IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sukhwinder Singh
Jan 11 2023

Outline

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

Executive Summary

- In this presentation we will review the finding about the prediction if the SpaceX Falcon 9 first stage will land successfully using different algorithms
- Finding in plot graphs will show different correlation of rocket launches and the outcomes of the launches such as success or failure
- Steps used
  - Data collections
    - Data Wrangling
    - Data formatting
  - Data Analysis
  - Data Visualization
  - Machine Learning Prediction

Introduction

- SpaceX Falcon 9 rockets cost approx. around $62 million which is cheaper than other provider cost up to $165 million.
- Cost difference is due to the fact that SpaceX can land then be re used for again
- We need to determine if the first stage will land, which will allow use to determine the cost of the launch
- All the information collected will help us with our new company Space Y

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - We used requests such as GET to extract data from SPACEX REST API
  - Also used web scraping from Wikipedia
- Perform data wrangling
  - Collected the information and then calculating the number of launches and mission after removing all the NaN values from the data
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Used SQL language and Python Libraries to export visual charts
- Perform interactive visual analytics using Folium and Plotly Dash
  - Created interactive dashboards

Data Collection

- Data was collected via SPACEX REST API
  - https://api.spacexdata.com/v4/rockets/ was used to collect data
  - Within this data collection we cleaned the requested data
    - Data needed to filter for only falcon 9
    - Every missing value was replaced with the mean

- Data was collected via Webscraping – Wikipedia
  - Data was obtained from - https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
  - The website only contained data for Falcon launches

Data Collection – SpaceX API

Task 1 – API calling

- Used get command to get the data from the API
- Converted the response to .json() which then we used to result into a data frame

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

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

Check the content of the response

```
print(response.content)
```

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

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

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

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

```
response.status_code
```

```
200
```

Now we 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
df = response.json()
data = pd.json_normalize(df)
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

Data Collection – SpaceX API

Task 2 – Creating List and filtering

- Created a list for the new dataframe
- Stored the values in that dataframe
- Filtered the dataframe for only Falcon 9 Launches

The data from these requests will be stored in lists and will be used to create a new dataframe.

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

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

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

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

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

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

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/indexing.py:1773: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	1	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	2	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	3	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	4	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	5	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	86	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	87	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	88	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	89	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	90	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

Data Collection – Webscraping

- Requested data from wiki page
- Created a beautifulsoup object
- Extracted all the column headers
- Created our own dataframe with these values

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

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

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

```
200
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text)
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
print(soup.title)
```

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

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

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

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

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

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

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

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
launch_dict= dict.fromkeys(column_names)

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

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

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

Usually, HTML tables in Wiki pages are likely to contain unexpected annotations and other types of noises, such as reference links `B0004.1[8]`, missing values `N/A [e]`, inconsistent formatting, etc.

To simplify the parsing process, we have provided an incomplete code snippet below to help you to fill up the `launch_dict`. Please complete the following code snippet with TODOs or you can choose to write your own logic to parse all launch tables:

```
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])

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

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

Data Wrangling

- SpaceX data contains all the facilities in the Launchsite column
  - We checked for all the launches in each site
- Calculated the number of orbit and the outcome
- Created a landing outcome

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

```
# Apply value_counts() on column LaunchSite

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

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

TASK 2: Calculate the number and occurrence of each orbit

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

```
# Apply value_counts on Orbit column

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
```

TASK 3: Calculate the number and occurence of mission outcome per orbit type

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

```
# landing_outcomes = values on Outcome column

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

```
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
```

EDA with Data Visualization

- EDA was used to show different types of chart which allowed us to clearly see and compare variables

Charts used

- Scatter Plots – ex. Flight Number v Launch site
- Bar Charts – ex. Success rate v Orbit type
- Line Charts – ex. Success rate v Year

EDA with SQL

- Loaded the dataset into the IBM DB2 database
- Used python to query for the data
- Allowed SQL queries to understand the data

Task 1

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

