# IBM DATA SCIENCE CAPSTONE PROJECT

Space X Falcon 9 Landing Prediction

David Alkass

2023-02-06

# OUTLINE

# EXECUTIVE SUMMARY

## Summary of Methodologies:

Project includes the following:

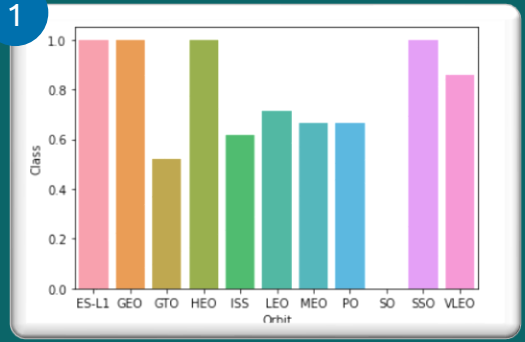
- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis by Classification

## Summary of Results:

Project yields and with visual examples:

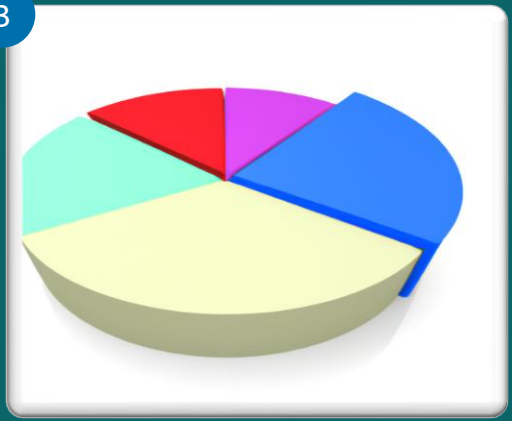
1. Exploratory Data Analysis (EDA)
2. Geospatial analysis
3. Interactive dashboard
4. Predictive classification models

1

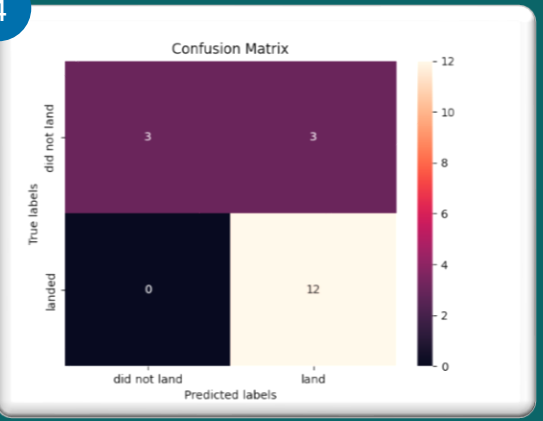



2

3



4

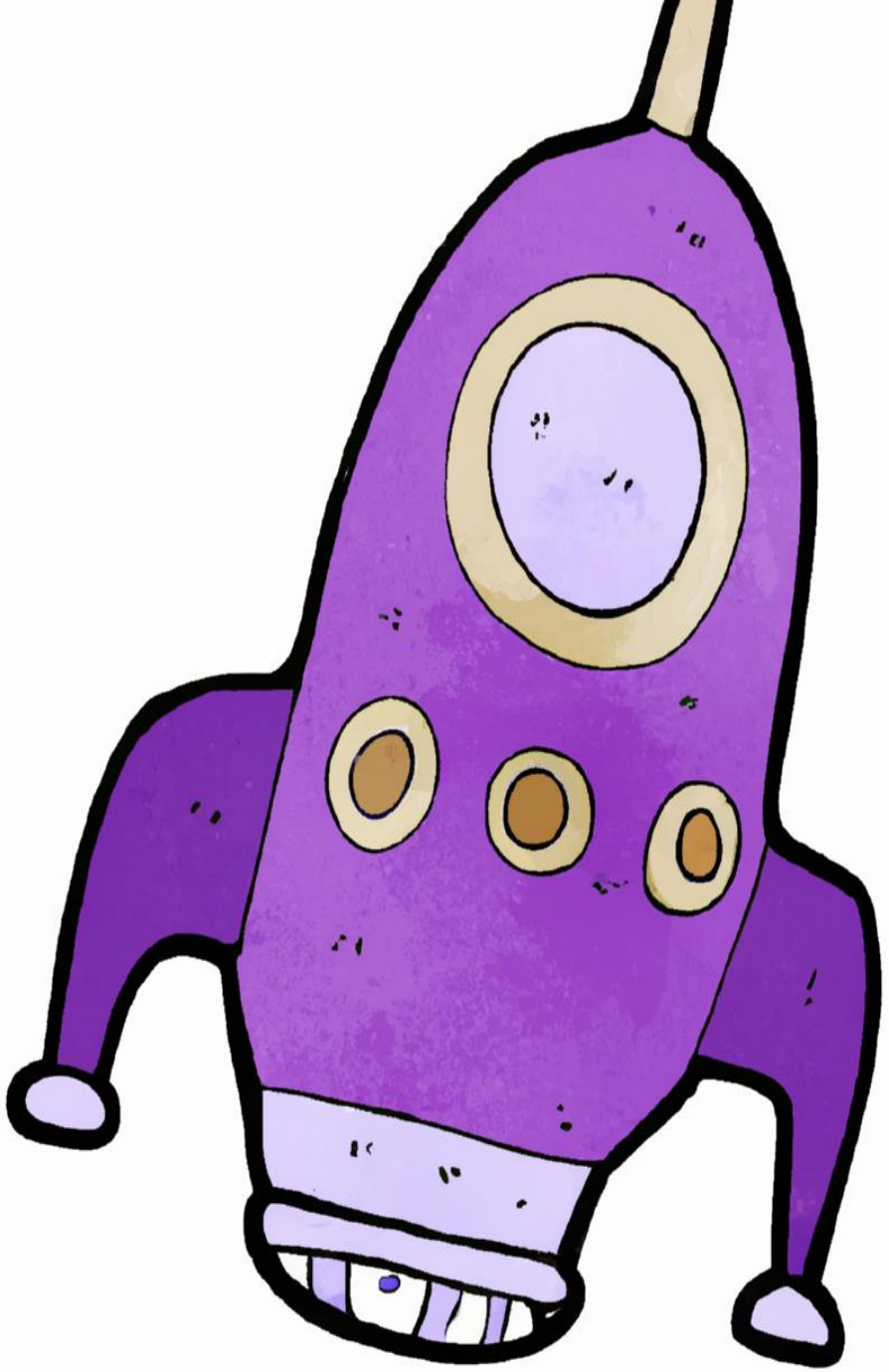


# INTRODUCTION

- SpaceX has massively decreased costs of rocket launches to around $62m. The reduction in cost is due to a property of the Falcon 9 rocket that allows it to land and reuse its first stage after detaching it during launch. This however is not always the case and sometimes it crashes and burns.

- If we can predict whether or not the stage one will be retrieved or at least land successfully, that would allow us to provide a more truthful cost prediction for potential customers of SpaceX.

- So the purpose of this project is to develop the ability of predicting cost based of the factor that is the stage one of Falcon 9´s construction.



# METHODOLOGY SUMMARY

1. Data Collection
   - Making GET requests to the SpaceX REST API
   - Web Scraping
2. Data Wrangling
   - Implement pandas method to remove NaN values
   - Apply `.value_counts()` for following information:
     - Number of launches on each site
     - Number and occurrence of each orbit
     - Number and occurrence of mission outcome per orbit type
   - Append Boolean column value to our data frame:
     - 0 when the booster did not land successfully
     - 1 when the booster did land successfully
3. Exploratory Data Analysis
   - Using SQL queries to manipulate and explore the SpaceX dataset
   - Matplotlib to visualize relationships between attributes, and determine patterns
4. Interactive Visual Analytics
   - Geospatial analysis using Folium
   - Creating an interactive dashboard using Plotly Dash
5. Data Modelling and Evaluation
   - Using Scikit-Learn to:
     - Pre-process the data
     - Split data for training with `train_test_split`
     - Apply and train classification models
     - Find hyperparameters using `GridSearchCV`
   - Plot and explore confusion matrices
   - Determine accuracy of models and comparing

# DATA COLLECTION – SPACE X REST API

Calling the SpaceX API to retrieve data regarding launches, containing various information about the rocket name, payload, launch details, type of landing, and landing success.

1

- Make a GET response to the SpaceX REST API
- Convert response to json-file for Pandas usability

2

- Create global list that that will serve as arrays within an array
- Create a dictionary of lists so that we may use their key-values for column names and thus we have an (array of arrays)

3

- Array of arrays layout is supported for conversion to DataFrame object in pandas

4

- Filter the DataFrame to only include Falcon 9 launches by excluding ‘Falcon 1’
- Replace NaN values of PayloadMass with the mean of PayloadMass

1

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.
```

```
response = requests.get(static_json_url)
data = pd.json_normalize(response.json())
```

2

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
BoosterVersion[0:5]
```

```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falc
```

we can apply the rest of the functions here:

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

```
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}
```

3

```
df = pd.DataFrame.from_dict(launch_dict)
```

4

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

```
data_falcon9 = data_falcon9.fillna(value={'PayloadMass': data_falcon9['PayloadMass'].mean()})

data_falcon9['PayloadMass'].isnull().sum()
```

RestAPI@GitHub – Click Me

# DATA COLLECTION – WEB SCRAPING

Web scraping to collect Data on Falcon 9 launches from Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.

1
- Request the HTML format Wikipedia page from the static URL
- Store response in variable data

2
- Format response to BeautifulSoup object for manipulation
- Find all tables in our “soup-object”

3
- Make a list of all column header names that we can apply to future DataFrame

4
- Use the column names as keys in a dictionary

5
- Transform the dictionary to a Pandas DataFrame

1
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
data = response.text
```

2
```
soup = BeautifulSoup(data, 'lxml')
html_tables = soup.find_all('table')
```

3
```
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name is not None and len(name) > 0):
        column_names.append(name)
```

4
```
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'] = []
```

5
```
df=pd.DataFrame(launch_dict)
```

Webscrp@GitHub – Click Me

# DATA MANIPULATION/WRANGLING – PANDAS

Intro:

- Dataset from SpaceX contains varying launch sites, and their names are stored in our `LaunchSite` column.
- Launches are set to path a certain orbit, we display the different values and number of occurrences in picture 2. Orbit value is stored in the `Orbit` column.

Data Exploration:

Apply `.value_counts()` to examine:

1. Amount of launches from each location
2. Name and frequency of orbits
3. Landing outcomes, types and once again occurrence

1

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

```
CCSFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

2

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

```
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
```

3

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

```
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
```

Wrangling@GitHub – ClickMe

# DATA MANIPULATION/WRANGLING – PANDAS

Context:

- The landing `Outcome` will be assigned 1 for success and 0 for failure:
  - `True Ocean` – 1
  - `False Ocean` – 0
  - `True RTLS` – 1
  - `False RTLS` – 0
  - `True ASDS` – 1
  - `False ASDS` – 0
  - `None ASDS` and `None None` – 0

Data Wrangling:

- Boosters success is now represented by a Boolean value in our DataFrame, column is called Class for Classification.
- This is done by:
  1. Separating based of names and putting failure values in `bad_outcome`
  2. Creating a list, `landing_class` where we examine Outcome values and assign them 1 or 0 based of string comparison to our failure list.
  3. `Class` column is now derived and applied from list `landing_class`

1

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

```
False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'
```

2

```
# 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)
```

3

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

Wrangling@GitHub – ClickMe

# EXPLORATORY DATA ANALYSIS (EDA) – VISUALIZATION

## SCATTER CHARTS

We used these to portray relation between the following:

- Flight Number and Launch Site
- Payload and Launch Site
- Orbit Type and Flight Number
- Payload and Orbit Type

Scatter charts are best applied to visually explore the relation between numeric values.

