Winning Space Race with Data Science

Giuliano Grigolin
2023-02

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - Exploratory Data Analysis (EDA) with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

Project background and context

Predict if the Falcon 9 first stage will land successfully. 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

- What influences if the rocket will land successfully?
- The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing.
- What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

Section 1

Methodology

Methodology

- Data collection
  - collect data on the Falcon 9 first-stage landings, at the use a RESTful API and web scraping.
- Data wrangling
  - review some of the attributes, converting 8 values of outcomes to classes y (0 = bad outcome, or 1 = good outcome), for each launch.
- Exploratory data analysis (EDA) using visualization and SQL
  - perform some Exploratory Data Analysis using a database.
  - perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - use of the machine learning to determine if the first stage of Falcon 9 will land successfully, spliting data into training and test data to find the best Hyperparameter for SVM, classification trees, and logistic regression. Then find the method that performs best using test data.

Data Collection

Github - Data Collection

Work with SpaceX REST API, that delivery data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

The goal was predict whether SpaceX will attempt to land a rocket or not.

Data Collection – SpaceX API

1- Define a series of helper functions to extract information using identification numbers in the launch data;

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
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'])
```

```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
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'])
```

2- Request rocket launch data from SpaceX API with the URL;

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

3- Decode the response content as a JSON using *".json()"* and turn it into a Pandas dataframe using *".json_normalize()"*

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

4- Filter the dataframe to only include `Falcon 9` launches

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[df['BoosterVersion'] == 'Falcon 9']
data_falcon9
```

Data Collection – SpaceX API

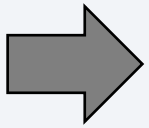
Dataframe with only Falcon 9 launches

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561857
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561857
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829	34.632093
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561857
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89	102	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1060	-80.603956	28.608058
90	103	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	13	B1058	-80.603956	28.608058
91	104	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1051	-80.603956	28.608058
92	105	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	12	B1060	-80.577366	28.561857
93	106	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	8	B1062	-80.577366	28.561857

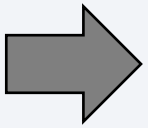
90 rows × 17 columns

Data Collection – SpaceX API

helper functions to extract data from SpaceX API



Decode the response as a normalized JSON dataframe



Clean and Filter the data to obtain Falcon 9 launches

Data Collection – Web Scraping

Github - Web Scraping

1- Define some helper functions to process web scraped HTML table;

```
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass
```

2- Example of web scrap Falcon 9 launch records with `BeautifulSoup`;

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(resp.text, "html.parser")
```

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

3- iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one;

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

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

Data Collection – Web Scraping

4- Parse the table and convert it into a Pandas data frame;

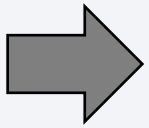
```
launch_dict= dict.fromkeys(column_names)

# 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- fill up the `launch_dict` with launch records extracted from table rows;

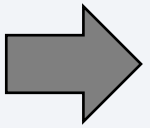
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if 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')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            datatimelist=date_time(row[0])
            launch_dict['Flight No'] ='flight_number'
            print(flight_number)


            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'] ='date'
            print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'] ='time'
            print(time)
```

Data Collection – Web Scraping

helper functions to process web scraped HTML table



web scrap Falcon 9 launch records with `BeautifulSoup`



Parse the table into a Pandas data frame

Data Wrangling - Introduction

There are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident:

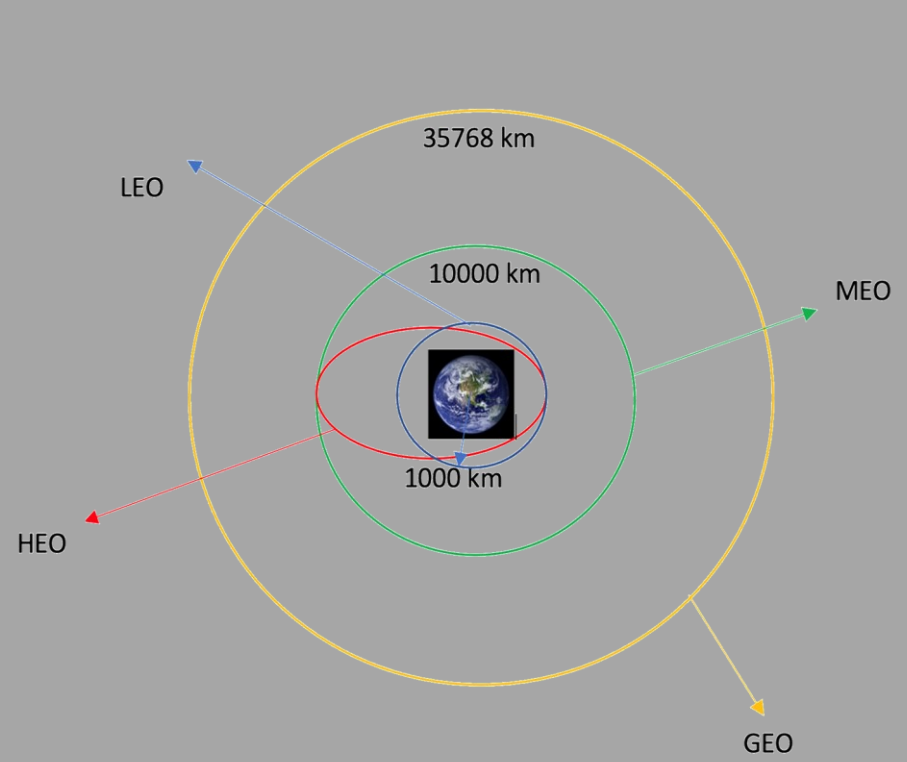
- True Ocean means the mission outcome was successfully landed to a specific region of the ocean;
- False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean;
- True RTLS means the mission outcome was successfully landed to a ground pad;
- False RTLS means the mission outcome was unsuccessfully landed to a ground pad;
- True ASDS means the mission outcome was successfully landed on a drone ship;
- False ASDS means the mission outcome was unsuccessfully landed on a drone ship.

The goal of this step was convert those outcomes into Training Labels:

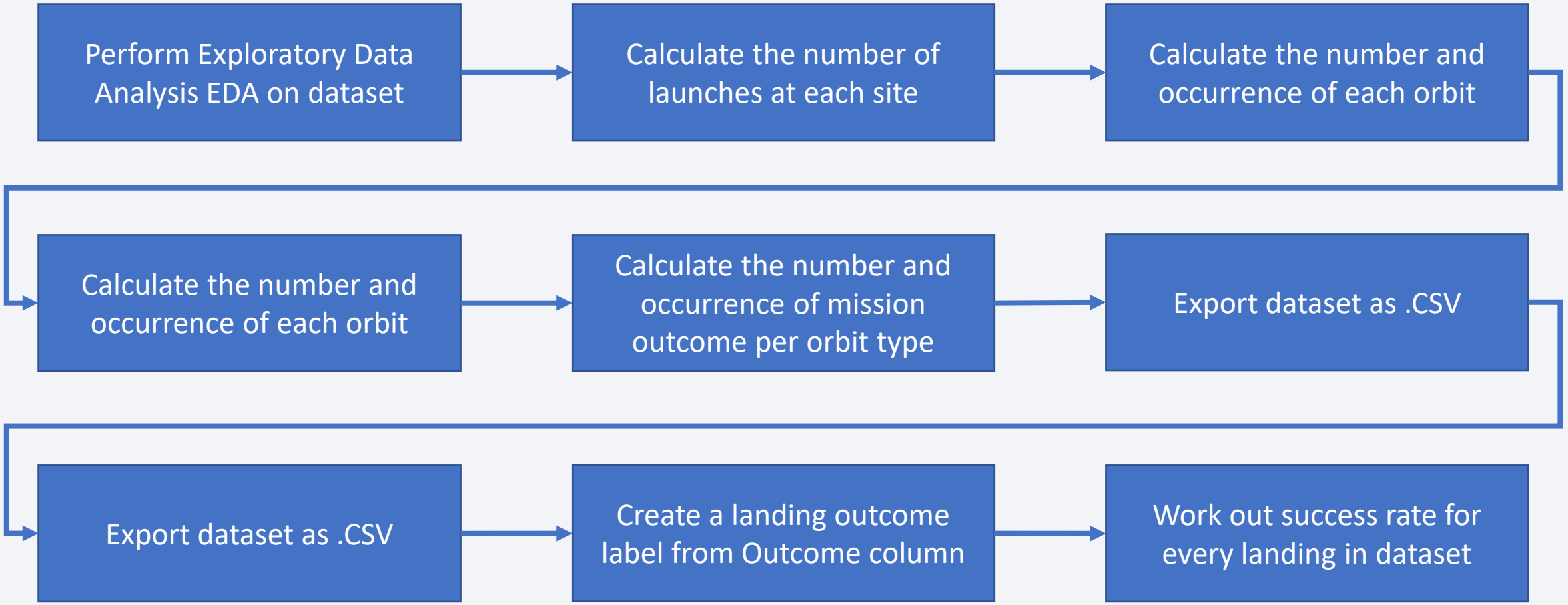
- 1 means the booster successfully landed and;
- 0 means it was unsuccessful.

Each launch aims to an dedicated orbit.

In the figure below we see some common orbit types:



Data Wrangling - Process



Data Wrangling - Process

Github - Data Wrangling

1- Get Response form .csv file:

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
```

2- Calculate the number of launches on each site :

```
df['LaunchSite'].value_counts()
```

3- Calculate the number and occurrence of each orbit

```
df['Orbit'].value_counts()
```

4- Calculate the number and occurrence of mission outcoming per orbit type

```
landing_outcomes = df['Outcome'].value_counts()
```

5- Create a landing outcome label from outcome column

```
dict={}
for i in set(df['Outcome']):
    if i in bad_outcomes:
        dict[i]=0
    else:
        dict[i]=1
#print(dict)

landing_Class=df['Outcome'].replace(dict, inplace = False)
```

EDA with Data Visualization

Github - EDA Visualization

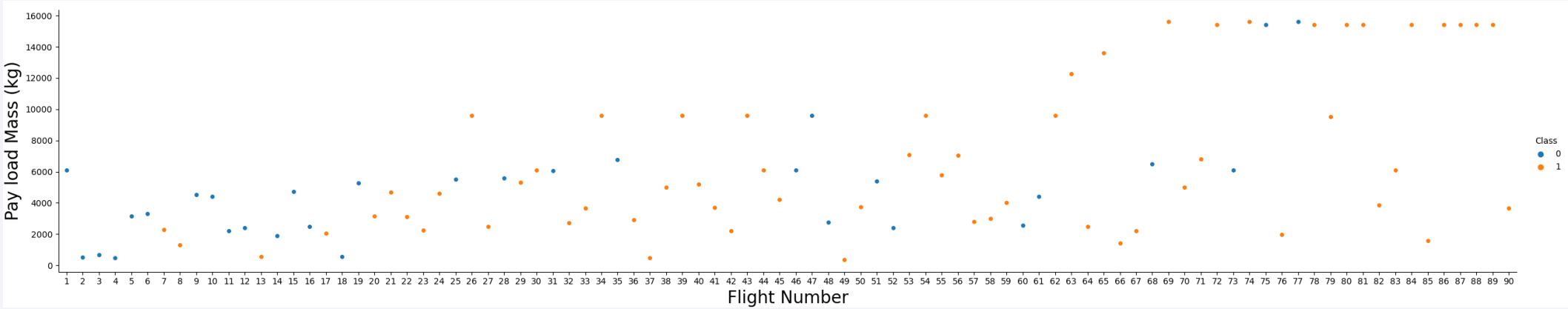
Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation.

