IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Yoko Takishita
January 6, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - Exploratory Data Analysis (EDA) with SQL
  - EDA with Matplotlib
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis
- Summary of all results
  - Exploratory Data Analysis
  - Interactive analysis
  - Predictive analysis

Introduction

- Project background and context
  - SpaceX advertises Dlacon9 rocket launches on its website with a cost of 62 million dollars, while other providers cost upward of 165 million dollars. The saving was achieved by SpaceX because Space X can reuse the first stage.
  - If we can predict whether the first stage can successfully land, we can determine the cost of a launch

- Problems you want to find answers
  - What factors determine if the first stage of the rocket launch can land successfully
  - Build a predictive model

Section 1

Methodology

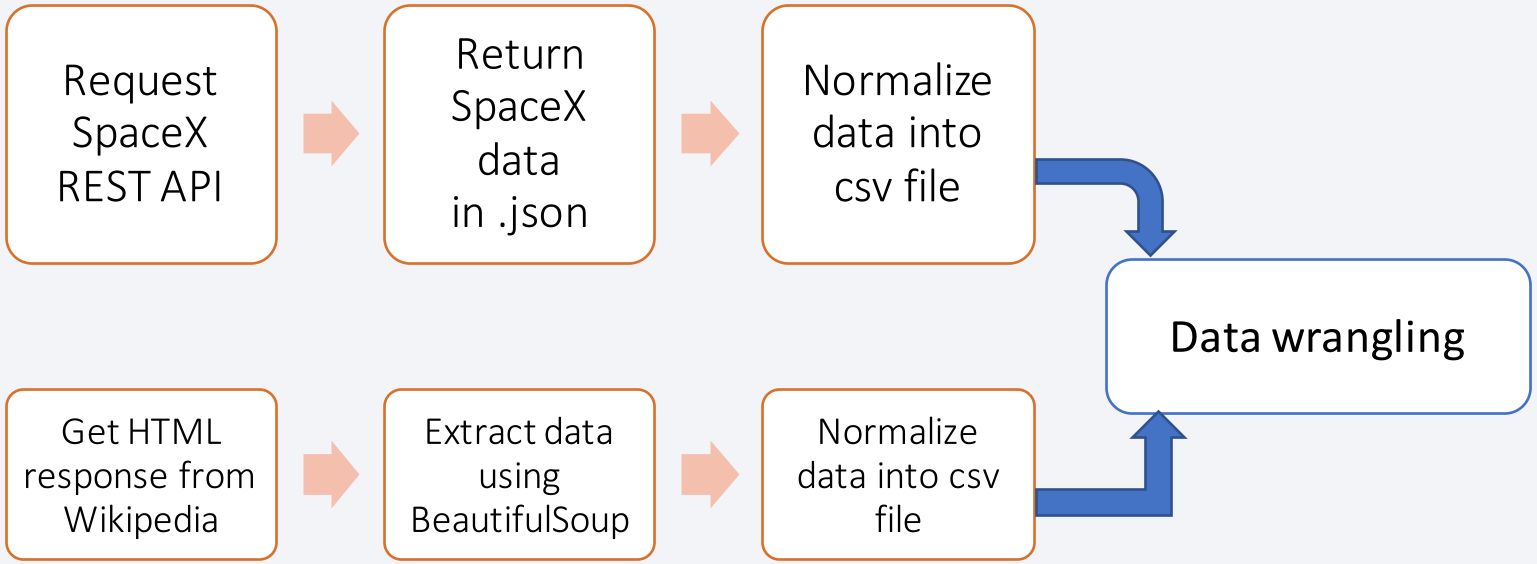
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - One Hot Encoding applied to categorical variables
  - Data cleaning of null values and irrelevant columns
- Perform exploratory data analysis (EDA) using SQL and Matplotlib
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - LR, KNN, SVM, DT models and evaluation

Data Collection

- SpaceX launch data was obtained from the SpaceX REST API
- Falcon9 launch data was obtained by web scraping Wikipedia using BeautifulSoup



Data Collection – SpaceX API

- Get request to obtain the data
- Normalize by json_normalize()

https://github.com/PomuPomu8/Testrepo/blob/master/REST_API.ipynb

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:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spa
```

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

In [10]:
```
response.status_code
```

Out[10]:
```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Using the dataframe `data` print the first 5 rows

In [13]:
```
# Get the head of the dataframe
print(data.head())
```

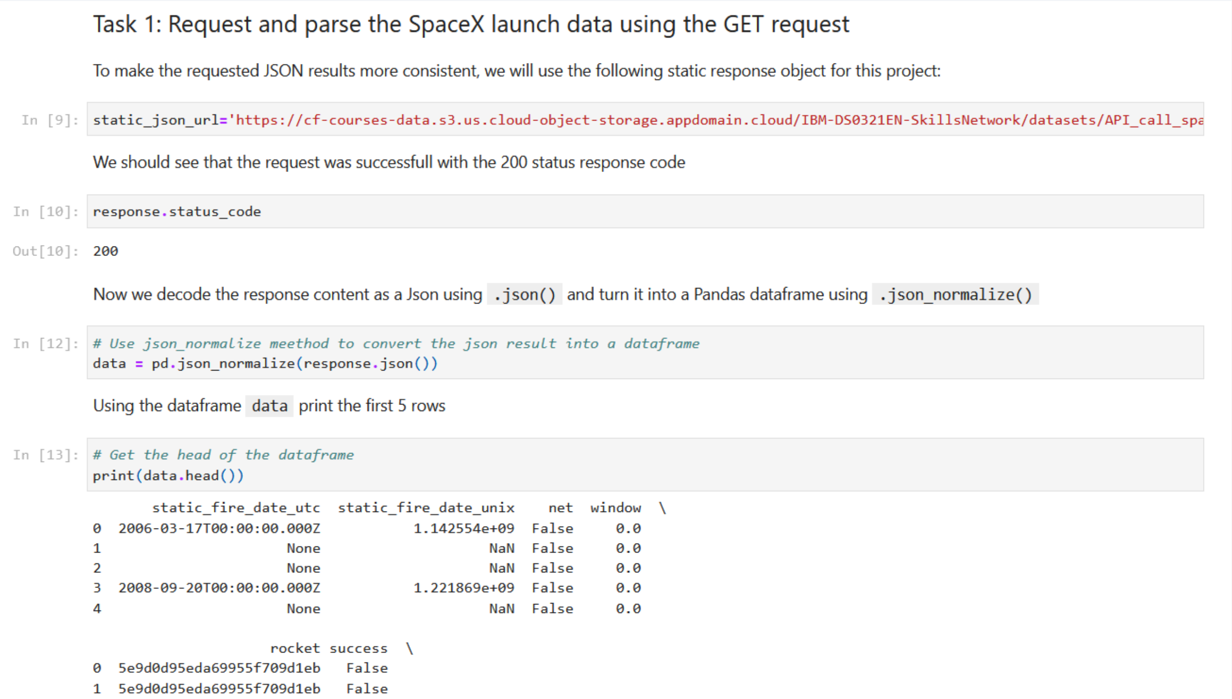
```
       static_fire_date_utc  static_fire_date_unix    net  window  \
0  2006-03-17T00:00:00.000Z           1.142554e+09  False     0.0
1                      None                    NaN  False     0.0
2                      None                    NaN  False     0.0
3  2008-09-20T00:00:00.000Z           1.221869e+09  False     0.0
4                      None                    NaN  False     0.0

                     rocket success  \
0  5e9d0d95eda69955f709d1eb   False
1  5e9d0d95eda69955f709d1eb   False
```