## BAR CHART

We created a bar char to paint the relationship between Success Rate and Orbit.

Bar charts are best used to portray relation between categorical values and numerical ones.

## LINE CHARTS

Line charts were used to visualize the relation between Success Rate and Year.

Line charts are most commonly used to show the changing value of a numerical variable over time.

Visual@GitHub – ClickMe

# EXPLORATORY DATA ANALYSIS (EDA) – SQL

To extract information from our DataSet, we constructed SQL queries.

Some of the SQL queries used on the data set were meant to:

1. Retrieve unique names of launch sites found in the dataset
2. Show 5 records where launch sites begin with ‘CCA’
3. Return the total payload mass carried by boosters from by NASA (CRS)
4. Return an average value of payload mass carried by booster F9 v1.1
5. Tell the date when the first successful landing outcome of type (ground pad) was recorded
6. Return name values of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
7. Display a total of successful and failed missions
8. Tell the booster version value that had carried highest payload mass
9. From the year 2015, list failed landings on drone ships. Including their booster versions and launch sites
10. Rank descending the count of landing outcomes such as Failure (drone ship) and Success (ground pad) between the date 2010-06-04 and 2017-03-20

SQL@GitHub – ClickMe

# GEOSPATIAL ANALYSIS – FOLIUM

The following tasks were performed to visualize launch data on an interactive map:

## 1. Mark all launch sites on map

- Initialise the map using a Folium `Map` object.
- Add a `folium.Circle` and `folium.Marker` at each launch site on the launch map.

## 2. Mark the success and failed launches for each site on the map

- Many launches share site coordinates, therefore we cluster them on our map object.
- Prior to clustering, assign a marker colour of successful (green), and unsuccessful (red).
- Clustering the launches by for each adding a `folium.Marker` to the `MarkerCluster()` object.
- Create an icon as a text label, assigning the `icon_color` as the `marker_colour` determined previously.

## 3. Calculate distances between launch sites

- Proximities of launch sites are given by calculations of distances between points using the `Lat` and `Long` values.
- Take the `Lat` and `Long` values of two points, create a `folium.Marker` object to show the distance.
- Display line between two points, draw a `folium.PolyLine` and add to map.

Folium@GitHub – ClickMe

# INTERACTIVE DASHBOARD – PLOTLY DASH

We created a Plotly Dash with the following interactive charts and graphs:

1. Pie chart (`px.pie()`) depicting successful launches by sites
   - This visually gives instant confirmation on how successful each site is
   - The chart could also be filtered using a `dcc.Dropdown()` to display the success ratio for a single site

2. Scatter graph `px.scatter()` shows the correlation between outcome and payload mass
   - Mass can be filtered `RangeSlider()` to a certain range of mass
   - It can also be filtered via booster version

PlotDash@GitHub – ClickMe

# PREDICTIVE ANALYSIS - CLASSIFICATION

Here is what was done to evaluate the best performing classification model:

## Model Development

- Preparing given dataset for model interaction:
  - Retrieve dataset
  - Standardize, pre-process and transform dataset
  - Split data into training and test data: `train_test_split()`
  - Figure out suitable model candidates for this given task
- For each suitable algorithm:
  - Initiate a `GridSearchCV` object and a dictionary of parameters
  - Fit this object to parameters
  - Train the model with training set

## Model Evaluation

- For each algorithm:
  - Using the output GridSearchCV object:
    - Examine your hyperparameters (`best_params_`)
    - Find the accuracy (`score` and `best_score_`)
  - Plot Confusion Matrix to portray accuracy

## Finding the Best Classification Model

- Compare accuracy of all algorithms
- The models performance is based of its accuracy so the higher the better