Scatter Graphs being drawn:

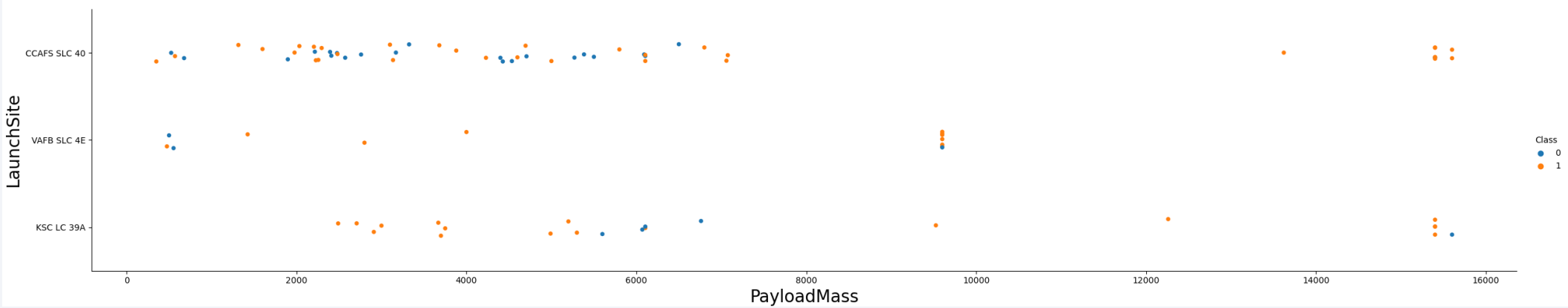
- Flight Number 🔀 Payload Mass
- Flight Number 🔀 Launch Site
- Payload 🔀 Launch Site
- Orbit 🔀 Flight Number
- Payload 🔀 Orbit Type
- Orbit 🔀 Payload Mass

EDA with Data Visualization

Flight Number  Payload Mass

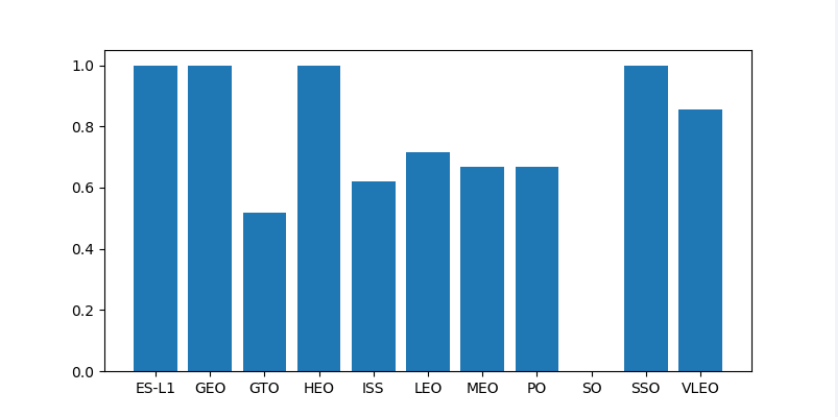


Flight Number  Launch Site

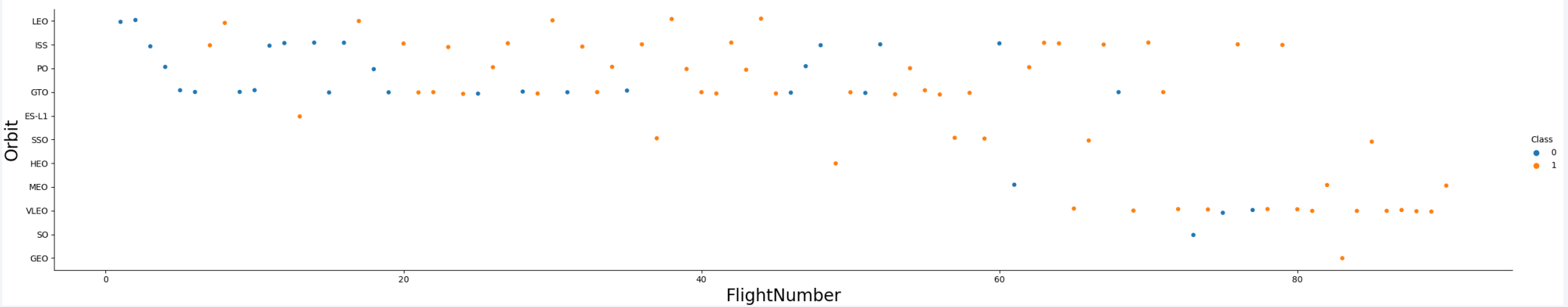


EDA with Data Visualization

Bar chart for the sucess rate of each orbit

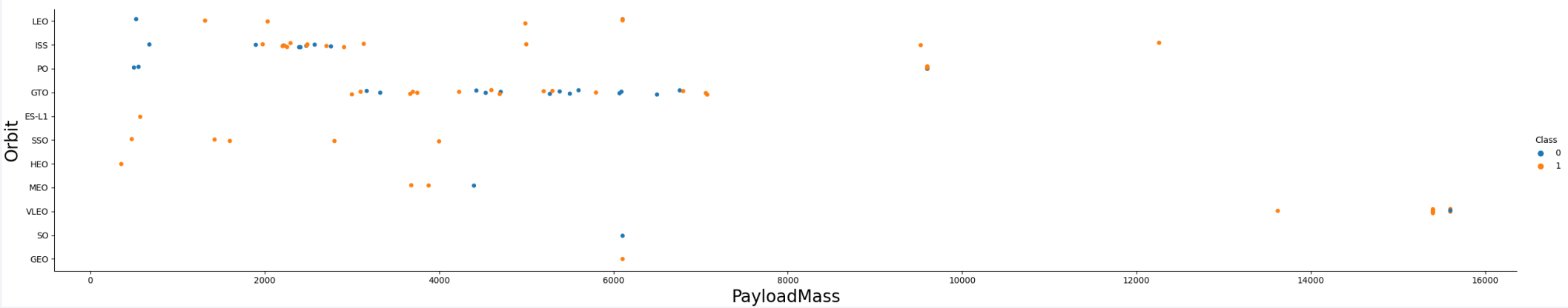


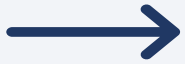
Flight Number Orbit Type

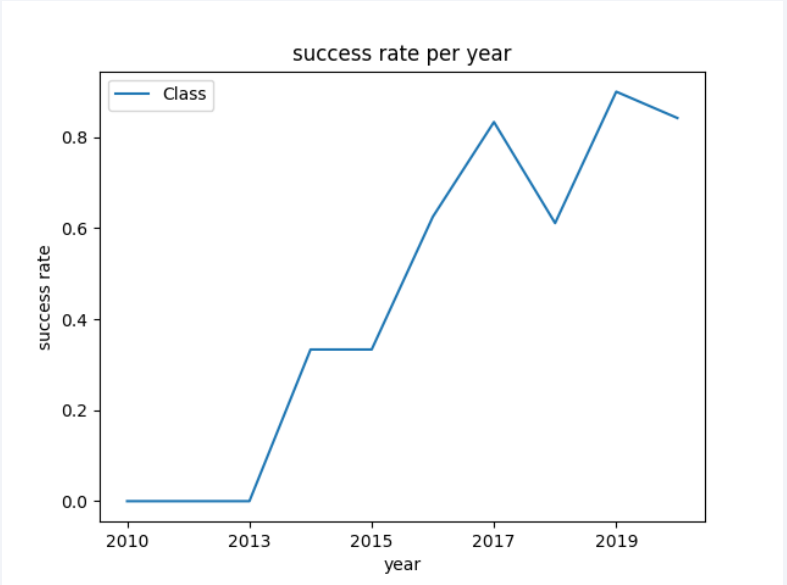


EDA with Data Visualization

Payload Mass  Orbit Type



Success rate per year 



EDA with SQL


Github - EDA with SQL

Performed SQL queries to gather information about the dataset.

We are using SQL queries to get the answers in the dataset:

- Displaying the names of the unique launch sites in the space mission;
- Displaying N records where launch sites begin with the string 'KSC';
- Displaying the total payload mass carried by boosters launched by NASA (CRS);
- Displaying average payload mass carried by booster version F9 v1.1;
- Listing the date where the successful landing outcome in drone ship was achieved.;
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000;
- Listing the total number of successful and failure mission outcomes;
- Listing the names of the booster_versions which have carried the maximum payload mass;
- Listing the records which will display the month names, successful landing_outcomes in ground pad, booster versions, launch_site for the months in year 2017;
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Interactive Map with Folium


Github - Map with Folium

The launch success rate may depend on many factors such as payload mass, orbit type, and so on.

It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories.

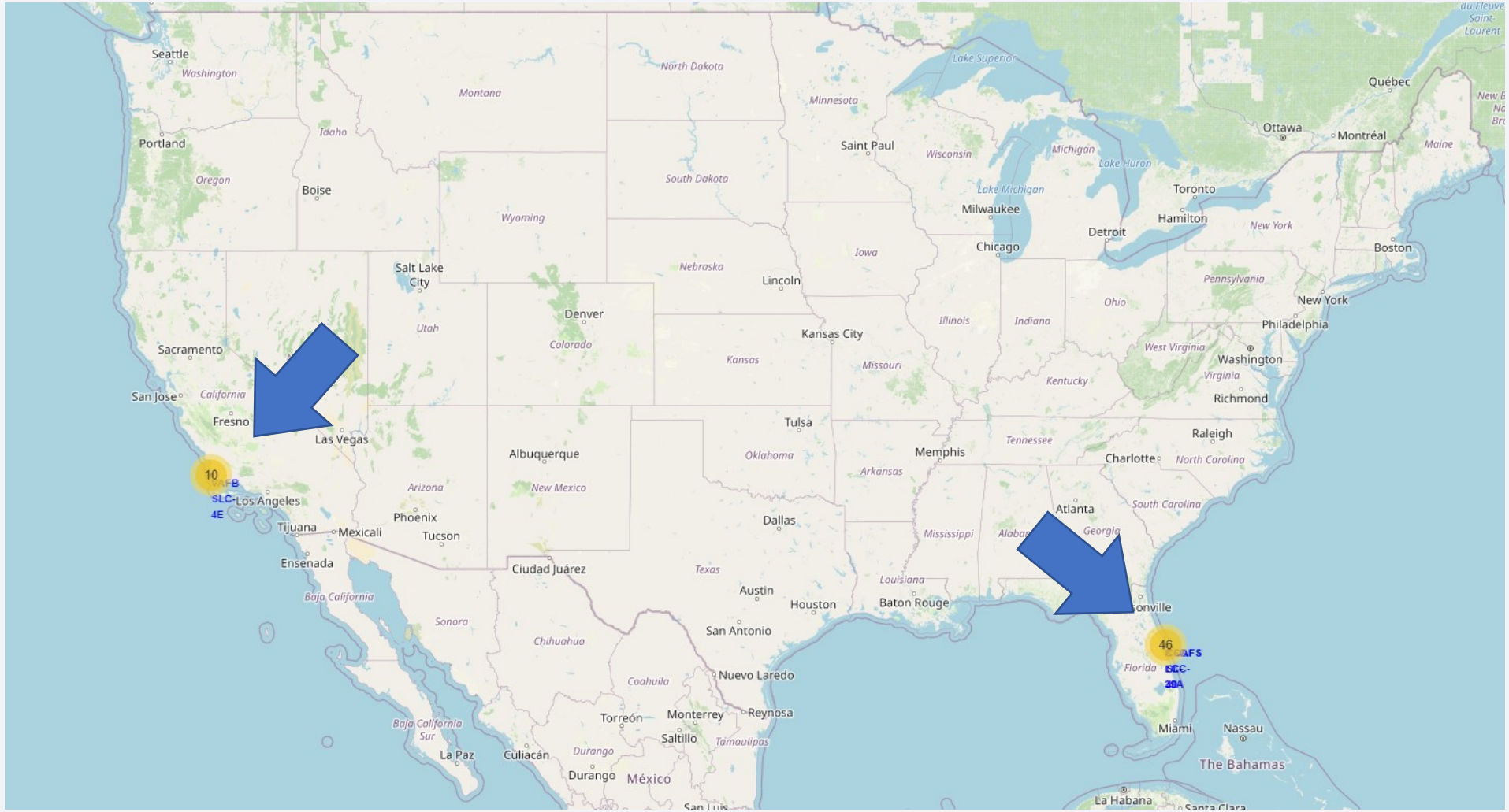
Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations.

