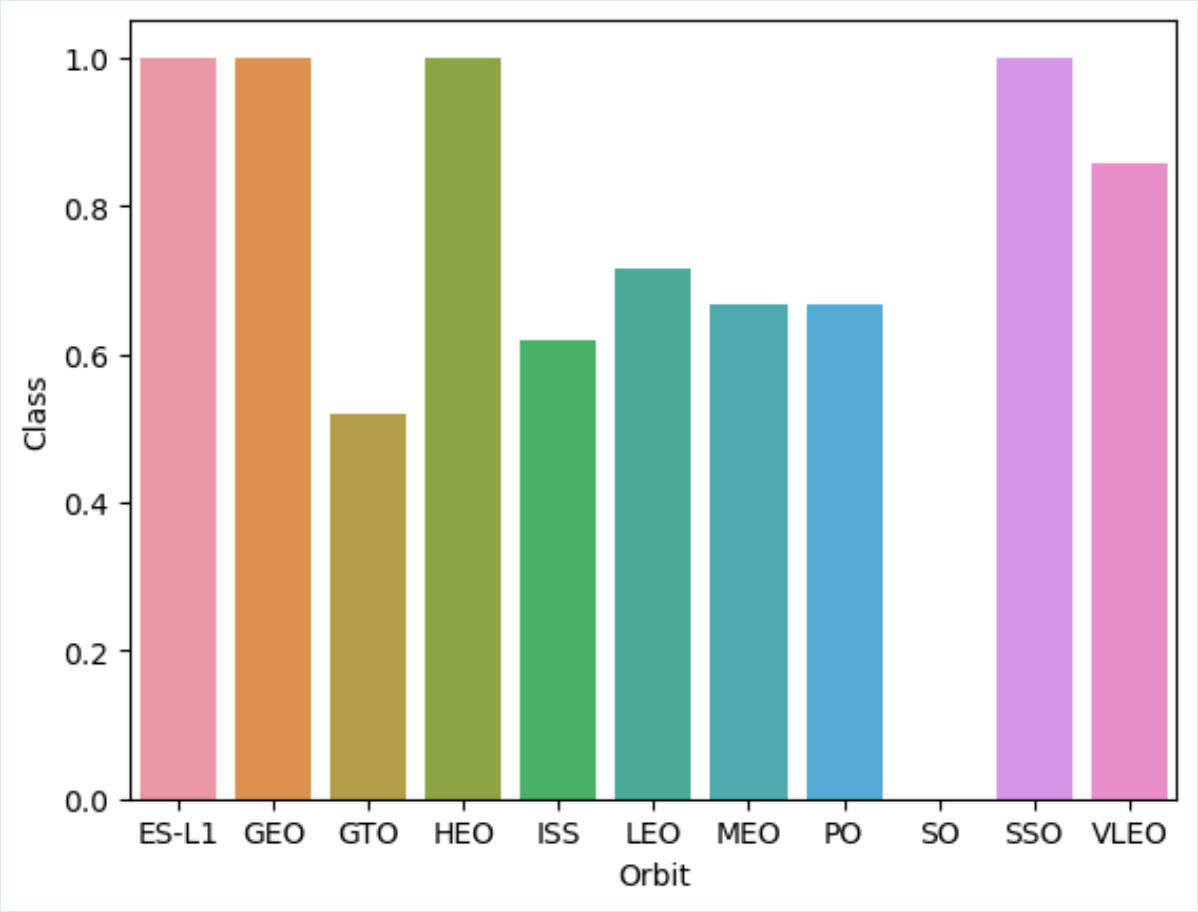


Using cat plot instead of scatter plot because we are working with categorical attributes.

Insights:

- Different launch sites appear to specialize in different payload mass categories. KSC LC 39A has 100% success rate for payloads under 5000kg
- A higher payload mass appears to positively affect the success of a mission, especially above the 9000kg point. While the success rate is satisfactory even at the 7000kg mark.
- However, payloads over 1200kg are only supported by CCFAS SLC 40 and KSCLC 39A at the time of this current report.

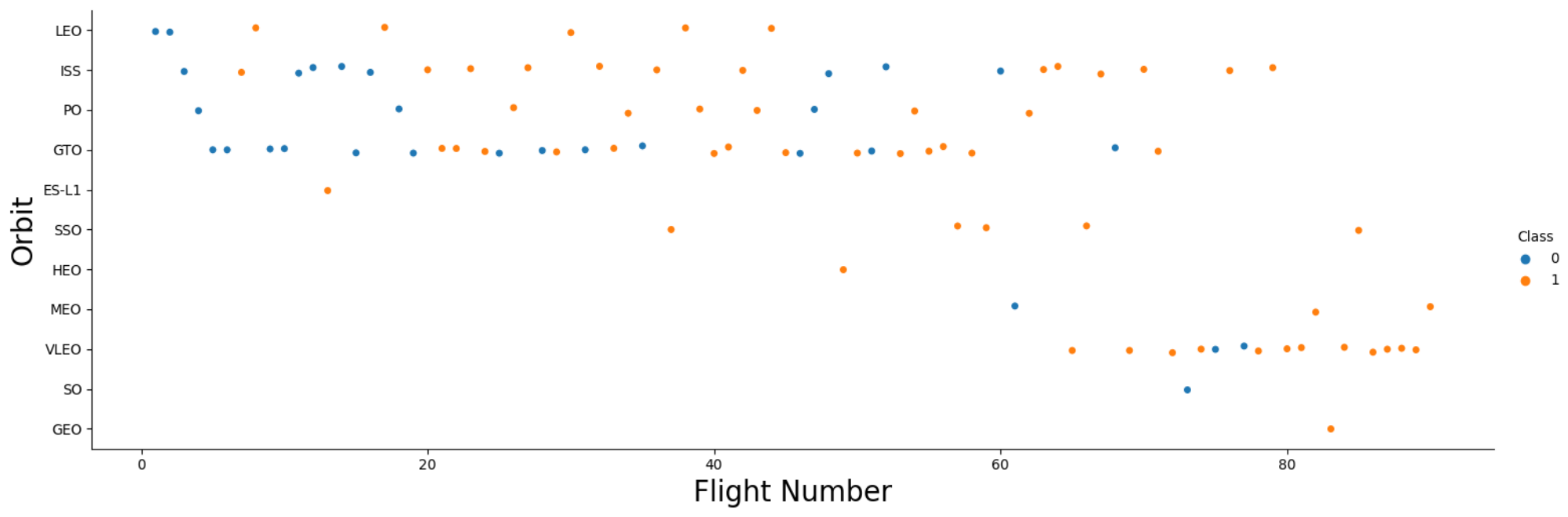
Success Rate VS Orbit Type



Insights:

- **100%** success rates for **ES-L1, GEO, HEO and SSO** orbit types. Sample sizes: 1, 1, 1 and 5 launches, respectively.
- ✓ **SSO orbit type has the best actual success rate**
- **0%** success rate for orbit type **SO** out of 1 launch.
- Out of 27 launches, **GTO** orbit type has a success rate of about **50%**
- Out of 21 launches, **ISS** orbit type has a success rate of about **60%**
- Success rates between **40%-85%**: **GTO, ISS, LEO, MEO, PO, VLEO**. Sample sizes: 27, 21, 7, 3, 9 and 14 launches, respectively.

Flight Number VS Orbit Type

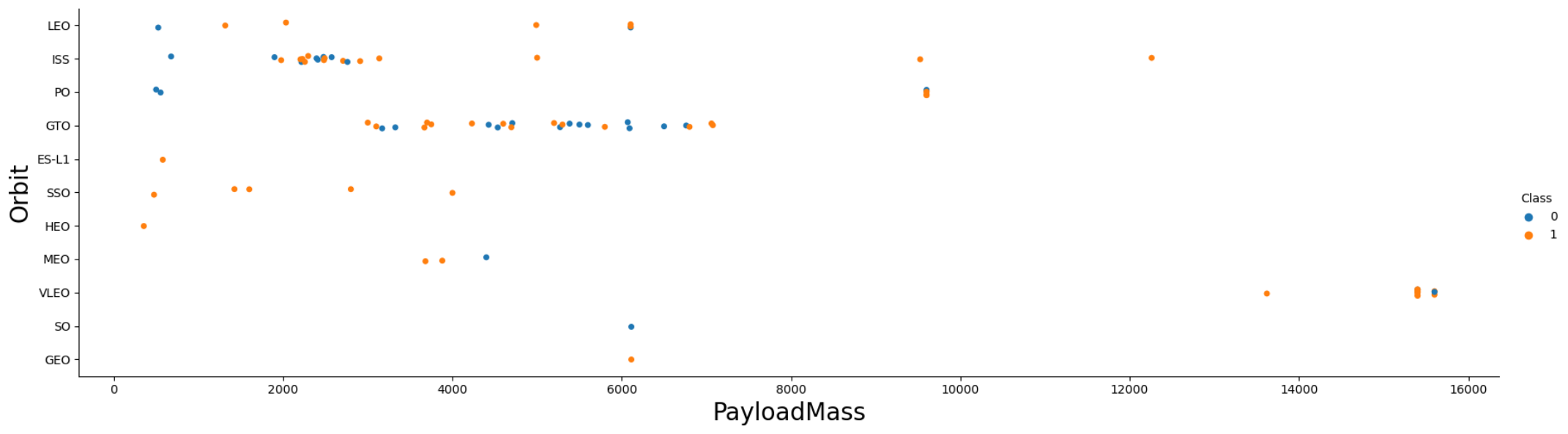


Using cat plot instead of scatter plot because we are working with categorical attributes.

Insights:

- Flight Number only seems to affect some orbit types.
- LEO appears to be affected by the number of flights
- GTO does not appear to be affected by the number of flights
- VLEO seems an interesting opportunity due to recent success of launches in that orbit
- SSO and HEO orbit types have 100% success rates probably because of previously acquired know-how on other orbit types

Payload VS Orbit Type



Using cat plot instead of scatter plot because we are working with categorical attributes.