```
%sql select DISTINCT LAUNCH_SITE as "Launch_Sites" from SPACEX;
```

```
 * ibm_db_sa://xnr69837:***@98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30875/bludb
Done.
```

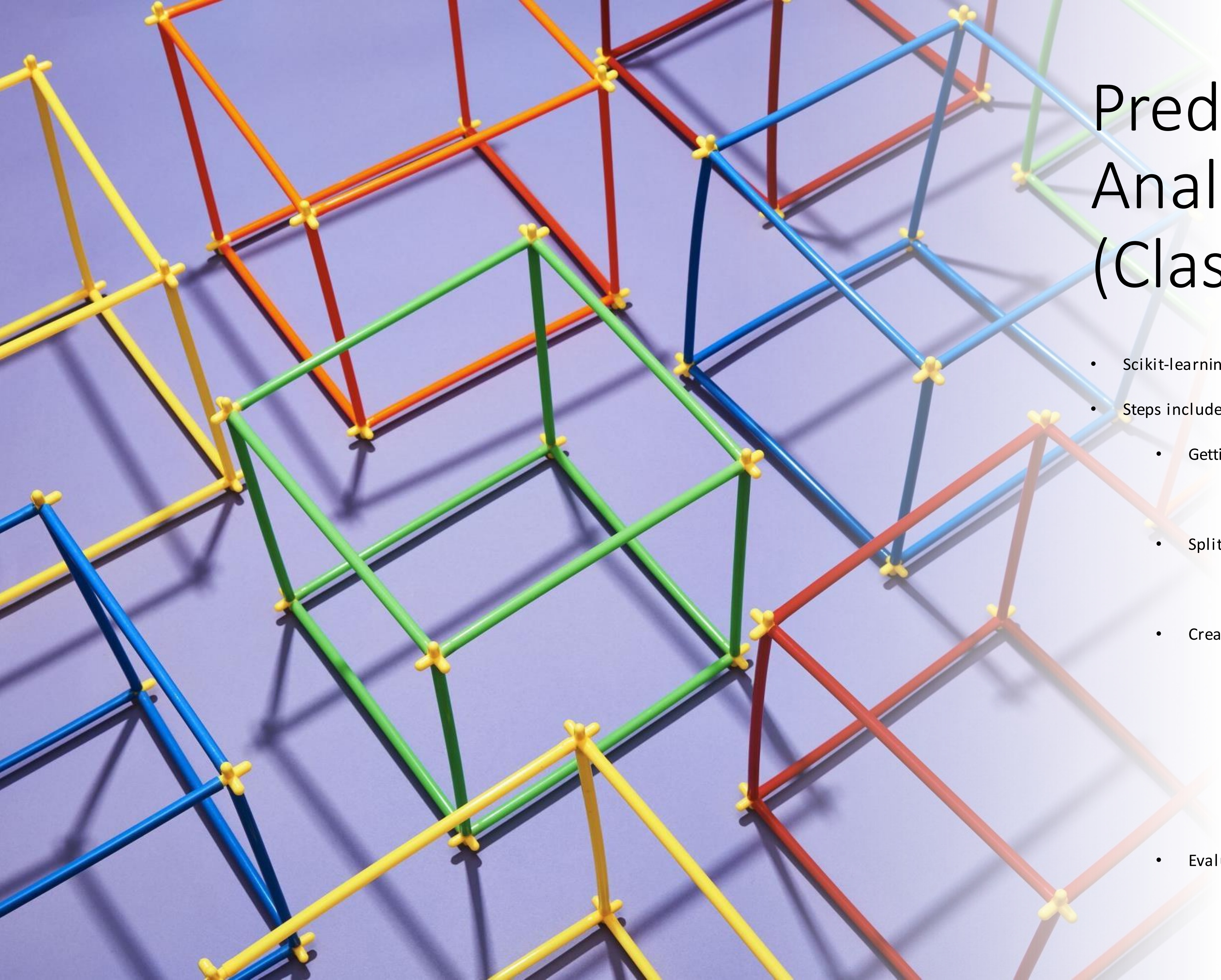
Launch_Sites
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Build an Interactive Map with Folium

- Used Folium to visual data on interactive Map
    - This allowed us to do and see the following on the map
        - Launch sites
        - Successful and unsuccessful landings
        - Distances from certain places to the launch sites

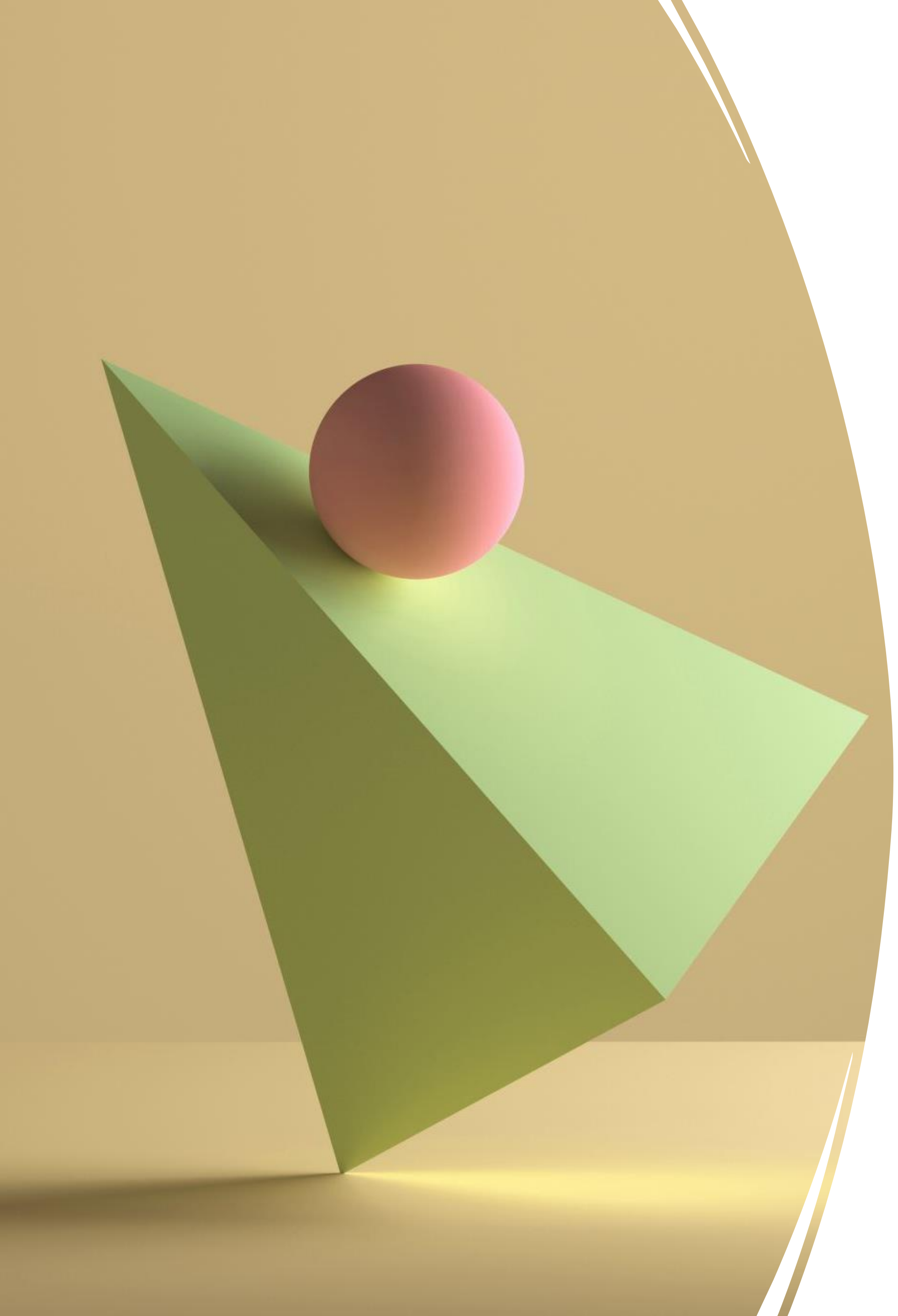
Build a Dashboard with Plotly Dash

- Created interactive pie charts and scatterplots
- Pie Charts
    - Showed the successful and unsuccessful launches for all the sites
- Pie Charts
    - Shows the correlation between payload and success for all the launch sites



Predictive Analysis (Classification)

- Scikit-learning library was used to create machine learning model
- Steps included
  - Getting the data