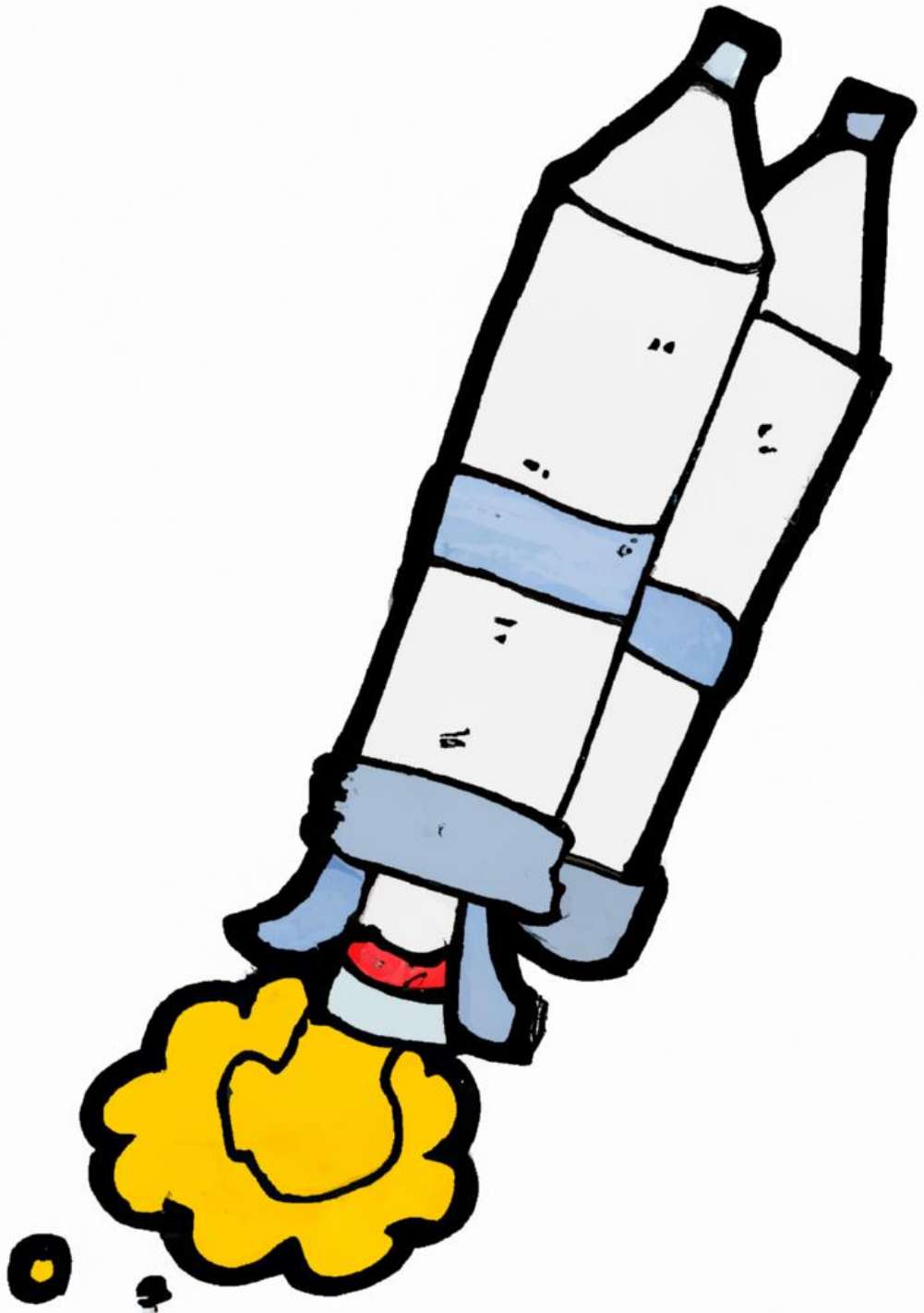
Predict@GitHub – ClickMe

# RESULTS

Exploratory Data Analysis

Interactive Analytics

Predictive Analysis

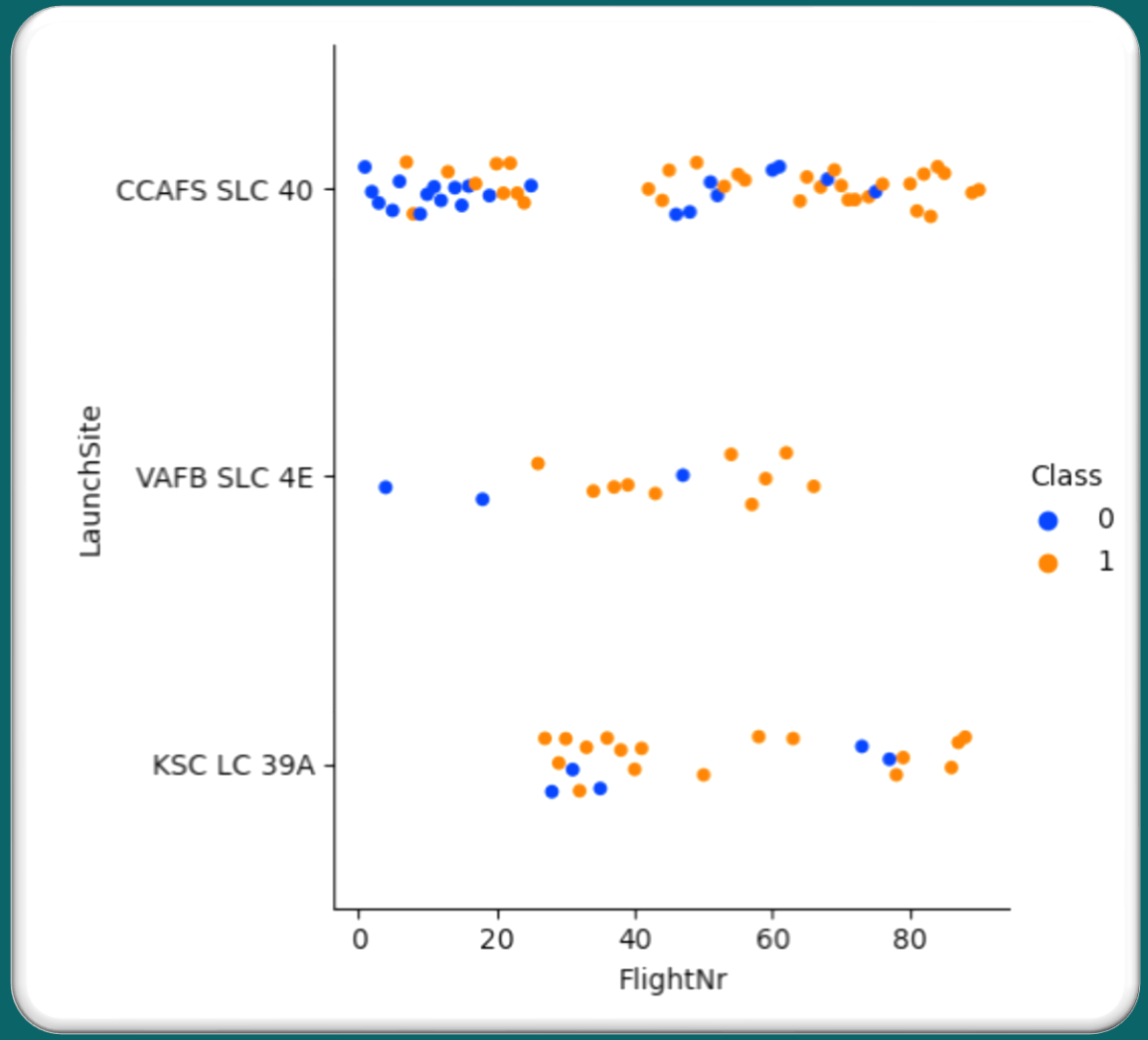


# EDA - WITH VISUALIZATION

# LAUNCH SITE VS. FLIGHT NUMBER

Scatter plot of Launch Site compared Flight Number:

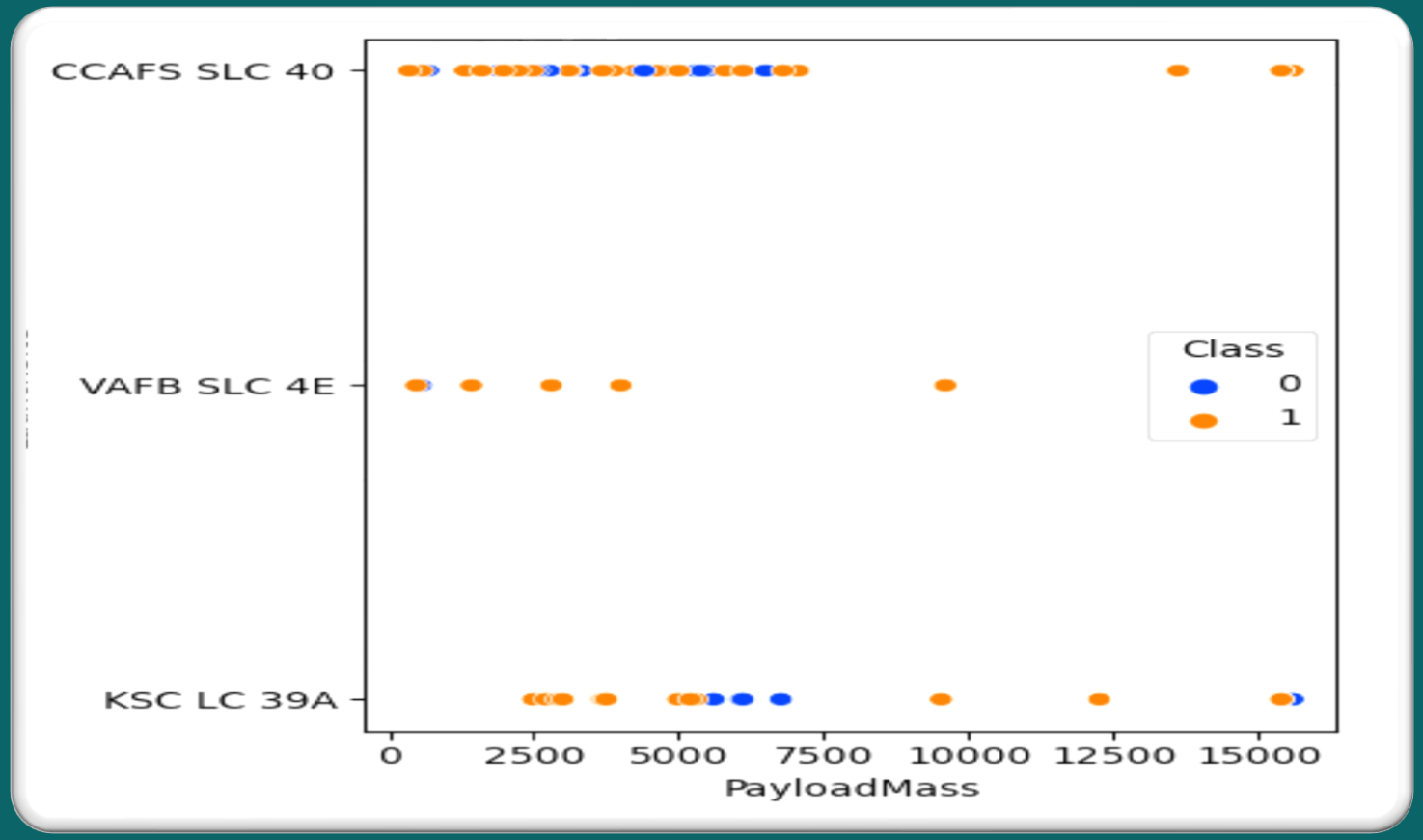
- More flights launched from a site leads to higher success rate overall for that site.
- CCAFS SLC 40 seem to have launched most of the early flights and the success rate is fairly low.
- VAFB SLC 4E shows the same pattern with early flights.
- No early flights were launched from KSC LC 39A, so the launches from this site are more successful.
- With more than 20-35 flights the success rates increase dramatically.



# LAUNCH SITE VS. PAYLOAD MASS

Scatter plot of Launch Site vs. Payload:

- Above 7000 kg, there are much fewer successful landings. Although we don't have much data for this category.
- The highest variance in payload mass can be found at KSC LC 39A.
- CCAFS SLC 40 generally carries lighter payloads with the exception of just a handful of launches.
- VAFB SLC 4E has a widespread variety of payload masses and it can also be noticed that it´s success rate suffers, perhaps from the high volatility of this particular variable.



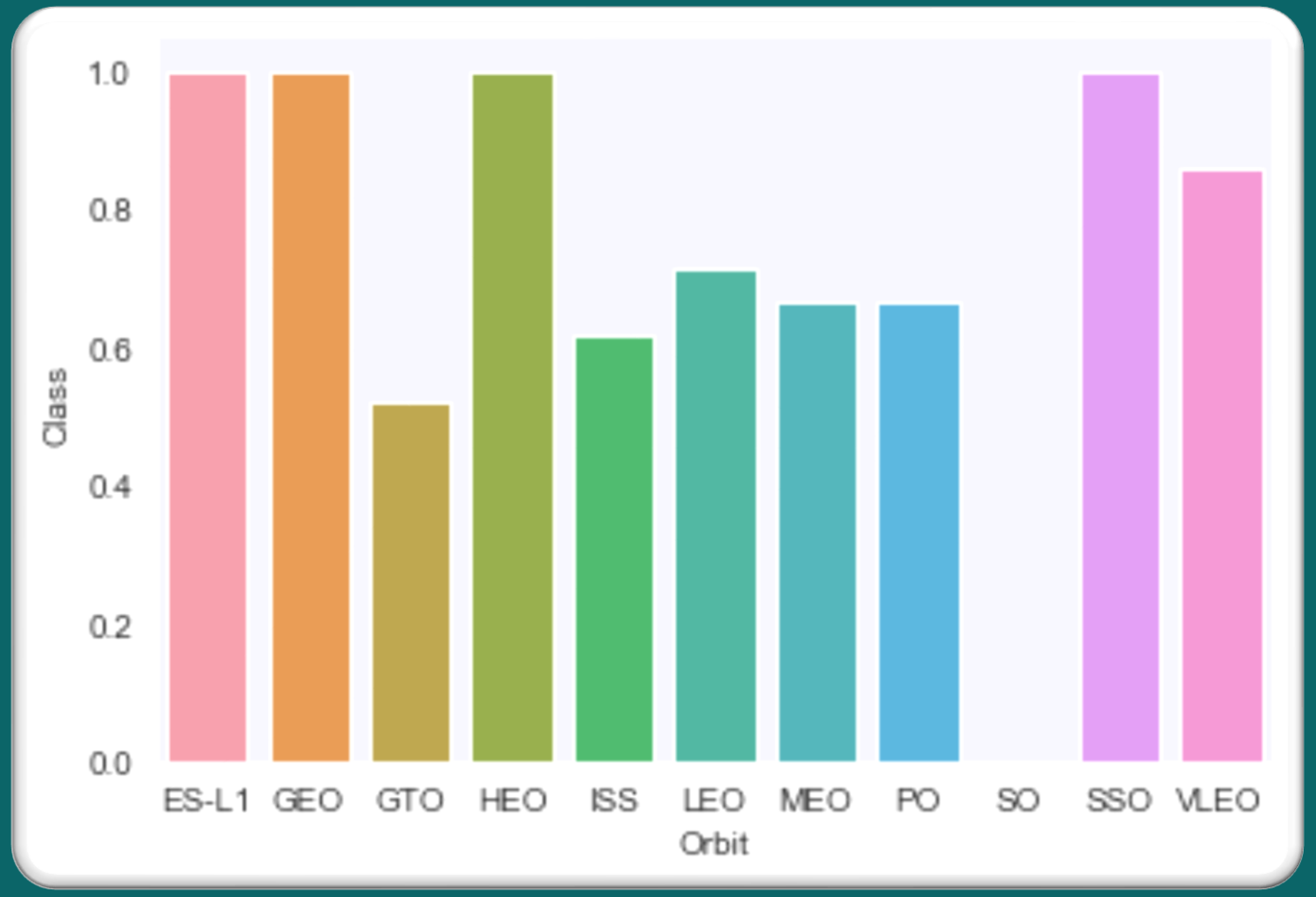
# SUCCESS RATE VS. ORBIT TYPE

Bar chart show that the following orbits have a 100% success rate:

- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)

One orbit has the opposite rate of success with a whopping 0%:

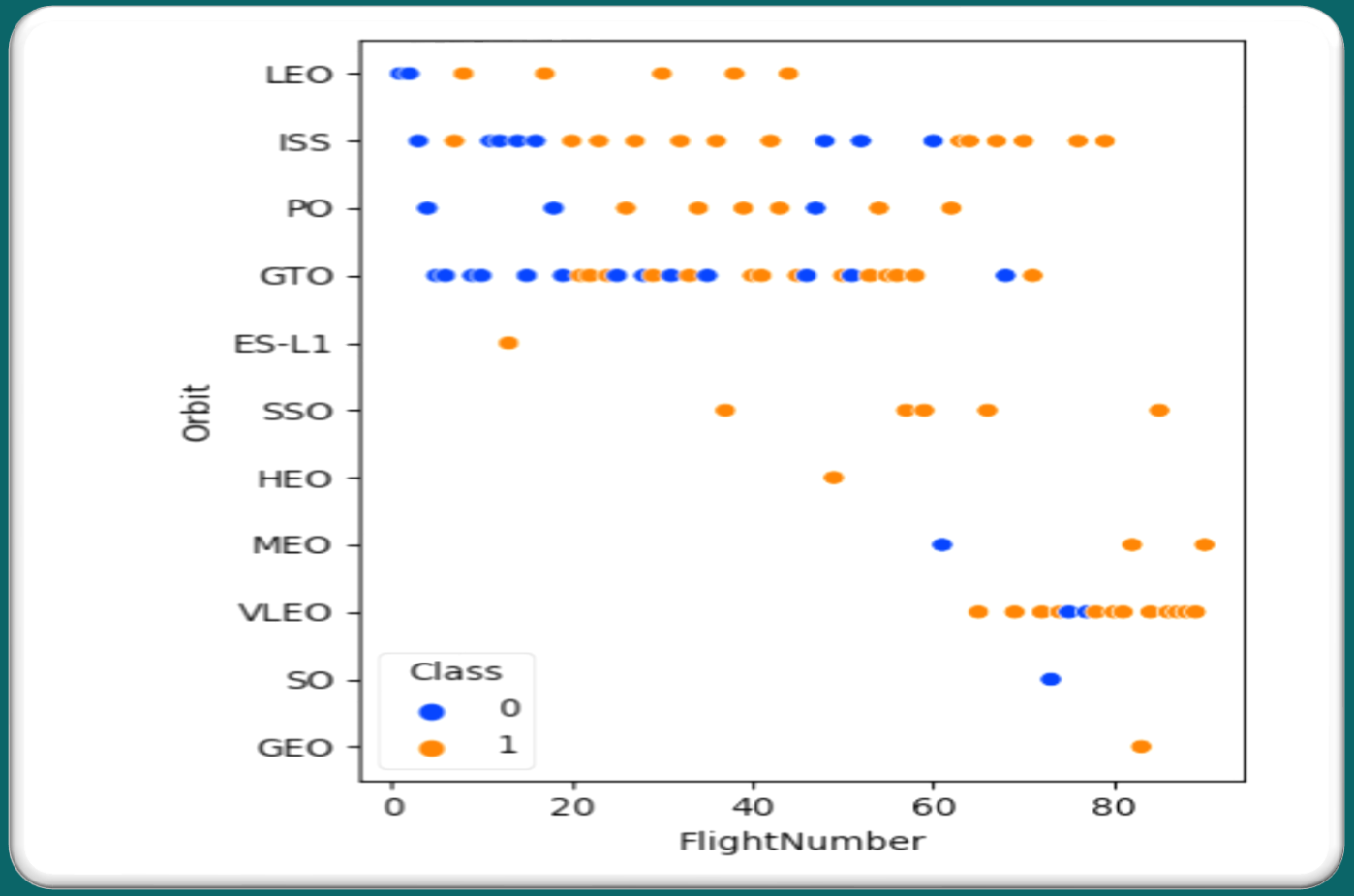
- SO (Heliocentric Orbit)



# ORBIT TYPE VS. FLIGHT NUMBER

This scatter plot derives some new conclusions and insight:

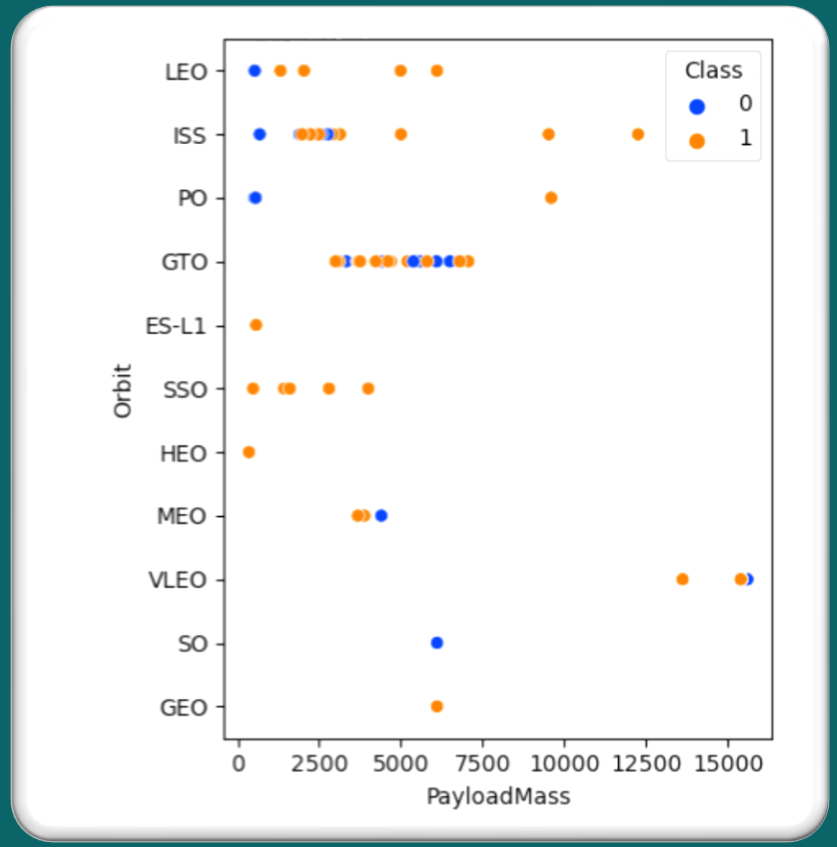
- The 100% success rates of certain orbits can be explained by the fact that they have only 1 or very few flights.
- SSO still maintains 100% despite already having 5 launches.
- GTO has not really improved over time as its success rate does not seem to follow a trend related to time or amount of flights.
- In other cases more commonly success rate increases by the factor of “practice makes perfect”.



# ORBIT TYPE VS. PAYLOAD MASS

This scatter plot gives that:

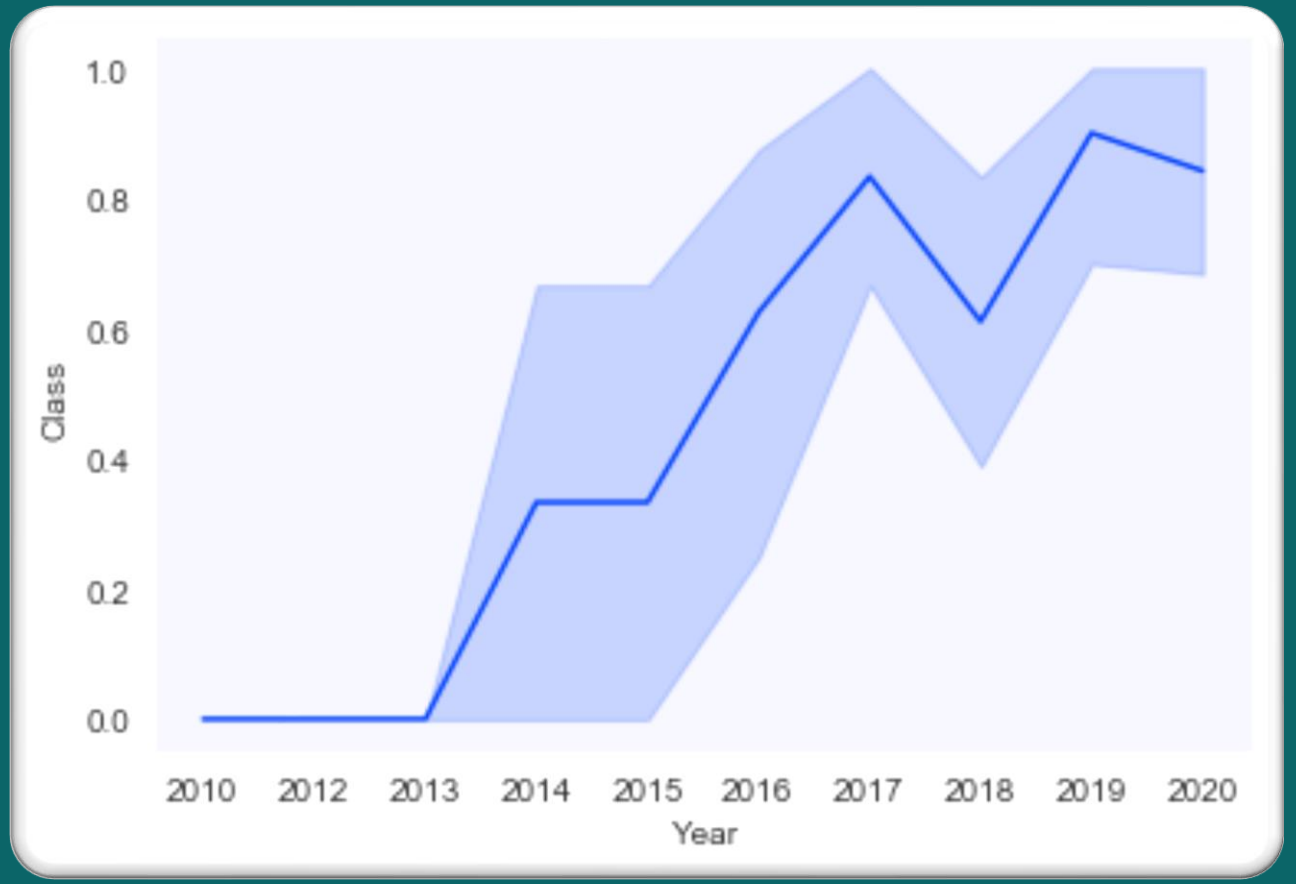
- The following orbits are more successful with heavier loads:
  - PO
  - ISS
  - LEO
- VLEO (Very Low Earth Orbit) launches are typically have heavier payloads.