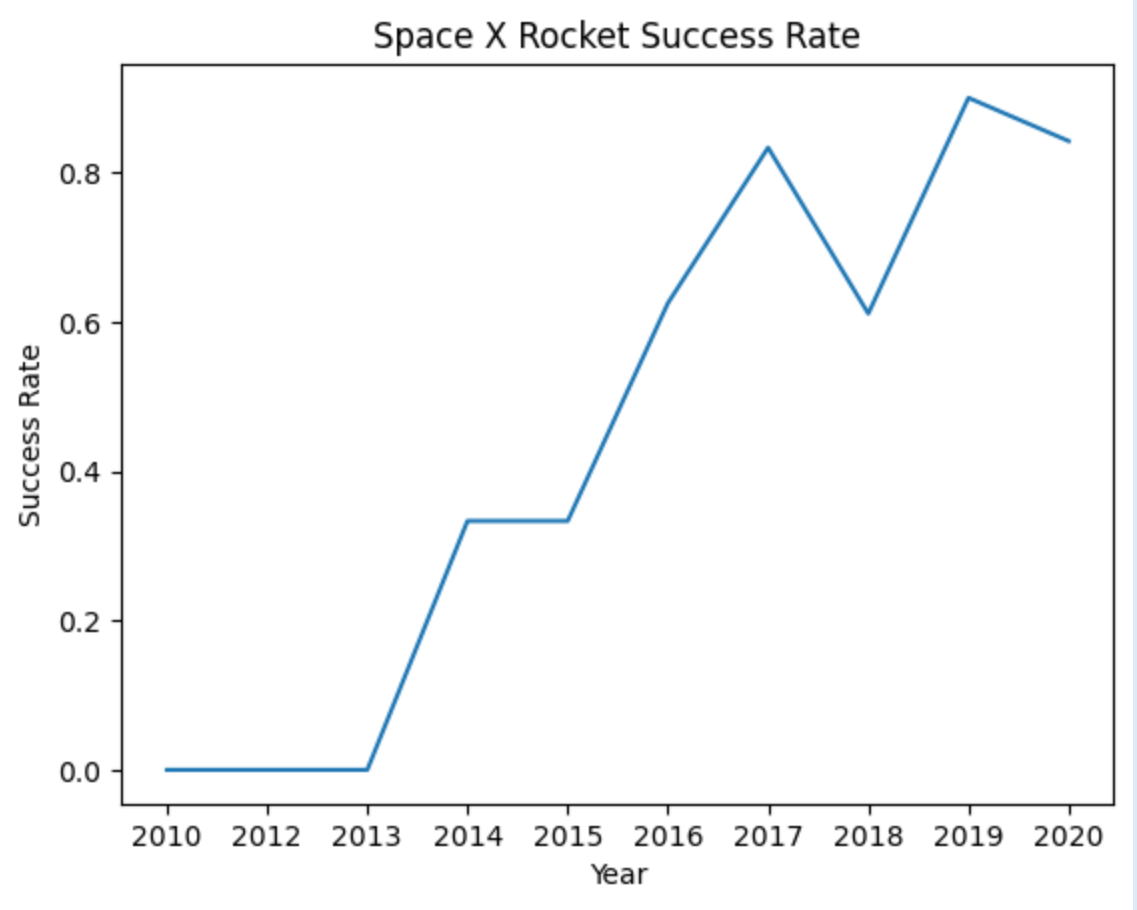
Insights:

- Payload Mass appears to greatly affect the success of launches in certain orbit types.
- For the **LEO** orbit type, **heavier payloads positively affect** the success rate of missions., while similar results are observed for the **PO** and **ISS** orbit types
- For the GTO orbit type specifically, lighter payloads appears to have a marginally positive effect on the success rate, but that effect is not particularly strong due to small sample size for low payload weights. Otherwise, **GTO is unaffected by Payload Mass.**
- **SSO** and LEO launches require **lower payload masse**s, but are highly **successful.**

Launch Success Yearly Trend

Insights:

- **General trend of success rate increasing between the years 2013-2020.**
- **Dip around 2018**, that could be explained by unsuccessful VLEO orbit type launches carried out at the KSC LC 39A launch site.
- **2010-2013** appears to be **unsuccessful**, but it can be assumed that during this 3 year period **critical research and improvements** were made that explain the later increasing trend in success rate of missions.



Exploratory Data Analysis with SQL Queries Results

All Launch Site Names

Query

```
[27]: %sql SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTBL
       * sqlite:///my_data1.db
      Done.
```

Result

[27]:

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

Comments:

The **SELECT DISTINCT** statement is used to return only unique (different) values of Launch Sites.

Launch Site Names Beginning with 'CCA'

Query

```
[9]: %sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5
 * sqlite:///my_data1.db
Done.
```

Result

[9]:

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)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	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

Comments:

- The **WHERE** clause is used to filter results.
- The **LIKE** operator is used in a **WHERE** clause to search for a specified pattern in a column, in this case **'CCA%'** in **Launch_Site.**

Total Payload Mass

Query

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

Result

[10]:	SUM("PAYLOAD_MASS__KG_")
	45596

Comments:

- The **SUM()** function returns the **total sum** of a specified numeric column.
- In this case we also specified the Customer to be **NASA (CRS)**.

Average Payload Mass by F9 v1.1

Query

```
[11]: %sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%'
 * sqlite:///my_data1.db
Done.
```

Result

[11]:	AVG("PAYLOAD_MASS__KG_")
	2534.6666666666665

Comments:

- The **AVG()** function returns the **average value** of a specified numeric column.
- In this case we also specified the **Booster_version** to be **F9 v1.1**.

First Successful Ground Landing Date

Query

```
[12]: %sql select min(date) as Date from SPACEXTBL where landing_outcome like 'Success (ground pad)'

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

Result

[12]:

Date
2015-12-22

Comments:

- The **AVG()** function returns the **smallest value** of a selected column.
- The **AS** command is use to **rename** a result with an **alias** and it exists only for the duration of the query.
- In this case we also specified **Landing_outcome** to resemble **'Success (ground pad)'** to find the first successful ground landing

Successful Drone Ship Landing with Payload between 4000 and 6000

Query

```
[13]: %%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)'
      and PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000

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

Result

[13]:

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

Comments:

- The **AND** operator is used to filter records based on more than one condition
- The **WHERE** clause of a query can contain one or many **AND** operators
- The **BETWEEN** operator selects values within a given range (numbers, text, dates). Begin and end values are included

Total Number of Successful and Failed Mission Outcomes (1)

Query 1

```
[14]: %%sql
SELECT MISSION_OUTCOME, COUNT(*) AS MOCount FROM SPACEXTBL
GROUP BY MISSION_OUTCOME ORDER BY MISSION_OUTCOME;

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

Result 1

[14]:

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

Query 2

```
[15]: %%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL
                    WHERE "MISSION_OUTCOME" LIKE '%Success%')
                    AS SUCCESS, (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL
                                 WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE

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

Result 2

[15]:

SUCCESS	FAILURE
100	1

Total Number of Successful and Failed Mission Outcomes (2)

Query 2

```
[15]: %%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL
                    WHERE "MISSION_OUTCOME" LIKE '%Success%')
                    AS SUCCESS, (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL
                                 WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE
 * sqlite:///my_data1.db
Done.
```

Result 2

[15]:

SUCCESS	FAILURE
100	1

Comments:

- The **COUNT()** function returns the number of rows that matches a specified criterion
- A **subquery** is an SQL query **nested** inside a larger query that can occur in any of the **SELECT**, **FROM** or **WHERE** clauses.
- The **GROUP BY** statement groups rows that have the same values into summary rows and is most often used along with an aggregate function.
- The **ORDER BY** keyword is used to sort the result-set in ascending or descending order. Default is ascending order, while for descending order **DESC** keyword must be specified.

Boosters Carried Maximum Payload

Query

```
[15]: %%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTBL
      WHERE "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTBL)

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

Result

[15]:

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

Comments:

- Use of **subquery** in the **WHERE** clause of the main query

2015 Launch Records

Query 1

```
[21]: %sql Select * from SPACEXTBL ORDER BY Date
 * sqlite:///my_data1.db
Done.
```

Result 1

ordered by date **YYYY-MM-DD**

07-09			40						
2014-07-14	15:15:00	F9 v1.1	CCAFS LC-40	OG2 Mission 1 6 Orbcomm-OG2 satellites	1316	LEO	Orbcomm	Success	Controlled (ocean)
2014-09-21	05:52:00	F9 v1.1 B1010	CCAFS LC-40	SpaceX CRS-4	2216	LEO (ISS)	NASA (CRS)	Success	Uncontrolled (ocean)
2015-02-03	03:50:00	F9 v1.1 B1014	CCAFS LC-40	ABS-3A Eutelsat 115 West B	4159	GTO	ABS Eutelsat	Success	No attempt
2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-04-27	23:03:00	F9 v1.1 B1016	CCAFS LC-40	Turkmen 52 / MonacoSAT	4707	GTO	Turkmenistan National Space Agency	Success	No attempt
2015-06-28	14:21:00	F9 v1.1 B1018	CCAFS LC-40	SpaceX CRS-7	1952	LEO (ISS)	NASA (CRS)	Failure (in flight)	Precluded (drone ship)
2015-10-01	09:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-11-02	23:03:00	F9 v1.1 B1013	CCAFS LC-40	DSCOVR	570	HEO	U.S. Air Force NASA NOAA	Success	Controlled (ocean)

Query 2

```
[23]: %%sql select substr(DATE, 6, 2) as Month, landing_outcome, booster_version, launch_site
from SPACEXTBL where DATE like '2015%' AND landing_outcome = 'Failure (drone ship)'
order by Date
 * sqlite:///my_data1.db
Done.
```

Result 2

[23]:

Month	Landing_Outcome	Booster_Version	Launch_Site
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40

April

October

Comments:

- The **SUBSTR()** function is used to extract a substring from a string starting at any position

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

Query

```
[24]: %%sql select landing_outcome, count(*) as count from SPACEXTBL
      where Date >= '2010-06-04' AND Date <= '2017-03-20'
      GROUP by landing_outcome ORDER BY count Desc

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

Result

[24]:

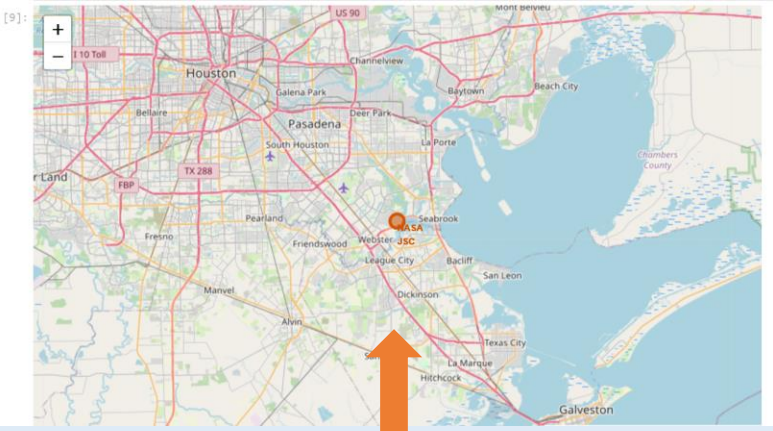
Landing_Outcome	count
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

Comments:

- The **DESC** command keyword is used to sort the data in the result-set in **descending order.**
- In this case we are ranking the Landing Outcomes by absolute frequency between the specified dates

