IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Grzegorz Rudecki
2022-01-31

Outline

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

Executive Summary

- The aim of this project is to estimate the cost of a launch of SpaceX Falcon 9. We are focused to determine if the first stage will land, because the reuse of the first stage creates a business advantage, as it can lower the cost dramatically from 165 millions $ to 62 millions $ per launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. The data was gathered from SpaceX REST API and Wikipedia (Web Scraping). We have performed Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models. Therefore, we have explored data sources with SQL and did visualization to show relationships between variables and find patterns. Finally we have managed to create Machine Learning models to predict future outcomes.
- We have concluded from the results that the success of landing is dependent on the launch site, the orbit, the mass of payload and some other technical factors. The success rate has been increasing since 2013.

Introduction

- On July 20, 1969, American astronauts Neil Armstrong and Edwin Aldrin became the first humans ever to land on the moon. The space rocket industry has made a huge progress since then. However, the cost of one rocket launch is still consider highly priced. One of the reason are not reusable rockets. SpaceX has been trying to reduce the cost of one launch by proposing reusable, the most expensive, first stage of the launch. The analysis and making predictions about this feature could be advantageous for companies in the space industry.
- The problem that we want to answer is what factors determine the success of a rocket landing and dependance between these variables based on available data. The results could be used to determine a probability of a success or a failure of a launch and in principle, help to keep increasing the success rate with this new knowledge.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data was gathered from SpaceX REST API and Wikipedia (Web Scraping)
- Perform data wrangling
  - The data was cleaned (missing values) and unified. The data was also limited to our needs.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

Data sets were collected via API given by SpaceX itself and Webscrapping from Wikipedia.org

Sources:

- SpaceX API `https://api.spacexdata.com/v4/`

- `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021` `https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922`

https://github.com/grzegorzrud/ibm_data_science/blob/55fe136901646d47d4175d617b700ba936ba6cf8/Week%201%20-%20Data%20Collection%20API.ipynb

Data Collection – SpaceX API

- Getting response from API
- Converting response to json file
- Clean the data with custom functions
- Create dictionary and finally dataframe
- Filter dataframe to our needs (limiting data to Falcon 9)
- Export dataframe to CSV file

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [7]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [8]: response = requests.get(spacex_url)
```

```
In [12]: # Use json_normalize meethod to convert the json result int.json_normalize()o a dataframe
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

```
In [19]: # Call getLaunchSite
getLaunchSite(data)
```

```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.047721
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.047721
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.047721
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.047721
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

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

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

https://github.com/grzegorzrud/ibm_data_science/blob/55fe136901646d47d4175d617b700ba936ba6cf8/Week%201%20-%20Data%20Collection%20with%20Web%20Scraping.ipynb

Data Collection - Scraping

- Getting response from HTML
- Using BeautifulSoup to parse response
- Finding tables
- Getting column names
- Creating dictionary
- Extracting rows to dictionary
- Converting dictionary to dataframe
- Exporting dataframe to CSV file

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

```
page = requests.get(static_url)
page.status_code
```

```
200
```

```
soup = BeautifulSoup(page.text, 'html.parser')
```

```
html_tables = soup.find_all('table')
```

```
temp = first_launch_table.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

```
[16]: 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']=[]
```

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

Out[18]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example. We have converted all outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Perform exploratory data analysis on dataset

Calculate the number of launches per site

Calculate the number of launches from orbits

Calculate the number of mission outcomes

Create landing outcome label for all cases (success = 1, failure = 0)

Export data to CSV file

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

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

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

GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
HEO       1
SO        1
GEO       1
ES-L1     1
Name: Orbit, dtype: int64
```

```
# 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
```

```
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

We can use the following line of code to determine the success rate:

```
df["Class"].mean()

0.6666666666666666
```

https://github.com/grzegorzrud/ibm_data_science/blob/55fe136901646d47d4175d617b700ba936ba6cf8/Week%201%20-%20EDA.ipynb

EDA with Data Visualization

I have used 3 different types of graph: scatter, bar, and line to show trend.

Scatter Graphs to show how much one variable is affected by another, to determine if there is a correlation and its type (linear/exponational etc.)

1. Flight Number vs. Launch Site
2. Flight Number vs Payload Mass
3. Payload vs. Launch Site
4. Orbit vs. Flight Number
5. Payload vs. Orbit Type
6. Orbit vs. Payload Mass

Bar Graph to visually check if there are any relationship between success rate and orbit type.

LinePlot Graph with x axis to be year and y axis to be average success rate, to get the average launch success trend. We can also to approximate the future from this line.

https://github.com/grzegorzrud/ibm_data_science/blob/55fe136901646d47d4175d617b700ba936ba6cf8/Week%202%20-%20EDA%20with%20Visualization.ipynb

EDA with SQL

I performed Explorary Data Analysis with SQL, in order to gather information about the datasets.

https://github.com/grzegorzrud/ibm_data_science/blob/55fe136901646d47d4175d617b700ba936ba6cf8/Week%202%20-%20EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

An interactive map was created with Folium library:

Folium is a powerful Python library that helps you create several types of Leaflet maps. By default, Folium creates a map in a separate HTML file. Since Folium results are interactive, this library is very useful for dashboard building.

I use the latitude and longitude coordinates for each launch, in order to add Circle Marker with a label on the map.

Cluster object of markers was added to show launch outcomes, with Green being successful and Red unsuccessful launch.

Additionally, polyline objects were created to show distance of the launches from various landmarks.

https://github.com/grzegorzrud/ibm_data_science/blob/55fe136901646d47d4175d617b700ba936ba6cf8/Week%203%20-%20Interactive%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

Dash *is a python framework created by plotly for creating interactive web applications. Dash is written on the top of Flask, Plotly. js and React. js. With Dash, you don't have to learn HTML, CSS and Javascript in order to create interactive dashboards, you only need python.*

I used this library, in order to create a dashboard with Pie Chart and Scatter Graph, with interactive filters.

Pie chart is showing the total success rate of all sites or filter by one launch site via combobox.

Scatter Graph is showing the correlation between Payload and Success for the different Booster Versions. The user can narrow down the range of Payload with a slider, to zoom-in/out graph for better visibility.

https://github.com/grzegorzrud/ibm_data_science/blob/55fe136901646d47d4175d617b700ba936ba6cf8/Dashboard%20plotly%20(2).py

Predictive Analysis (Classification)

1. Load data into Pandas dataframe and Numpy
2. Standarize and Transform data
3. Split data into train and test datasets
4. List the machine learning algorithms
5. Set parameters and train algorithm with GridSearch analysis for each algorithm
6. Check accuracy for each algorithm
7. Plot confusion matrix for each algorithm

https://github.com/grzegorzrud/ibm_data_science/blob/55fe136901646d47d4175d617b700ba936ba6cf8/Week%204%20-%20Machine%20Learning%20Prediction.ipynb