# LAUNCH SUCCESS YEARLY TREND

The line chart depicting yearly average success tells:

- 2010 to 2013, yielded a success rate of 0%
- After 2013, success rate increased in a semi steady pattern as it decreases in 2018 and again 2020.
- As of 2016 the chance of a successful launch is in our favour as it surpasses 50%.



# EDA - WITH SQL

# ALL LAUNCH SITE NAMES

Find the names of the unique launch sites.

```
%%sql
SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
```

Python

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

The word UNIQUE returns only unique values from the LAUNCH_SITE column of the SPACEXTBL table.

# LAUNCH SITE NAMES BEGIN WITH 'CCA'

Find 5 records where launch sites begin with ‘CCA’.

```
%%sql
SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

Python

| C) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome |
|---|---|---|---|---|---|---|---|
| 00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success |
| 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 |
| 00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success |
| 00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success |
| 00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success |

`LIMIT 5` retrieves 5 records, and `LIKE` is used with `‘CCA%’` to get string values starting with ‘CCA’.

# TOTAL PAYLOAD MASS

Calculate the total payload carried by boosters from NASA.

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL
WHERE CUSTOMER = 'NASA (CRS)';
```

Python

| total_payload_mass |
| --- |
| 45596 |

SUM keyword is used to calculate the total numeric value of the LAUNCH column, and WHERE condition filters the results boosters from NASA (CRS).

# AVERAGE PAYLOAD MASS BY F9 V1.1

Calculate the average payload mass carried by booster version F9 v1.1.

```
%%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

Python

| AVERAGE_PAYLOAD_MASS |
| --- |
| 2928.4 |

AVG is used to calculate the average of the PAYLOAD_MASS__KG_ column, and the WHERE filters the results to the F9 v1.1 booster version.

# FIRST SUCCESSFUL GROUND LANDING DATE

Find the dates of the first successful landing outcome on ground pad.

```
%%sql SELECT MIN(Date) AS FIRST_SUCCESSFUL_GROUND_LANDING FROM SPACEXTBL
WHERE `Landing _Outcome` = 'Success (ground pad)';
```

Python

| first_successful_ground_landing |
| --- |
| 2015-12-22 |

MIN is used to calculate the minimum or earliest value of DATE column, WHERE filters the results to only successful ground pad landings.

# SUCCESSFUL DRONE SHIP LANDING WITH PAYLOAD BETWEEN 4000 AND 6000

List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000.

```python
%%sql SELECT BOOSTER_VERSION FROM SPACEXTBL
WHERE (`Landing _Outcome` = 'Success (drone ship)')
AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000);
```

| booster_version |
| --- |
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

`WHERE` to filter the results to include only those that satisfy both conditions. The `BETWEEN` keyword ensures only payloads between numeric values 4000 and 6000 are retrieved.

# TOTAL NUMBER OF SUCCESSFUL AND FAILURE MISSION OUTCOMES

Calculate the total number of successful and failure mission outcome.

```
%%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

Python

| Mission_Outcome | TOTAL_NUMBER |
|---|---|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

COUNT is used to calculate the total number of mission outcomes, GROUPBY is used to group these results by value of mission outcome.

# BOOSTERS CARRIED MAXIMUM PAYLOAD

List the names of the booster which have carried the maximum payload mass.

```
%%sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

Python

| 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 |

`SELECT` statement on second row finds the maximum value of payload, and this value is applied in our `WHERE` condition. `DISTINCT` keyword is to retrieve only unique booster versions values.

# 2015 LAUNCH RECORDS

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015. And list the month “names”. (These will be listed as numeric month names).

```
%%sql
SELECT BOOSTER_VERSION, LAUNCH_SITE, substr(Date,4,2) AS MONTH FROM SPACEXTBL
WHERE (`Landing _Outcome` = 'Failure (drone ship)') AND substr(Date,7,4)='2015';
```

Python

| Booster_Version | Launch_Site | MONTH |
|---|---|---|
| F9 v1.1 B1012 | CCAFS LC-40 | 01 |
| F9 v1.1 B1015 | CCAFS LC-40 | 04 |

`WHERE` to filter the results for only failed landings, `AND` only for the year of 2015.
`substr(Date,4,2)` is to also get the column value of month.

# RANK LANDING OUTCOMES BETWEEN 2010-06-04 AND 2017-03-20

Rank the count of successful landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order.