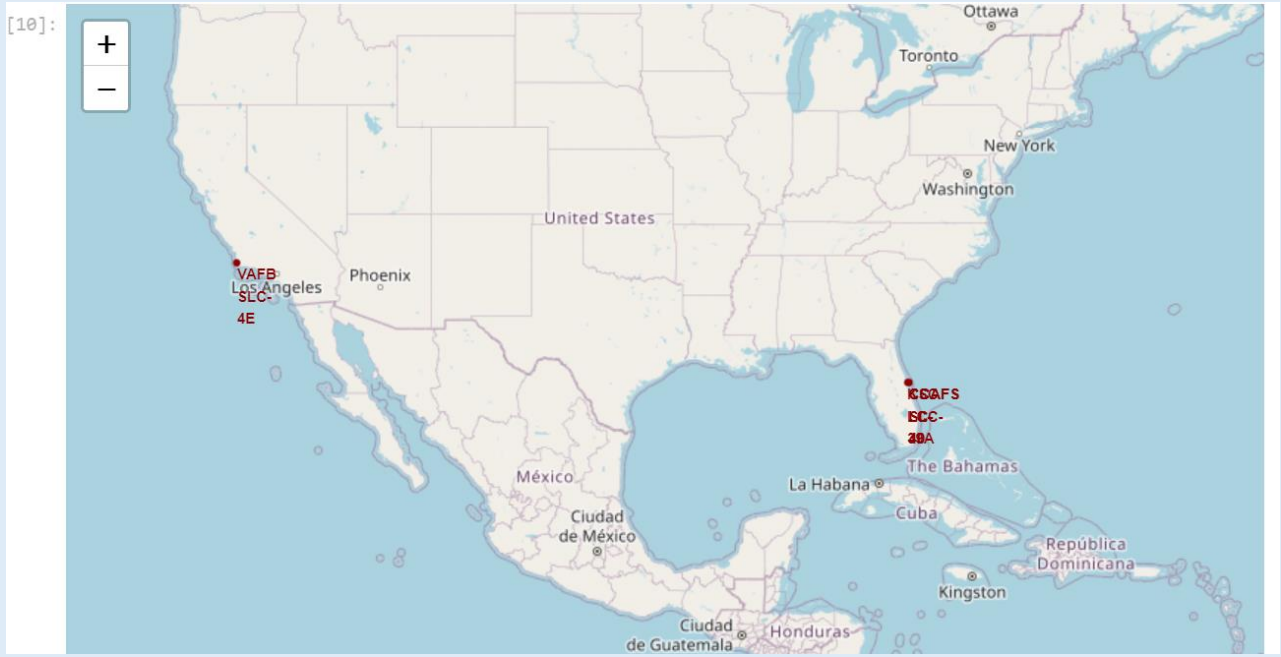

Section 3
Launch Sites
Proximities Analysis

Launch Site Markers on Map (1)

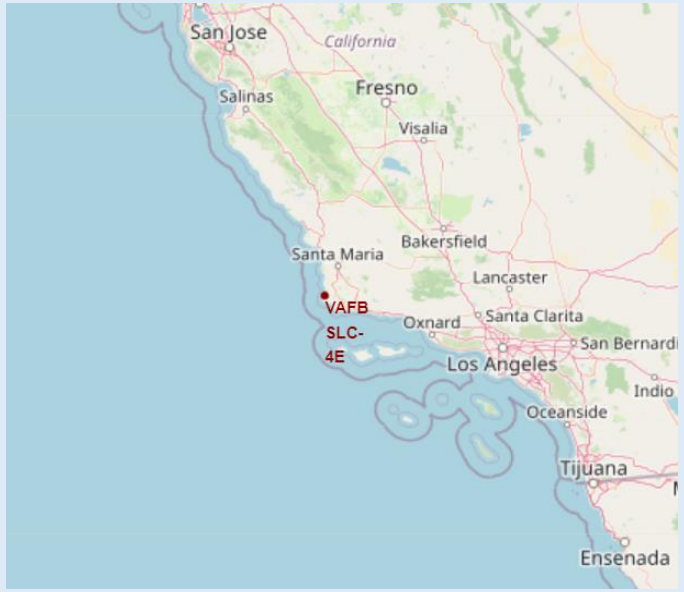


The **NASA Johnson Space Center** is located near Houston, Texas.

The SpaceX Launch Sites are located in **California** (VAFB SLC-4E) and in **Florida** (KSC LC-39A, CCAFS SLC-40, CCAFS LC-40).
Interestingly, all sites are **located relatively near the Equator**, in the southern part of the US.



Launch Site Markers on Map (2)



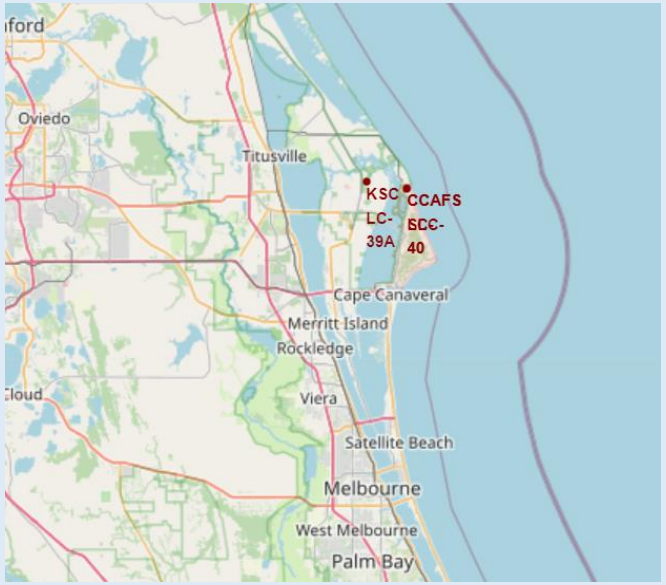
California

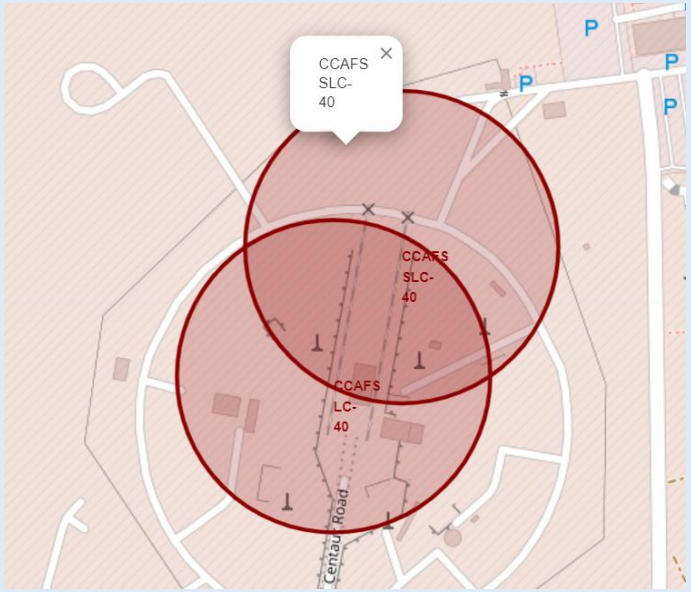
We can clearly observe the **VAFB SLC- 4E** near **Los Angeles, CA.**

Close-ups of launch site markers

Florida

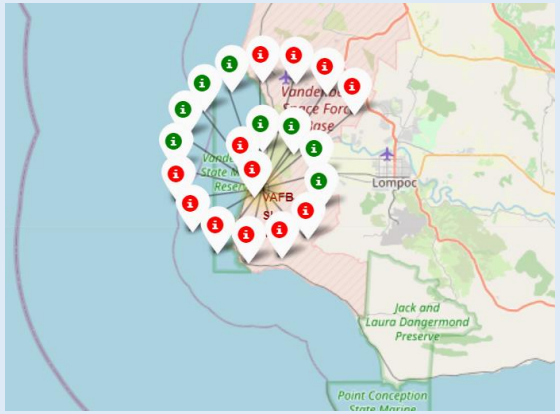
KSC LC-39A, CCAFS LCS-40 and **CCAFS LC-40** on **Merritt Island, Fl.**





CCAFS LCS-40 and **CCAFS LC-40** next to each other on the **Cape Canaveral Space Force Station.**