    - Preprocessing the data to standardize our data
  - Splitting the data
    - Training and test data
  - Create machine learning models
    - Logistic Regression
    - SVM
    - Decision Trees
    - K Nearest neighbor
  - Evaluate all the scores



Results

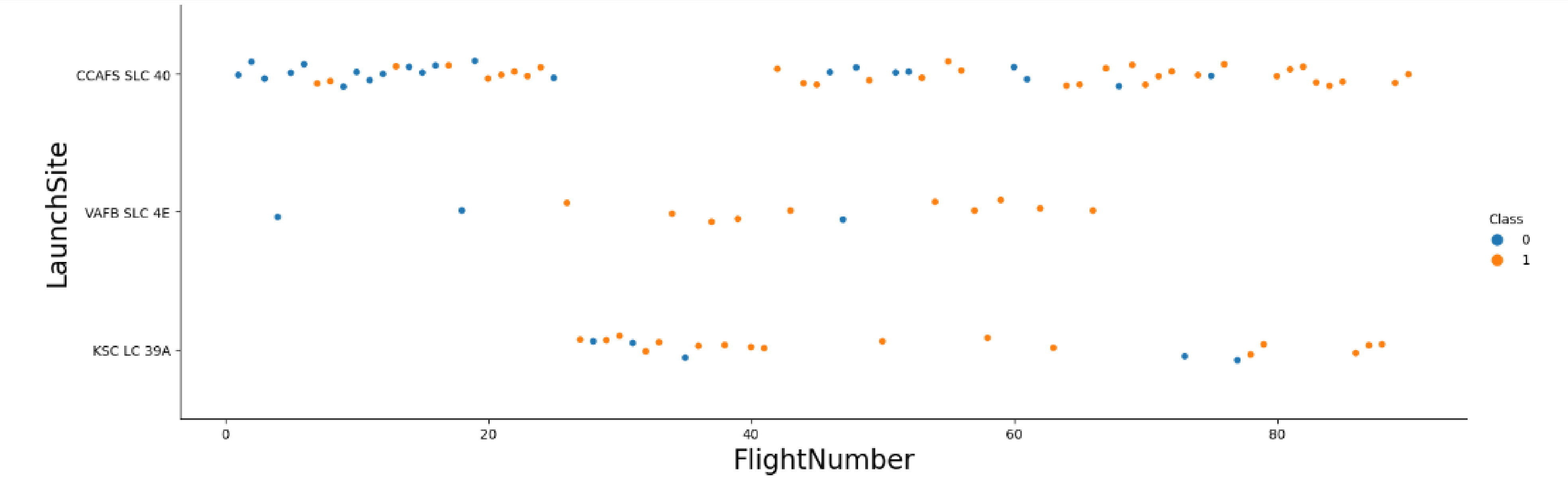
- EDA – Visualization
- EDA – SQL
- Interactive Visualizations
- Machine learning

Section 2

Insights drawn from EDA

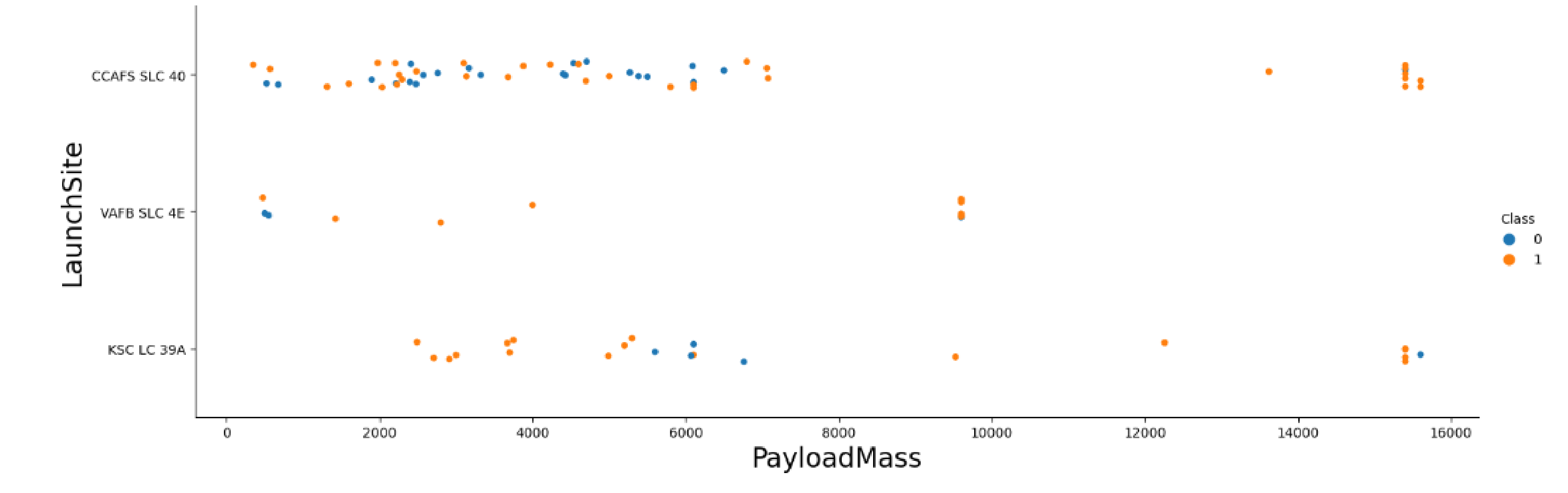
Flight Number vs. Launch Site

- Here you can see Flight Numbers v Launch Site
  - As the number of flights the success rate as increased
  - Most of all CCAFS SLC 40 early flights were unsuccessful



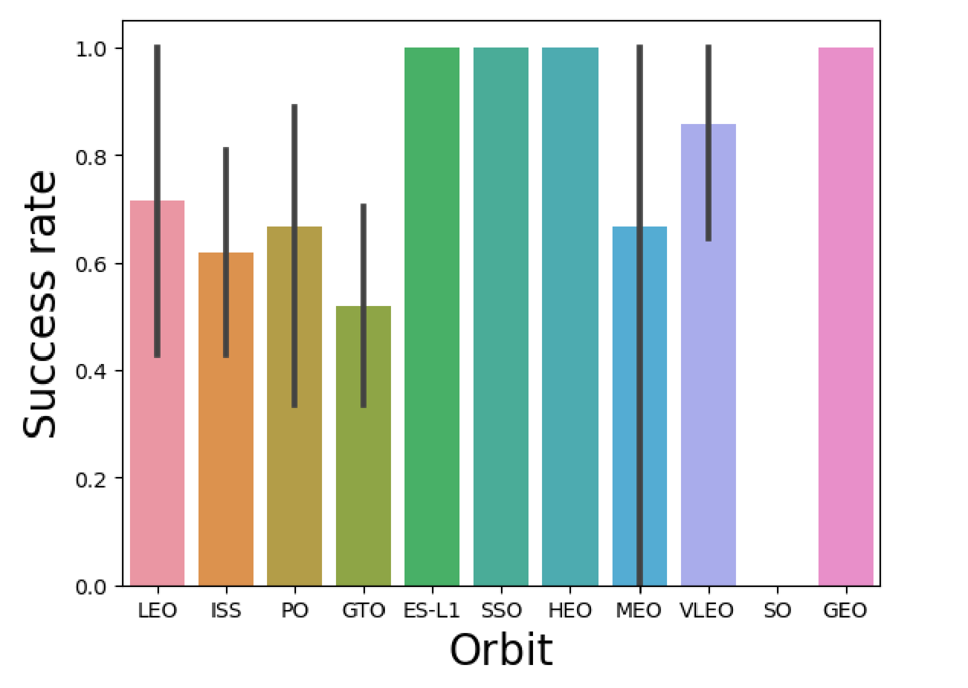
Payload vs. Launch Site

- Here you can see Payload v Launch Site
  - Most the data shows that there is correlation between Launchsites and payloadmass



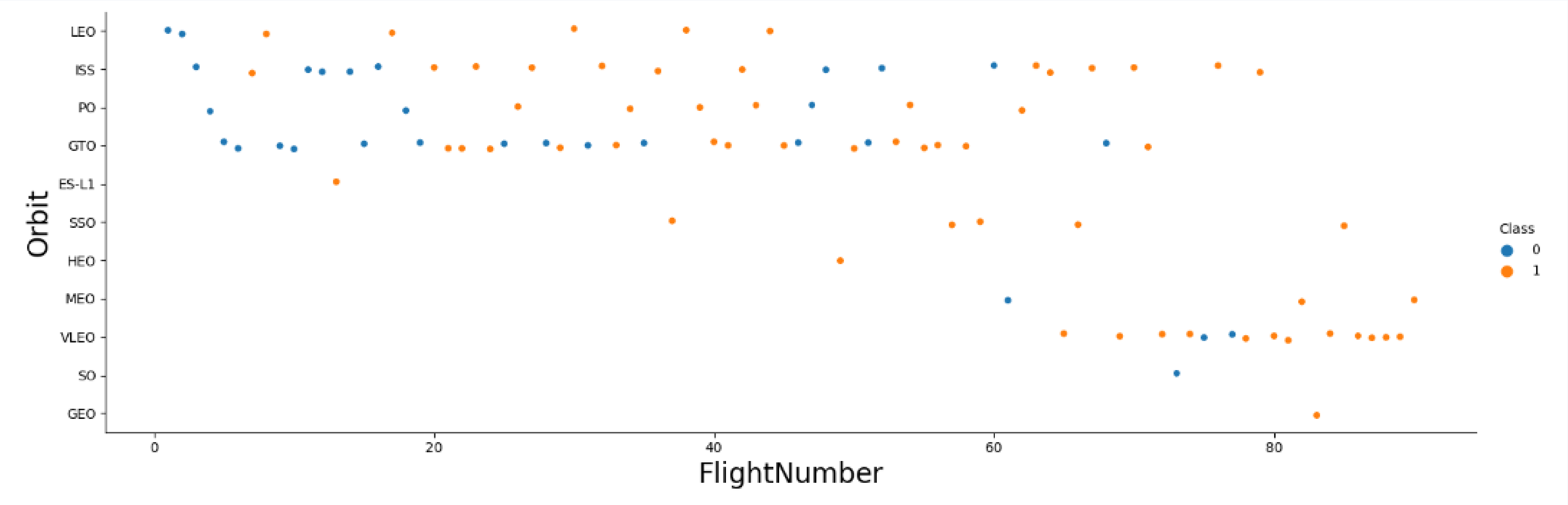
Success Rate vs. Orbit Type

- Here you can see Flight Numbers v Launch Site
  - As we can see Orbits that were 100% successful are