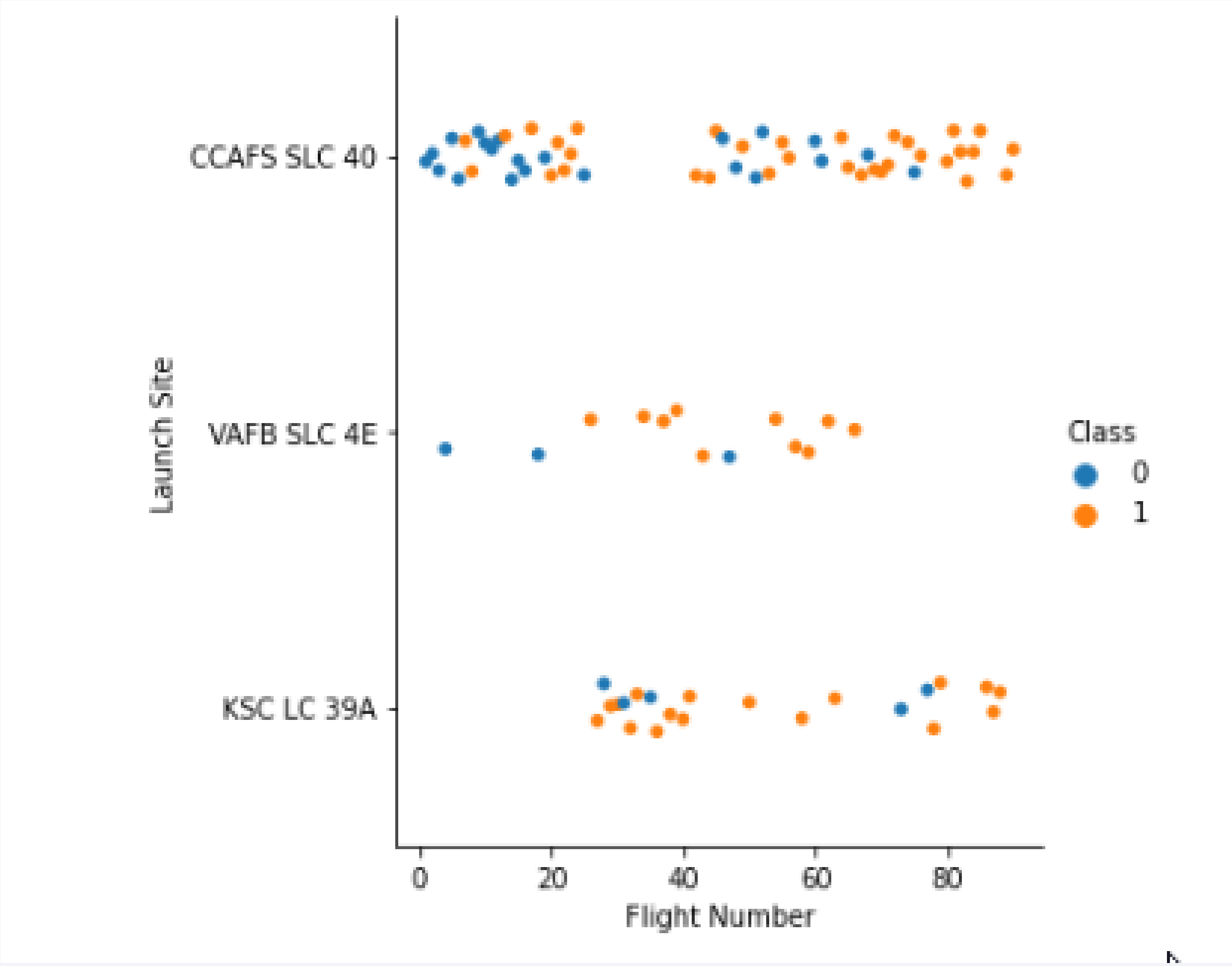
Results

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

Section 2

Insights drawn from EDA

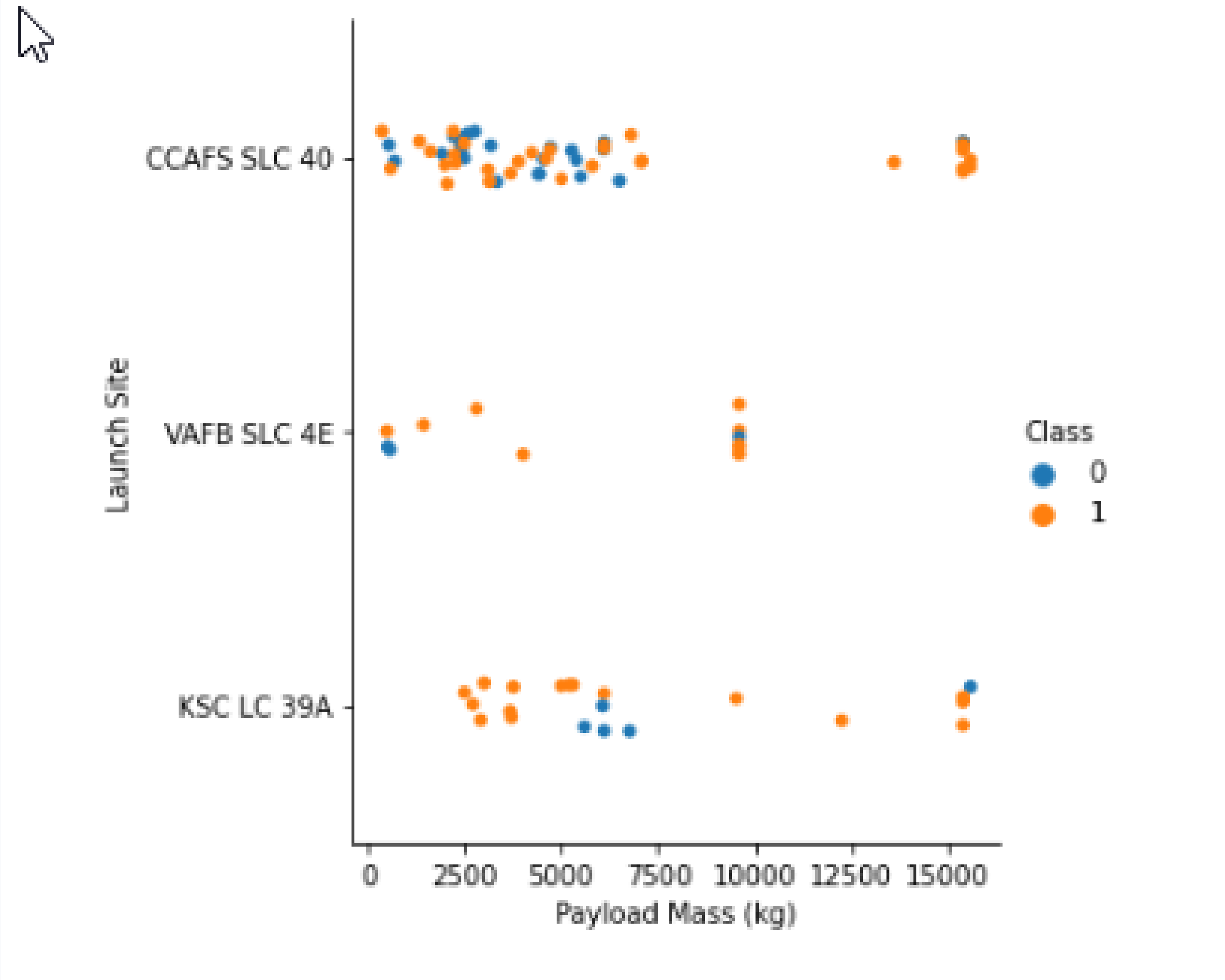
Flight Number vs. Launch Site



The success rate is increasing with flight number, espacially after ~30th flight.

Launch site seems to be less relevant than flight number.

Payload vs. Launch Site

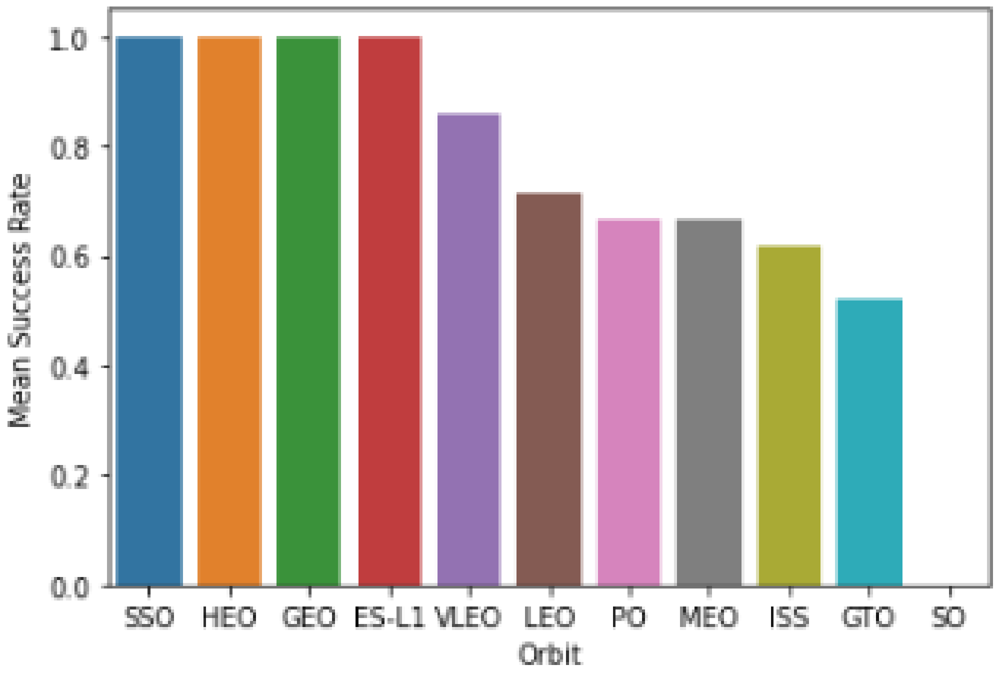


There is no clear pattern, but we can see that when the payload is higher the success rate is also higher.