Our goal was find some geographical patterns about launch sites completing the following tasks:

- Mark all launch sites on a map;
- Mark the success/failed launches for each site on the map;
- Calculate the distances between a launch site to its proximities;

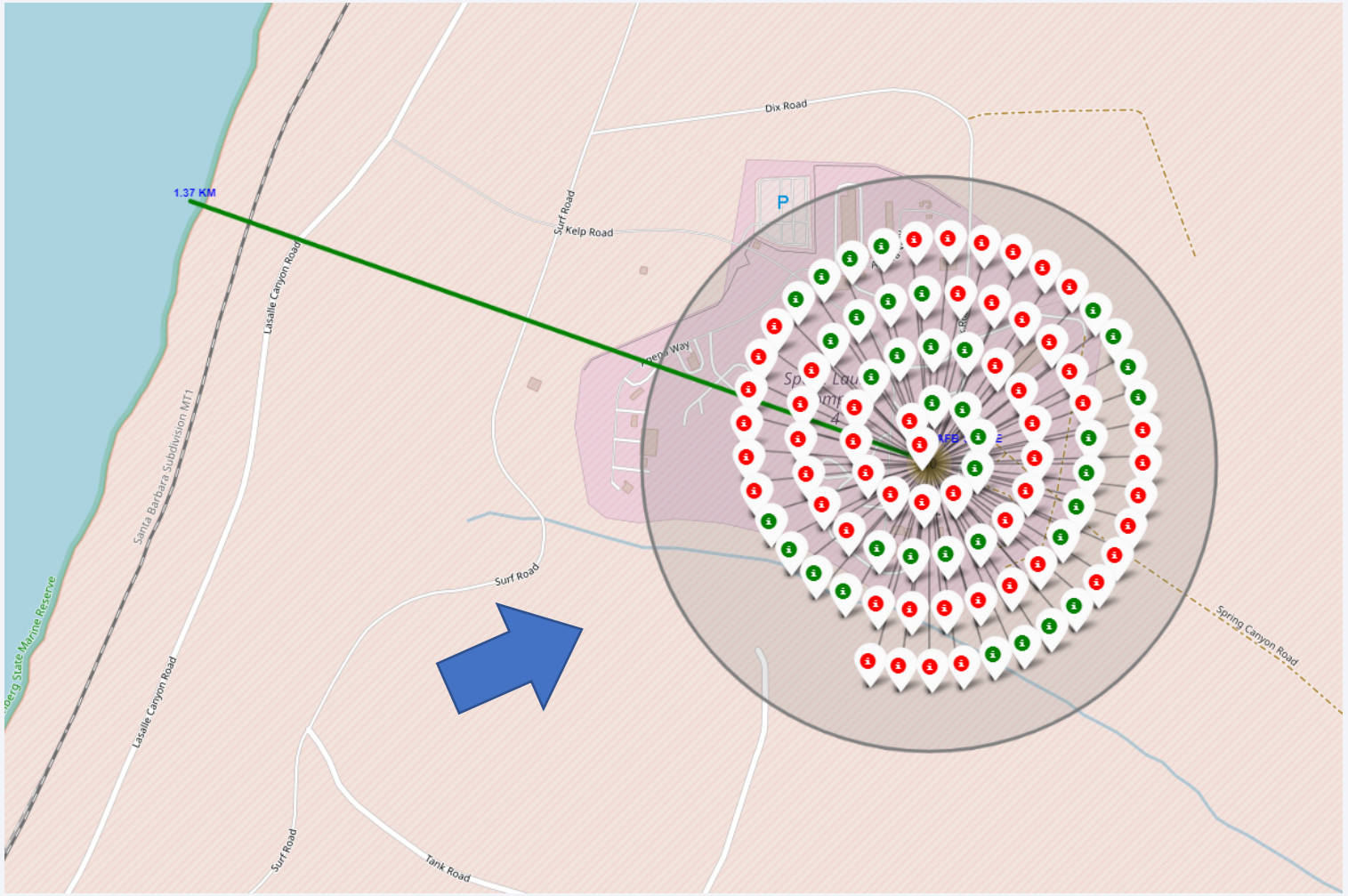
Interactive Map with Folium

Create and add `folium.Circle` and `folium.Marker` for each launch site on the site map



Interactive Map with Folium

A ‘folium.Marker’s in a ‘marker_cluster’ created for each launch result in ‘spacex_df’ data frame.



Build a Dashboard with Plotly Dash

Github - Dashboard with Plotly

The dashboard is built with Flask and Dash web framework.

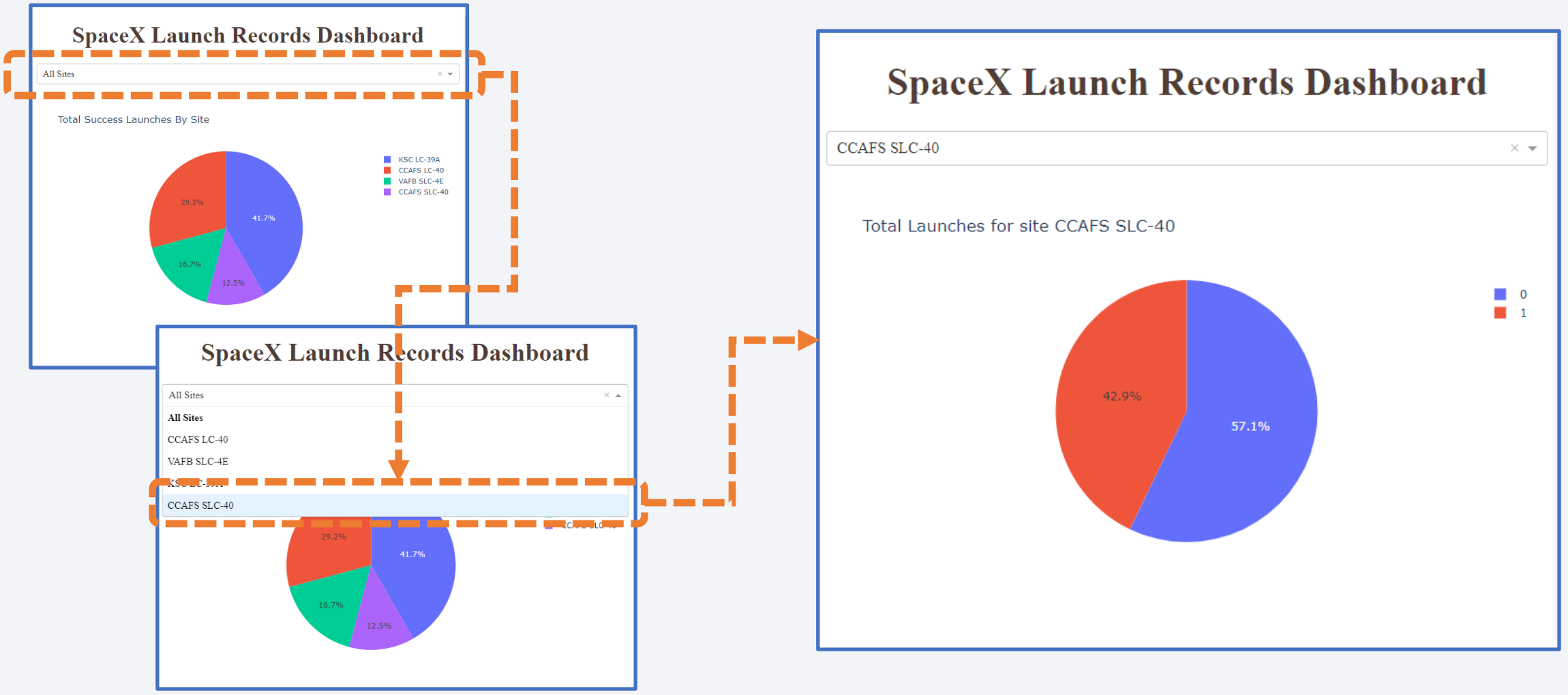
Graphs

- Pie Chart show the total launches by a specific site or all sites;
- Display relative proportions of multiple classes of data.

Scatter Graph show the relationship with Outcome and Payload Mass (Kg) to different Booster Versions:

- Shows the relationship between two variables;
- The best method to show a non-linear pattern;
- The range of data flow, i.e. maximum and minimum value, can be determined;
- Observation and reading are straightforward.

Build a Dashboard with Plotly Dash



Build a Dashboard with Plotly Dash

Predictive Analysis (Classification)

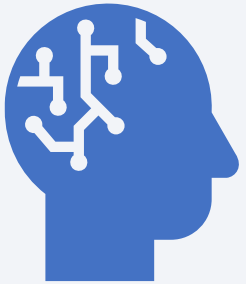
Github - Predictive Analysis

BUILDING MODEL
- Load dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our dataset.

EVALUATING MODEL
- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

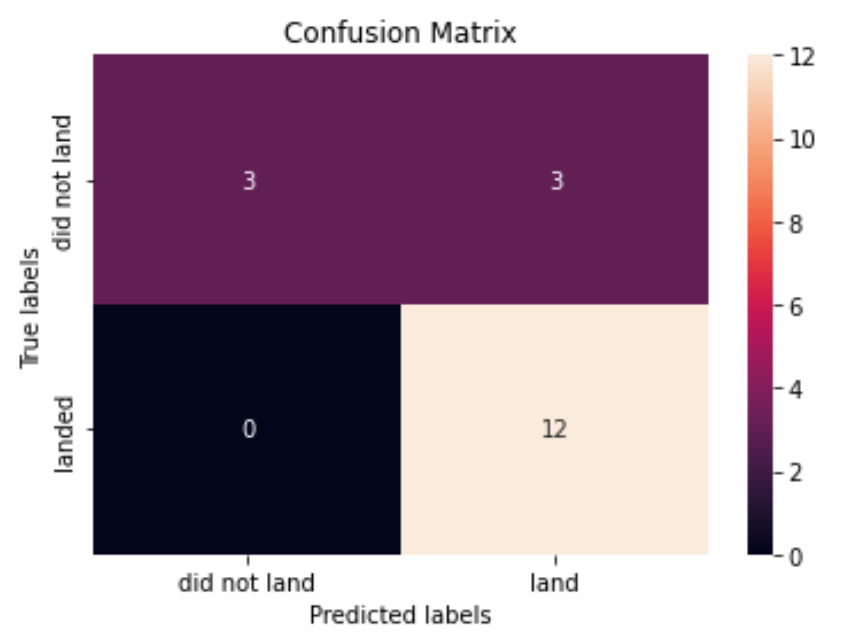
IMPROVING MODEL
- Feature Engineering
- Algorithm Tuning

FINDING THE BEST PERFORMING CLASSIFICATION MODEL
- The model with the best accuracy score wins the best performing model
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook.