Data Collection - Scraping

- Get request from HTML
- Beautifulsoup

https://github.com/PomuPomu8/Testrepo/blob/master/jupyter-labs-webscraping(2).ipynb



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:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spa
```

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

In [10]:
```
response.status_code
```

Out[10]:
```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Using the dataframe `data` print the first 5 rows

In [13]:
```
# Get the head of the dataframe
print(data.head())
```

```
       static_fire_date_utc  static_fire_date_unix    net  window  \
0  2006-03-17T00:00:00.000Z           1.142554e+09  False     0.0
1                      None                    NaN  False     0.0
2                      None                    NaN  False     0.0
3  2008-09-20T00:00:00.000Z           1.221869e+09  False     0.0
4                      None                    NaN  False     0.0

                     rocket  success  \
0  5e9d0d95eda69955f709d1eb    False
1  5e9d0d95eda69955f709d1eb    False
```

Data Wrangling

- Calculate the number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate the number of occurrence of mission outcome per orbit type
- Create a landing outcome label from outcome column

https://github.com/PomuPomu8/Testrepo/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

In [12]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

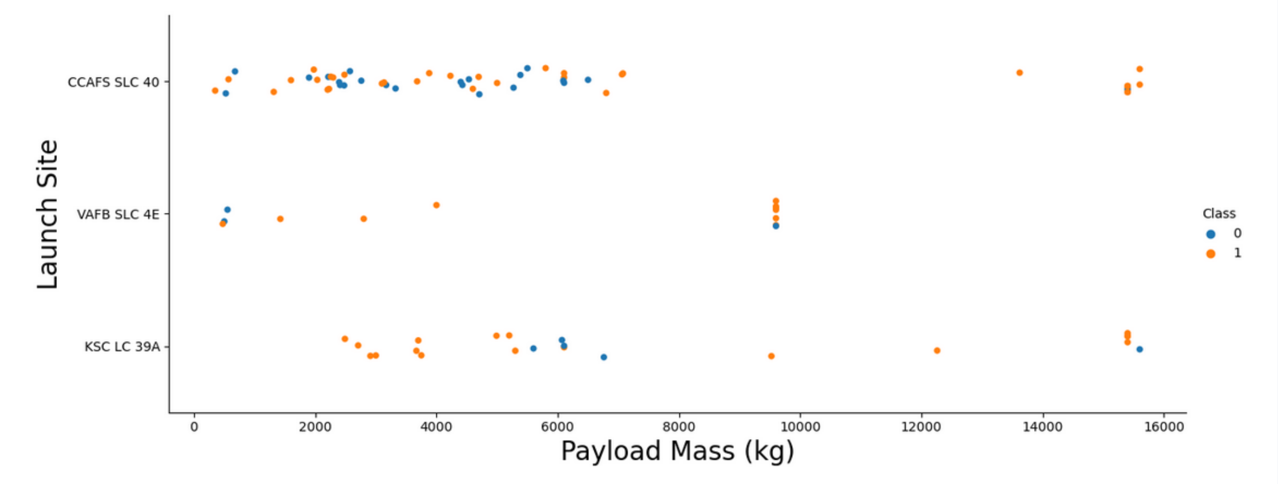
In [16]:
```
landing_class = [0 if x in bad_outcomes else 1 for x in df['Outcome']]
# landing_class
df['Class']=landing_class
```

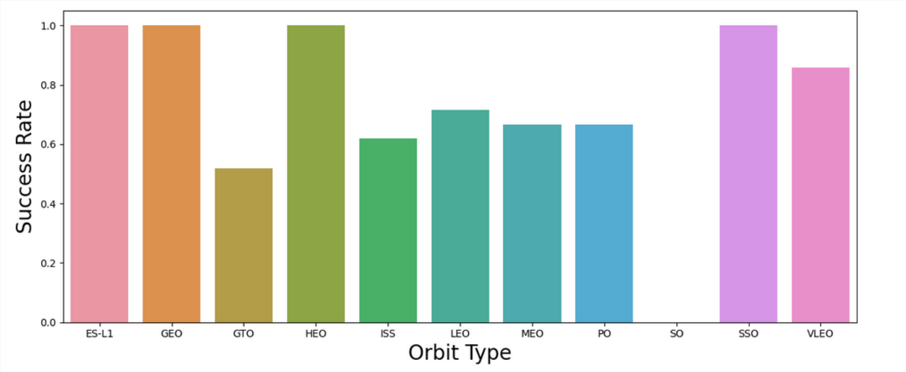
In [17]:
```
df.head(5)
```

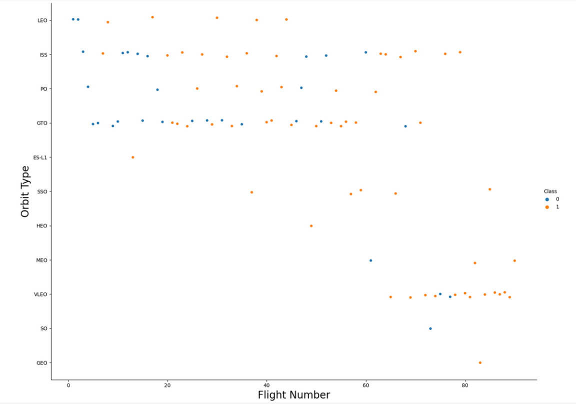
Out[17]:

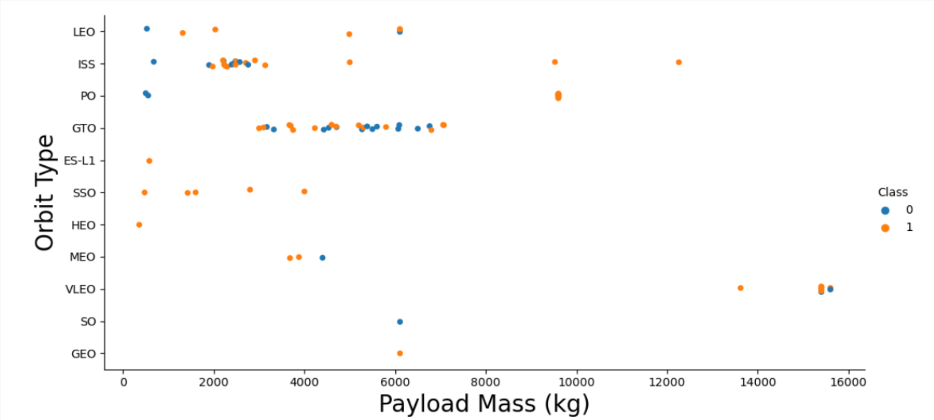
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	R
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	

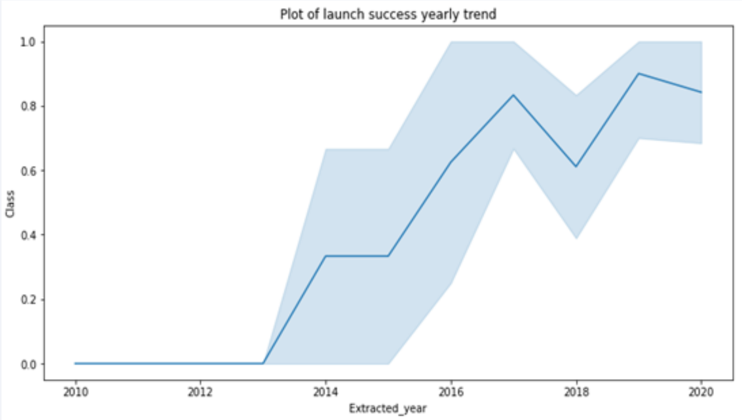
EDA with Data Visualization - Matplotlib











EDA with Data Visualization

- Create dummy variables to categorical columns

https://github.com/PomuPomu8/Testrepo/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

In [23]:
```
### TASK  7: Create dummy variables to categorical columns
features_one_hot = pd.get_dummies(features[['Orbit', 'LaunchSite', 'LandingPad', 'Serial']])
features_one_hot = pd.concat([features[['FlightNumber', 'PayloadMass', 'Flights', 'GridFins', 'Reused', 'Legs', 'Block', 'Re
features_one_hot.head(10)
```

Out[23]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Class	Orbit_ES-L1	...	Serial_B1048	Serial_B1049	Serial_B1050
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0	0	0
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
5	6	3325.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
6	7	2296.000000	1	False	False	True	1.0	0	1	0	...	0	0	0
7	8	1316.000000	1	False	False	True	1.0	0	1	0	...	0	0	0
8	9	4535.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
9	10	4428.000000	1	False	False	False	1.0	0	0	0	...	0	0	0

10 rows × 81 columns

EDA with SQL

- Load SpaceX data into a PostgreSQL database
- SQL queries such as:
- Name of unique launch sites in the space mission
- Total payload mass carried by boosters launched by NASA(CRS)
- Average payload mass carried by booster version F9 v1,1
- Total number of successful and failure mission outcomes

Task 8

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

In [42]:
```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPA
```

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

Out[42]:

boosterversion
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

https://github.com/PomuPomu8/Testrepo/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Marked all launch sites, and added map objects to mark the success or failure of launches for each site
- Create a cluster map to find the locations with high landing sucess
- Calculate the distances between a launch site to its proximities:
    - Are launch sites near railways, highways and coastlines?
    - Do launch sites keep distance away from cities?

https://github.com/PomuPomu8/Testrepo/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Interactive Dashboard with Plotly Dash

- Interactive dashboard with Plotly dash
- Dropdown list: 'Select a launch site here'
- Plot pie charts showing the total launches by each site
- Plot scatter graph showing the relationship with outcome and payload mass for the different booster version

https://github.com/PomuPomu8/Testrepo/blob/master/spacex_dash_app.py

Predictive Analysis (Classification)

- Create Numpyarray
- Split the data into training and testing sets (20%)
- Different models:
  - Logistic regression
  - SVM
  - Decision trree
  - KNN
- Evaluation by GridSearchCV
- Determine the best model by highest accuracy (R2 score)

https://github.com/PomuPomu8/Testrepo/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

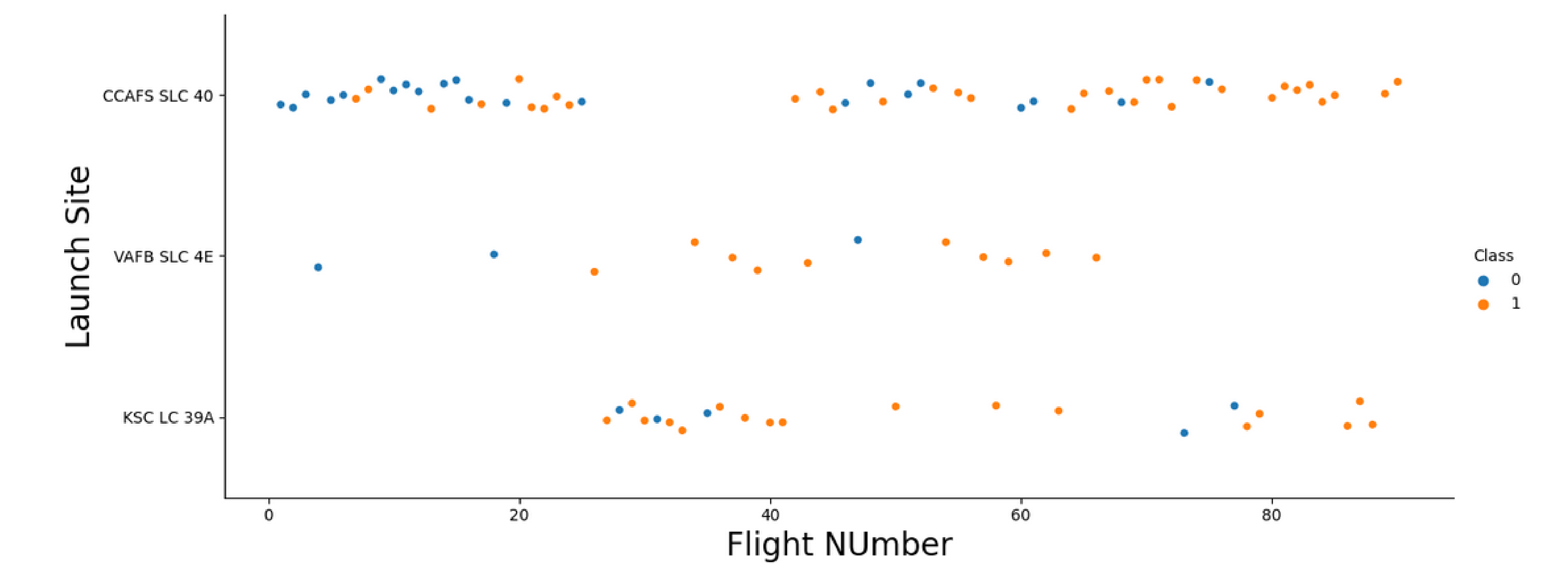
Results

- Logistic regression, SVM and KNN models were the best in terms of accuracy
- Lower payload mass performed better, compared to heavier payload
- The success of SpaceX launch has been increasing since 2013
- Orbit GEP, HEO, SSO, ED L2 gave best success rate.
- KSL LC 39A was the highest success rate