However in this area we have less data points.

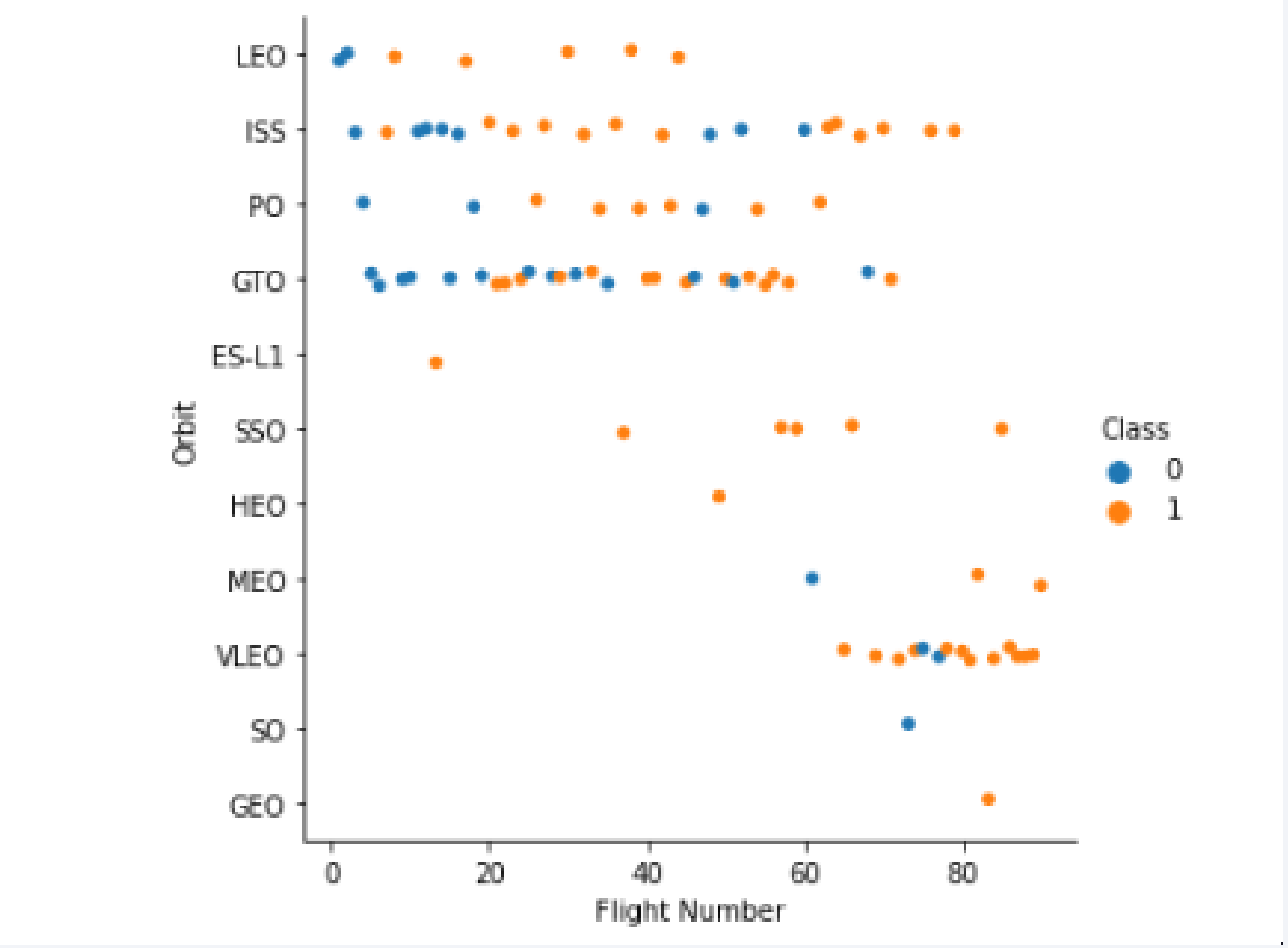
Success Rate vs. Orbit Type

```python
success_rate = df.groupby('Orbit')['Class'].mean()
success_rate.sort_values(ascending = False, inplace = True)
sns.barplot(x = success_rate.index, y = success_rate, palette = 'tab10')
plt.xlabel('Orbit')
plt.ylabel('Mean Success Rate')
plt.show()
```



There are differences between success rate by orbits. The highest success rate have SS0, HEO, GEO, ES-L1 and the lowest: ISS or GTO.