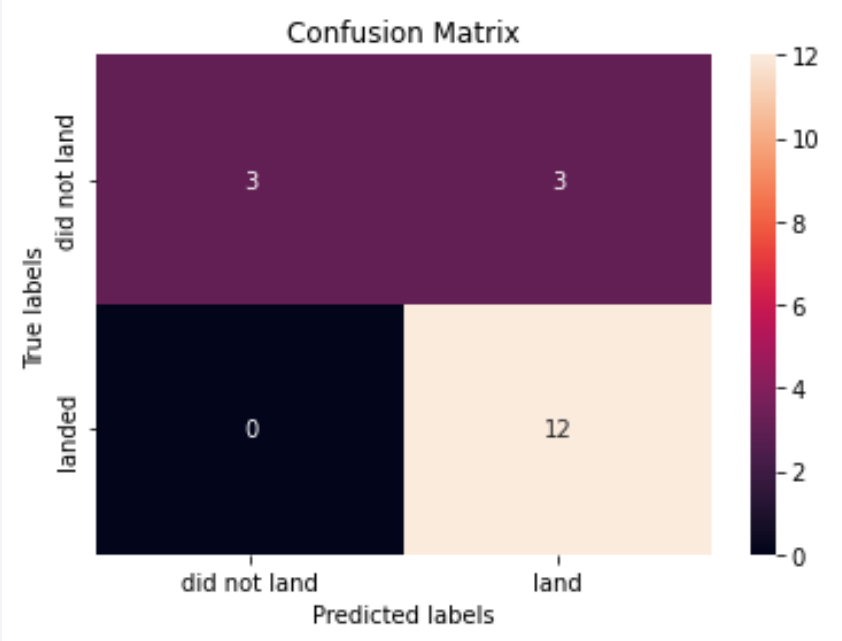


Predictive Analysis (Classification)

Logistic Regression accuracy 0.8333

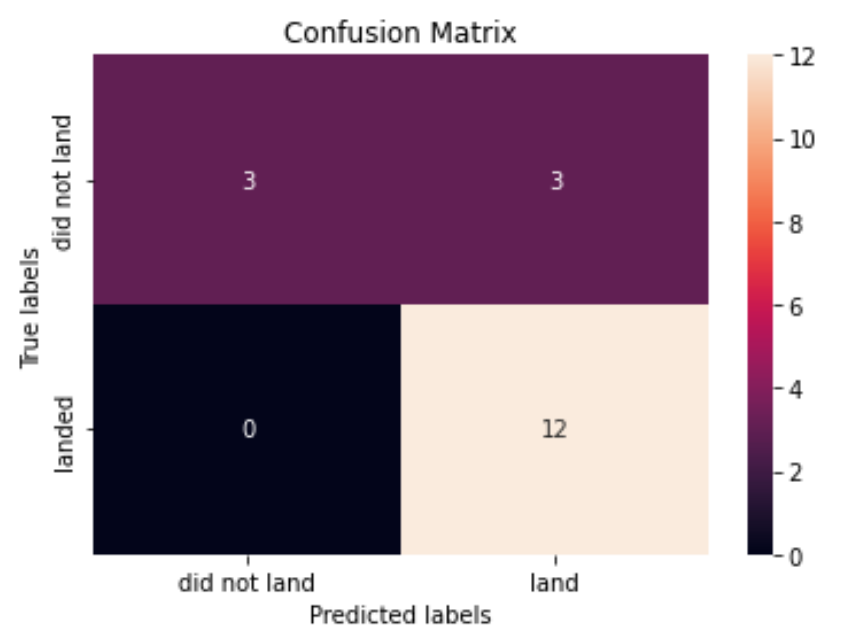


SVM test data accuracy 0.8333

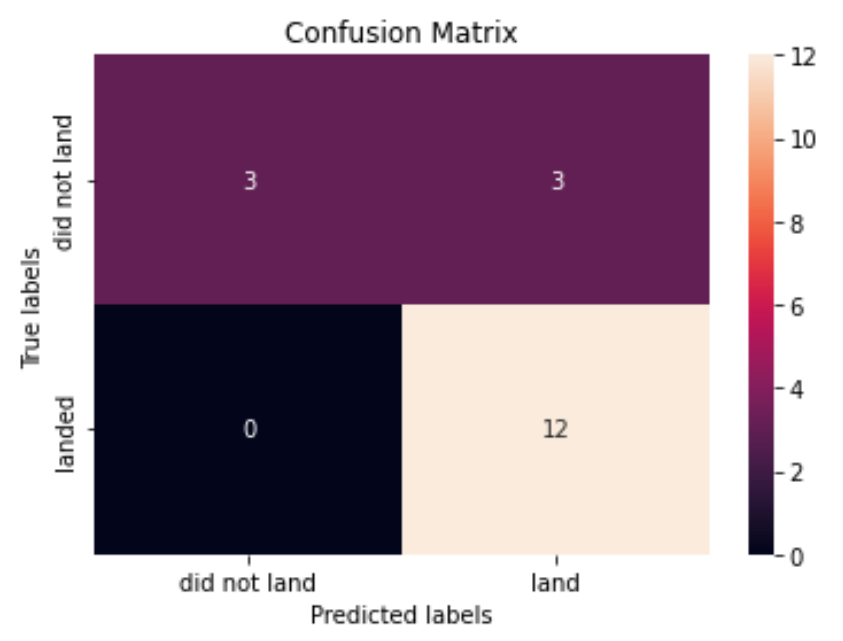


Predictive Analysis (Classification)

Decision Tree accuracy 0.8333



KNN CV accuracy 0.8333

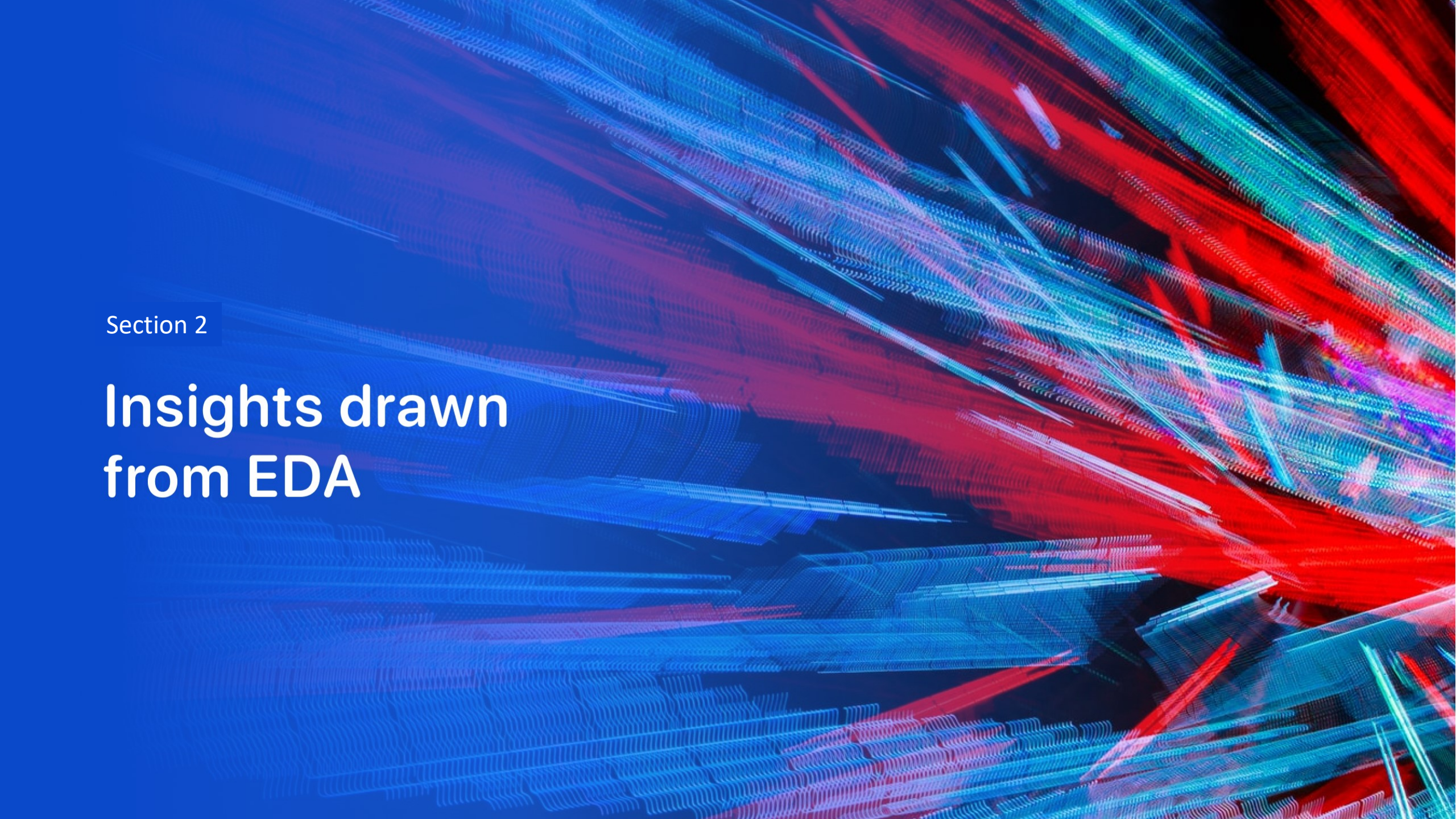


Results

After comparing accuracy of methods, they all preformed practically the same with test data, except for Tree CV witch fit train data slightly better but test data worse.

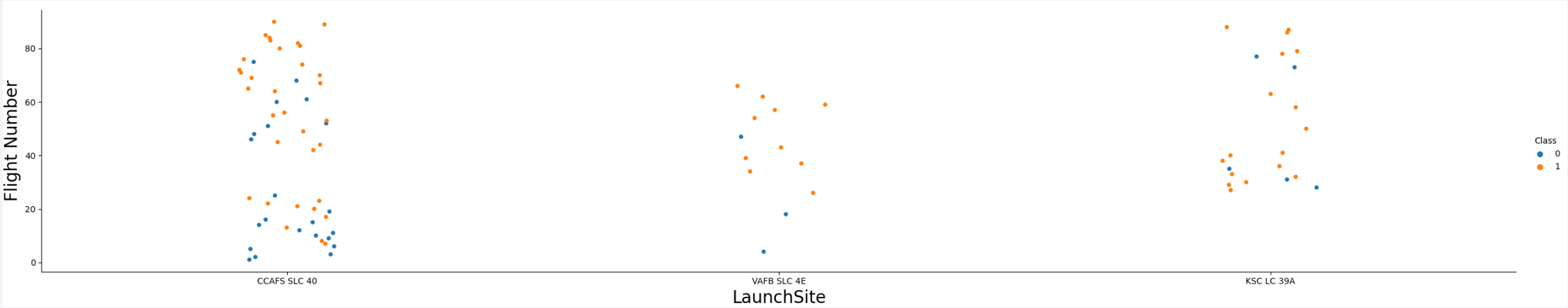
Accuracy per method:

	Logistic Regression	SVM	Decision Tree	KNN
Train data	0.8464	0.8482	08767	0.8482
Test data	0.8333	8.3333	0.8333	0.8333


Section 2
Insights drawn from EDA

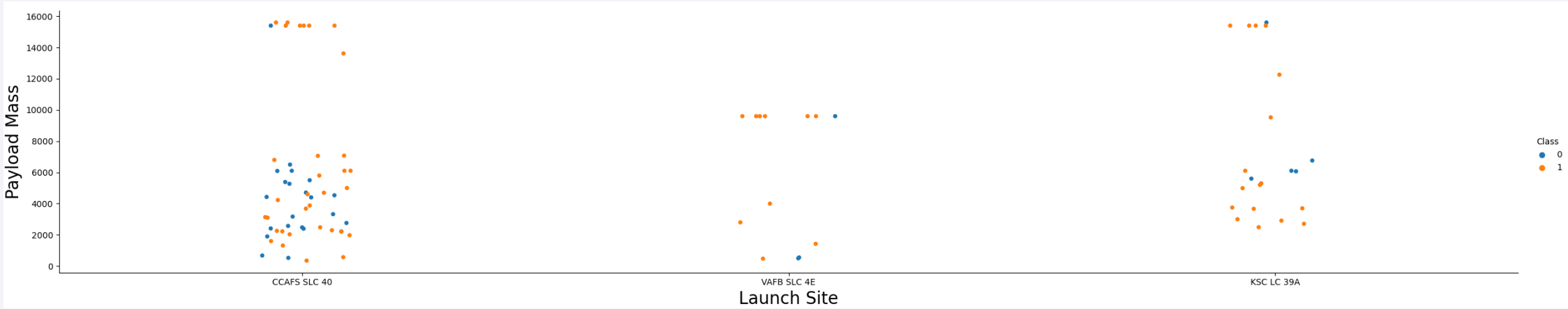
Flight Number vs. Launch Site

CCAFS SLC 40 present Flight Number values higher than the other sites.