    - SSO
    - ES-L1
    - HEO
    - GEO
  - Orbits with 0%
    - SO



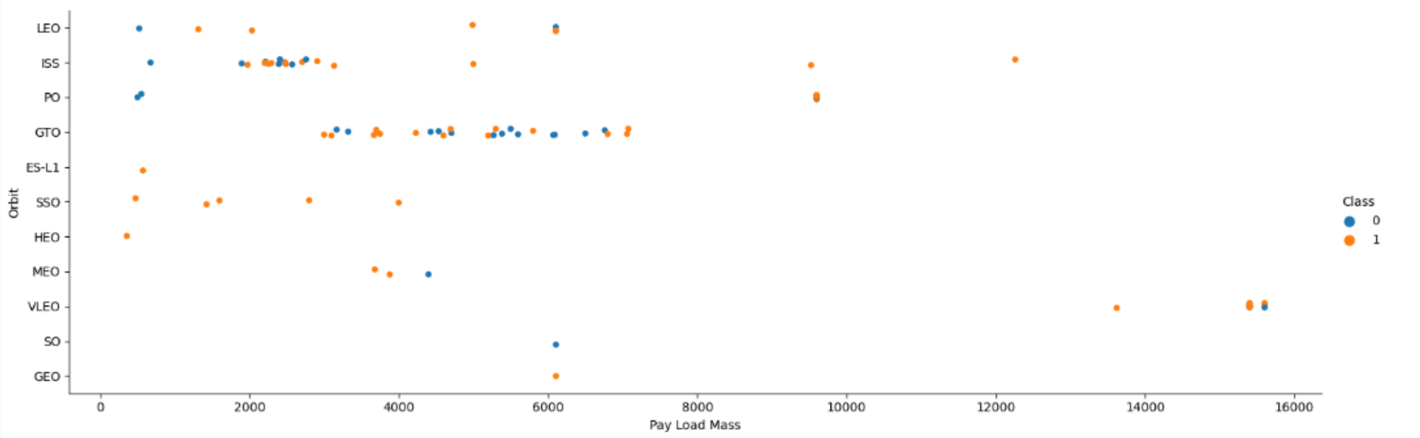
Flight Number vs. Orbit Type

- Here you can see Flight Numbers v Launch Site
  - Here is better visualization of all the 100% orbit success rate have only one launch
  - SSO is pretty impressive cause it has 5 launches



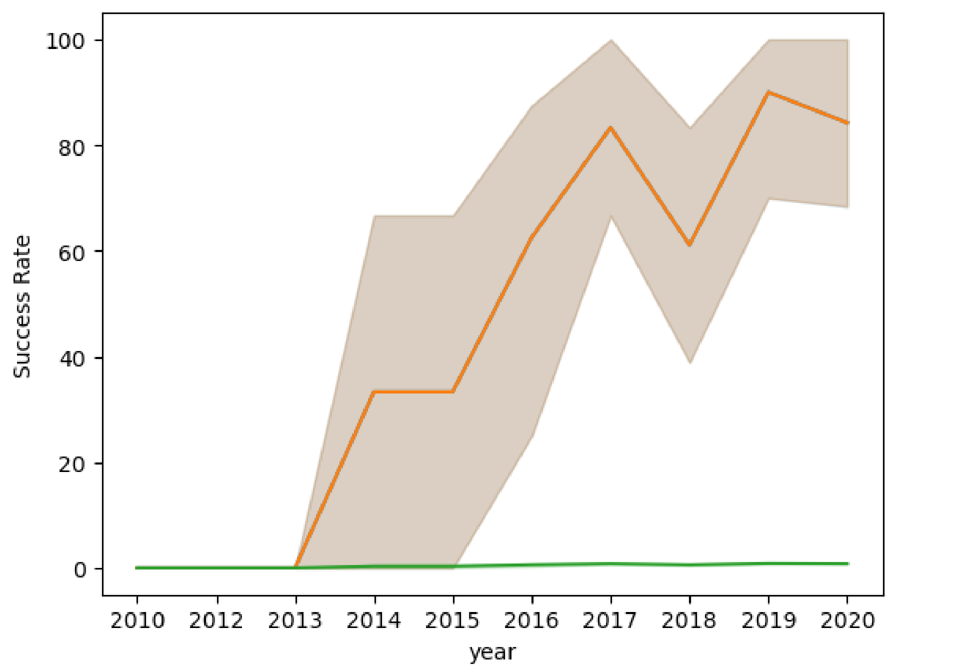
Payload vs. Orbit Type

- Here you can see Payload v Orbit Type
    - Orbit such as ISS, PO have more success with high payloadmass but the sample there is low
    - GTO is undetermined with the relation of success rate orbit and payloadmass



Launch Success Yearly Trend

- Here you can see Success Rate v Year
  - From 2010 to 2013 success was 0
  - After 2013, success rate started to increase
  - There was a dip in success 2018



All Launch Site Names

Below are all the launch sites in SpaceX

Only showing unique launch site by using DISTINCT

Task 1

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