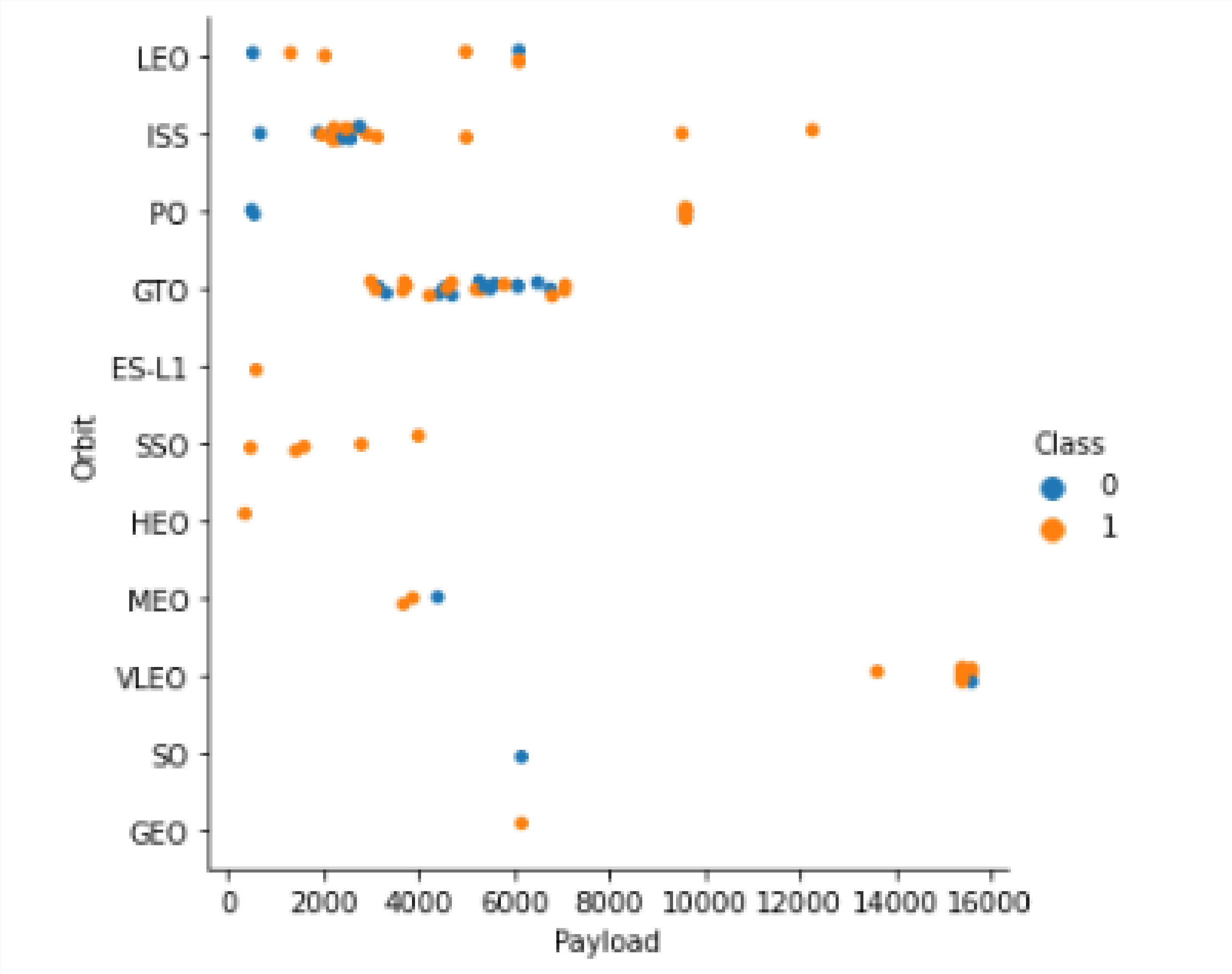
Flight Number vs. Orbit Type



The success rate is increasing with flight number, espacially after ~30th flight.

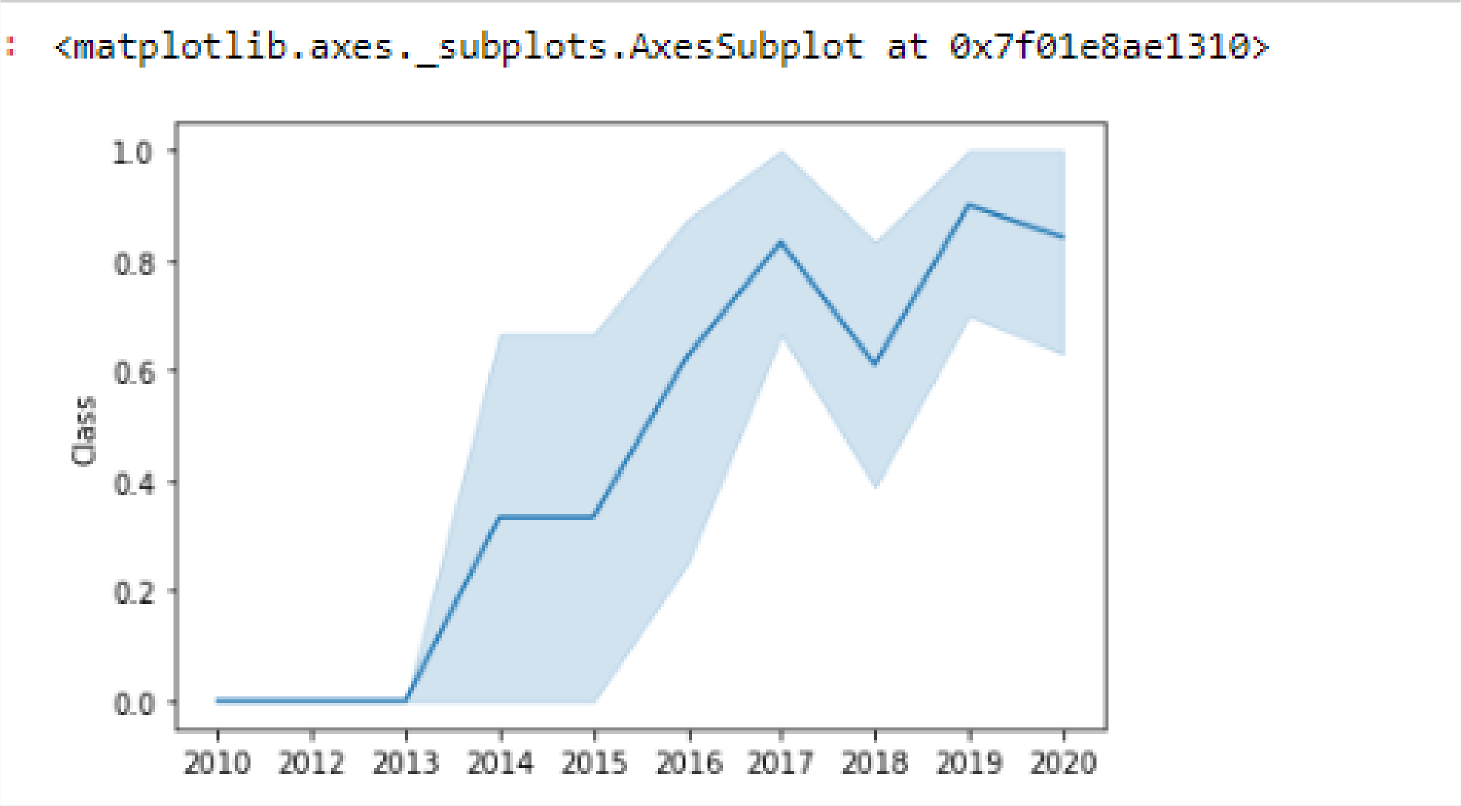
Orbit type seems to be less relevant than flight number, but we can see for example that LEO orbit after two failure, has only successful launches.

Payload vs. Orbit Type



We can see that heavy payloads cause problem to GTO, MEO and VLEO orbits, but not for LEO and ISS orbits.

Launch Success Yearly Trend



We can see that the success rate keep increasing since 2013 to 2017, with some dips after, but general trend stays.

All Launch Site Names

```
%sql select distinct launch_site from SPACEXTBL;

 * ibm_db_sa://wqr92448:***@55fbc997-9266-4331-afd3-888b0
Done.
```

5]:

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

Using *distinct*, I was able to list all unique site names within the table.

We can see that there are only 4 launch sites.

Launch Site Names Begin with 'KSC'

Task 2

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

In [6]:
```
%sql select * from SPACEXTBL where LAUNCH_SITE like 'KSC%' limit 5
```

```
 * ibm_db_sa://wqr92448:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/BLUDB
Done.
```

Out[6]:

DATE	Time (UTC)	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	Landing _Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Using *limit*, I was able to list 5 records from the table where launch site starts with KSC. I used *like condition* to filter out the data.

We can see that there are in fact at least 5 records for that condition.

Total Payload Mass

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

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER = 'NASA (CRS)'
```

```
 * ibm_db_sa://wqr92448:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.
Done.
```

[7]:

1
45596

Using *sum*, I was able to calculate total payload mass from the table for customer NASA (CRS).

We can see that total mas is equal to 45596 kg.

Average Payload Mass by F9 v1.1

```
]: %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'

    * ibm_db_sa://wqr92448:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.dat
   Done.

[8]:
```

1
2928

Using *avg*, I was able to calculate average payload mass from the table for booster version F9 v1.1.

We can see that the average payload mass is equal to 2928 kg.

First Successful Ground Landing Date

```
%sql select min(DATE) from SPACEXTBL where "Landing _Outcome" = 'Success (ground pad)'

 * ibm_db_sa://wqr92448:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.ap
Done.

2]:
     1
2015-12-22
```

Using *min*, I was able to find first date of successful landing from ground pad.

We can see that the date is 22 December 2015.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select BOOSTER_VERSION from SPACEXTBL where "Landing _Outcome" = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
```

```
 * ibm_db_sa://wqr92448:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/BLUDB
Done.
```

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

Using *and*, I was able to filter the data for landind outcome as well as payload mass.

We can see that for the entered contidions, we have only four booster versions.