Color Labeled Markers for Launch Outcomes



VAFB SLC- 4E

Total Launches: 20

Successful: 8

Failed: 12

Success Rate: 40%

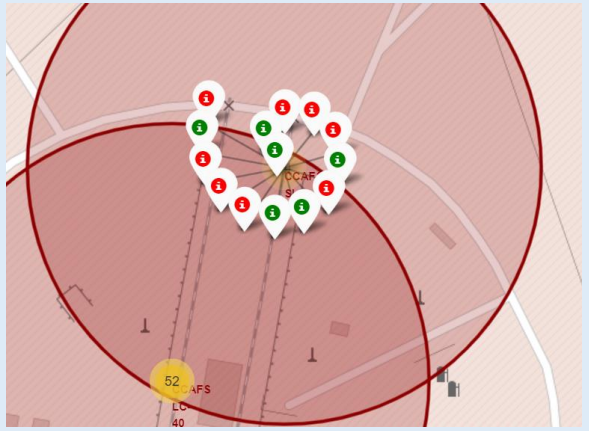
CCAFS SLC- 40

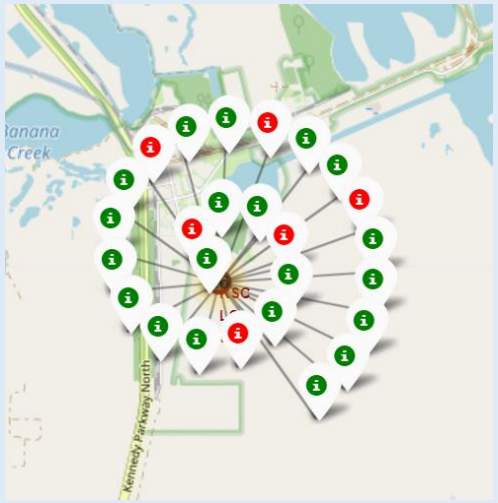
Total Launches: 14

Successful: 6

Failed: 8

Success Rate: 42.86%





KSC LC- 39A

Total Launches: 26

Successful: 20

Failed: 6

Success Rate: 76.92%

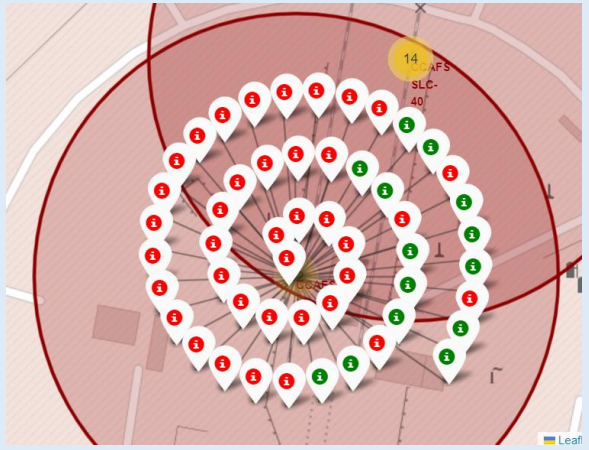
CCAFS LC- 40

Total Launches: 52

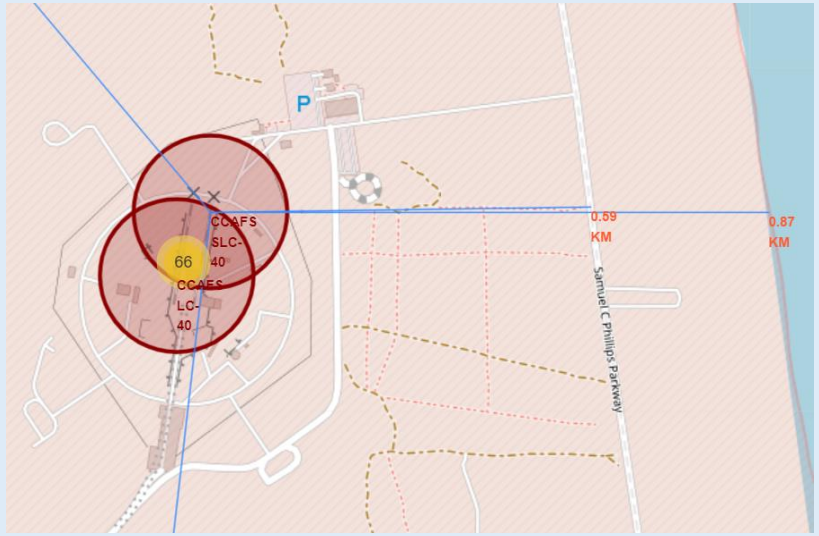
Successful: 14

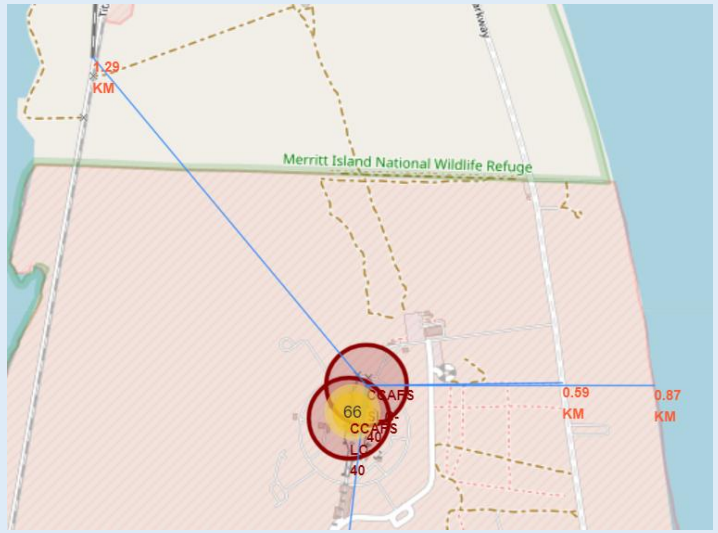
Failed: 38

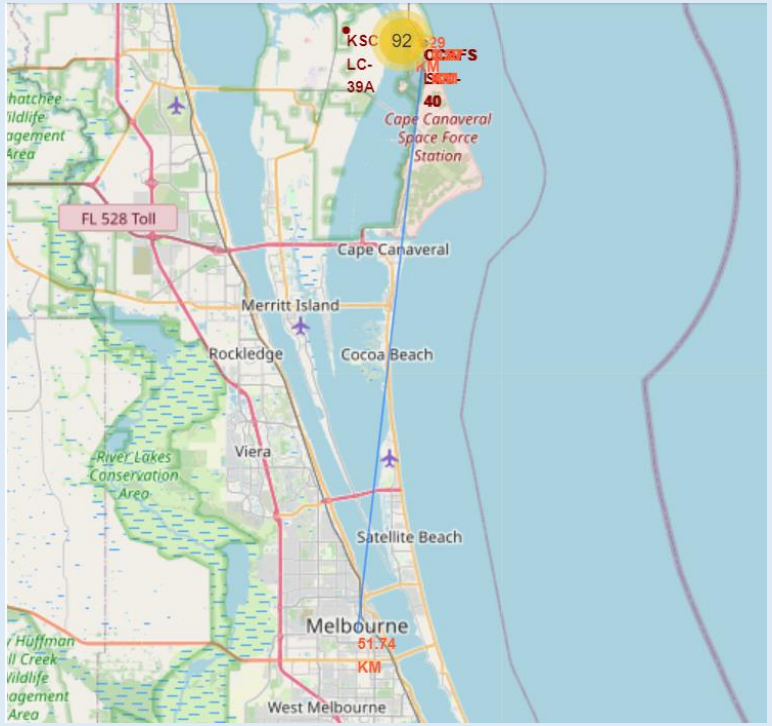
Success Rate: 26.92%



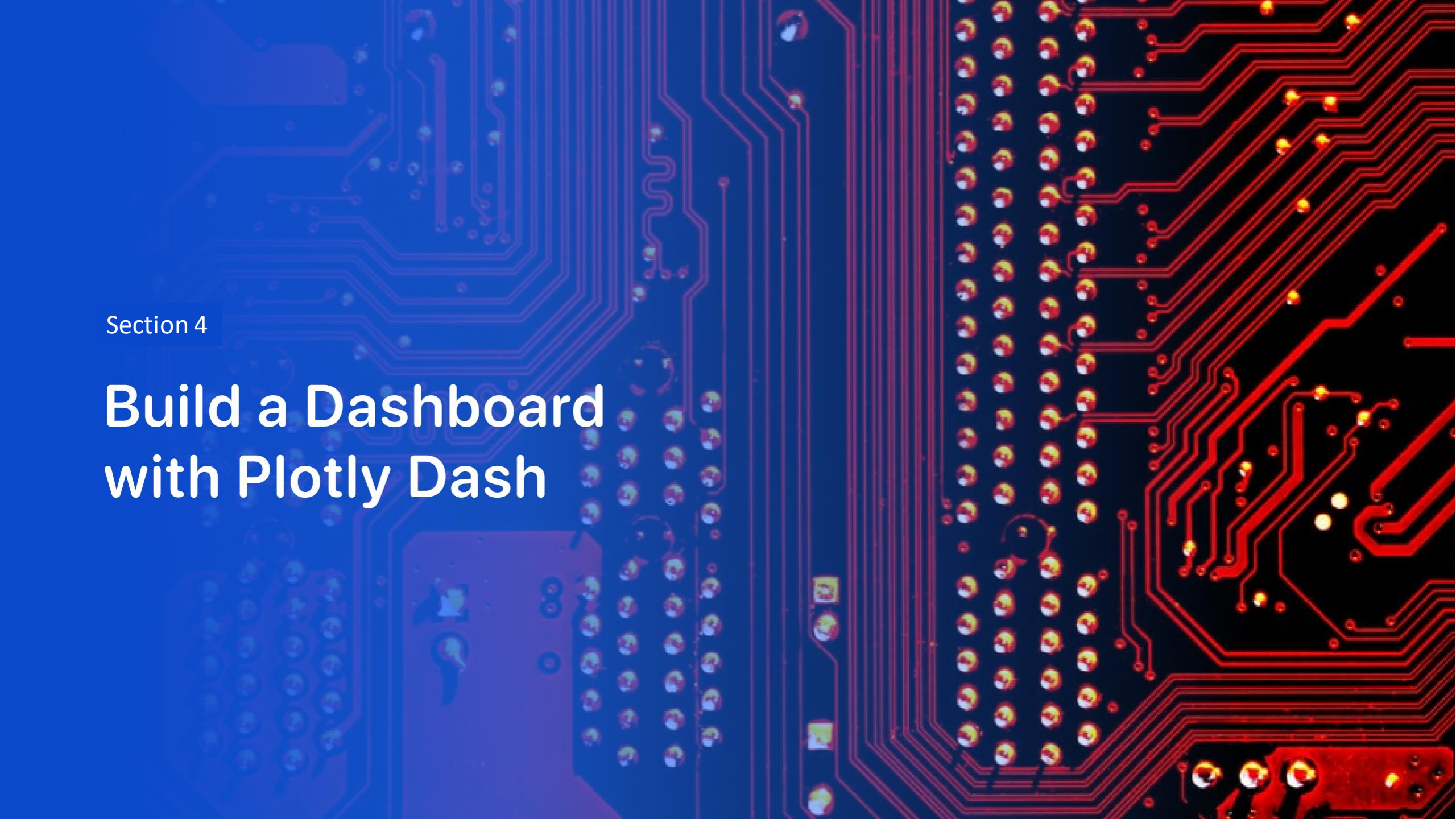
Distances between CCAFS SLC-40 and its proximities



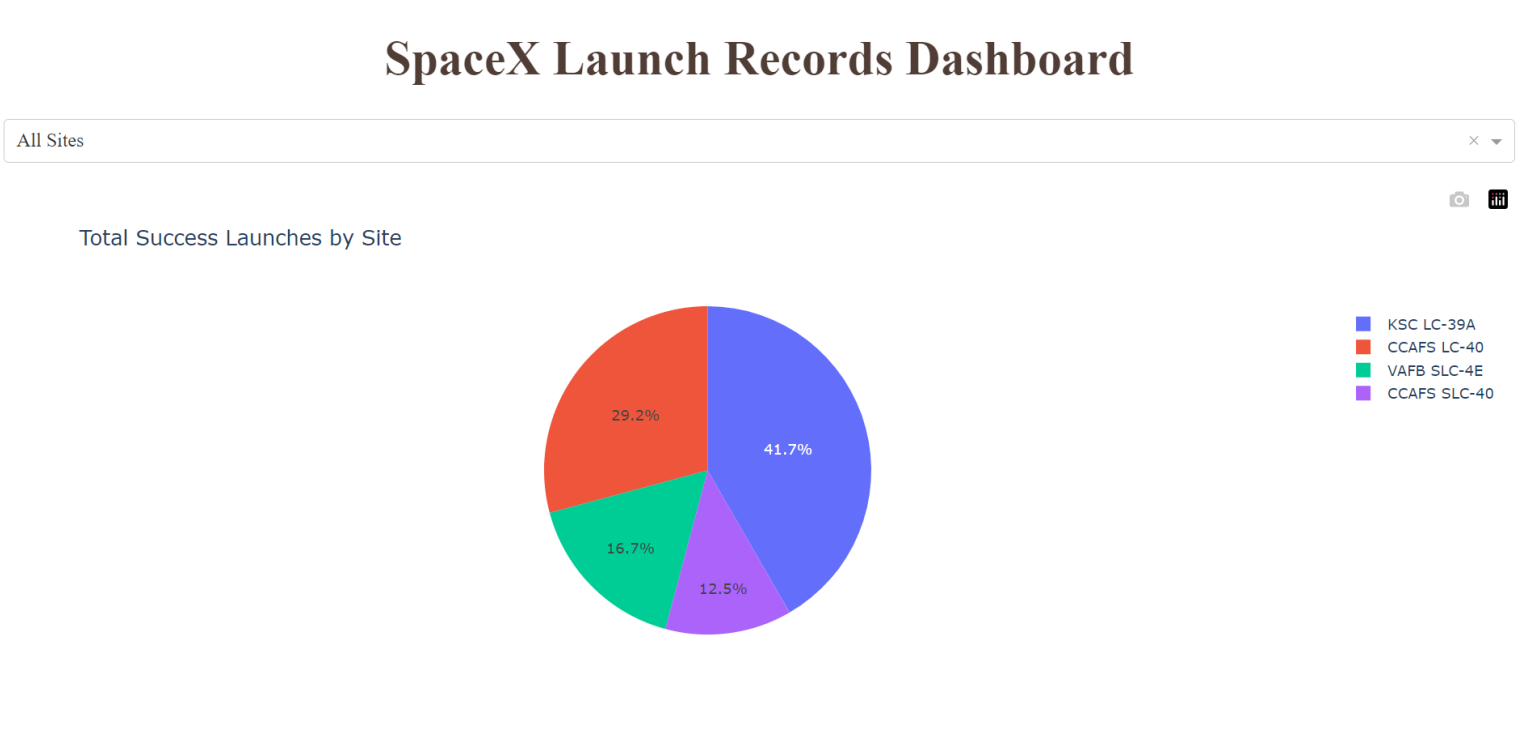




Closest coastline? 0.87km - CLOSE
Nearest highway? 0.59km - CLOSE
Closest railway? 1.29km - RELATIVELY CLOSE
Nearest city? Melbourne, Fl (51.74km) - FAR