```
%%sql
SELECT `Landing _Outcome`, COUNT(`Landing _Outcome`) AS NUMBER FROM SPACEXTBL
WHERE `Landing _Outcome` LIKE 'Success%' AND DATE BETWEEN'20-03-2010
ORDER BY NUMBER DESC;
```

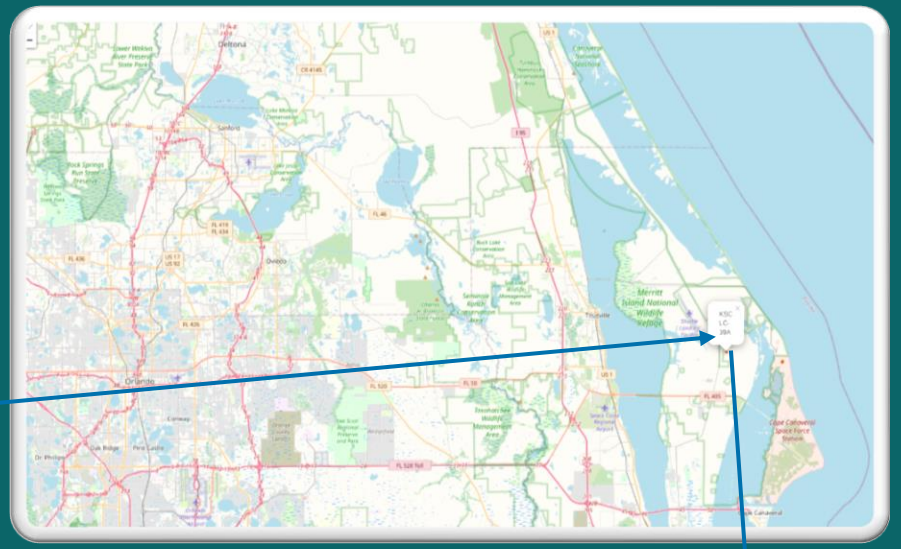
Python

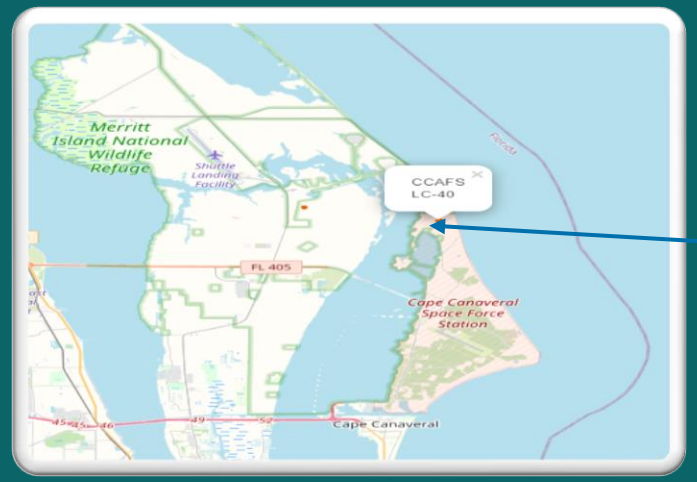
| Landing _Outcome | NUMBER |
|---|---|
| Success | 21 |
| Success (drone ship) | 8 |
| Success (ground pad) | 6 |

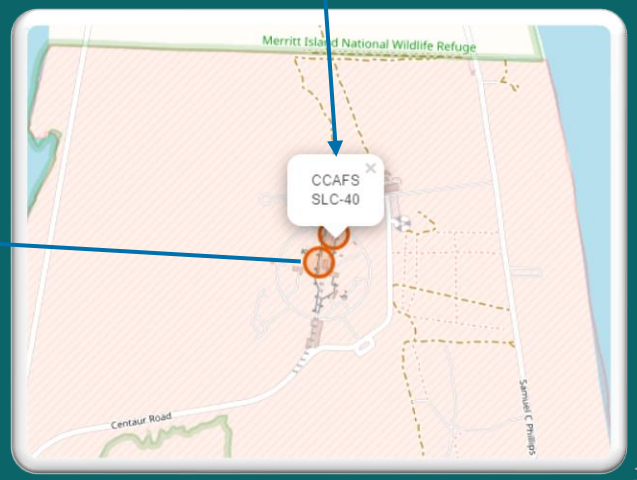
`WHERE` is used with the `BETWEEN` keyword to filter by dates given. The results are grouped and ordered by `GROUP BY` and `ORDER BY` where `DESC` causes display in descending order.

# LAUNCH SITES PROXIMITY ANALYSIS – FOLIUM INTERACTIVE MAP

# ALL LAUNCH SITES ON A MAP

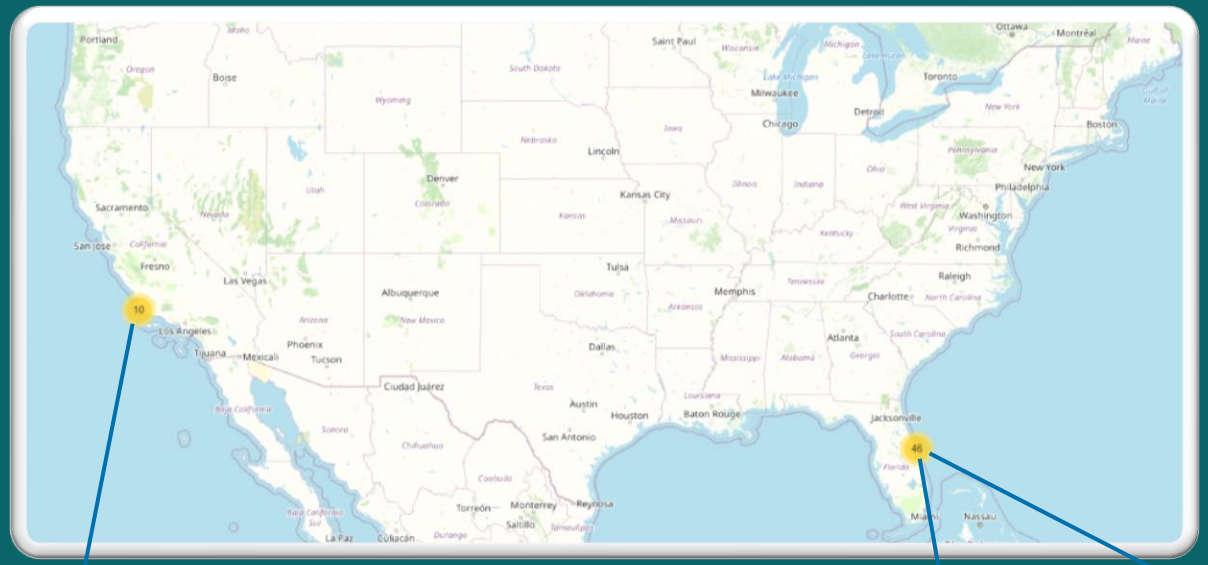






SpaceX launch sites are located by the coast, we can see them being clustered in Florida and California.

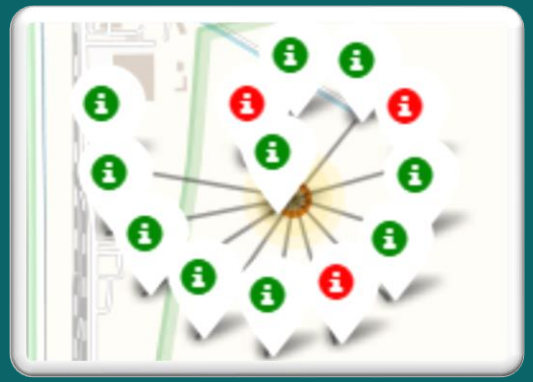
# SUCCESSFUL AND FAILED LAUNCHES AT EACH SITE



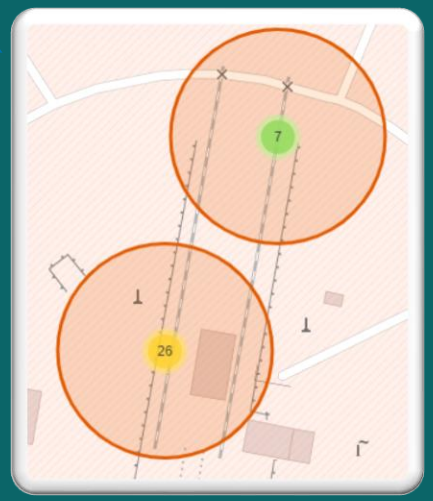

Launches that share a site have been clustered like the images below. The green icons are successful launches, and red icons represent failed launches.

VAFB SLC-4E

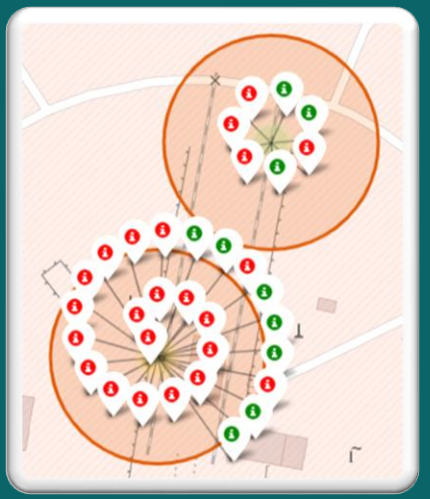
KSC LC-39A



CCAFS SLC 40




CCAFS LC 40



# PROXIMITY OF LAUNCH SITES TO VARIOUS POINTS

Using the CCAFS SLC 40 site for example, we can resonate regarding the selective locations of these sites.


Is the site close to a coastline?

- YES. The coastline is only 0.87 km due East.

Is the site close to a highway?

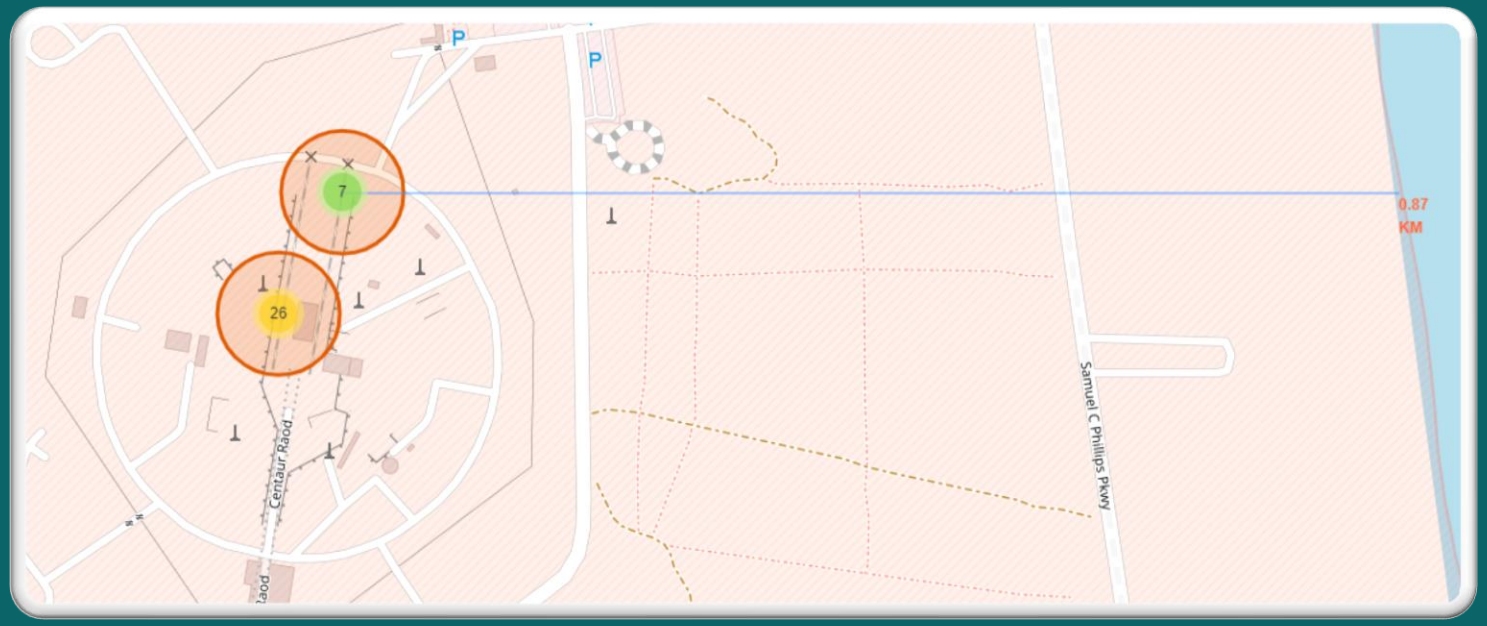
- YES. The nearest highway is only 0.59km away.

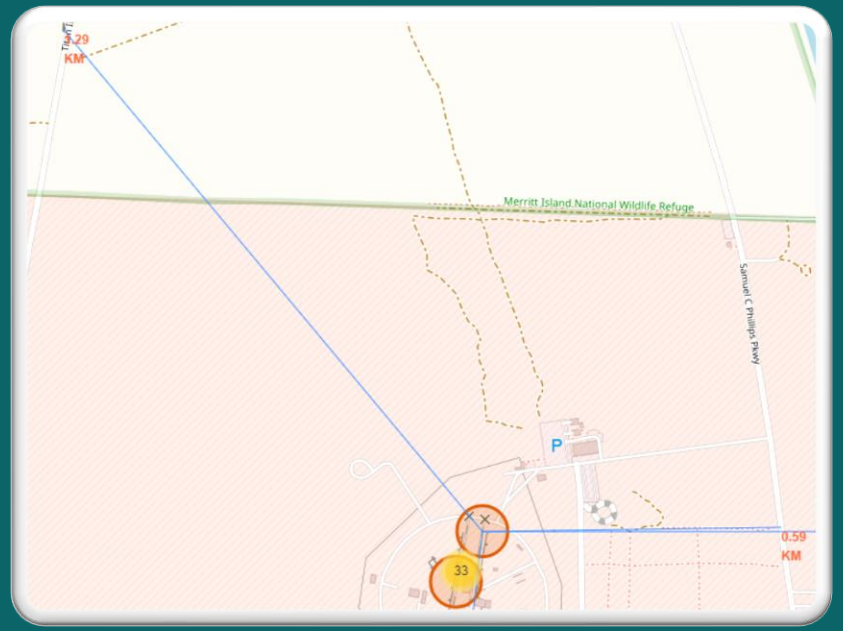
Is it near a railway?

- YES. The nearest railway is only 1.29 km away.

Is it relatively far from a city?

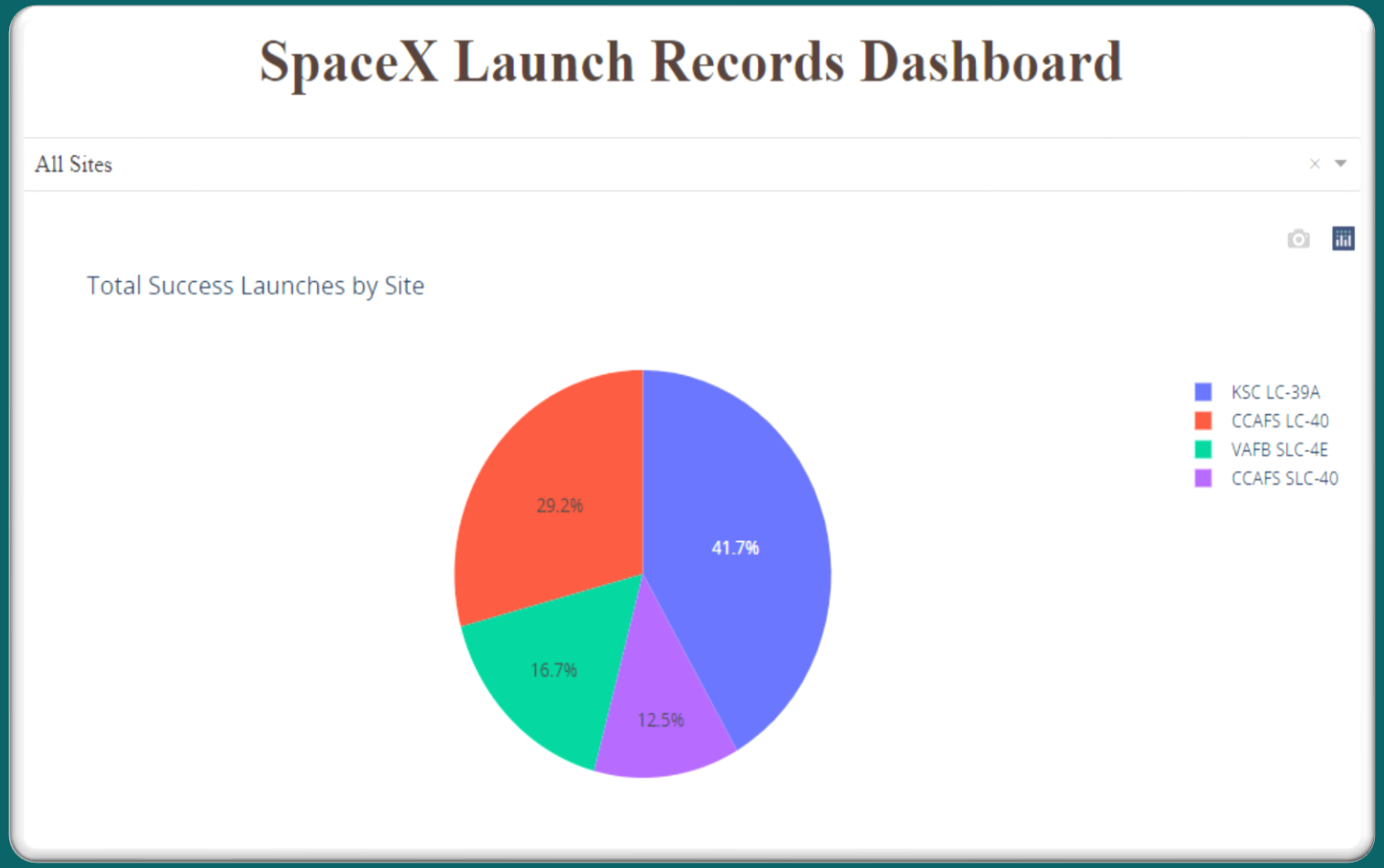
- YES. The nearest city is 51.74 km away.



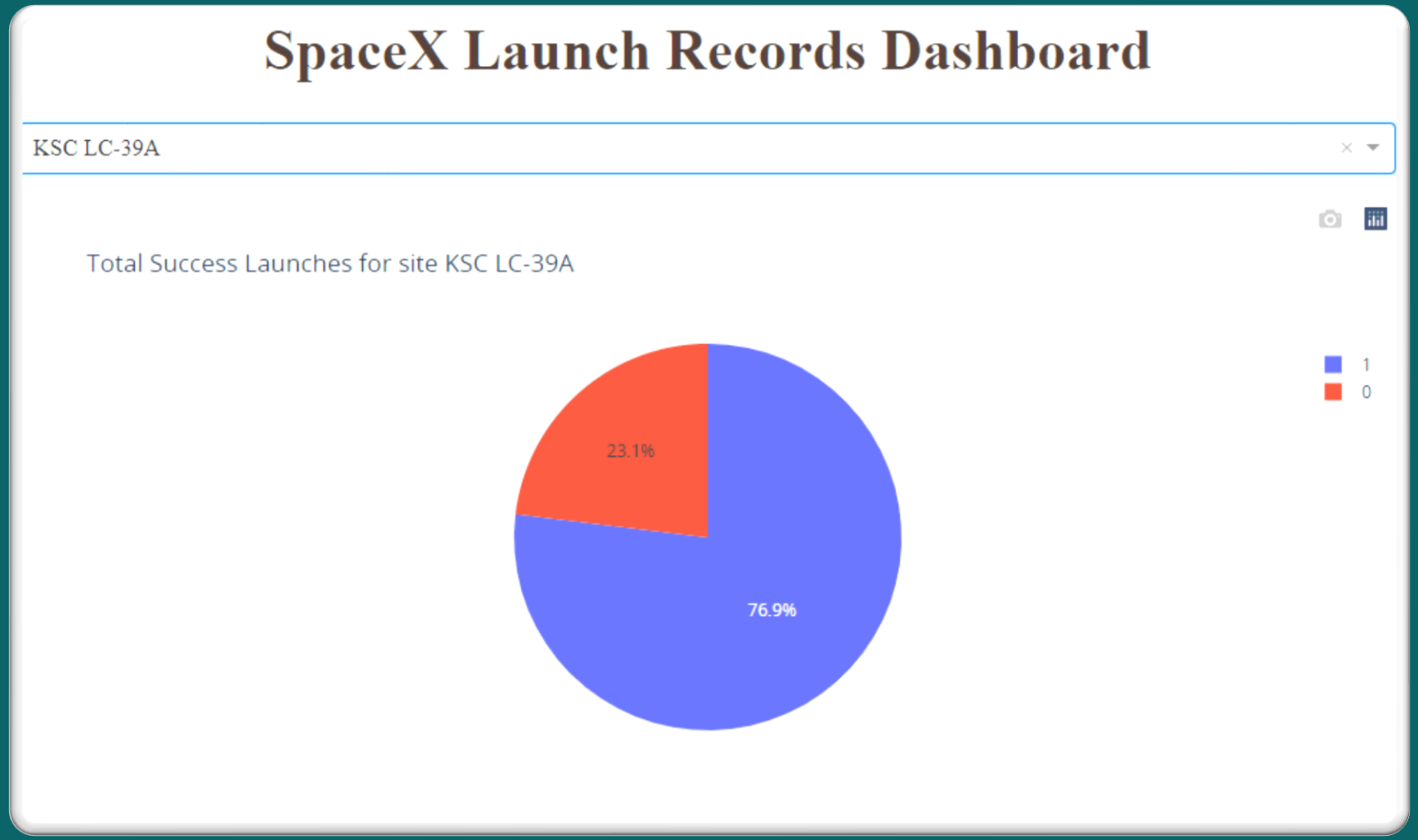