Total Number of Successful and Failure Mission Outcomes

In [19]:

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

```
 * ibm_db_sa://prd36480:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.
Done.
```

Out[19]:

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

```
%sql select case when MISSION_OUTCOME like '%Success%' then 'Success' else 'Failure' end, count(MISSION_OUTCOME) from SPACEXTBL GROUP BY  case when MISSION_OUTCOME like '%Success%' then 'Success' else 'Failure' end
```

```
 * ibm_db_sa://prd36480:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

20]:

1	2
Failure	1
Success	100

Using *group by* and *case when*, I was able to count all the outcomes.

There are 100 Success missions (99 success and one success unclear) and 1 failure.

Boosters Carried Maximum Payload

```
%sql SELECT distinct booster_version FROM spacextbl WHERE payload_mass__kg_ = (select max(payload_mass__kg_) FROM spacextbl)
```

```
 * ibm_db_sa://prd36480:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

1]:

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

Using *max* and *subquery*, I was able to list all booster version that were carried maximum payload.

There are several booster version that were carried maximum payload.

2017 Launch Records

```
In [32]: %sql SELECT {fn MONTHNAME(DATE)} as "Month", BOOSTER_VERSION, LAUNCH_SITE,landing__outcome FROM SPACEXTBL WHERE year(DATE) = '2017' AND \
         landing__outcome = 'Success (ground pad)' order by DATE

 * ibm_db_sa://prd36480:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

Out[32]:

Month	booster_version	launch_site	landing__outcome
February	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
May	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
June	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
August	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
September	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
December	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

Using *monthname* and *year*, I was able to list all lauches in 2017 with succesful landing_outcomes in ground pad, with month name.

We can see that there are 6 records, all in different month.

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

```
%sql select * from SPACEXTBL where landing__outcome like 'Success%' and (DATE between '2010-06-04' and '2017-03-20') order by date desc
```

```
 * ibm_db_sa://prd36480:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600	Polar LEO	Iridium Communications	Success	Success (drone ship)
2016-08-14	05:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-07-18	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	Thaicom 8	3100	GTO	Thaicom	Success	Success (drone ship)
2016-05-06	05:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	Success (drone ship)
2015-12-22	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

Using *sort* and *between*, I was able to list all lauches between given dates, in desceding order.