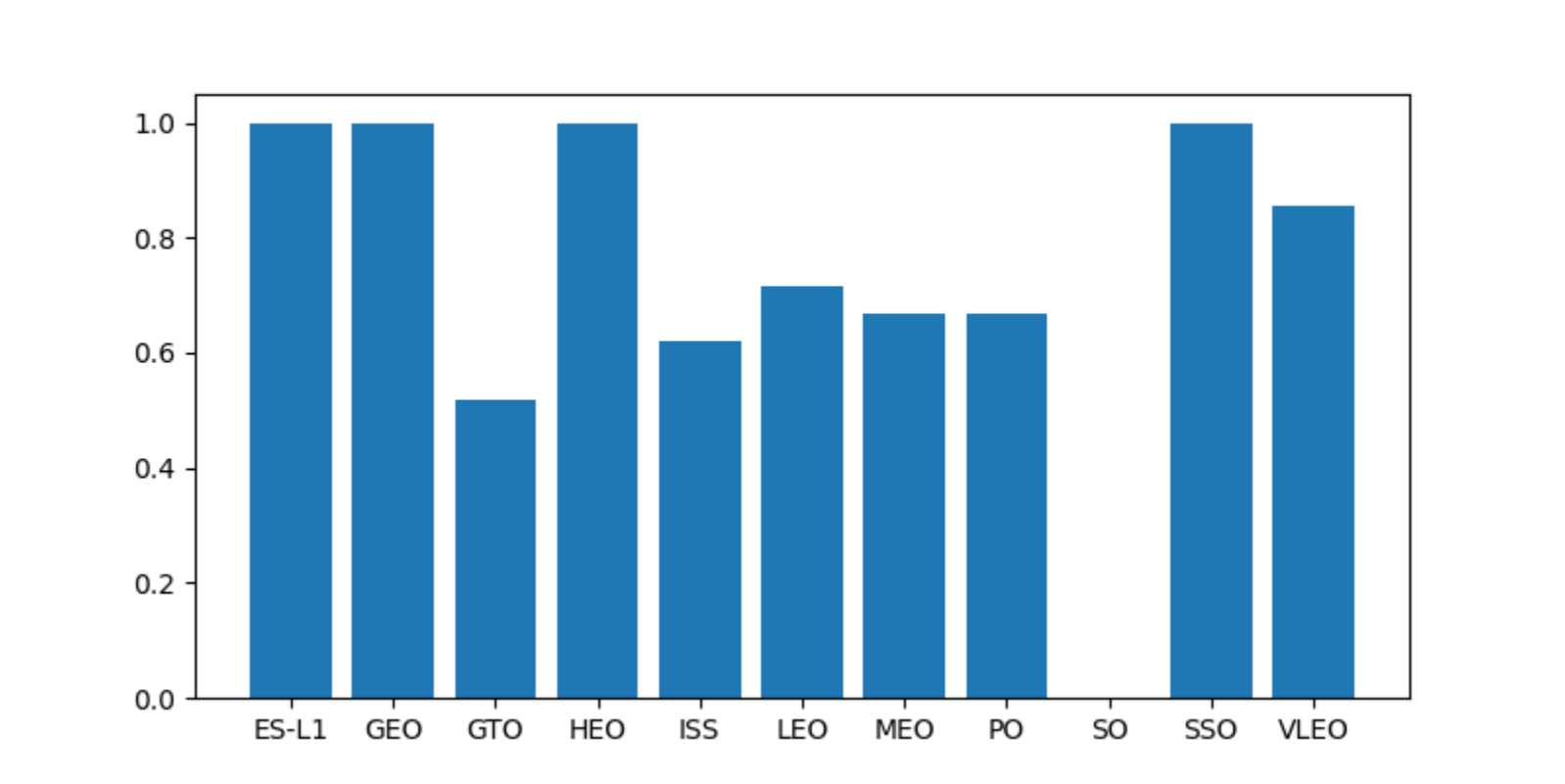
Payload vs. Launch Site

Higher Payload Mass values are related to more unsuccessful landings (0).



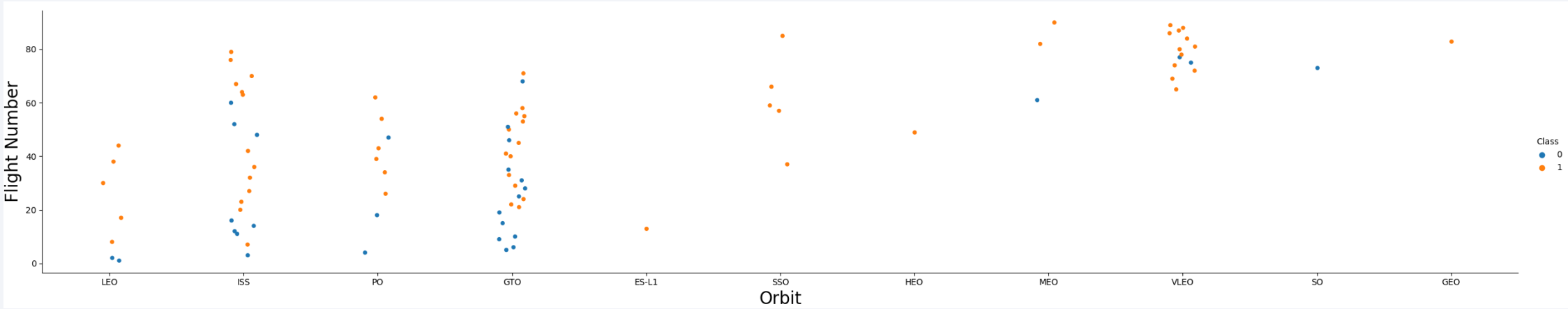
Success Rate vs. Orbit Type

The orbits ES-L1, GEO, HEO and SSO had higher success rates.



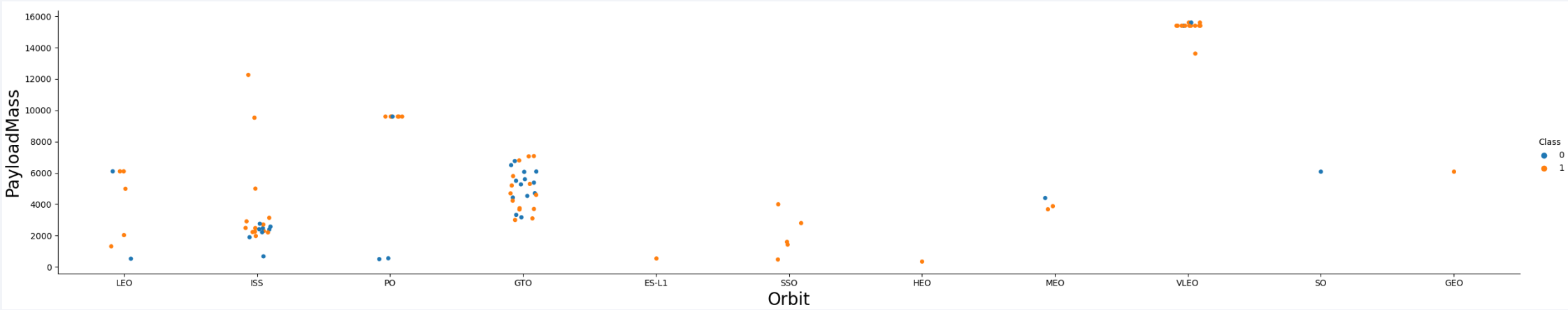
Flight Number vs. Orbit Type

The orbit VLEO had higher success rate of Flight Number.



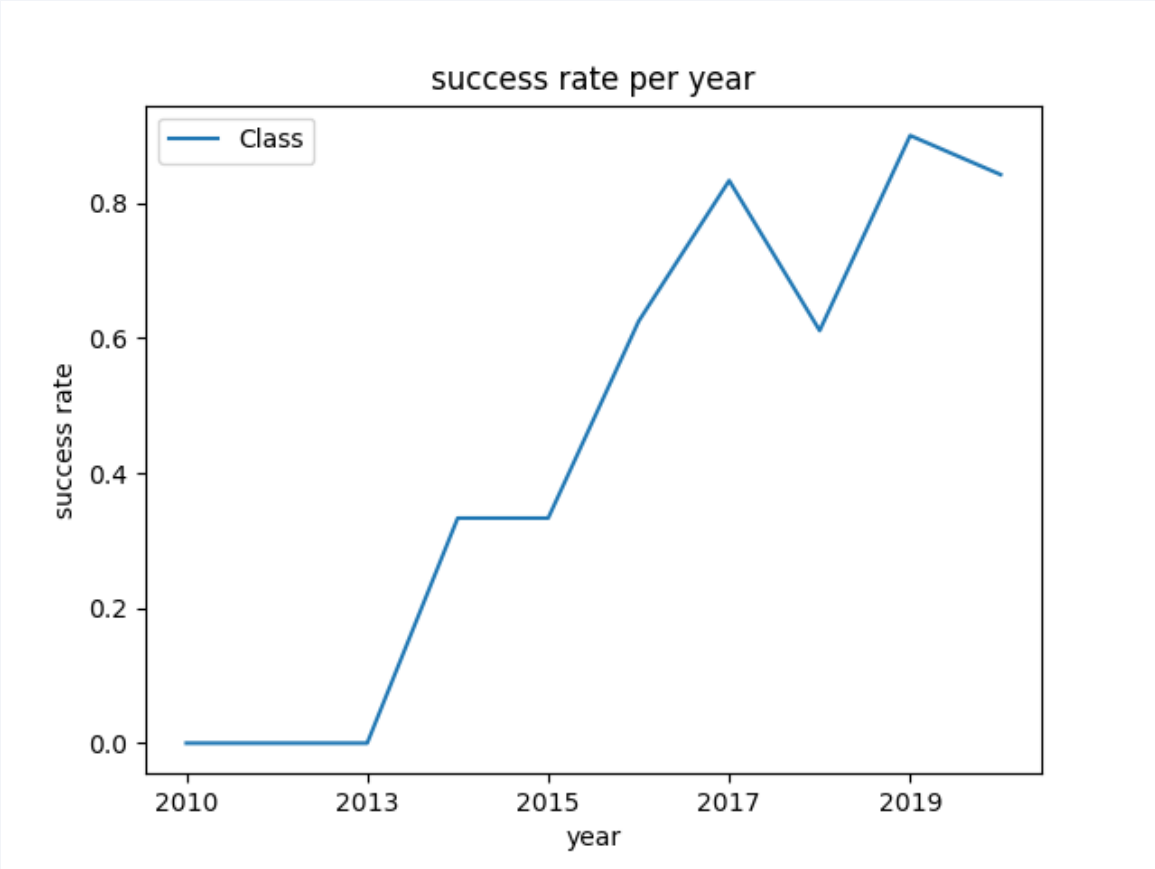
Payload vs. Orbit Type

The orbit VLEO had higher success rate of higher Payload Mass values.



Launch Success Yearly Trend

The launch success rate increases since 2010, with 2 decreases in 2018 and 2020.