Section 2

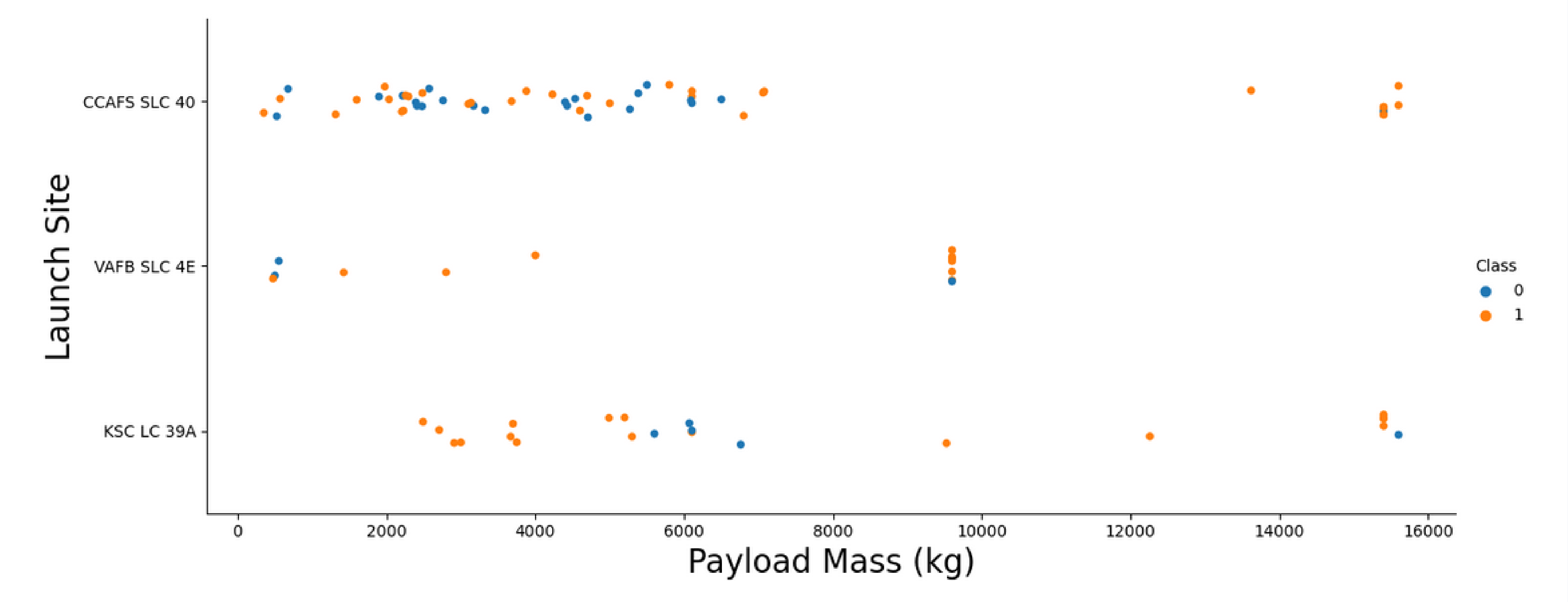
Insights drawn from EDA

Flight Number vs. Launch Site



- For CCAFS SLC 40 site, higher flight numbers have better success rate.