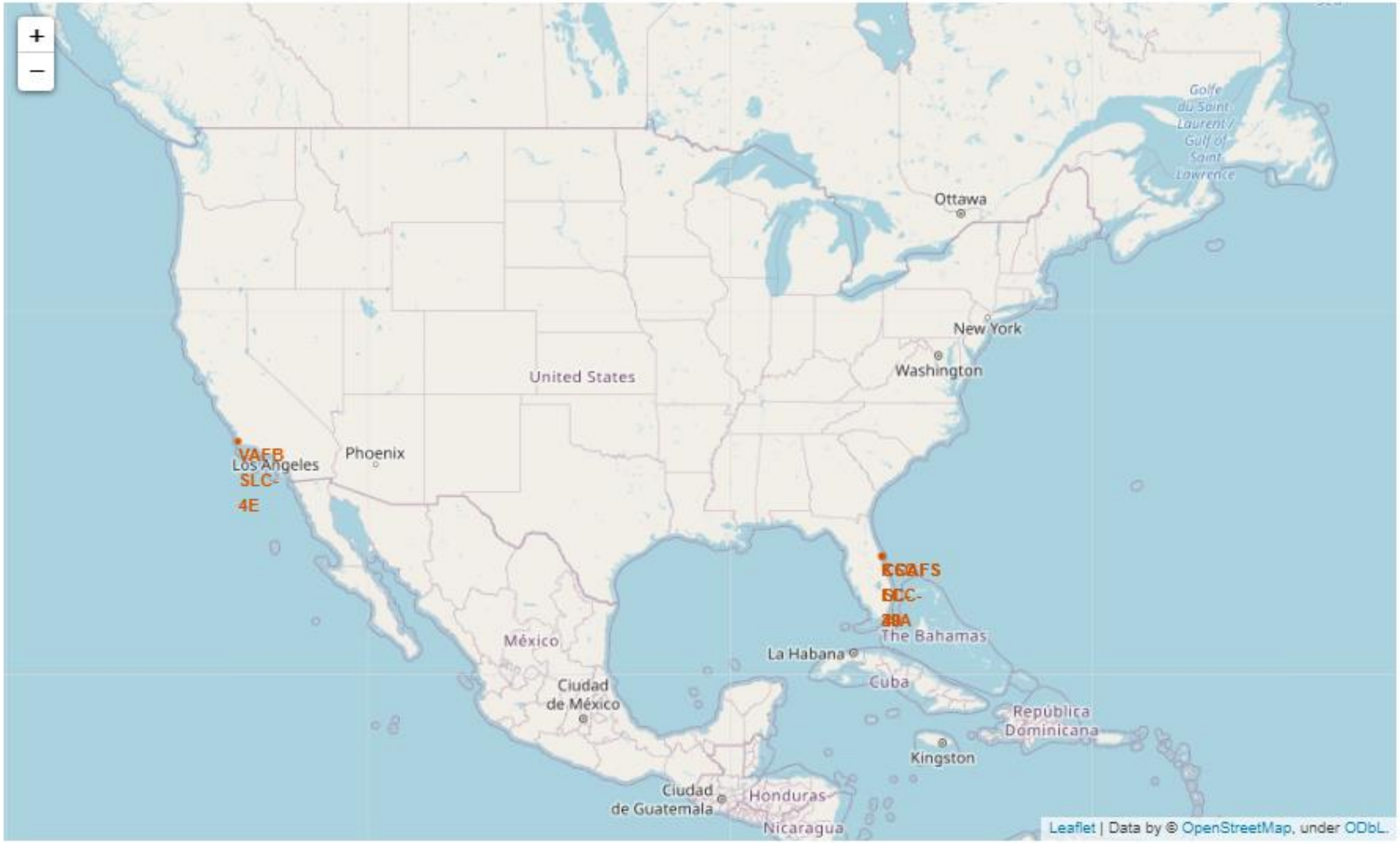
Section 3

Launch Sites Proximities Analysis

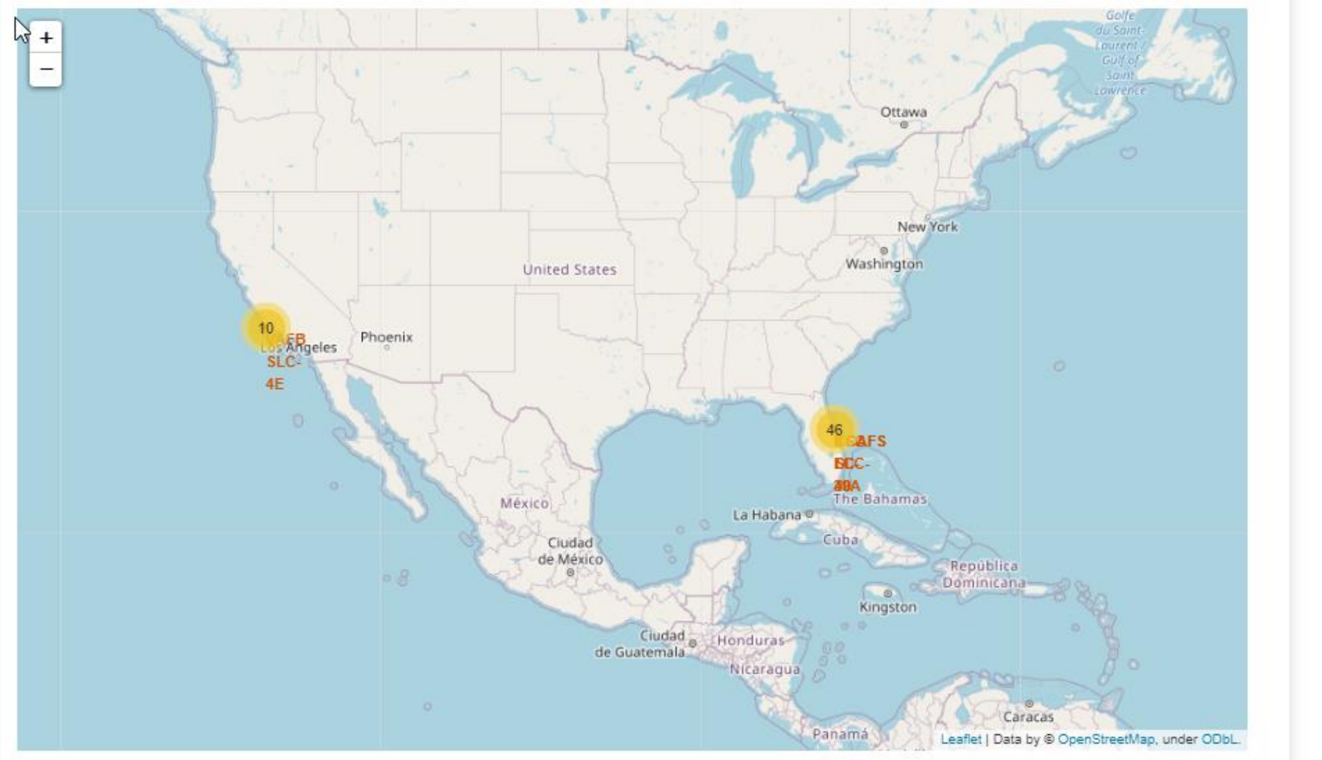
All Launch Sites on Folium Map

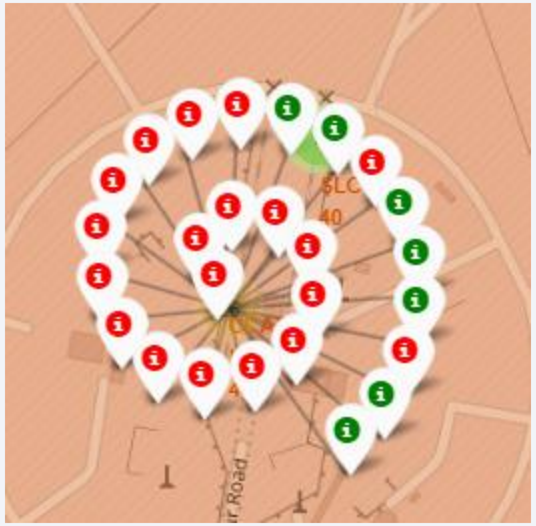
We can see that all Launch Sites are on the US coasts: California (x1) and Florida (x3).

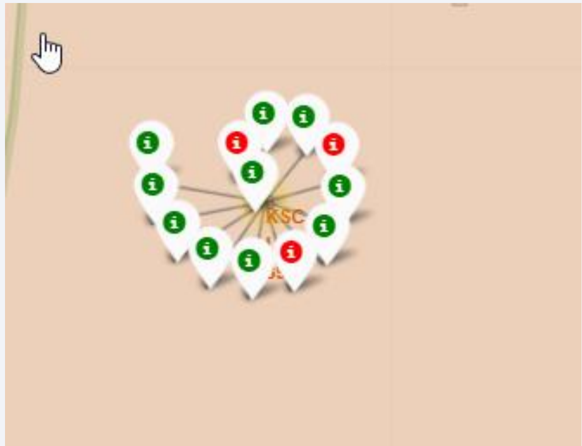
```
In [9]: site_map
Out[9]:
```



Lauch records – color labeled on Folium Map







Green marker shows succcessful launches and Red Marker shows failure launches.

KSC LC-39A has the best successful rate.

Launch Sites distances on Folium Map

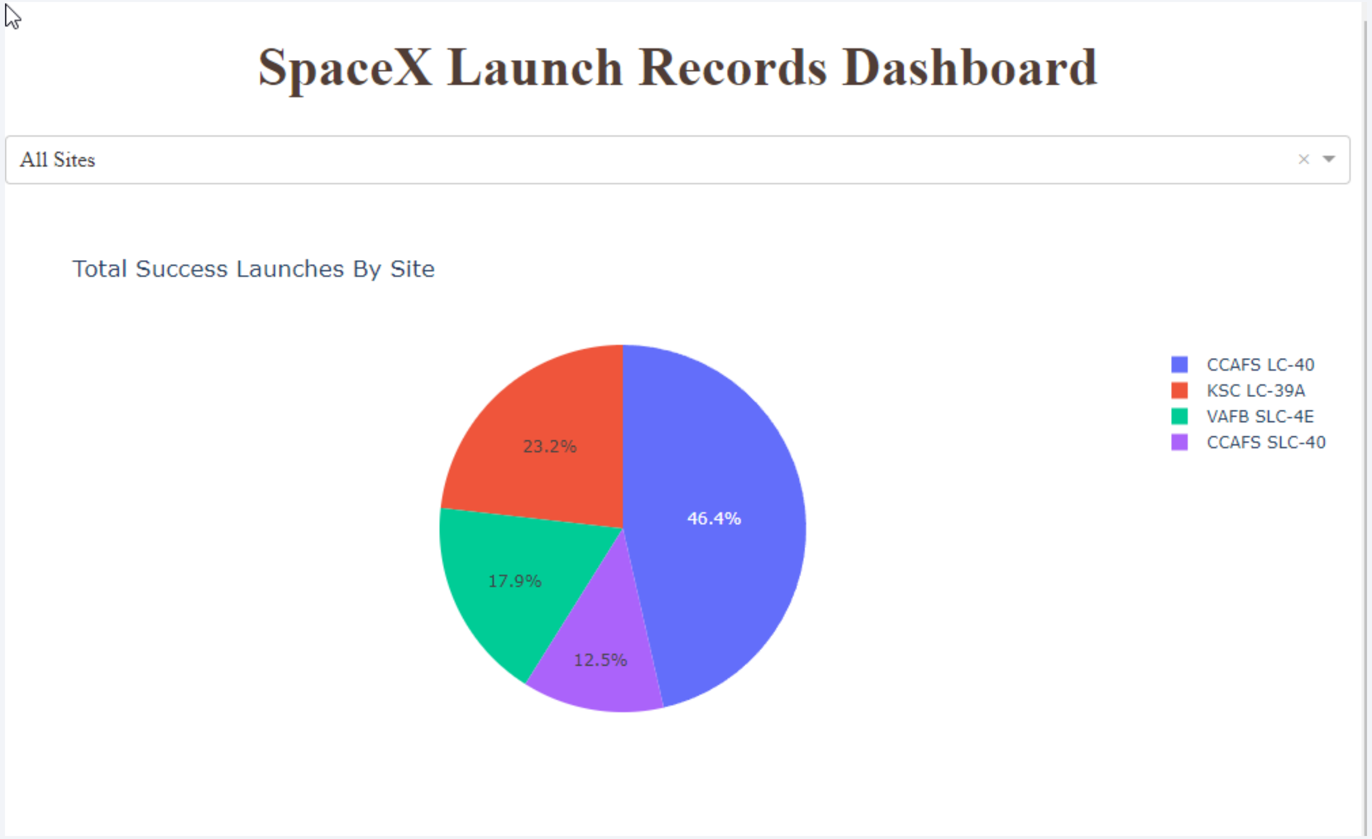
Example of calculating distance from the coast to one of the launch site:

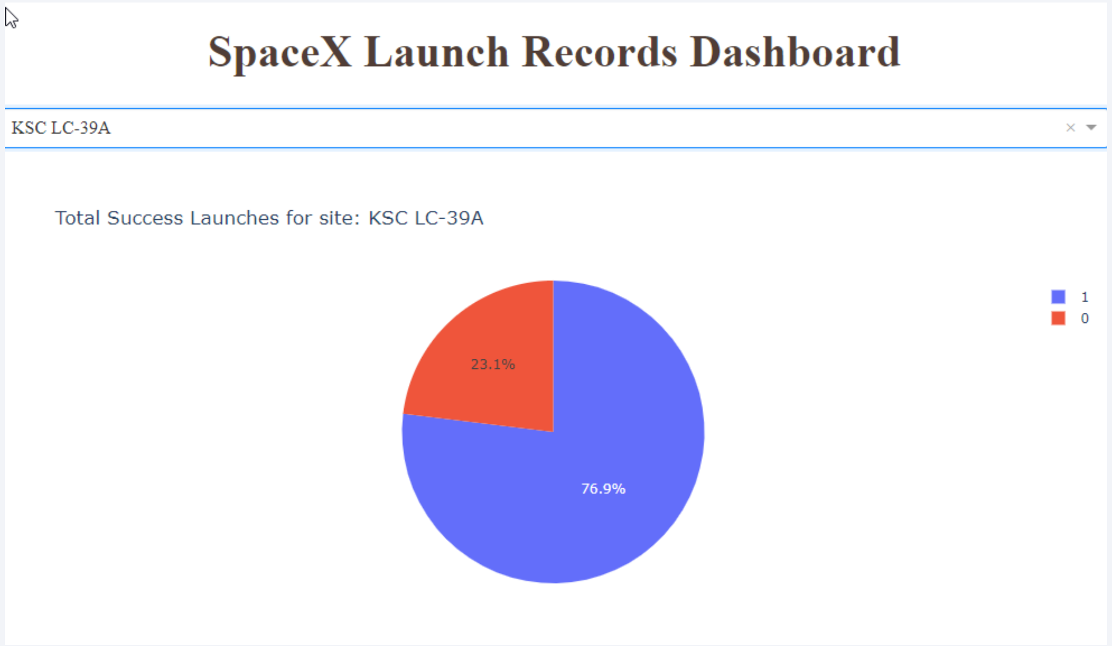
•Are launch sites in close proximity to railways?
Yes
•Are launch sites in close proximity to highways?
Yes
•Are launch sites in close proximity to coastline?
Yes
•Do launch sites keep certain distance away from cities?
Yes

Section 4

Build a Dashboard with Plotly Dash

Plotly Dashboard – Piechart

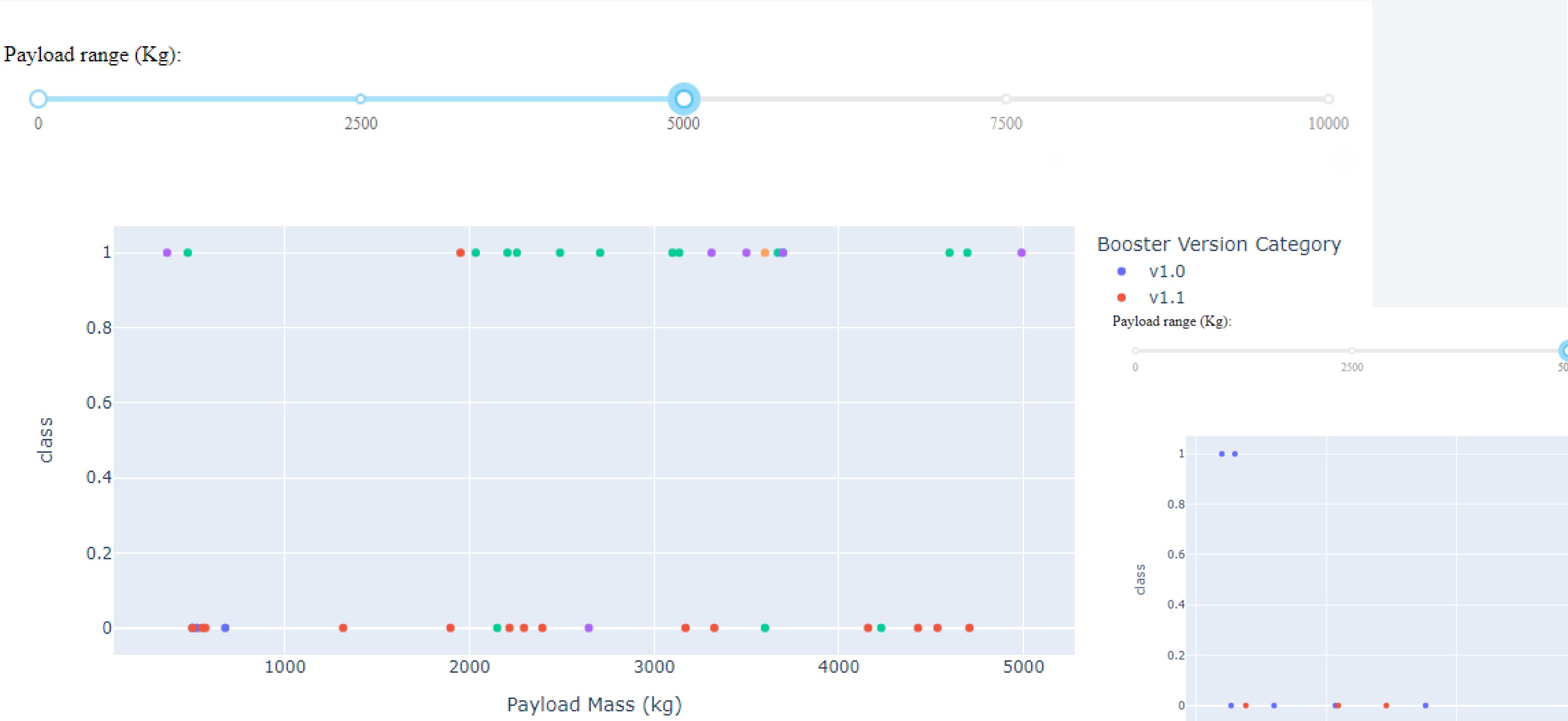




We can see that KSC LC-39A has the highest success rate.

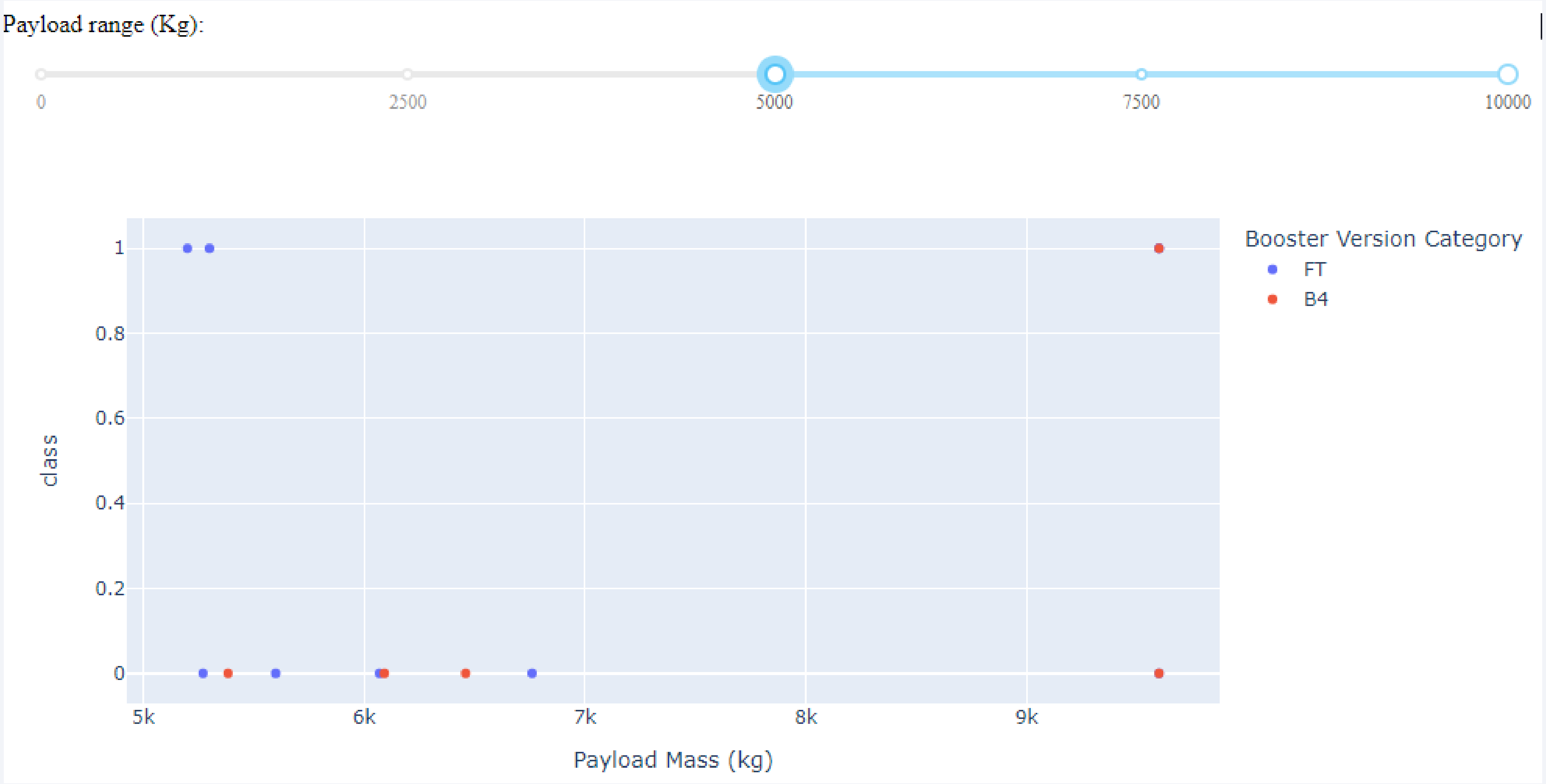
Plotly Dashboard – Payload vs Launch Outcome scatterplot

Payload range 0-5000 kg.