Section 4
Build a Dashboard with Plotly Dash

Total Successful Launches by Site



Total Successful Launches: 48

Successful in VAFB SLC-4E: 8

✓ 16.67%

Successful in KSC LC-39A: 20

✓ 41.66%

Successful in CCAFS SLC-40: 6

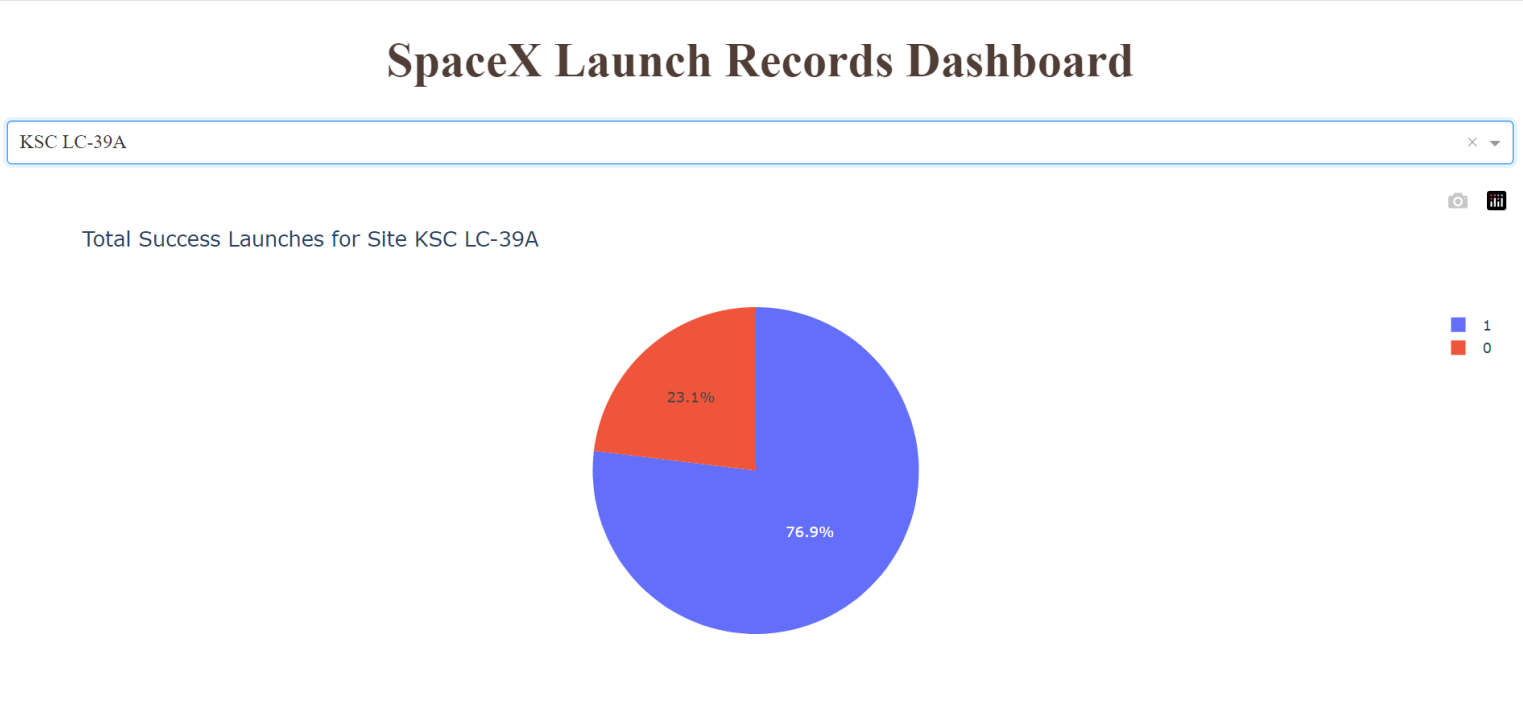
✓ 12.5%

Successful in CCAFS LC-40: 14

✓ 29.17%

- This pie chart depicts the **percentage of successful launches among all Launch Sites**
- **Most successful launches** have been carried out in **KSC LC-39A** in Florida, taking up **41.7%** of all successful launches

Successful VS Failed Launches for Site KSC LC-39A



KSC LC-39A

Total Launches: 26

Successful: 20

✓ Success Rate: 76.92%

Failed: 6

✓ Failure Rate: 23.08%

- **KSC LC-39A** is the SpaceX Launch Site with the **highest mission success rate** with an incredible **76.92%** of launches being successful

Successful VS Failed Launches for Site KSC LC-39A

- **Highest success rates** are observed for Payload Masses withing the **2000-5500kg** range.
- Generally **lower payloads** seem to make for **more successful** missions.
- Payloads **under 6000kg** are **especially successful** when the **FT Booster** is used.
- There is **not enough data** to make conclusive predictions for payloads **heavier than 7000kg**

Section 5

Predictive Analysis (Classification)

Classifier 1: Logistic Regression

Model Construction (Grid Search)

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

```
[14]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}#
      lr = LogisticRegression()
      logreg_cv = GridSearchCV(lr, parameters, cv=10)
      logreg_cv.fit(X_train, Y_train)
```

Best parameters and Training Set Accuracy

```
[15]: 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
```

Test Set Accuracy

```
[16]: lr_score=logreg_cv.score(X_test,Y_test)
      lr_score
```

```
[16]: 0.8333333333333334
```

Confusion Matrix

```
[57]: yhat=logreg_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```

Classifier 2: Support Vector Machine

Model Construction (Grid Search) &

Best parameters and Training Set Accuracy

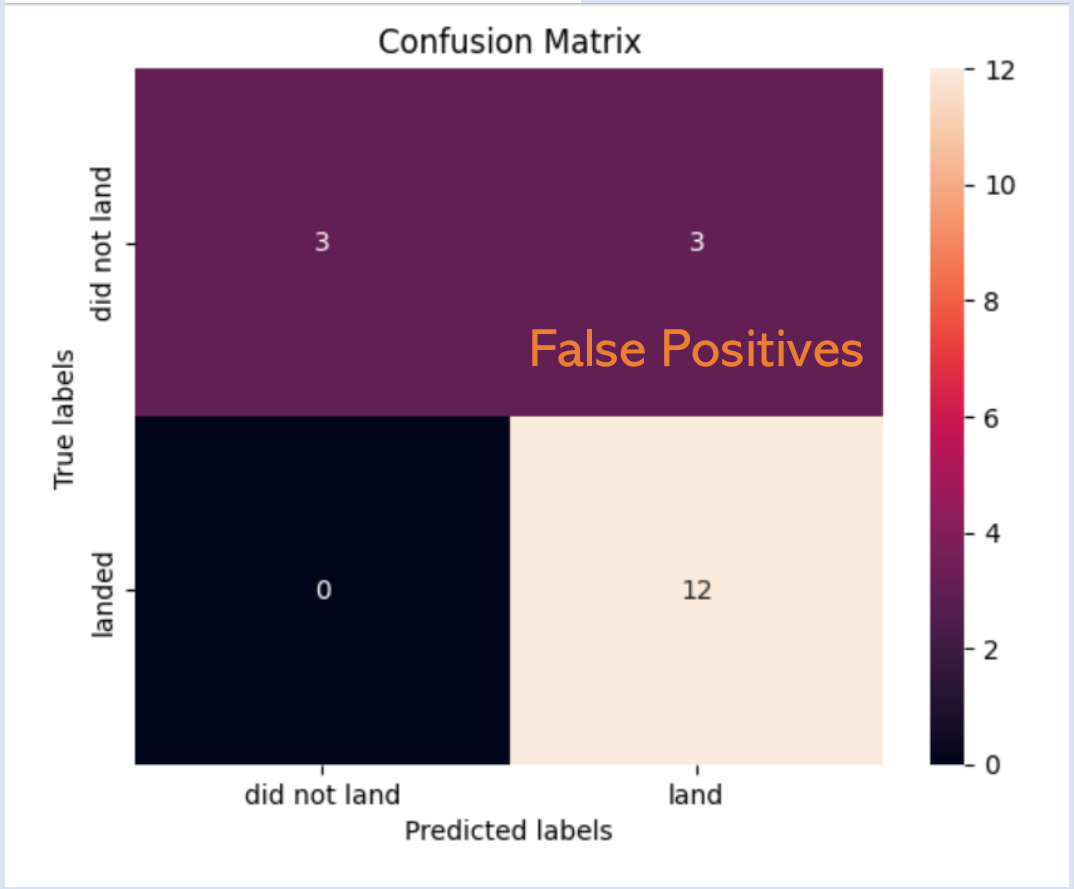
```
[18]: parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
                    'C': np.logspace(-3, 3, 5),
                    'gamma':np.logspace(-3, 3, 5)}
      svm = SVC()

[19]: gridsearch_cv_svm= GridSearchCV(svm, parameters, scoring='accuracy', cv=10)
      svm_cv= gridsearch_cv_svm.fit(X_train, Y_train)

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

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

Test Set Accuracy

```
[21]: svm_score= svm_cv.score(X_test,Y_test)
      svm_score

[21]: 0.8333333333333334
```

Confusion Matrix

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



Classifier 3: Decision Tree

Model Construction (Grid Search) &

Best parameters and Training Set Accuracy

```
[94]: parameters = {'criterion': ['gini', 'entropy'],
           'splitter': ['best', 'random'],
           'max_depth': [2*n for n in range(1,10)],
           'max_features': ['sqrt'], #auto has been removed as an option in python 1.3
           'min_samples_leaf': [1, 2, 4],
           'min_samples_split': [2, 5, 10]}

      tree = DecisionTreeClassifier()
```

```
[98]: gridsearch_cv_tree = GridSearchCV(tree, parameters, scoring='accuracy', cv=10)
      tree_cv = gridsearch_cv_tree.fit(X_train, Y_train)
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 18, 'max_features': 'sqrt', 'min_
samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.8892857142857142
```