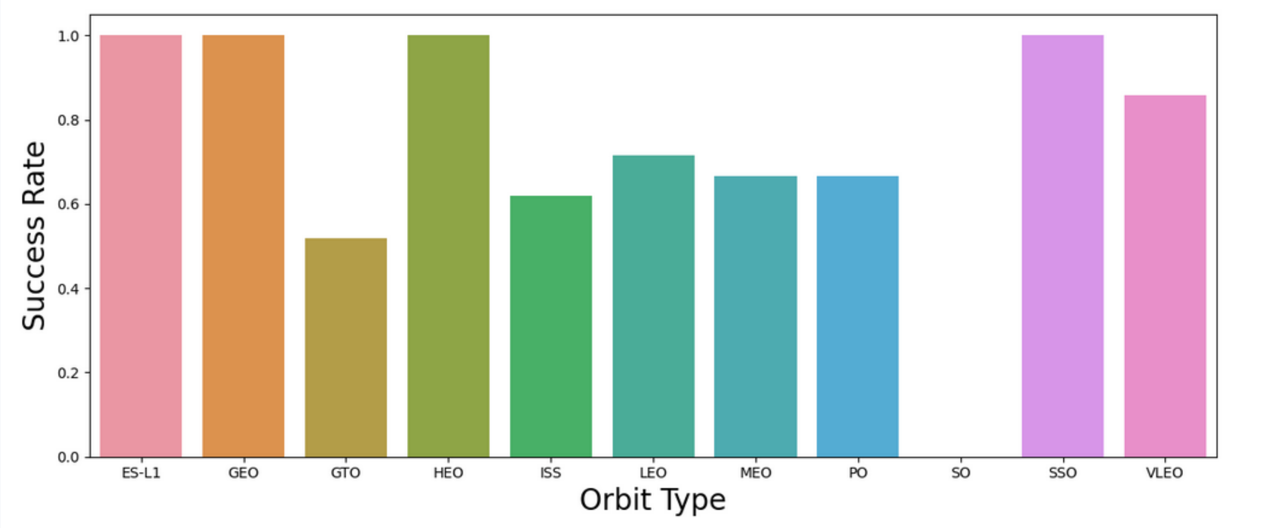
Payload vs. Launch Site



- For CCAFS SLC 40, heavier payload mass had better success rate

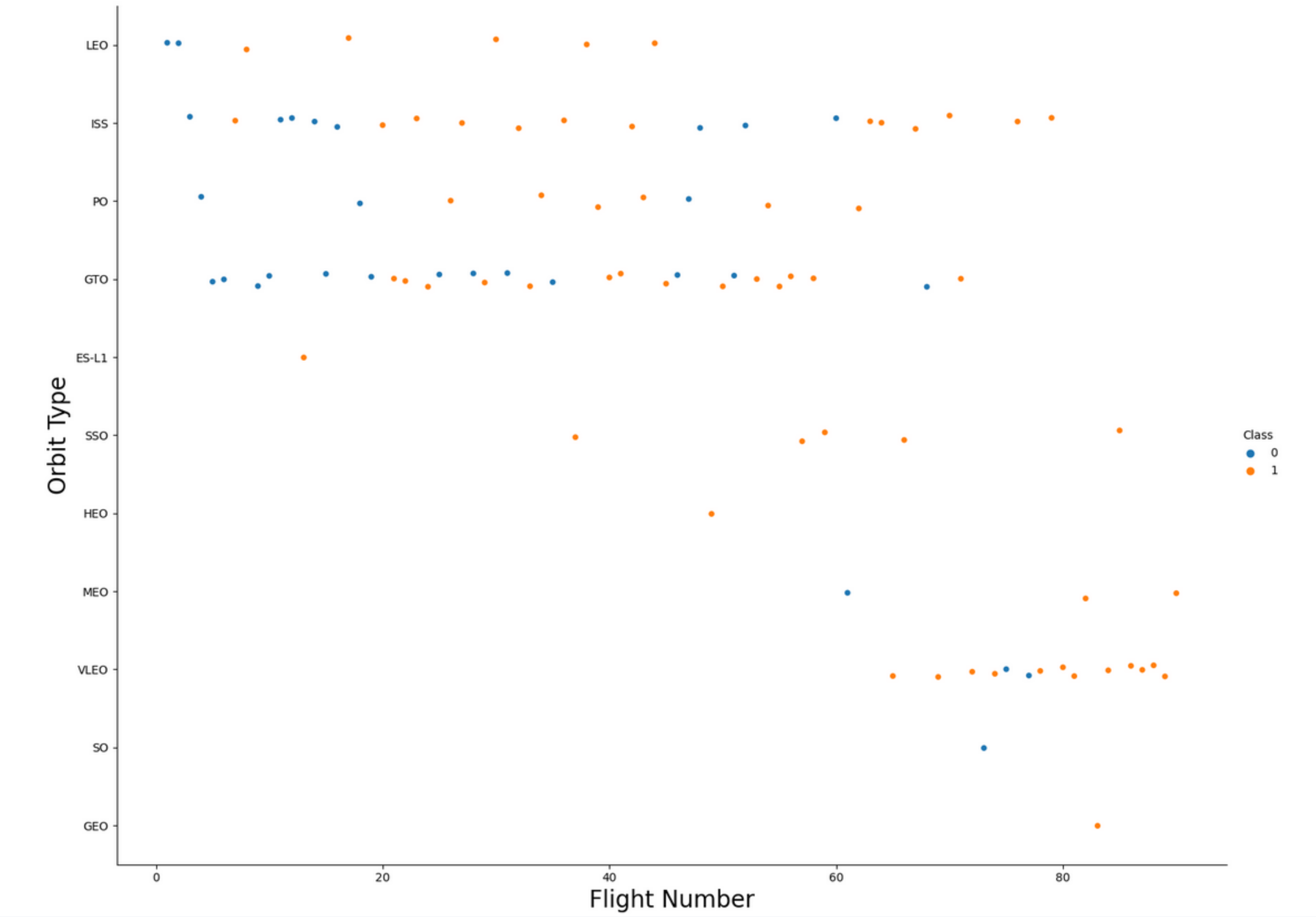
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO, VLEO had the best success rate

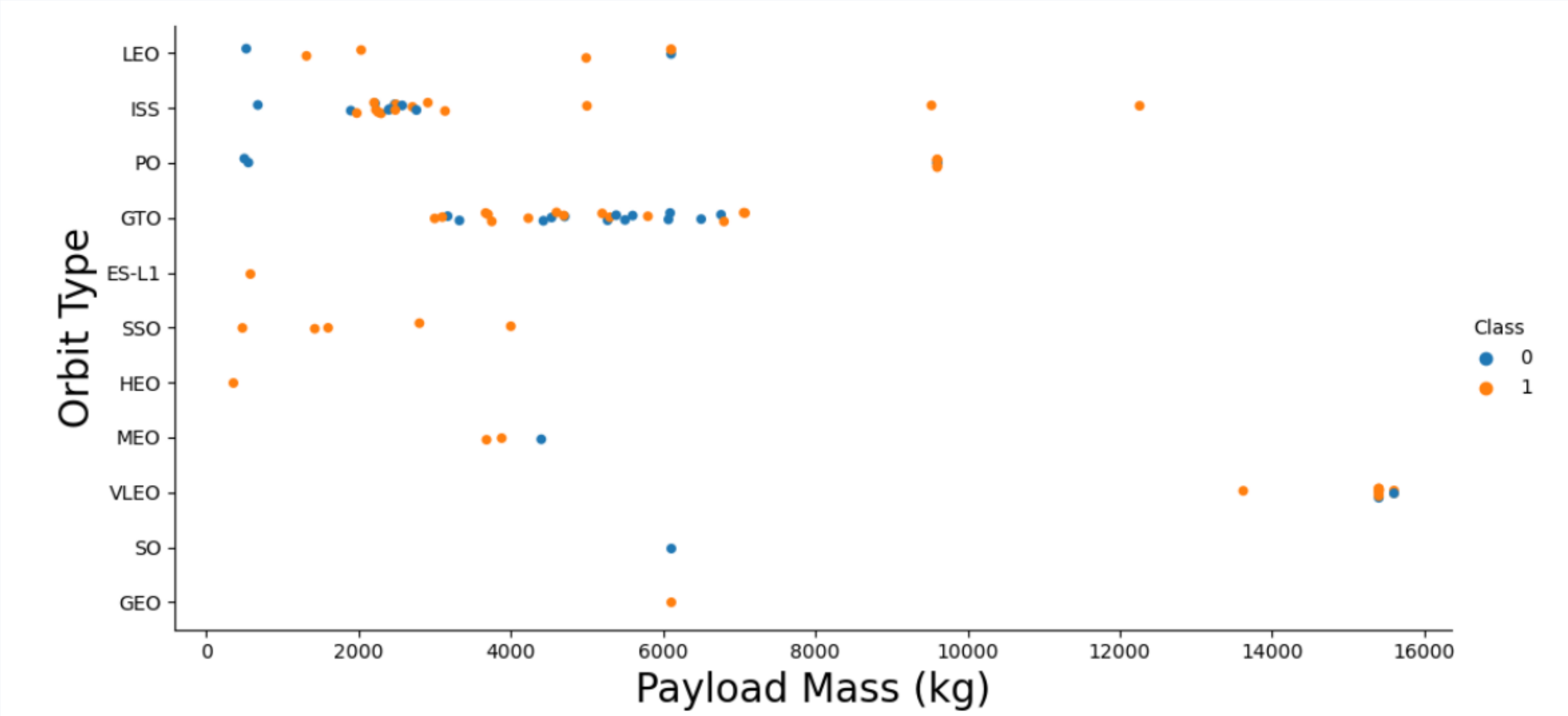


Flight Number vs. Orbit Type

- For VLEO, higher flight number had more success
- For GTP, there seemed no relationship between success rate and flight number