Plotly Dashboard – Payload vs Launch Outcome scatterplot

Payload range 5000-10000 kg.



Plotly Dashboard – Payload vs Launch Outcome scatterplot

We can see that success rate is higher for lower payload than higher payload, but we have smaller number of data points for that second range.

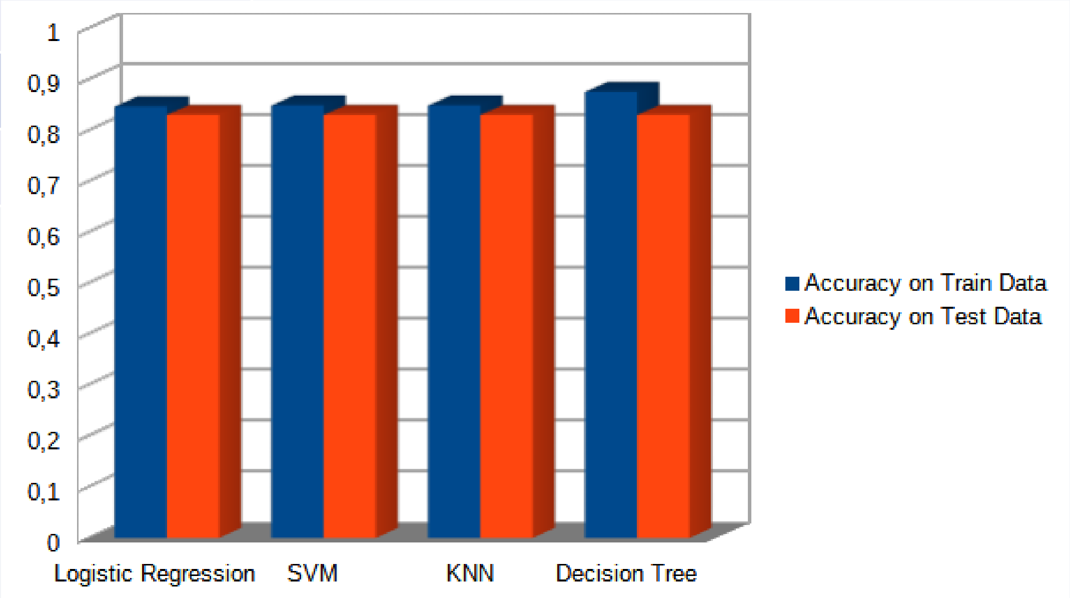
Section 5

Predictive Analysis (Classification)

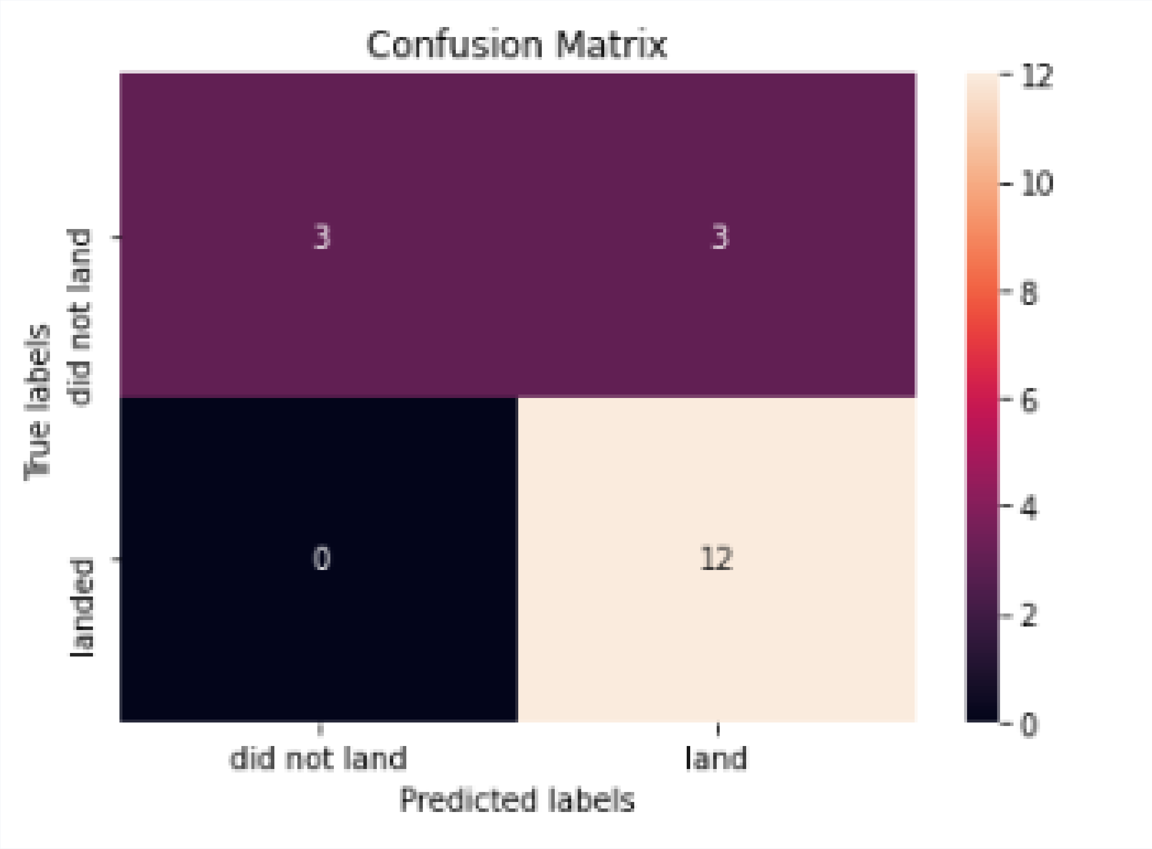
Classification Accuracy

All machine learning models (used algorithms) have 83.3% on test data, but decision tree is the best, taking into consideration train data (87.5%)

Algorithm	Accuracy on Train Data	Accuracy on Test Data
Logistic Regression	0.8464	0.8333
SVM	0.8482	0.8333
KNN	0.8482	0.8333
Decision Tree	0.8750	0.8333



Confusion Matrix



All models have the same confusion matrix.

As we can see True Positive, True Negative and False Negative are correct. The problem is only with False Positive – 3 records out of 18.

For that reason we have 83.3% accuracy on test data.

Conclusions

- We have concluded from the results that the success of landing is dependent on the launch site, the orbit, the mass of payload and some other technical factors.
- Low weighted payloads has higher success rate than heavier.
- KSC LC-39A site had the most successful launches.
- Orbit GEO,SSO,HEO,ES-L1 has the best Success Rate.
- The success rate is increasing with flight number, espacially after ~30th flight.
- The success rate has been increasing since 2013.
- Machine Learning Modelling gives very good results in predicting success or failure of the launch
- The best machine learning algorithm is Decision Tree Classifier.

Thank you!