```
%sql select DISTINCT LAUNCH_SITE as "Launch_Sites" from SPACEX;
```

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

Launch Site Names Begin with 'CCA'

- Displaying 5 launch sites that begin with CCA within the dataset.
- By using 'CCA%' that matched all the launch sites that start with CCA
- Limit 5 – only shows 5 launch sites

Task 2

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

```
%sql select * from SPACEX where LAUNCH_SITE like 'CCA%' limit 5;
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Used the SUM to add all the Payload Mass for NASA (CRS) Customer

Task 3

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

```
%sql select SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass" from SPACEX where CUSTOMER ='NASA (CRS)'
```

Total Payload Mass
45596

Average Payload Mass by F9 v1.1

- Average payloadmass carried by booster version F9 v1.1, used the AVG command with where Booster_Version = 'F9 v1.1'

Task 4

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

```
%sql select AVG(PAYLOAD_MASS__KG_) as "Average Payload Mass" from SPACEX where BOOSTER_VERSION = 'F9 v1.1';
```

Average Payload Mass
2928

First Successful Ground Landing Date

- MIN was used to the find the minimum date – which in this case is 2015-12-22
- Where was used to find Success ground pad

Task 5

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

Hint:Use min function

```
%sql select min(DATE) as Date from SPACEX where LANDING__OUTCOME = 'Success (ground pad)'
```

DATE
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Selected the Booster_Version column in the data

Used the where landing outcome was success – drone ship and where the payload was between 4000-6000

Task 6

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 SPACEX 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

Total Number of Successful and Failure Mission Outcomes

- Selected the Mission Outcome column from the database and then grouped them while counting them.

Task 7

List the total number of successful and failure mission outcomes

```
%sql select MISSION_OUTCOME, count(*) as Count from SPACEX group by MISSION_OUTCOME order by MISSION_OUTCOME
```

mission_outcome	COUNT
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- Selected the Booster_Version (unique) where the payload_mass__KG = the max payload

Task 8

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

```
%sql select DISTINCT BOOSTER_VERSION from SPACEX where PAYLOAD_MASS__KG_ =(select MAX(PAYLOAD_MASS__KG_) from SPACEX)
```

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

2015 Launch Records

- Selected Date, booster_version and launch_site
- Used the where command to find date=2015 and the landing_outcome =failure –drone ship

Task 9

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

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%sql select DATE, BOOSTER_VERSION, LAUNCH_SITE from SPACEX where year(DATE) = '2015' and LANDING__OUTCOME = 'Failure (drone ship)'
```

DATE	booster_version	launch_site
2015-01-10	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	F9 v1.1 B1015	CCAFS LC-40

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

Selected Landing_Outcome, and also counted the landing_outcome

Used where command to find date between 2010-06-04 and 2017-03-20

Grouped the finding and ordered the finding in descending order

Task 10

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