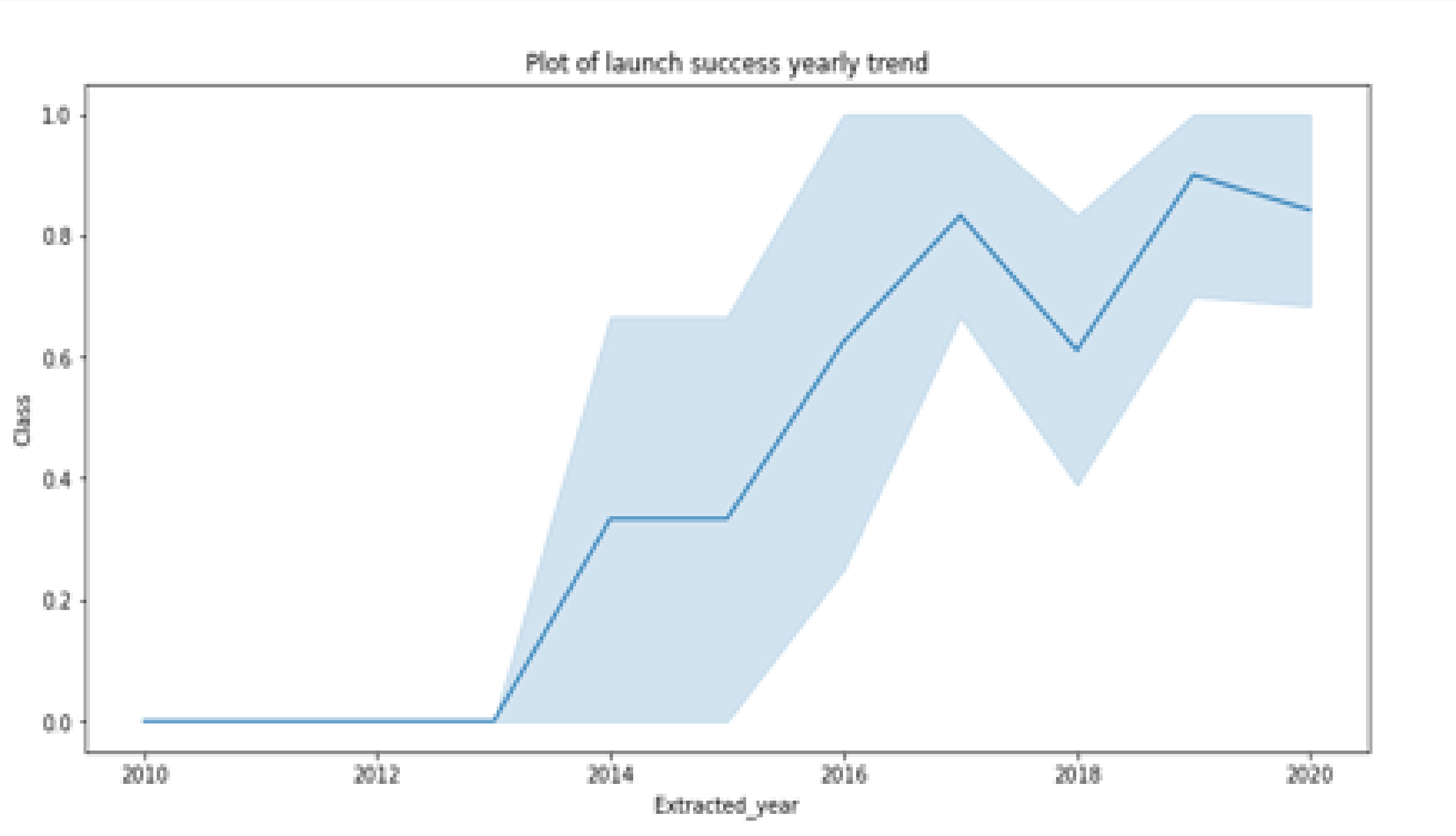
Payload vs. Orbit Type



- PO. LEO and ISS orbits are used with heavierr payload mass

Launch Success Yearly Trend

- Success of SpaceX launch has been increasing since 2013



All Launch Site Names

- Used DISTINCT clause to show unique launch sites

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

In [31]:
```
%sql SELECT distinct (LAUNCH_SITE) from SPACEXTBL;
```

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

Out[31]:

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

Launch Site Names Begin with 'CCA'

- Found 5 records where launch sites begin with `CCA`, using WHERE, LIKE and LIMIT clauses

Task 2

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

In [91]:
```
%sql SELECT * from SPACEXTBL where LAUNCH_SITE LIKE ('CCA%') LIMIT 5;
```

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

Out[91]:

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

- Calculated the total payload mass = 45596 kg

Task 3

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

In [89]:
```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmasskg from SPACEXTBL Where Customer = 'NASA (CRS)';
```

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

Out[89]:

payloadmasskg
45596

Average Payload Mass by F9 v1.1

- Calculated the average payload mass carried by booster version F9 v1.1 = 2928.4 kg

Task 4

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

In [88]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1'
```

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

Out[88]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Found the dates of the first successful landing outcome on ground pad

= December 22, 2015

Task 5

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

Hint:Use min function

In [87]:
```
%%sql
SELECT min(substr(Date,7,4) || substr(Date,4,2) || substr(Date,1,2))
from SPACEXTBL
where "Landing _Outcome" ='Success (ground pad)';
```

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

Out[87]:

min(substr(Date,7,4) || substr(Date,4,2) || substr(Date,1,2))
20151222

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, using AND and BETWEEN clauses

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

In [85]:
```
%%sql

SELECT BOOSTER_VERSION from SPACEXTBL where "Landing _Outcome"='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 an
```

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

Out[85]:

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

Total Number of Successful and Failure Mission Outcomes

- Total number of successful and failure mission outcomes

= Success: 100, Failure: 0

Task 7

List the total number of successful and failure mission outcomes

In [60]:
```
%sql SELECT substr(Mission_Outcome,1,7) as Mission_Outcome, count(*) from SPACEXTBL group by 1;
```

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

Out[60]:

Mission_Outcome	count(*)
Failure	1
Success	100

Boosters Carried Maximum Payload

- Listed the names of the booster which have carried the maximum payload mass, using MAX clause

Task 8

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

In [42]:
```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPA
```

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

Out[42]:

boosterversion
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

2015 Launch Records

- Listed the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015, using WHERE, AND clauses, and substr()

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.

In [92]:

```
%%sql
SELECT substr(Date,4,2) as Month,Booster_Version,Launch_Site,"Landing _Outcome"
FROM SPACEXTBL WHERE "Landing _Outcome" = 'Failure (drone ship)' and substr(Date,7,4)='2015';
```

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

Out[92]:

Month	Booster_Version	Launch_Site	Landing _Outcome
01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

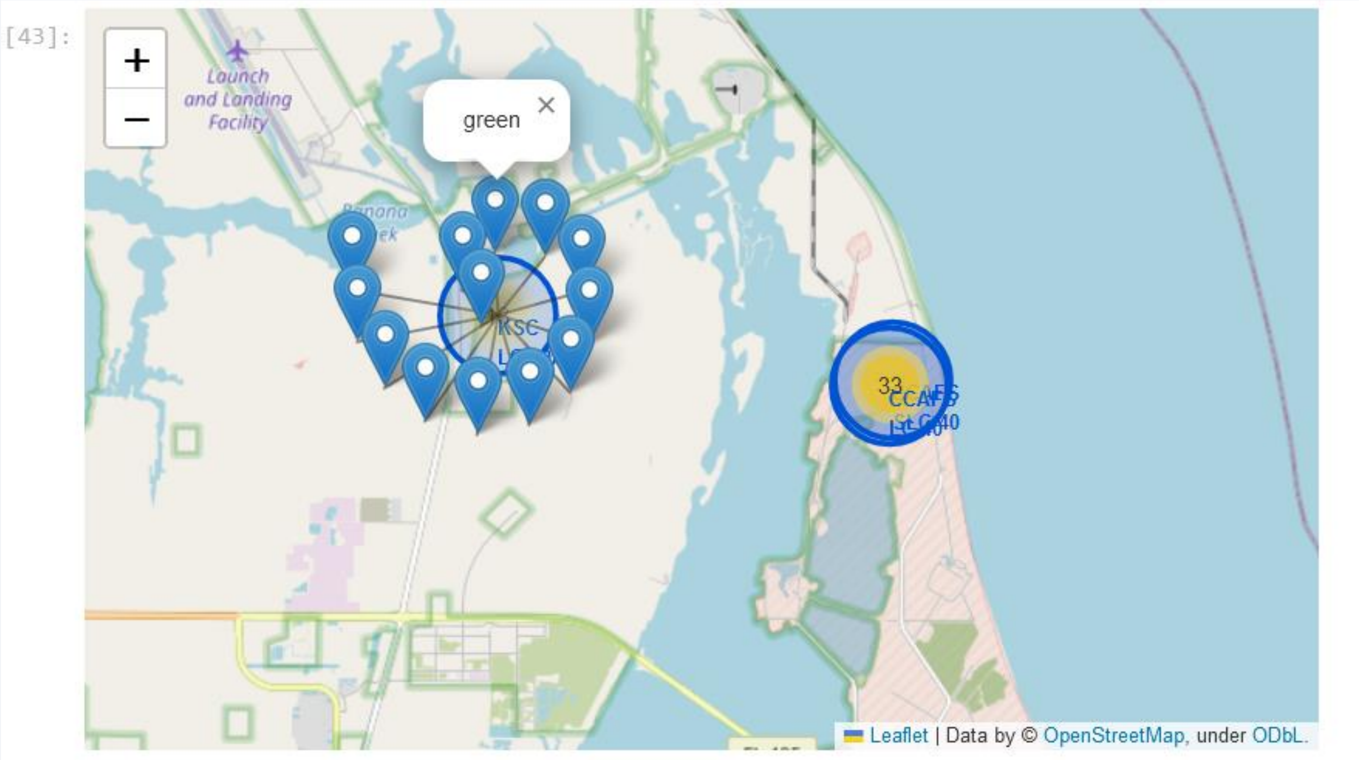
Section 3

Launch Sites Proximities Analysis