All Launch Site Names

%sql select distinct Launch_Site from SPACEXTBL

```
%sql select distinct Launch_Site from SPACEXTBL
```

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

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

Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with `CCA`

SELECT * FROM "YMM48264"."SPACEXTBL" WHERE "LAUNCH_SITE" LIKE 'CCA%' LIMIT 5; (syntax applied in IBM Cloud interface)

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

```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Calculate the total payload carried by boosters from NASA

SELECT SUM("PAYLOAD_MASS__KG_") FROM "YMM48264"."SPACEXTBL" WHERE customer= 'NASA (CRS)'; (syntax applied in IBM Cloud interface)

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

```
%sql select sum(payload_mass__kg_) from SPACEXTBL WHERE customer = 'NASA (CRS)'
```

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

sum(payload_mass__kg_)
45596

Average Payload Mass by F9 v1.1

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

SELECT AVG(payload_mass__kg_) FROM "YMM48264"."SPACEXTBL" WHERE booster_version = 'F9 v1.1'; (syntax applied in IBM Cloud interface)

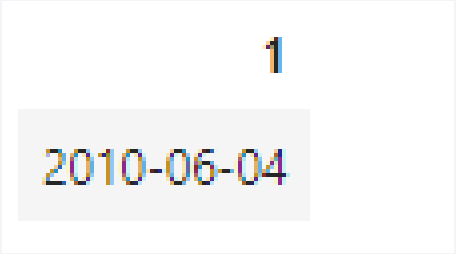
```
%sql select avg(payload_mass__kg_) from SPACEXTBL WHERE booster_version = 'F9 v1.1'
 * sqlite:///my_data1.db
Done.
```

avg(payload_mass__kg_)
2928.4

First Successful Ground Landing Date

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

SELECT MIN("DATE") FROM "YMM48264"."SPACEXTBL" WHERE "MISSION_OUTCOME" = 'Success'; (syntax applied in IBM Cloud interface)



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

SELECT "BOOSTER_VERSION" FROM "YMM48264"."SPACEXTBL" where "LANDING__OUTCOME" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" BETWEEN 4000 and 6000; (syntax applied in IBM Cloud interface)

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

```
%sql select booster_version from SPACEXTBL where "Landing _Outcome" = 'Success (drone ship)' \
and payload_mass__kg_ between 4000 and 6000
```

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

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

Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcomes

SELECT COUNT("MISSION_OUTCOME") FROM "YMM48264"."SPACEXTBL“ GROUP BY "MISSION_OUTCOME"; (syntax applied in IBM Cloud interface)

List the total number of successful and failure mission outcomes

```
%sql select mission_outcome, count(mission_outcome) from SPACEXTBL GROUP BY mission_outcome
```

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

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

Boosters Carried Maximum Payload

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

SELECT "BOOSTER_VERSION" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_"= (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL); (syntax applied in IBM Cloud interface)

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

```
%sql select Booster_Version, PAYLOAD_MASS__KG_ from SPACEXTBL\
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

%sql select substr(Date, 4, 2) as "Month", substr(Date, 7, 4) as "Year", Booster_Version as "Booster Version", launch_site as "Launch Site" from SPACEXTBL where landing_outcome = 'Failure (drone ship)' and substr(Date, 7, 4) = '2015'

Month	Year	Booster Version	Launch Site
01	2015	F9 v1.1 B1012	CCAFS LC-40
04	2015	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

%sql select count(“Landing _Outcome”) as "Count", “Landing _Outcome” as "Landing Outcome" from SPACEXTBL \

where Date between '04-06-2010' and '20-03-2017' group by “Landing _Outcome” \

order by count(“Landing _Outcome”) desc

```
%sql select count("Landing _Outcome") as "Count", "Landing _Outcome" as "Landing Outcome" from SPACEXTBL \
where Date between '04-06-2010' and '20-03-2017' group by "Landing _Outcome" \
order by count("Landing _Outcome") desc
```

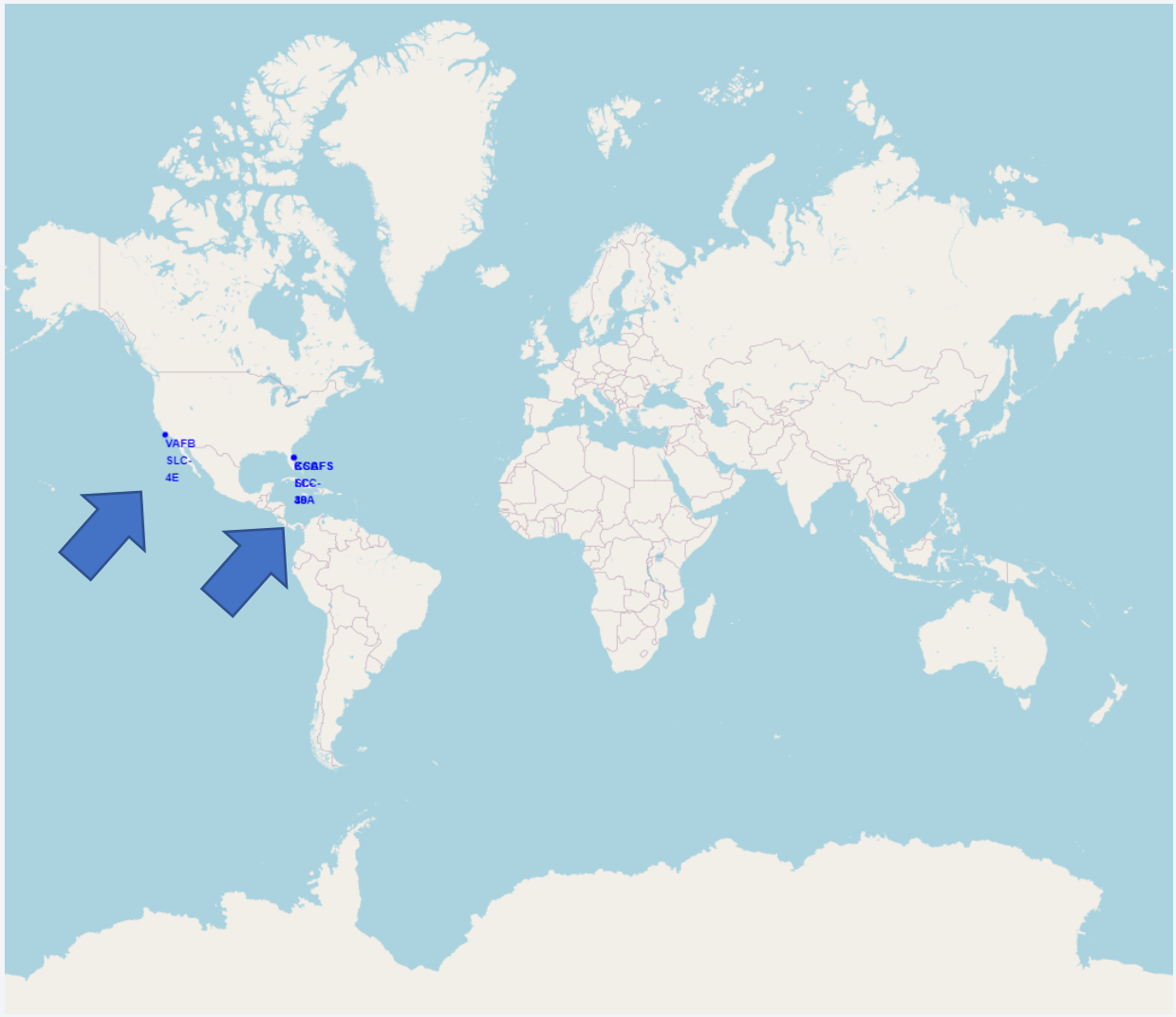
```
 * sqlite:///my_data1.db
Done.
```

Count	Landing Outcome
20	Success
10	No attempt
8	Success (drone ship)
6	Success (ground pad)
4	Failure (drone ship)
3	Failure
3	Controlled (ocean)
2	Failure (parachute)
1	No attempt

Section 3

Launch Sites Proximities Analysis

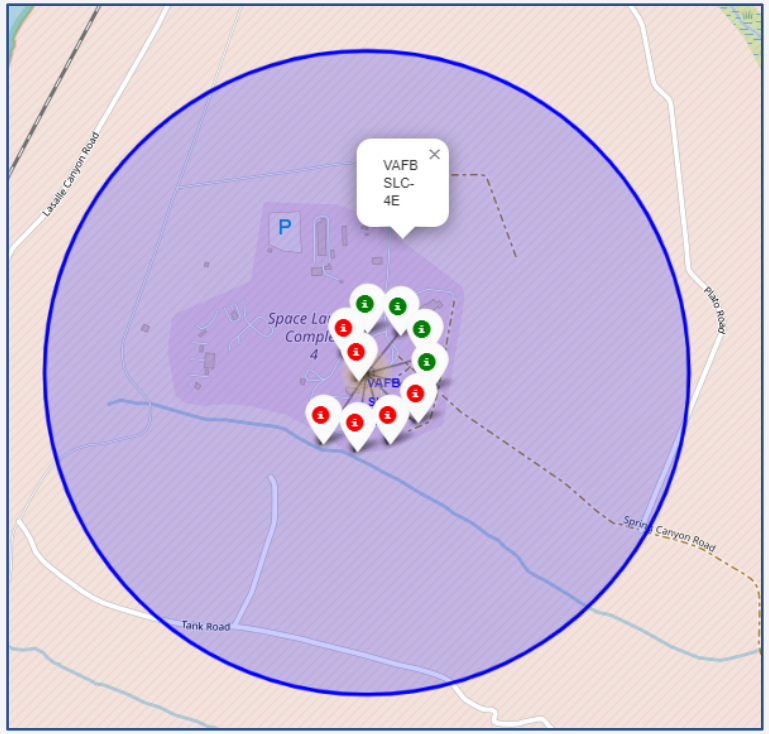
Launch Sites Location in the World



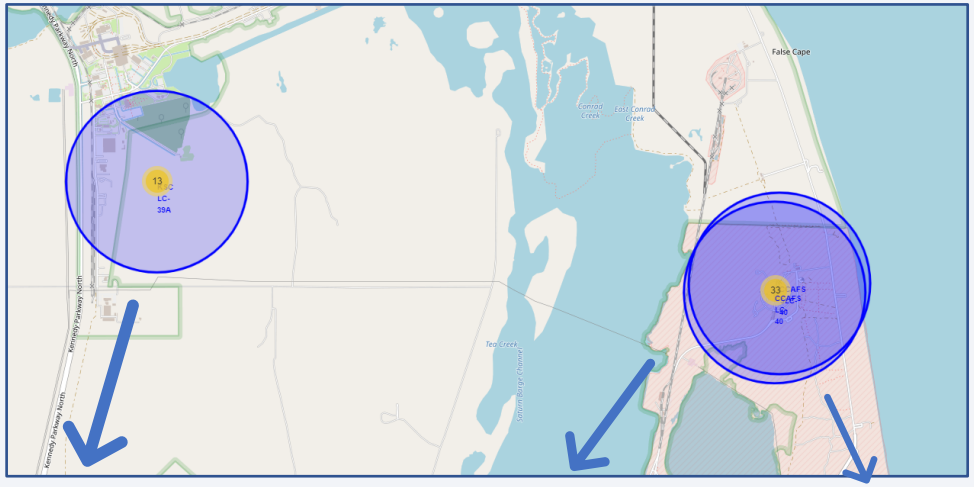
The SpaceX launch sites are in the USA, located in Florida and California coasts.