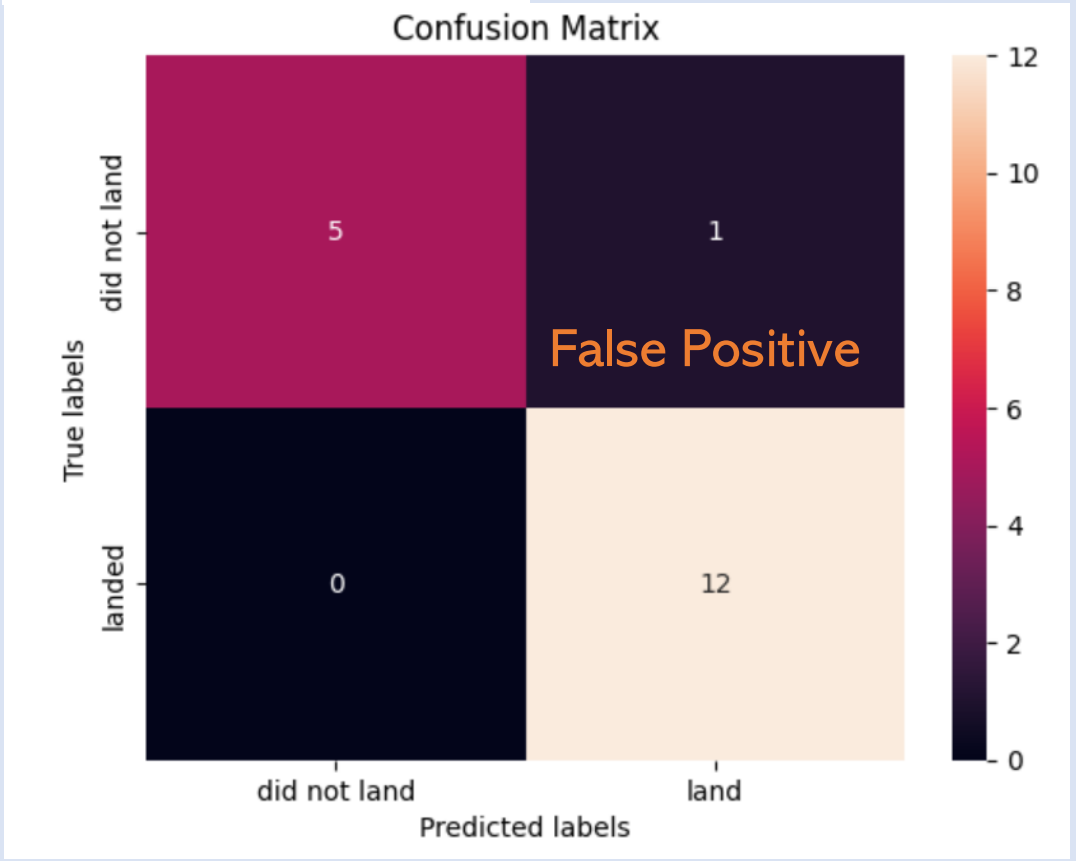
Test Set Accuracy

```
[102]: tree_score = tree_cv.score(X_test, Y_test)
       tree_score
```

```
[102]: 0.9444444444444444
```

Confusion Matrix

```
[101]: yhat = tree_cv.predict(X_test)
       plot_confusion_matrix(Y_test,yhat)
```



Classifier 4: K-Nearest Neighbors

Model Construction (Grid Search) &

Best parameters and Training Set Accuracy

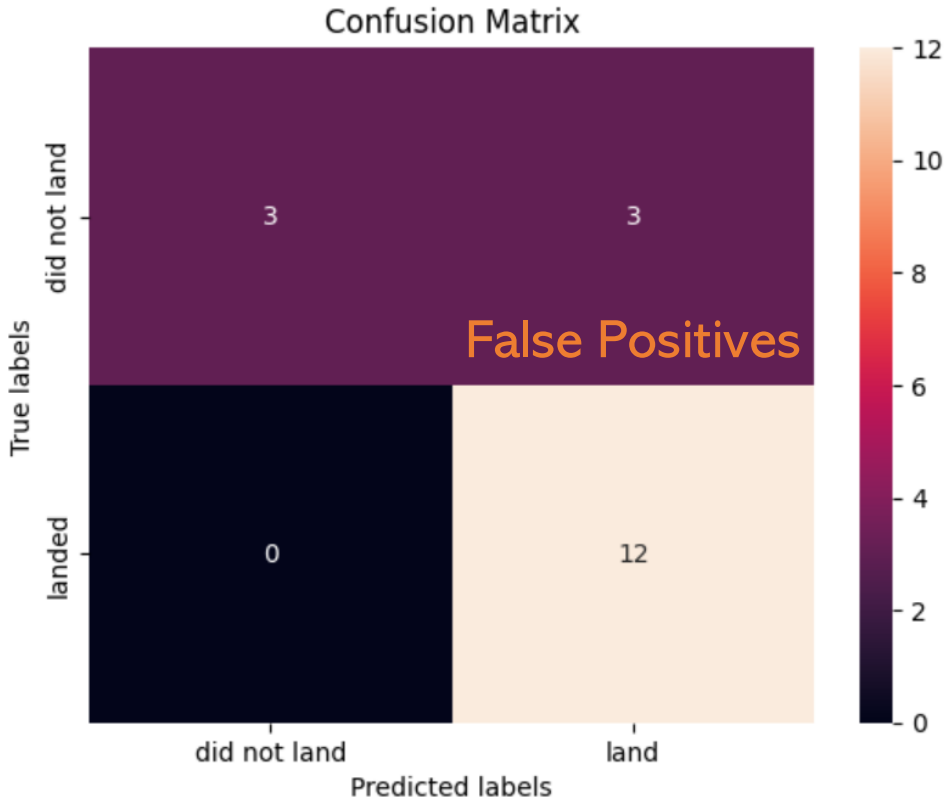
```
[103]: parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                     'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                     'p': [1,2]}

       KNN = KNeighborsClassifier()

[104]: gridsearch_cv_knn = GridSearchCV(KNN, parameters, scoring='accuracy', cv=10)
       knn_cv = gridsearch_cv_knn.fit(X_train, Y_train)

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

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

Test Set Accuracy