```
%sql select LANDING__OUTCOME, COUNT(LANDING__OUTCOME) as LandingCount from SPACEX where DATE BETWEEN '2010-06-04' and '2017-03-20' GROUP BY LANDING__OUTCOME ORDER BY COUNT(LANDING__OUTCOME) DESC;
```

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

Section 3

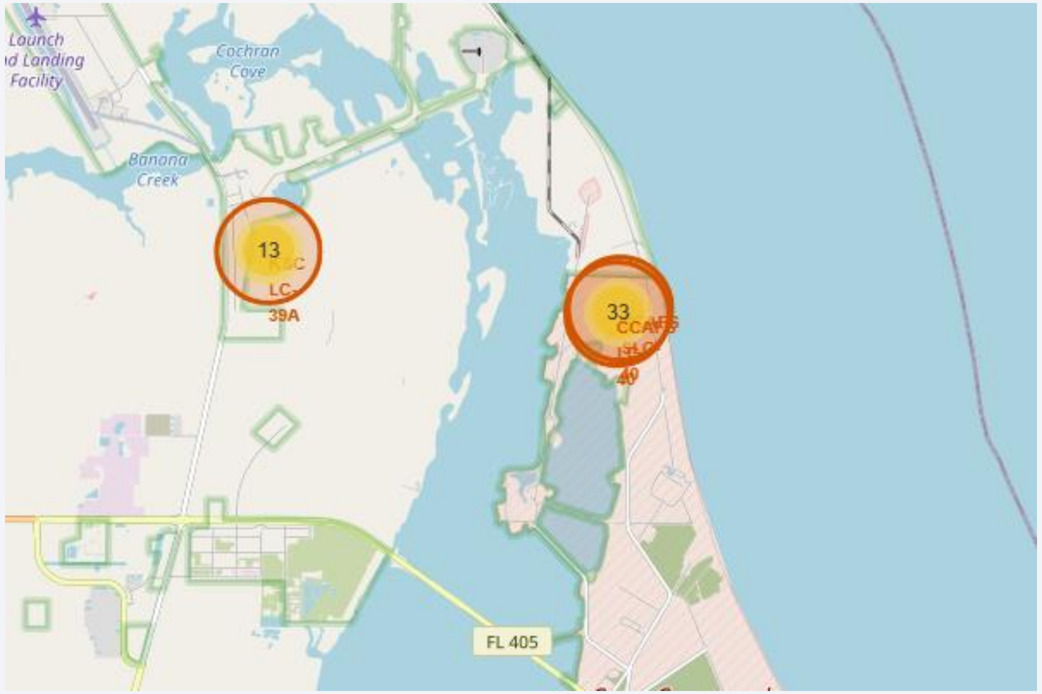
Launch Sites Proximities Analysis

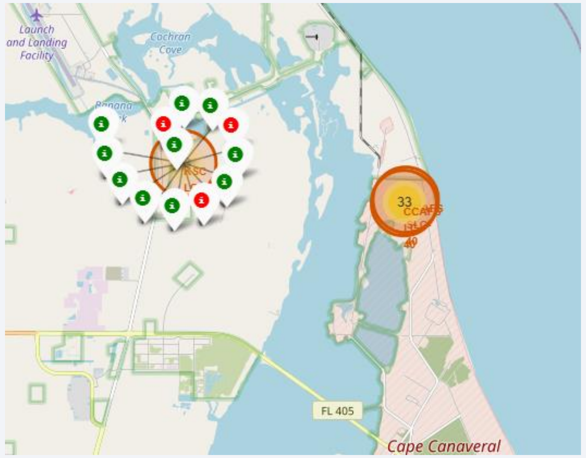
Launch Site locations

Here shows the launch site for Space X

As see they are on both sides of the United states.
3 in the East coast and 1 in the West coast

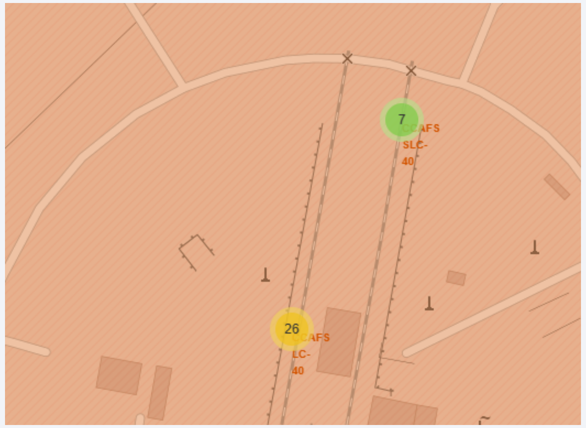
Success and Failed Launches



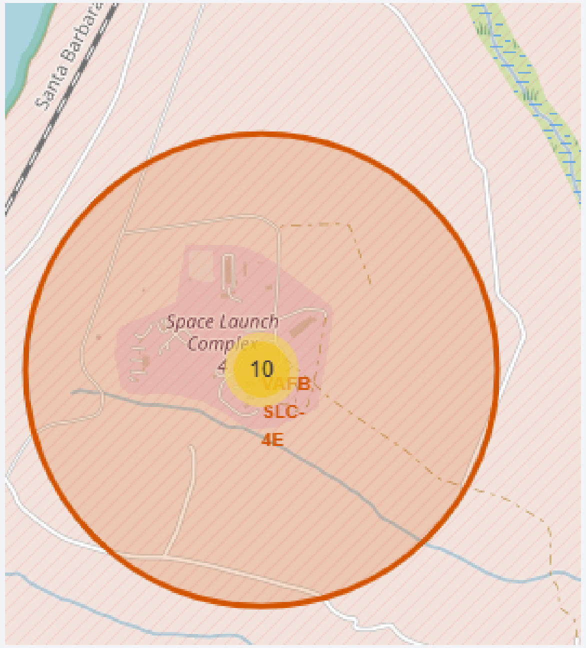


We can see the launch sites and there success rates. Green shows successful launches and Red shows failed launches.