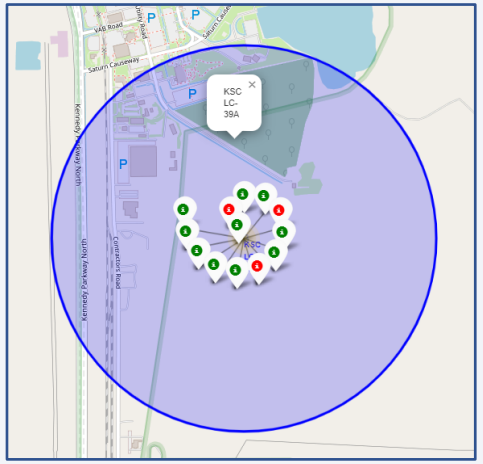
Colour Labelled Markers

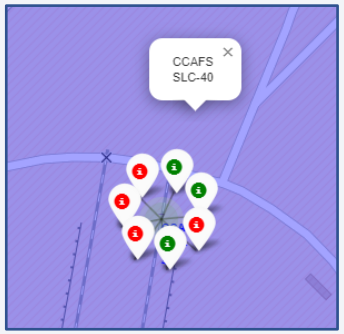
California Launch Site

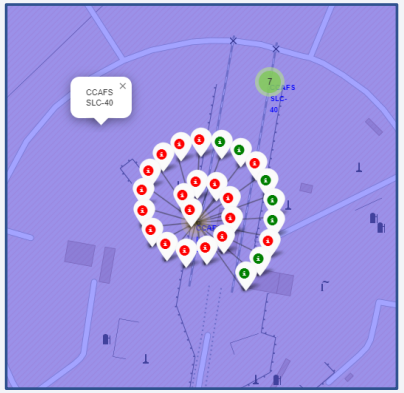


Florida Launch Sites









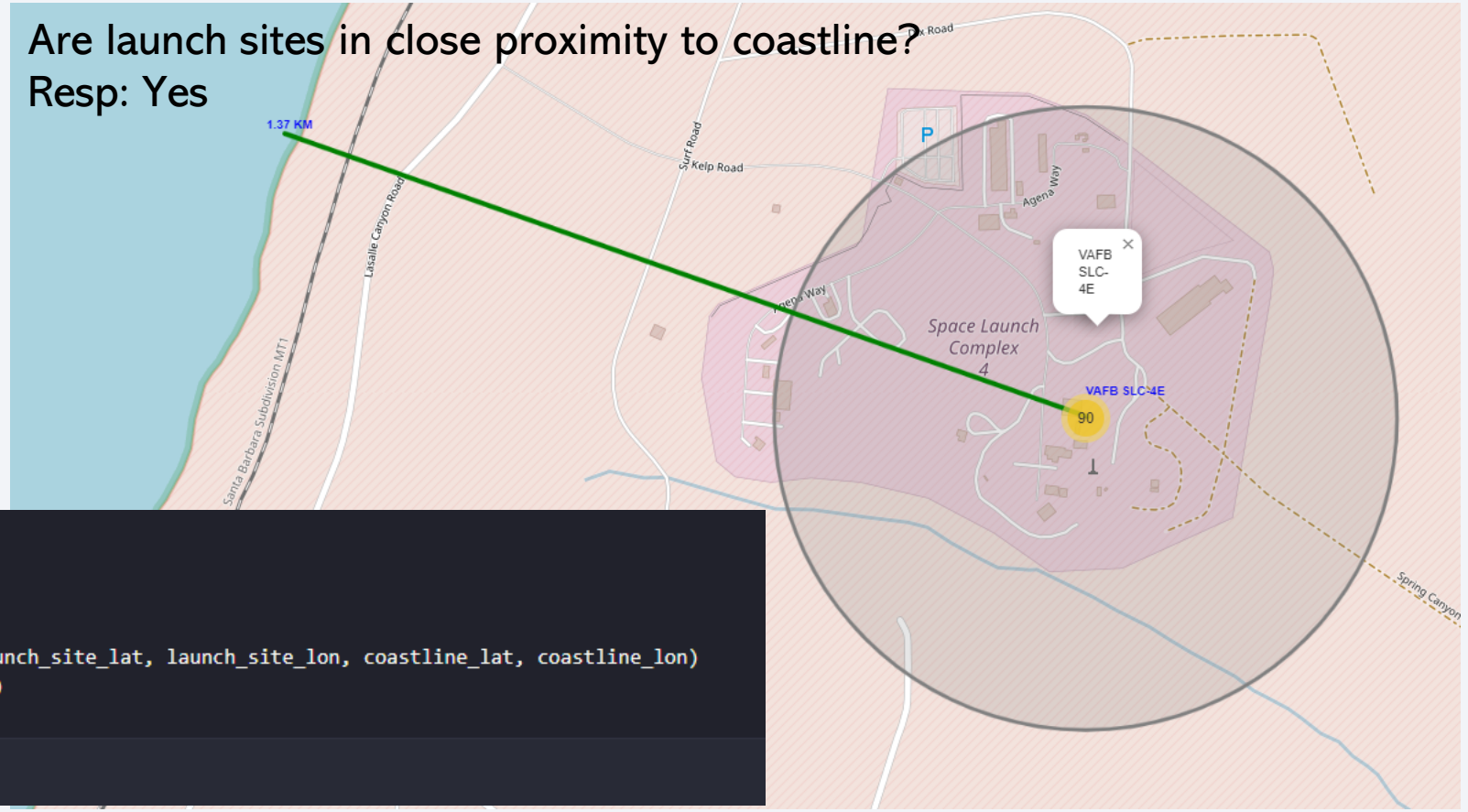
Green Marker -> successful launches

Red Marker -> Failures

Launch Sites Distance to Landmarks

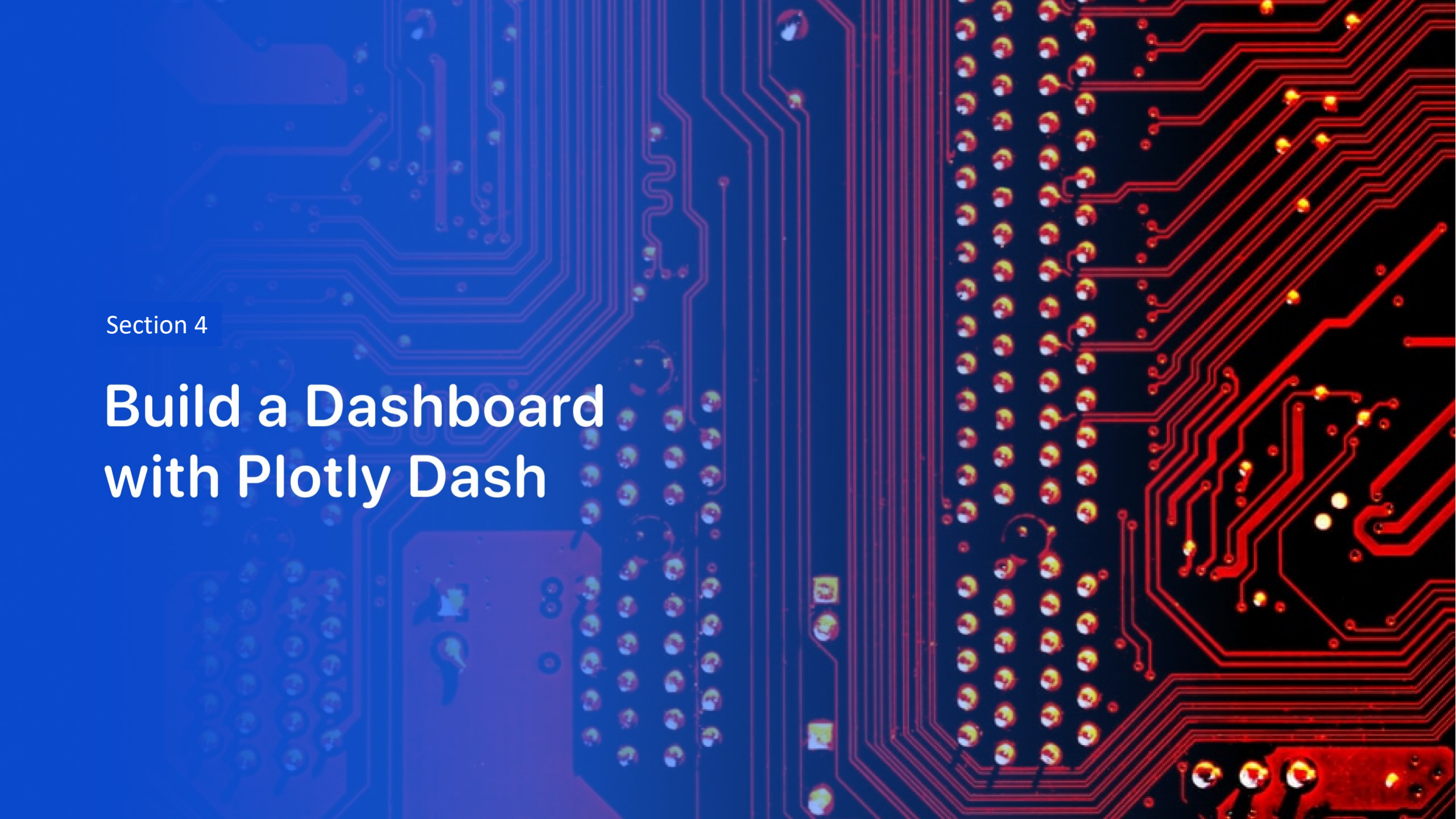
Distance of VAFB SLC-4E launch site to the coast.

Are launch sites in close proximity to coastline?
Resp: Yes



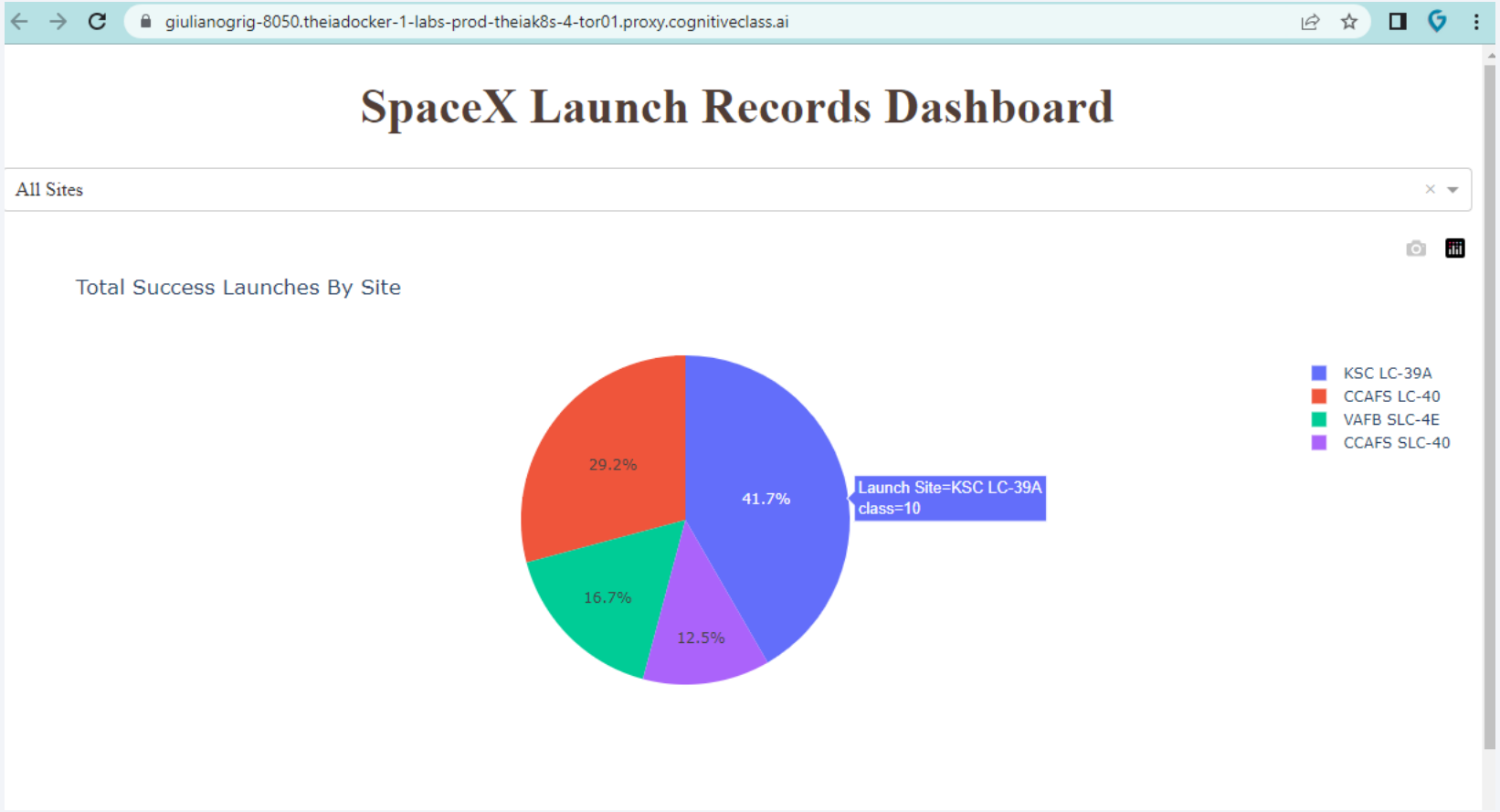

```
launch_site_lat = 34.63285
launch_site_lon = -120.6107
coastline_lat = 34.63695
coastline_lon = -120.62482
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
print(distance_coastline,'km to coastline')
```

```
1.3703554466682268 km to coastline
```