# INTERACTIVE DASHBOARD - PLOTLY DASH

# LAUNCH SUCCESS COUNT FOR ALL SITES




Site KSC LC 39 A had most successful launches, with 41.7% out of all successful launches.

# PIE CHART FOR THE LAUNCH SITE WITH HIGHEST LAUNCH SUCCESS RATIO




The launch site KSC LC 39 A was also most successful in by comparison of percentage statistics as an isolated entity, with a 76.9% success rate.

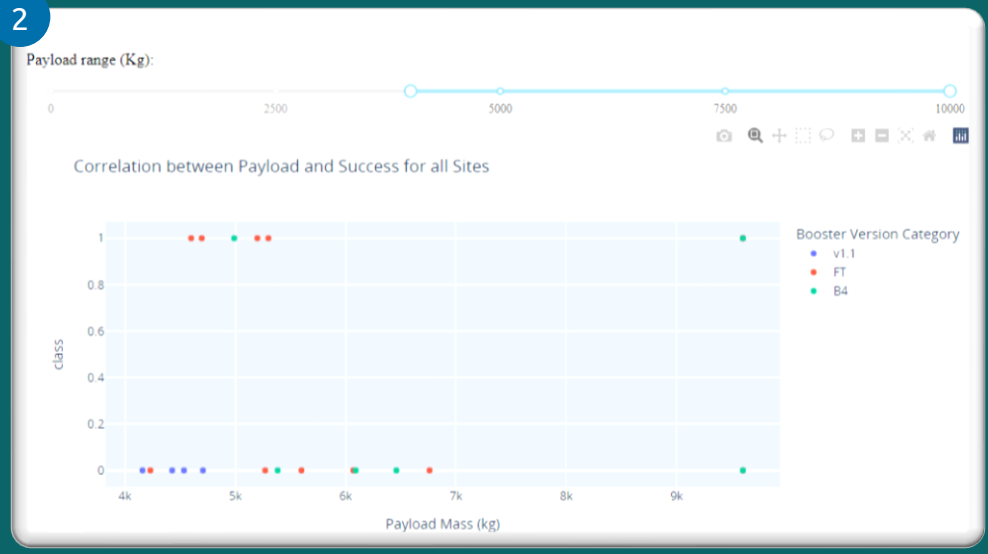
Note: $class \begin{cases} 0, Failure \\ 1, Success \end{cases}$

# LAUNCH OUTCOME TO PAYLOAD SCATTER PLOT: ALL SITES


- Plotting the launch outcome with regard to payload for all sites depicts a gap around 4000 kg, so we split the data into two categories as to not compare apples and pears:
  - 0 – 4000 kg (Lighter)
  - 4000 – 10000 kg (Heavier)
- The two plots above shows that the success for heavy payloads is lower than that for light payloads.

1


2




Note: $class \begin{cases} 0, Failure \\ 1, Success \end{cases}$

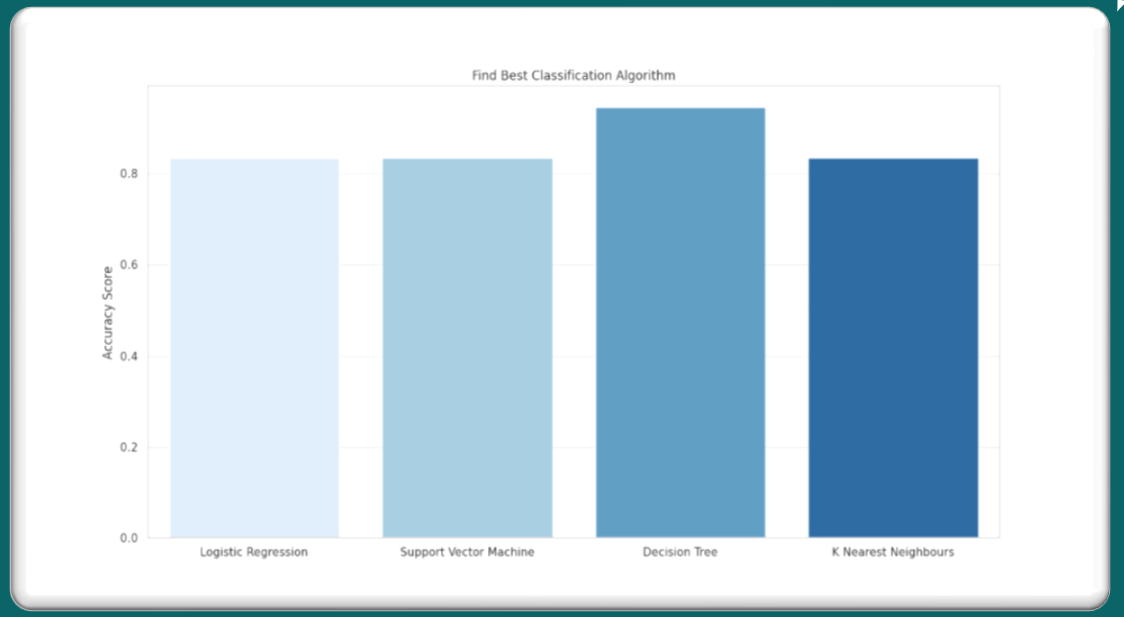
# PREDICTIVE ANALYSIS - CLASSIFICATION

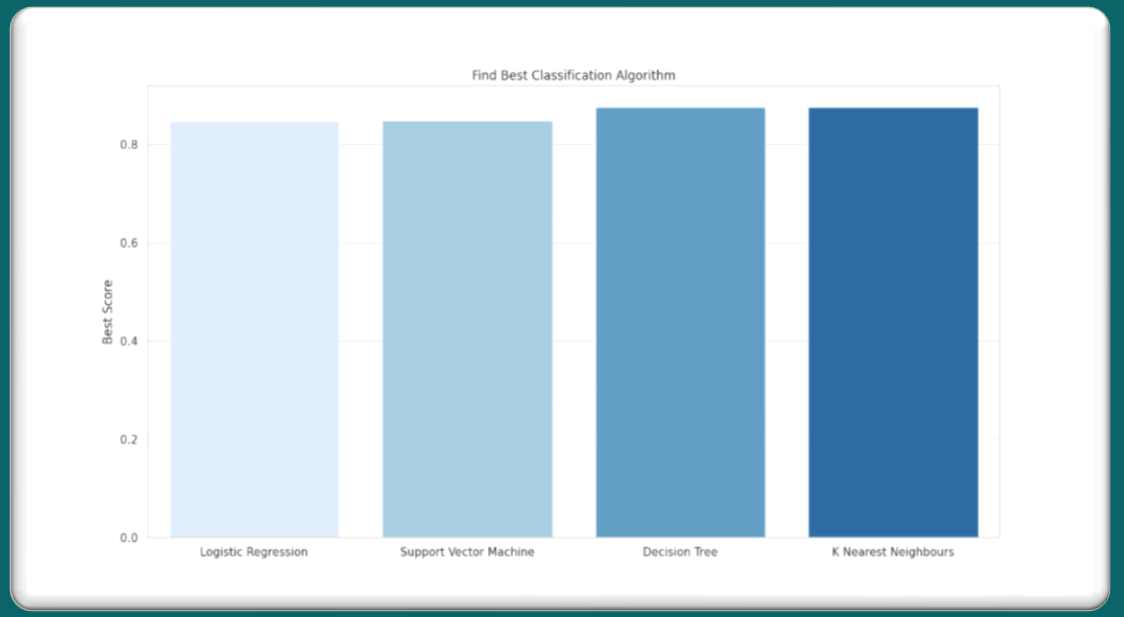
# CLASSIFICATION ACCURACY

Bar chart of Accuracy and Best Score for each algorithm produces leads to conclusion:

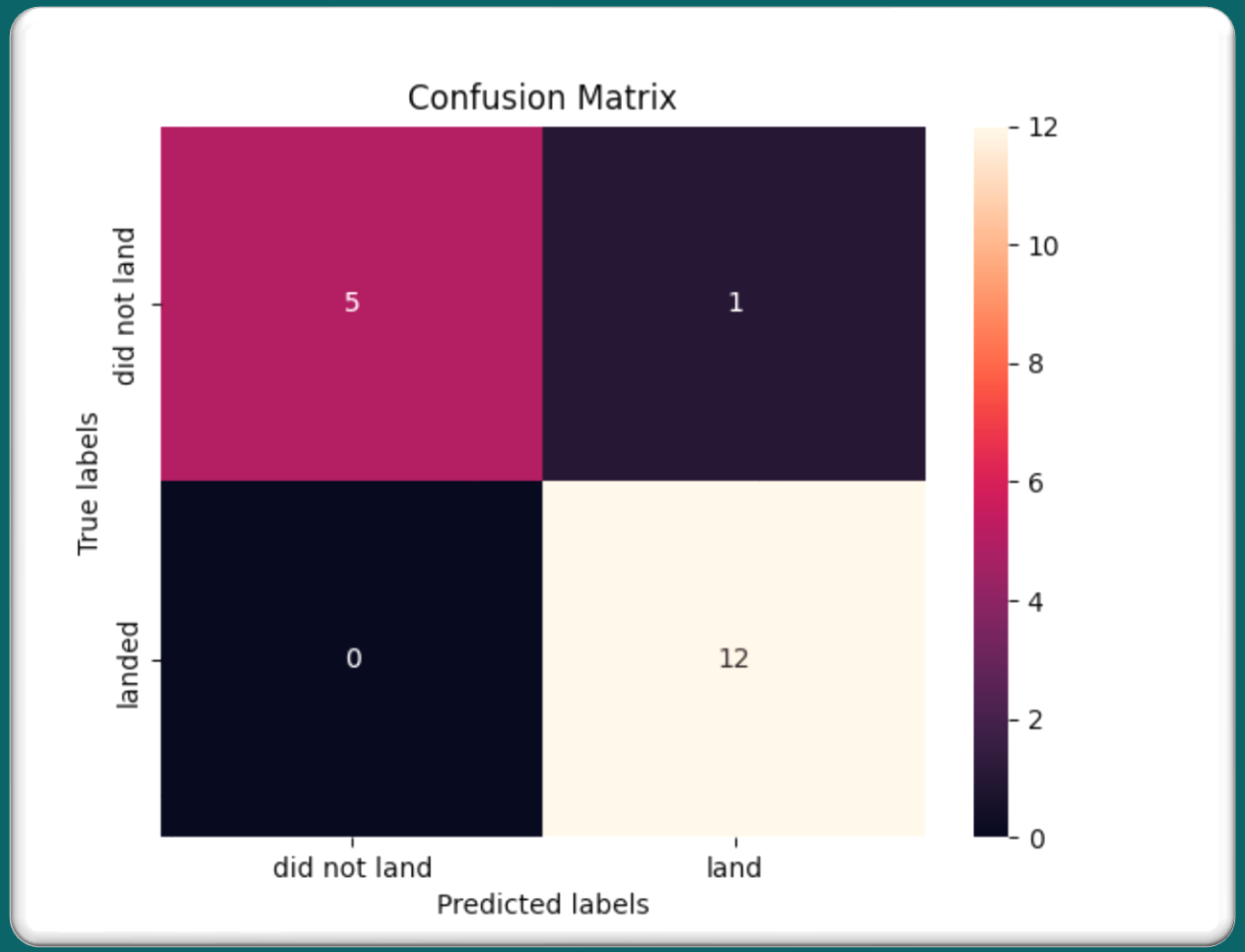
- Decision Tree model has the highest accuracy of the four
  - The Accuracy Score is 94.4…%
  - The Best Score is 87.5%

| | Model | Accuracy | Best Score |
|---|---|---|---|
| 0 | Logistic Regression | 0.833333 | 0.846429 |
| 1 | Support Vector Machine | 0.833333 | 0.848214 |
| 2 | Decision Tree | 0.944444 | 0.875000 |
| 3 | K Nearest Neighbours | 0.833333 | 0.875000 |



# CONFUSION MATRIX

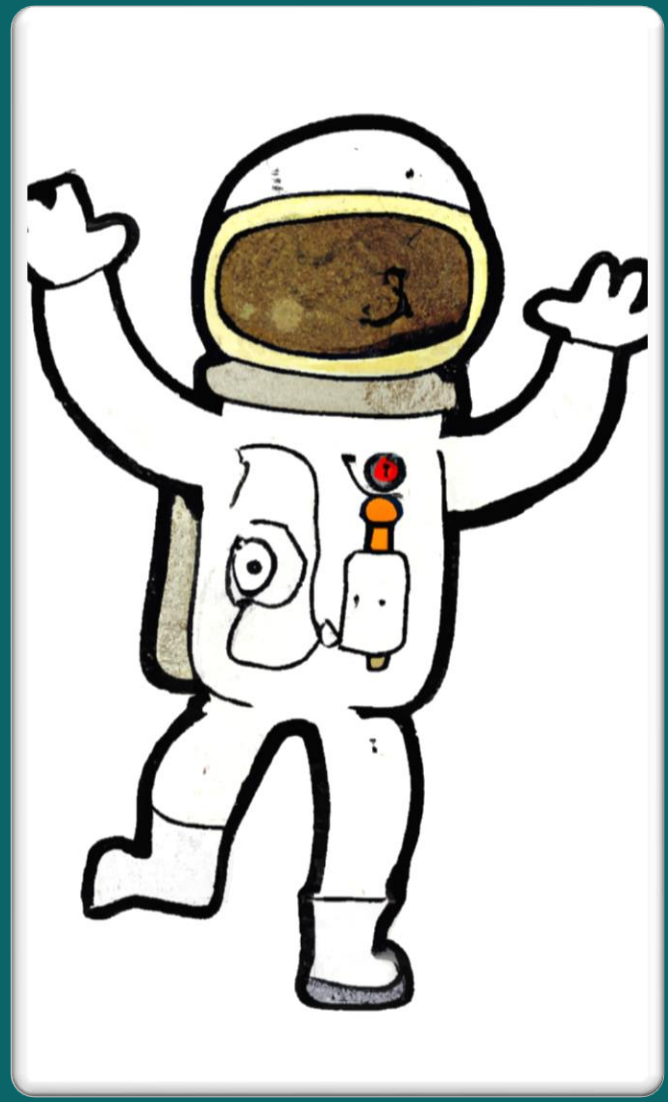



- So: the best preforming model is the Decision Tree, with an accuracy of 94.44%.
- The confusion matrix allows us to explain this further by seeing that only 1 prediction was a “miss” or “false positive” in our case.