Locations near the launch sites



Here we can see the railways near our launch sites

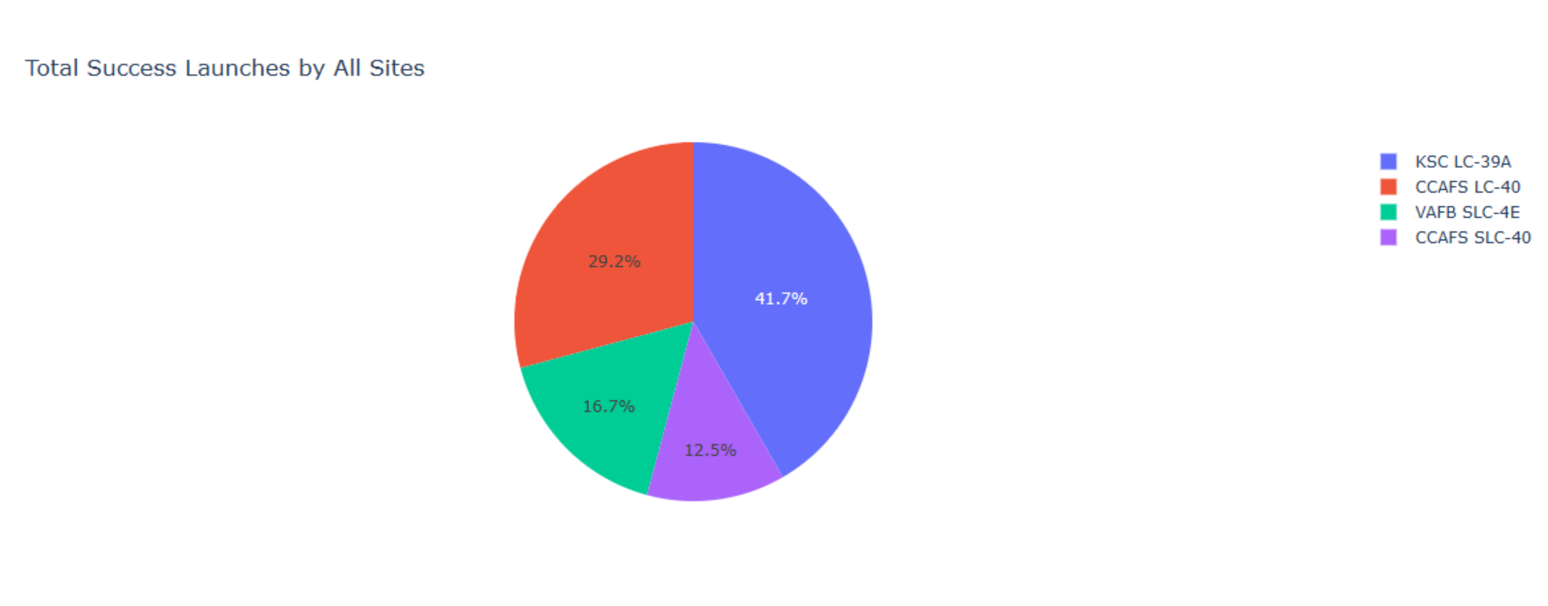


Section 4

Build a Dashboard with Plotly Dash

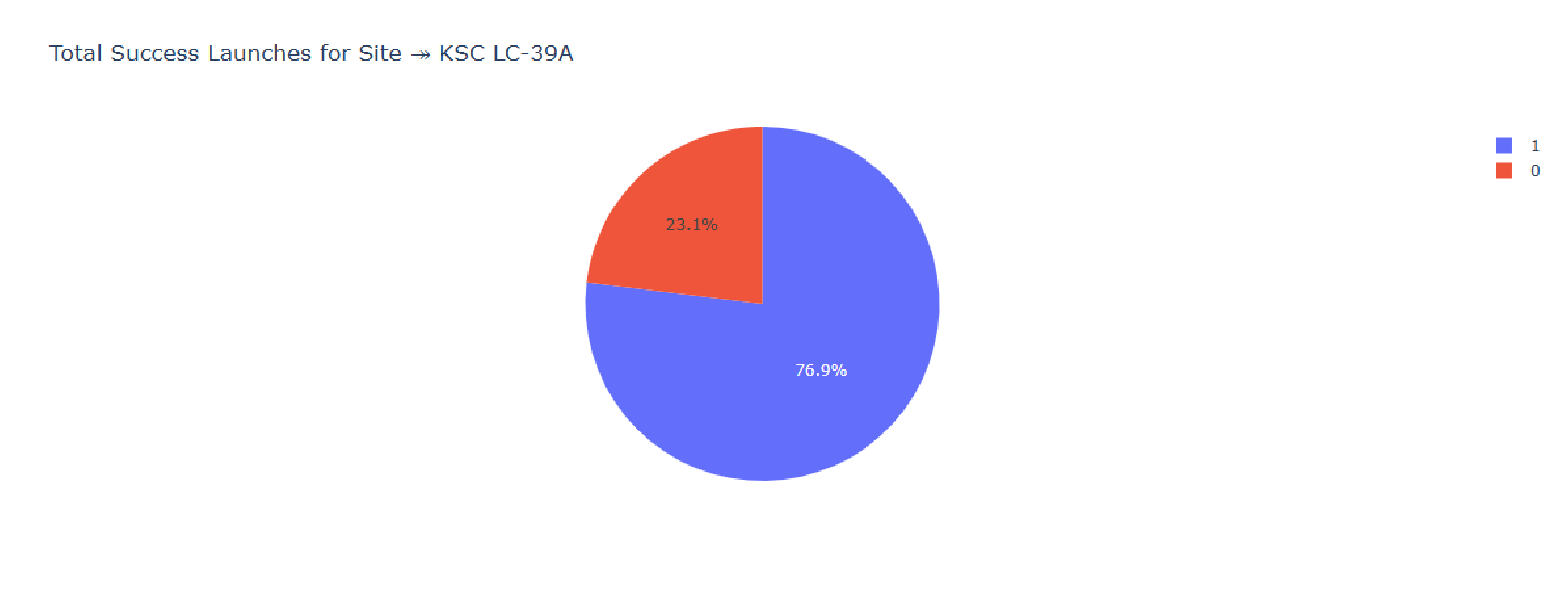
Launch Success Counts for all sites

- From here we can see that KSC LC – 39A has the most success launches by all sites.



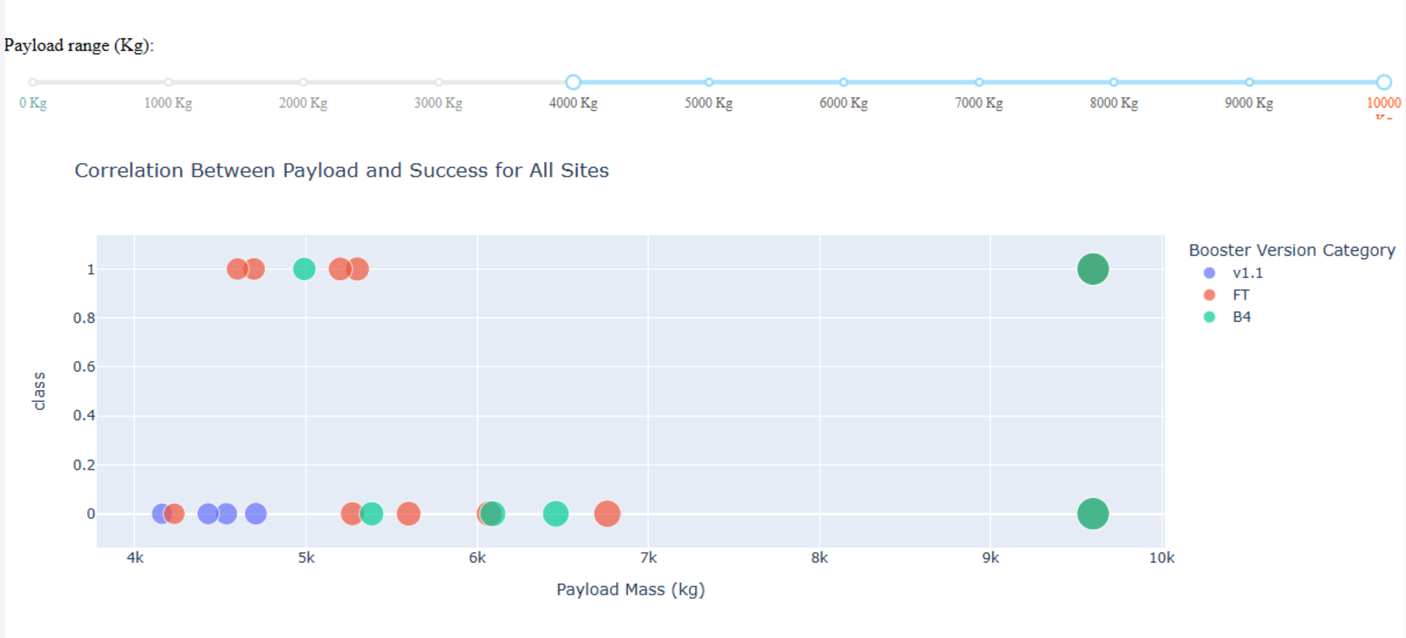
Highest Launching Site Ratio

KSC LC – 39 is the highest launching site and here we can see why. There success rate is at 76.9%



Payload vs. Launch Outcome scatter plot for all sites

- We can use different payload and see the outcomes changes for all the different booster version

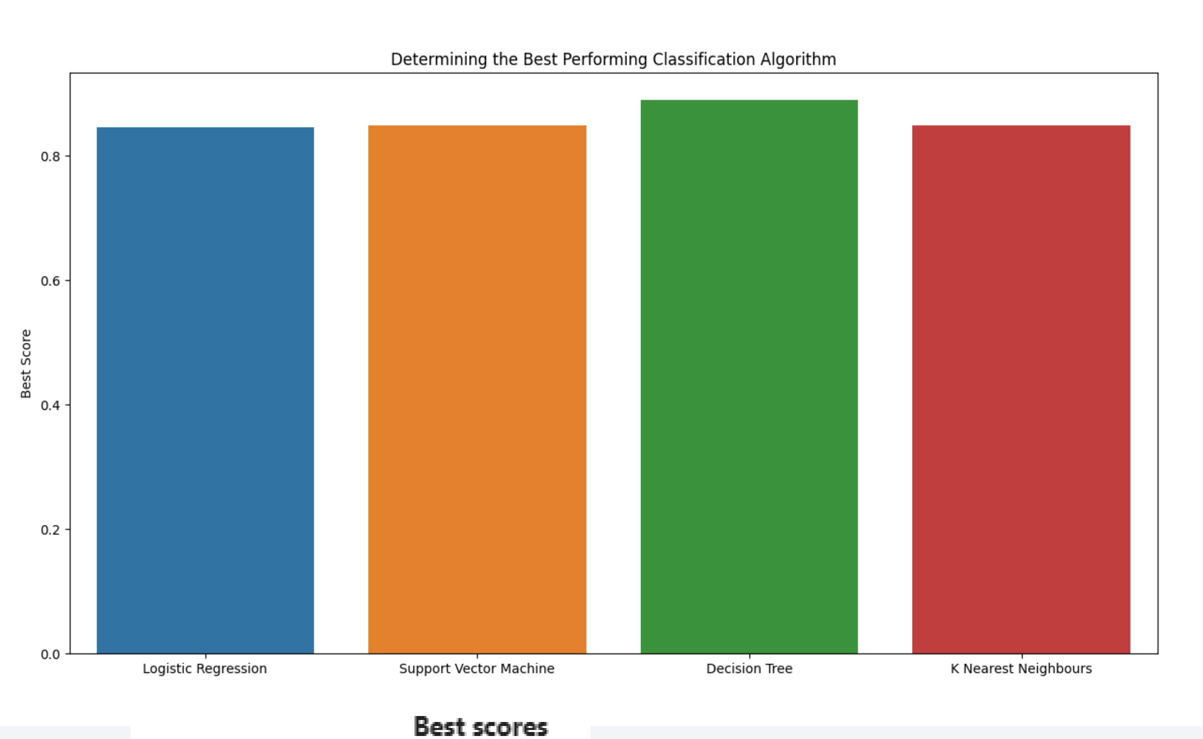


Section 5

Predictive Analysis (Classification)

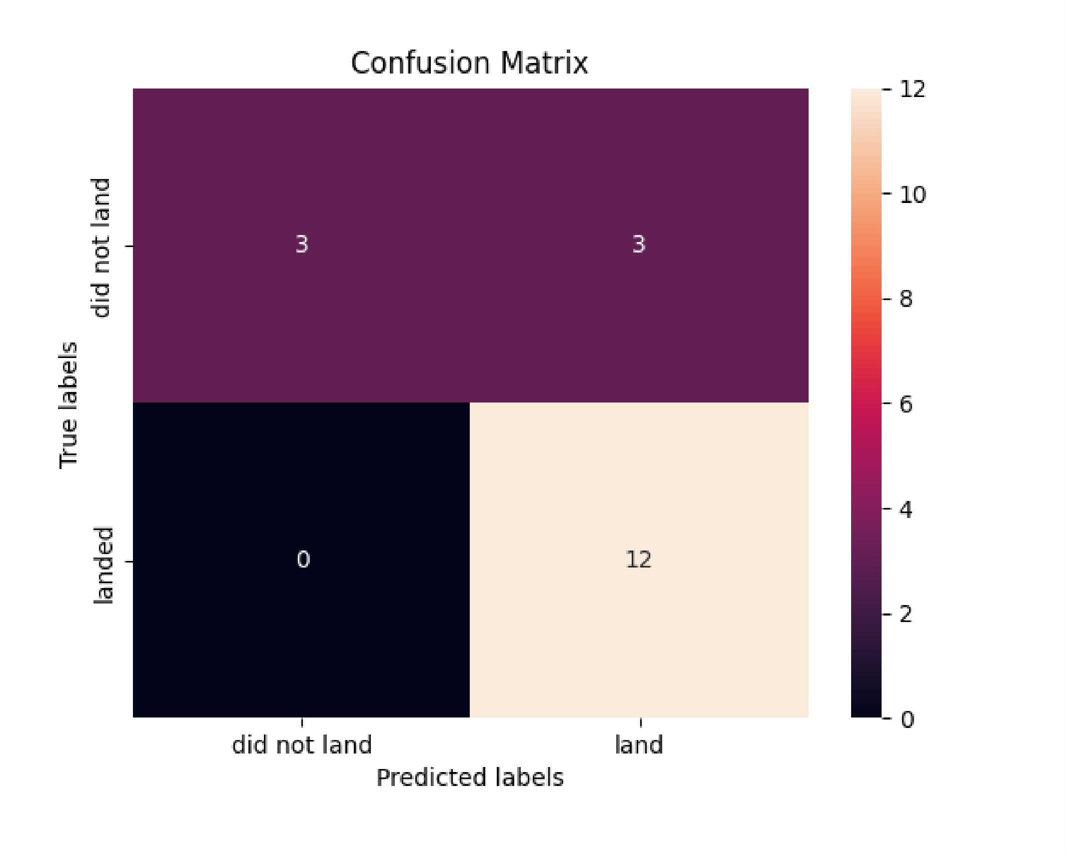
Classification Accuracy

- As you can see from these accuracy, that decision tree is the best option. It has the highest classification accuracy



	Best scores
Logistic regresssion	0.846429
SVM	0.848214
Decision tree	0.889286
KNN	0.848214

Confusion Matrix



- Confusion Matrix explains the following
  - 12 successful landings when true labels is landed
  - 0 when predicted did not land and true labels is landed

Conclusions

- 2010 – 2013 there were 0 successful landing
- After 2013 started the successful pattern
- 2018 dipped and saw more then usual unsucessful launches
- 4 100% orbits – ES-L1, GEO, HEO and SSO
- Launch site KSC LC-39 most successful launches
- As the numbers of flights increased the success rate increased.

Appendix

- https://github.com/sukhwinder2392/SpaceX

Thank you!