```
[106]: knn_score=knn_cv.score(X_test, Y_test)
       knn_score

[106]: 0.8333333333333334
```

Confusion Matrix

```
[107]: yhat = knn_cv.predict(X_test)
       plot_confusion_matrix(Y_test,yhat)
```

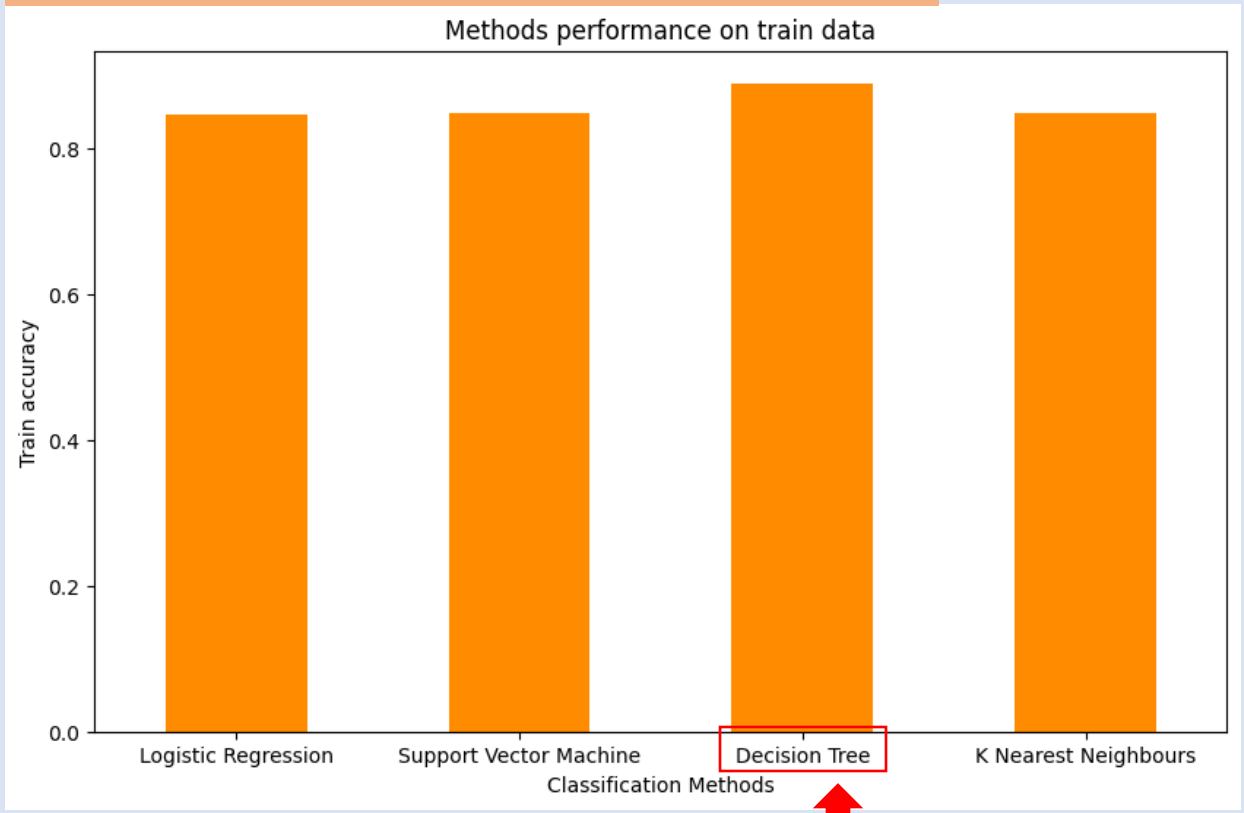


Determining the best Classification Model (1) : Training Set Accuracy

[108]:

	Classifiers	Accuracy Score (TEST)	Accuracy Score (TRAIN)
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.944444	0.889286
3	K Nearest Neighbours	0.833333	0.848214

Bar Plot for Training Set Accuracy



Sort by Training Set Accuracy

[109]:

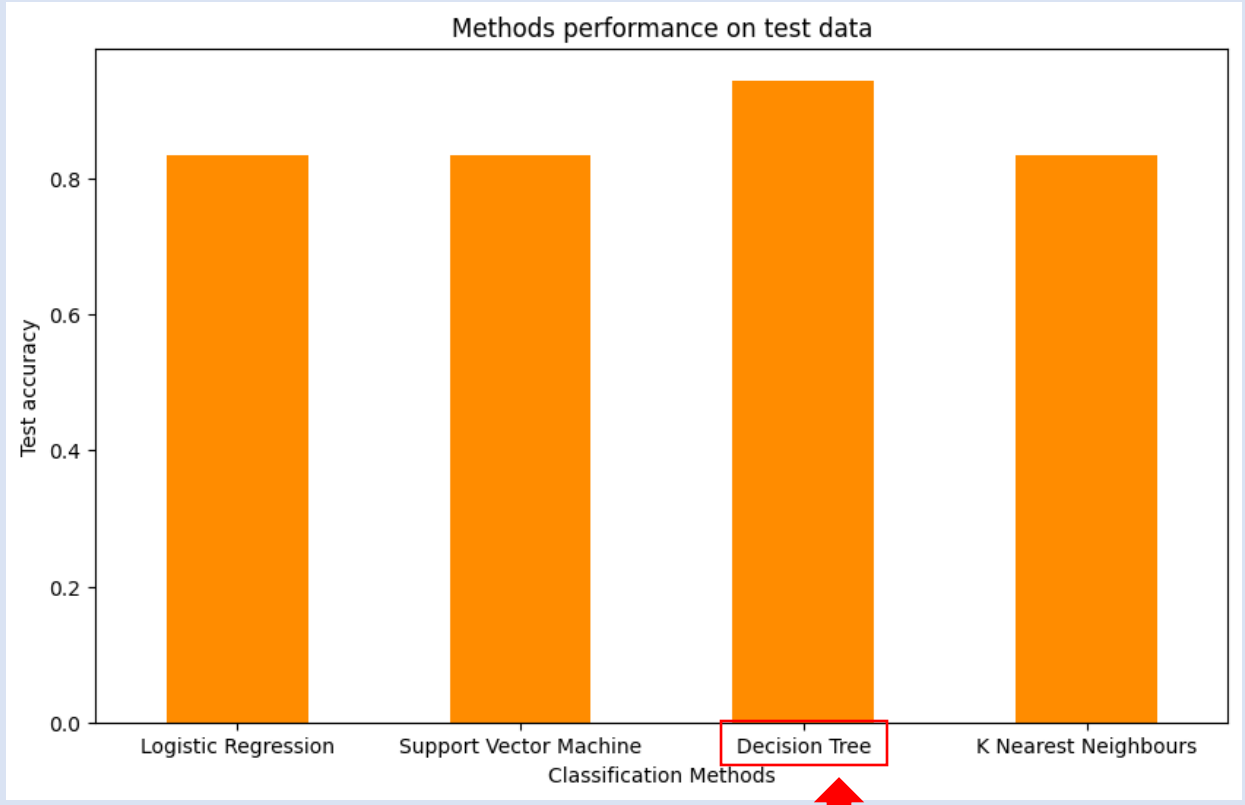
	Classifiers	Accuracy Score (TRAIN)
2	Decision Tree	0.889286
3	K Nearest Neighbours	0.848214
1	Support Vector Machine	0.848214
0	Logistic Regression	0.846429

Determining the best Classification Model (2) : Test Set Accuracy

[108]:

	Classifiers	Accuracy Score (TEST)	Accuracy Score (TRAIN)
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.944444	0.889286
3	K Nearest Neighbours	0.833333	0.848214

Bar Plot for Test Set Accuracy



Sort by Test Set Accuracy

[110]:

	Classifiers	Accuracy Score (TEST)
2	Decision Tree	0.944444
0	Logistic Regression	0.833333
1	Support Vector Machine	0.833333
3	K Nearest Neighbours	0.833333

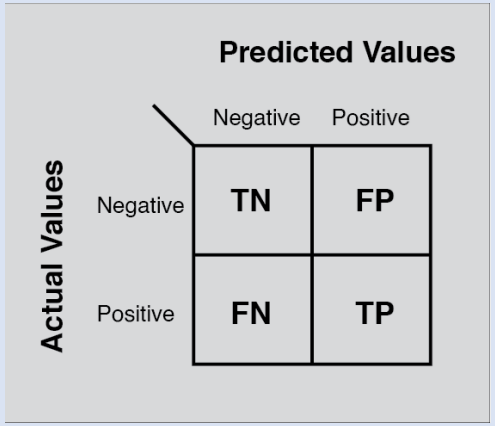
Classification Model Choice (1)

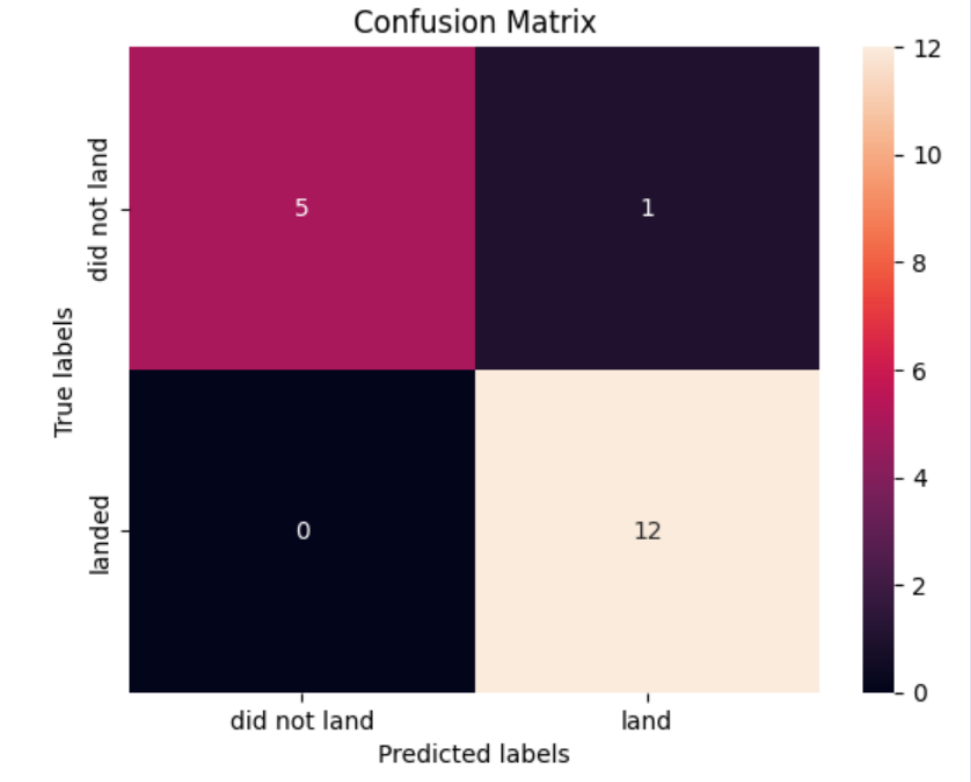
✓ The Model with the best performance is the **Decision Tree** with the parameters below:

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 18, 'max_features': 'sqrt', 'min_
samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.8892857142857142
```

✓ The **Confusion Matrix** of the chosen model is given again here.

✓ The only issue appears to be a **tendency towards false positives**, but that can be explained by the small size of the training and the testing set





Classification Model Choice (2)

Disclaimer
Although random state has been set to 2 during the dataset training/test split, **the construction of a decision tree still involves a degree of randomness**. Thus, by **executing the same code twice one could get completely different results** when it comes to the Decision Tree Model, maybe even worse performing than the other Classifiers who follow a deterministic constructive process.

In the next slide, we give examples of optimal Decision Tree Models for different code runs. We see a big degree of variation.

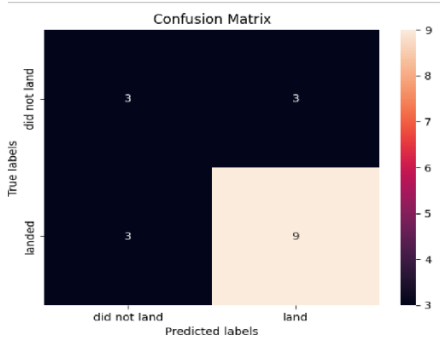
What this means is, that if you get the Decision Tree shown in the previous slide from the first run, you wouldn't have reason to suspect that **it is a somewhat random result**. However, this specific occurrence of output result is indeed the one of the best possible Decision Tree given the current dataset.

Classification Model Choice (3)

Examples of optimal Decision Tree Models for different code runs

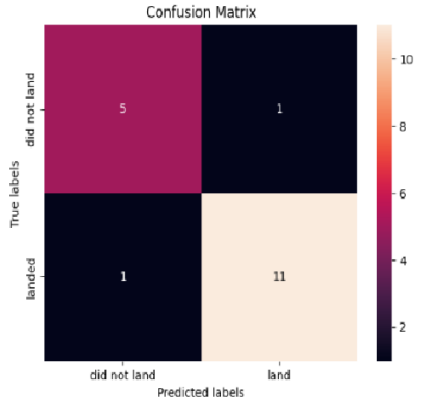
```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_
samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'best'}
accuracy : 0.8875000000000002
```

```
[116]: tree_score = tree_cv.score(X_test, Y_test)
       tree_score

[116]: 0.6666666666666666
```



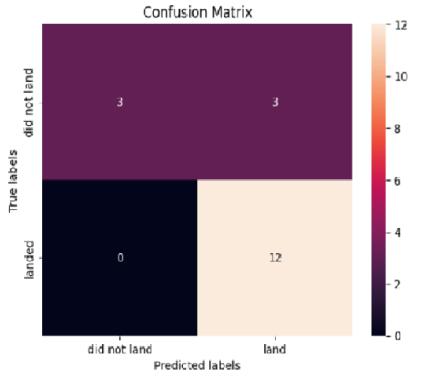

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samp
les_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.875
```

```
[127]: tree_score = tree_cv.score(X_test, Y_test)
       tree_score

[127]: 0.8888888888888888
```



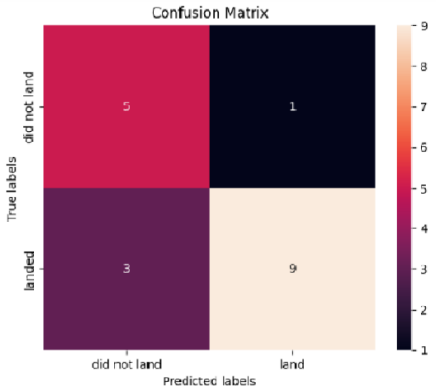

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_s
amples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.8892857142857145
```

```
[122]: tree_score = tree_cv.score(X_test, Y_test)
       tree_score

[122]: 0.8333333333333334
```




```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_sam
ples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
accuracy : 0.875
```

```
[137]: tree_score = tree_cv.score(X_test, Y_test)
       tree_score

[137]: 0.7777777777777778
```