- All other predictions are correct as 12 landed and 5 did not.

# CONCLUSION

# CONCLUSION

- The more flights attempted the higher success rate a given launch site seem to achieve, most early flights (lower flight numbers) are unsuccessful but we have derived a clear upwards trend as trial and error works in our favour.
  - 2010 to 2013, all landings failed.
  - After 2013, the success rate increased and maintains a growth trend to this day.
  - As of 2016 the “statistical odds” speaks to a likelihood of successful launches.
- Orbits ES-L1, GEO, HEO, and SSO have a success rate of 100%.
  - Success rates of GEO, HEO, and ES-L1 orbits can be explained by having 1 or very few flights.
  - The 100% success rate in SSO is more impressive, SSO has a total 5 successful flights.
  - Orbits PO, ISS, and LEO, are more successful with heavier payloads:
- The launch site KSC LC 39 A is most successful with 41.7% of the total successful launches and a 76.9% success rate on its own.
- The best performing classification model is the Decision Tree, with an accuracy of 94.4…%.



# APPENDIX

# RESOURCES AND REFERENCE – EMBEDDED LINKS

- Firstly, all material used in this PowerPoint can be found on GitHub-Capstone
- For the RestAPI Data collection refer to API-Data
- Regarding webscraping the entire notebook with all relevant information is at WebScrape
- Full project for data wrangling can be found at DataWrangling
- All SQL and database related content is performed in file: SQL
- For the EDA Visual section I refer to EDA-Visual
- Folium related content can be found at Folium
- Python script for the Plotly Dashboard can be found at PlotlyDash
- Material related to prediction and classification models PredictClass
- Lastly, all data files related to this project can be found in respective folders at GitHub.