Section 4
Build a Dashboard with Plotly Dash

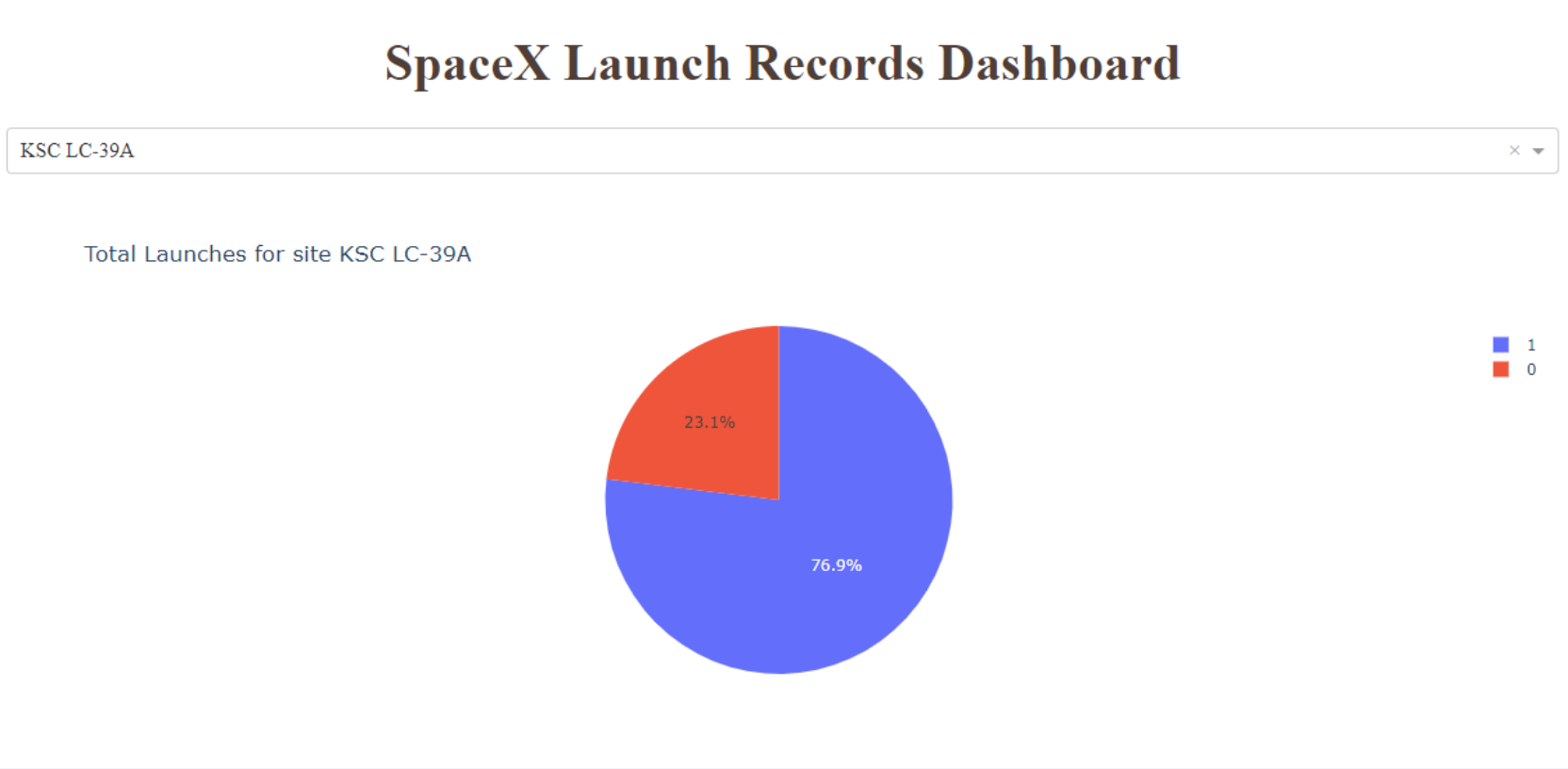
Success Launches by Site

The graph show that KSC LC-39A had the most successful launches in comparison of the other sites



KSC LC-39A Success Launch Percentage

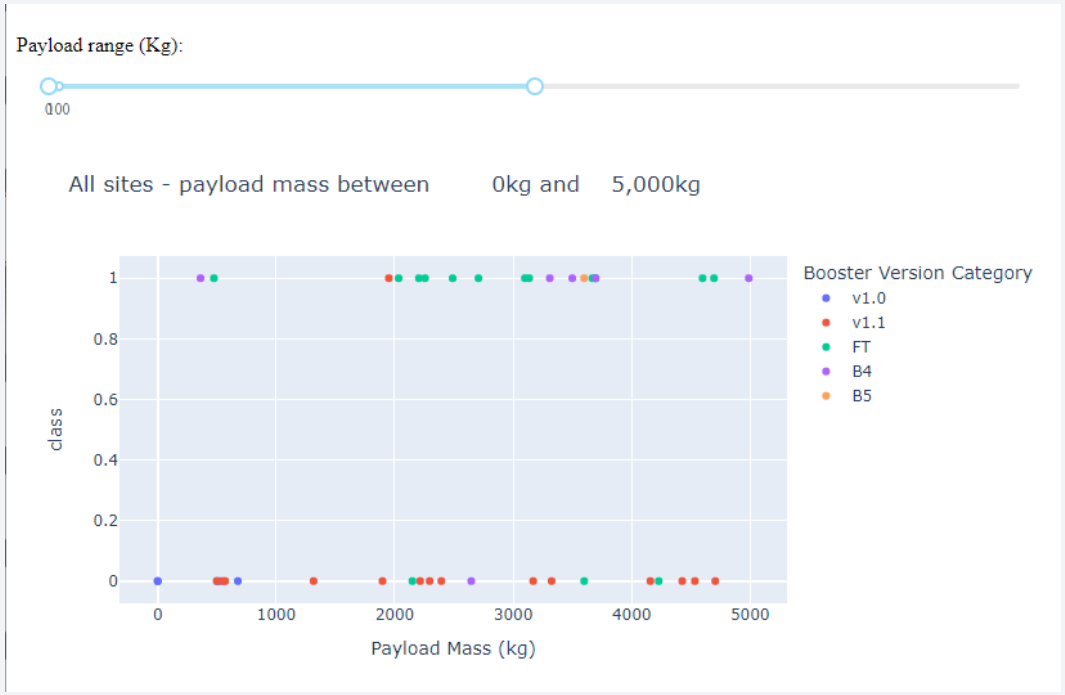
KSC LC-39A achieved a 76.9% success launch rate



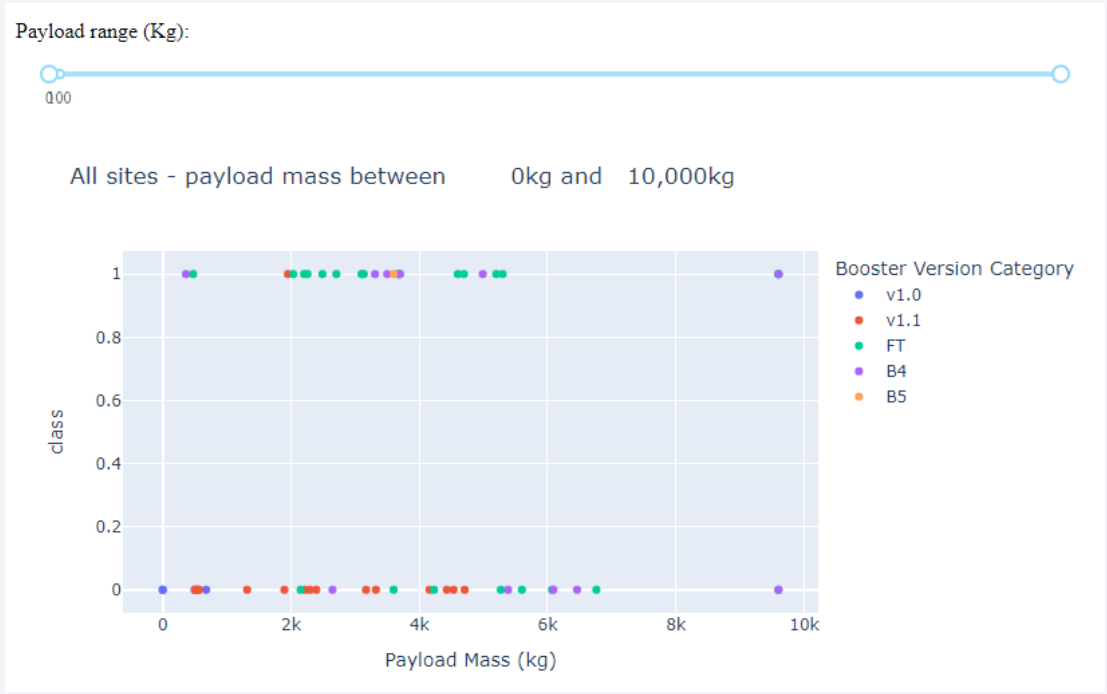
Payload vs. Launch Outcome Scatter Plot

Payload vs. Launch Outcome scatter plot for all sites, with different payload mass value selected in the range slider

Payload 0 – 5.000kg



Payload 0 – 10.000kg



The success rates for lower weighted payloads is higher than the heavy weighted payloads.

Section 5

Predictive Analysis (Classification)

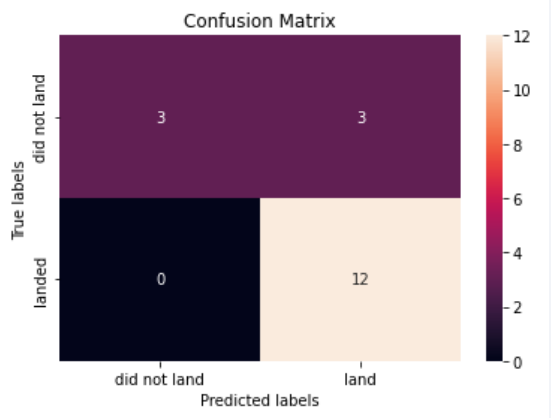
Classification Accuracy

	Logistic Regression	SVM	Decision Tree	KNN
Train data	0.8464	0.8482	08767	0.8482
Test data	0.8333	8.3333	0.7222	0.8333

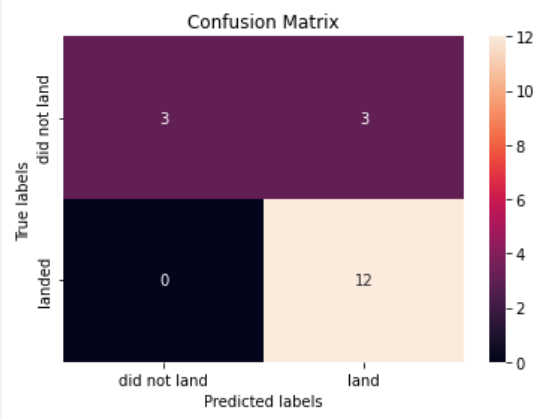
Confusion Matrix

Logistic Regression accuracy 0.8333

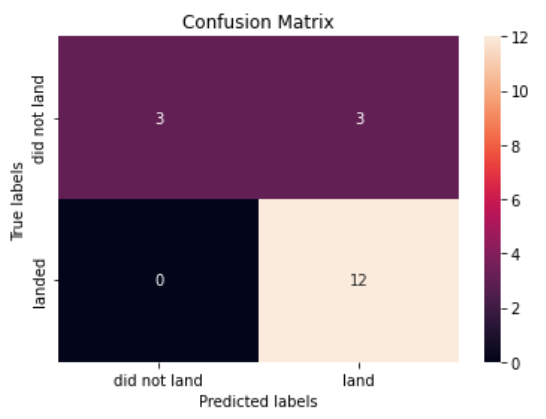
SVM test data accuracy 0.8333



Decision Tree accuracy 0.7222



KNN CV accuracy 0.8333



Conclusions

- Low weighted payloads perform better than heavier payloads;
- Success rates increases with year for SpaceX launches;
- KSC LC 39A had the most successful launches from all the sites;
- Orbit types GEO, HEO, SSO, ES L1 has the best success rate;
- SVM, KNN and Logistic Regression model are the best in terms of accuracy for this dataset;
- The Tree Classifier Algorithm is the best for Machine Learning for this dataset

Thank you!