Summary

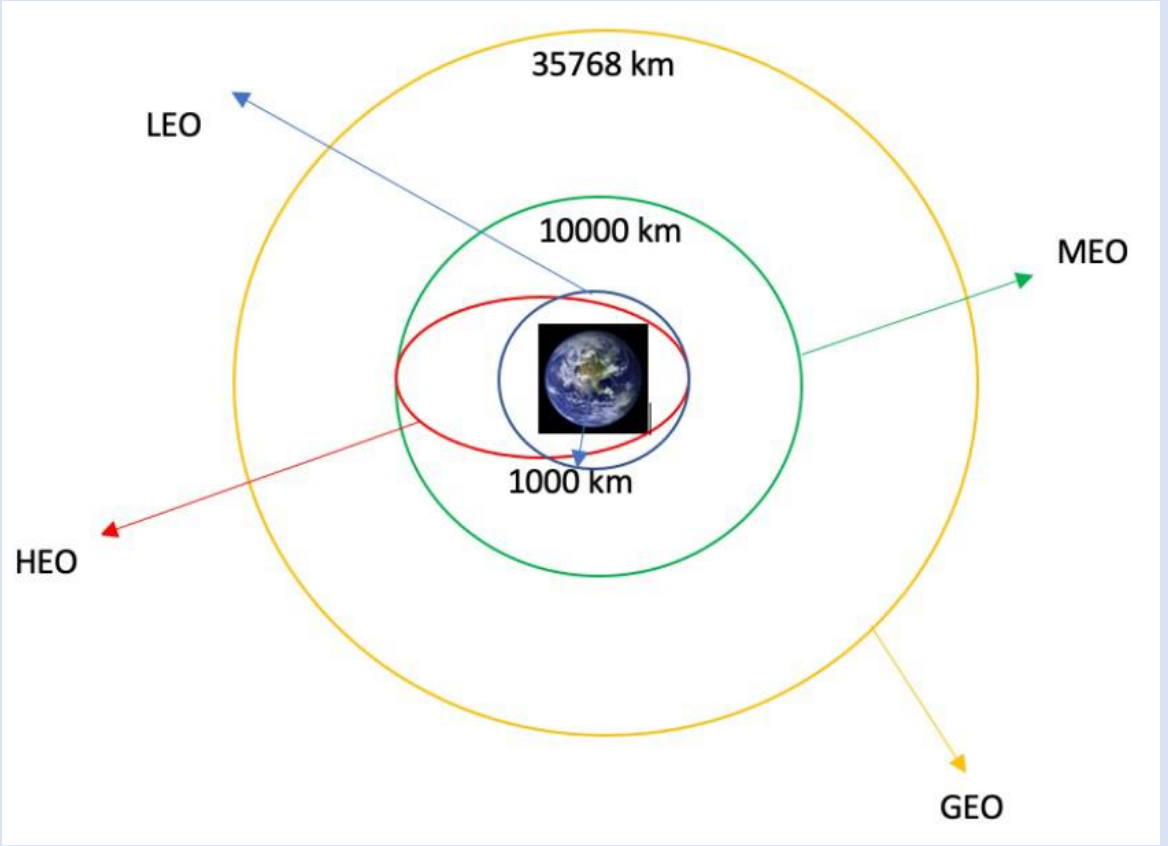
- The success of a SpaceX launch is a multifactorial problem, which we attempted to solve using Exploratory Data Analysis, static and interactive Data Visualizations and Predictive modeling, specifically Classification, since the variable 'Class' we are trying to predict is categorical.
- The purpose of this project was to determine the viability of a new competitor space company, called SpaceY. We do that by predicting whether the first stage of a rocket launch will land successfully or not, which is the main reason SpaceX's commercial costs are that much lower thn those of its competitors
- The data collected were public and freely available using the SpaceX REST API and Web Scraping the SpaceX Wikipedia page.

Conclusions (1)

- **Launch success rate** has significantly **increased between 2013-2020**, with a dip in 2018.
- As the overall success rate of launches has gone up over the years, we can assume that **valuable knowledge** has been obtained from previous missions, so that the **identified problems** encountered in past missions have presumably been **fixed**.
- Therefore **the larger the amount of total launches carried out the higher the probability of mission success.**
- The "**best performing**" launch site according to the current dataset is **KSC-LC-39A** (**Kennedy Space Center Launch Complex 39A**) with a success rate of **76.92%**, but it is unclear why. More information is required to decide whether a specific launch site is superior to others for some reason.

Conclusions (2)

- The orbit type of the mission also influence the success or failure of the first stage retrieval
- The orbit types with the **highest success rates** are **ES-L1** (100%), **GEO** (100%), **HEO** (100%), **SSO** (100%) and **VLEO** (~85%).
- However, since **sample sizes** are extremely **small**, we can deduce that **VLEO is the most successful orbit type** with a sample size of 14, **followed by SSO** with 5 total launches
- **Least successful orbit type is SO**, although only one launch has been attempted.
- In the **mid-tier performance** line-up, we have **LEO, PO, MEO, ISS, GTO**, sorted in descending success rate.

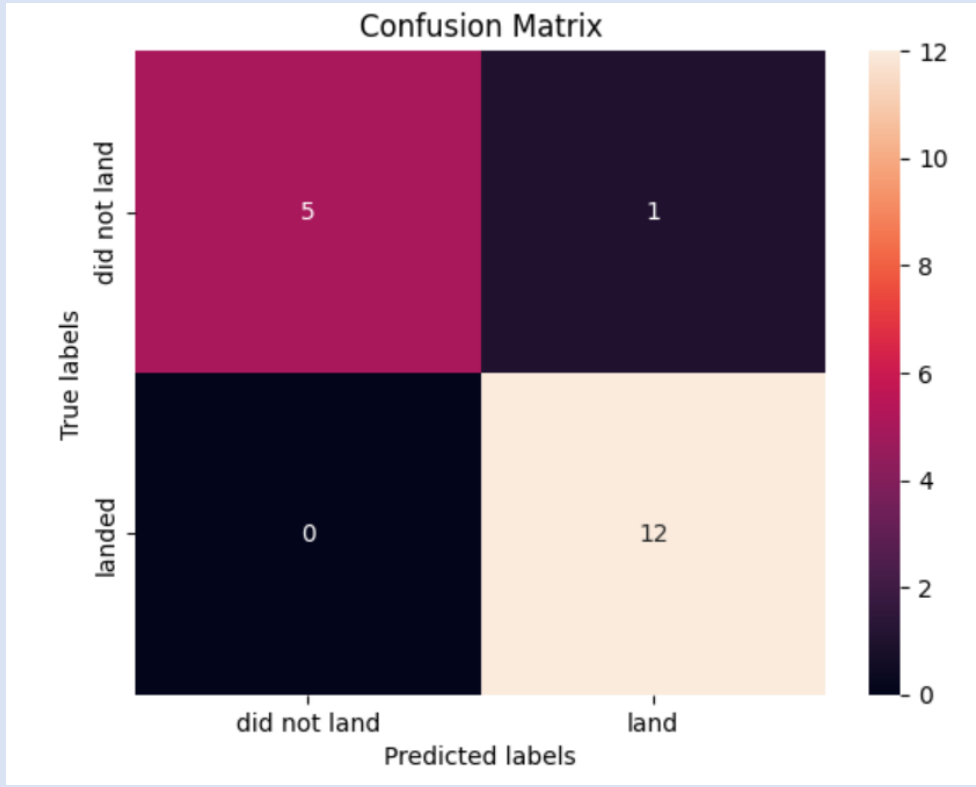


Conclusions (3)

- For different orbit types **Payload Mass** might **play an important role** in determining whether a launch will be successful or not.
- Generally **low and medium weight payloads perform better than high payloads.**
- However, payloads **over 7000kg positively affect** the first stage landing outcome
- **SSO** orbit type launches perform best for **lower** payload masses
- **LEO** orbit type success rate is positively affected by **higher** payloads.
- **GTO** orbit type is mostly **unaffected by payload mass.**

Conclusions (4)

- Out of all the **Predictive Models** we have constructed, the **Decision Tree Classifier** shown in this slide has the **best accuracy score on both training and test set data**.
- With this classifier we can predict whether Allon Mask's SpaceY launches will have **a successful first stage landing or not**.




```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 18, 'max_features': 'sqrt', 'min_
samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.8892857142857142
```

Notes (1)

- As we have previously observed, the SpaceX launch sites are located in California(1) and Florida(3) in the **southern part of the US**.
- The choice to build spacecraft launch sites **near the Equator** becomes evident when we consider that most of these rockets (at least the ones in Florida) **travel eastwards** towards the Atlantic, **benefiting from the Earth's rotation** during take-off.
- Considering that the Earth's **rotational speed is the highest at the Equator**, we can now understand why launch sites are built in the south of the US.
- Moreover, we should keep in mind that **Earth's gravitational pull is weaker the closer we are to the Equator** and the furthest from the poles, due to **Earth's oblateness (equatorial bulge).**

Notes (2)

- **Vandenberg Space Force Base Space Launch Complex** (**VSFB-SLC**), previously known as **Vanderberg Air Force Base Space Launch Complex** (**VAFB-SLC**) is located in Santa Barbara County, California. The **VSFB-SLC-4E** launch lad has been used for Falcon 9 launches since late 2013. Space X had previously used the VSFB-SLC-3W launch site for their Falcon 1 missions.
- **Kennedy Space Center Launch Complex** (**KSC-LC**) is located on Merritt Island, Florida. The **KSC-LC-39A** launch site has been leased by Space X since 2014 for exclusive use and has been used for their Falcon 9 and Falcon Heavy launches since 2017.

Notes (3)

- **Cape Canaveral Space Force Station** (**CCSFS**), previously known as **Cape Canaveral Air Force Launch Complex** (**CCAFS**) is located on Cape Canaveral, Florida.
- **CCSFS** is **not to be confused** with **Cape Cod Space Force Station**, which is located in Massachusetts. To avoid confusion and in accordance with the materials of this course, we have referred to **CCSFS** with its previous name **CCAFS** in previous slides.
- **Cape Canaveral Space Force Station Space Launch Complex 40** (**CCSFS-SLC-40**) was previously known as **Cape Canaveral Space Force Launch Complex 40** (**CCSFS-LC-40**). **CCSFS-SLC-40** and **CCSFS-LC-40** are presented as two different launch sites during analysis for reasons unclear.
- **CCSFS-SLC-40** launch site has been leased by SpaceX since 2008 and has been launching Falcon 9 rockets since 2010.

Appendix

- GITHUB LINK: CAPSTONE PROJECT REPOSITORY
- Folium maps do not render on GitHub when using a Jupyter Notebook, so all the maps included in this presentation are screenshots from code runs.
- Many thanks are due to the Instructors of this course.

SOURCES

- https://api.spacexdata.com/v4/rockets/
- https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
- https://www.scienceabc.com/eyeopeners/why-are-rockets-launched-from-areas-near-the-equator.html
- https://en.wikipedia.org/wiki/Vandenberg_Space_Force_Base
- https://en.wikipedia.org/wiki/Kennedy_Space_Center_Launch_Complex_39A
- https://en.wikipedia.org/wiki/Cape_Canaveral_Space_Launch_Complex_40

Thank you!