All launch sites marked on a global map

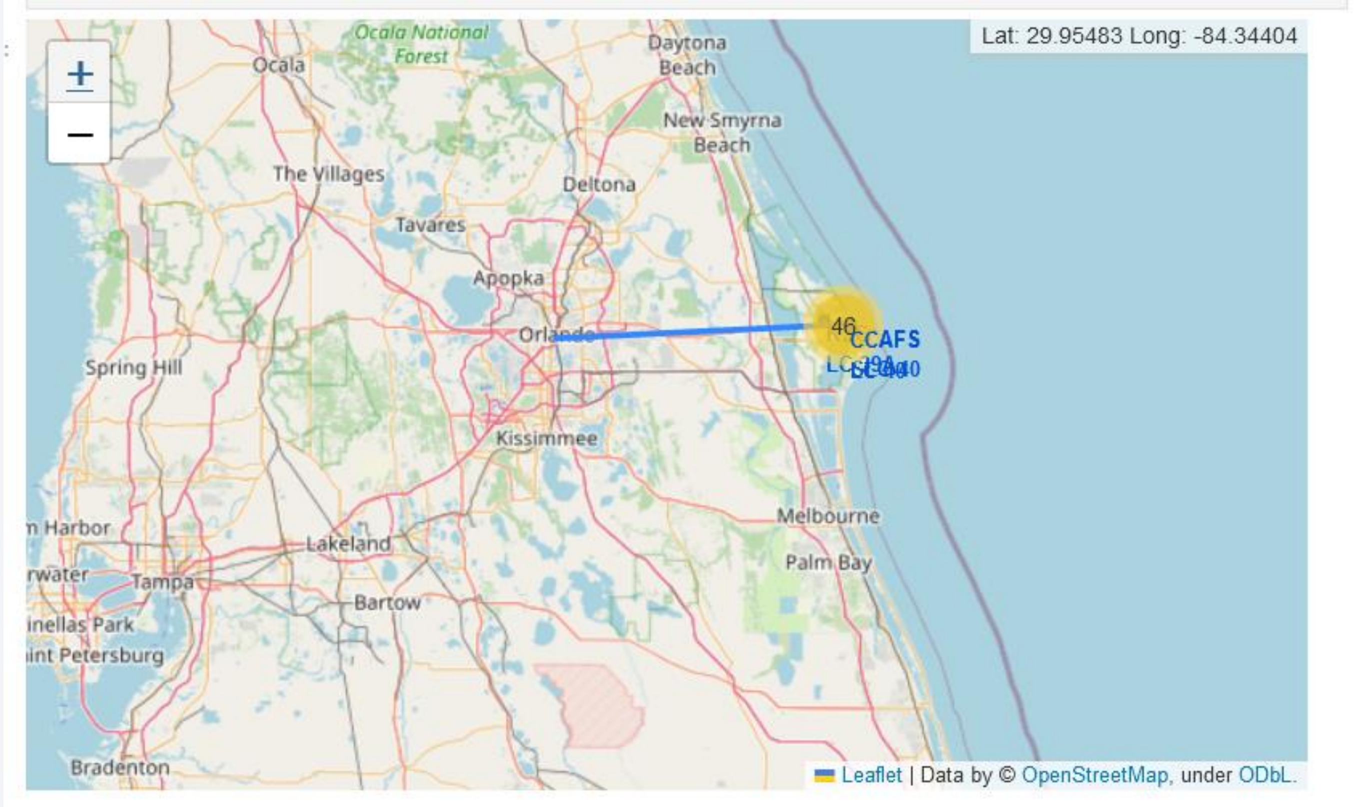
- Launch sites are coastal areas in California and Florida

Success/failed launches marked on the map



- Cluster marker showed sites with higher success

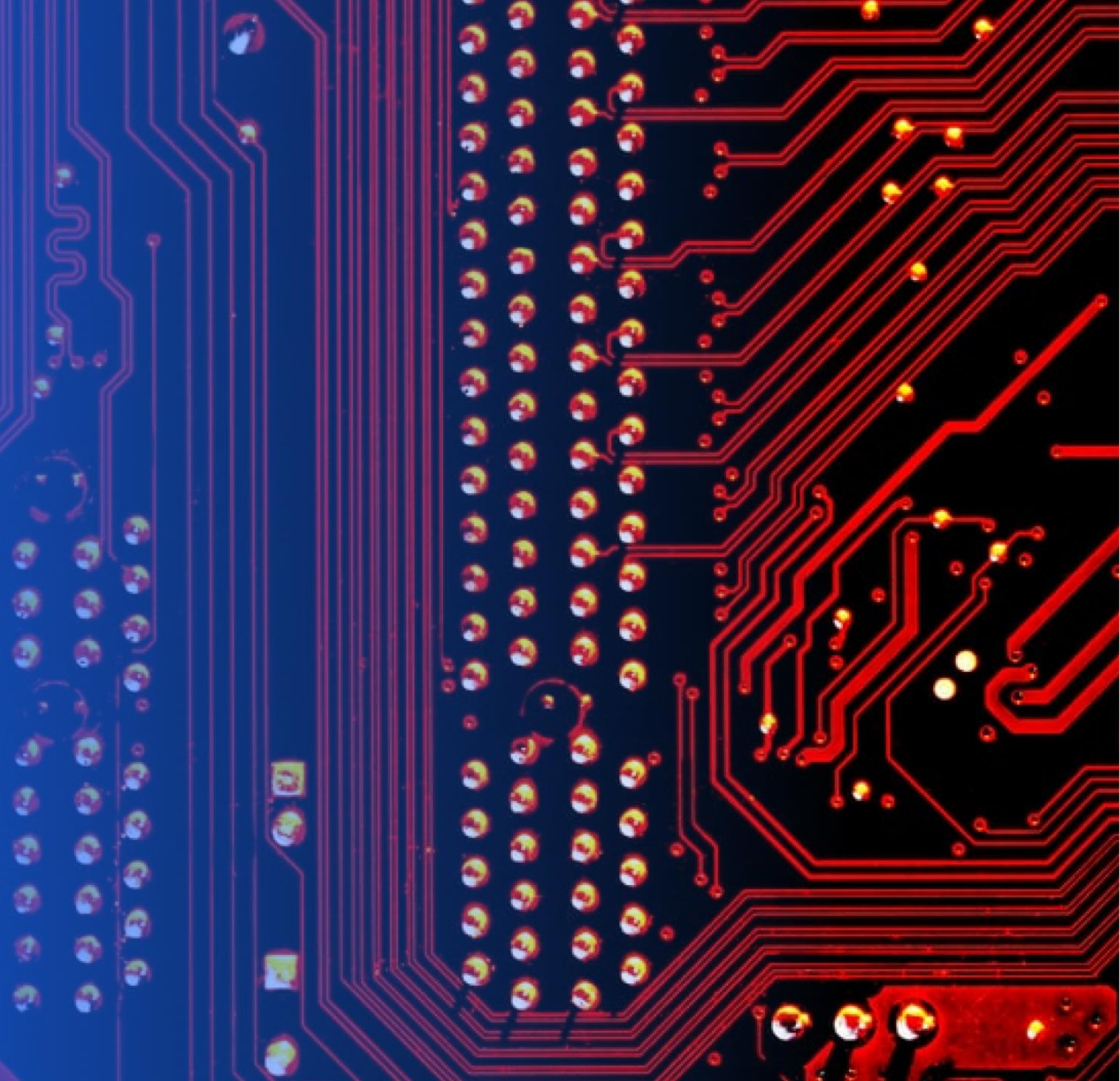
Distances between launch site to its proximites



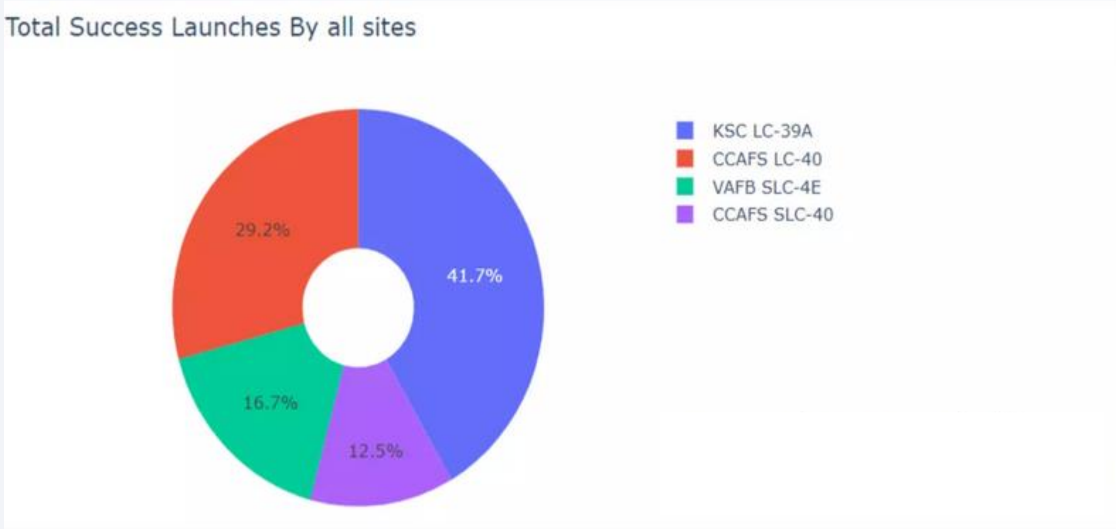
- Distance from the launch site to Orlando city

Section 4

Build a Dashboard with Plotly Dash

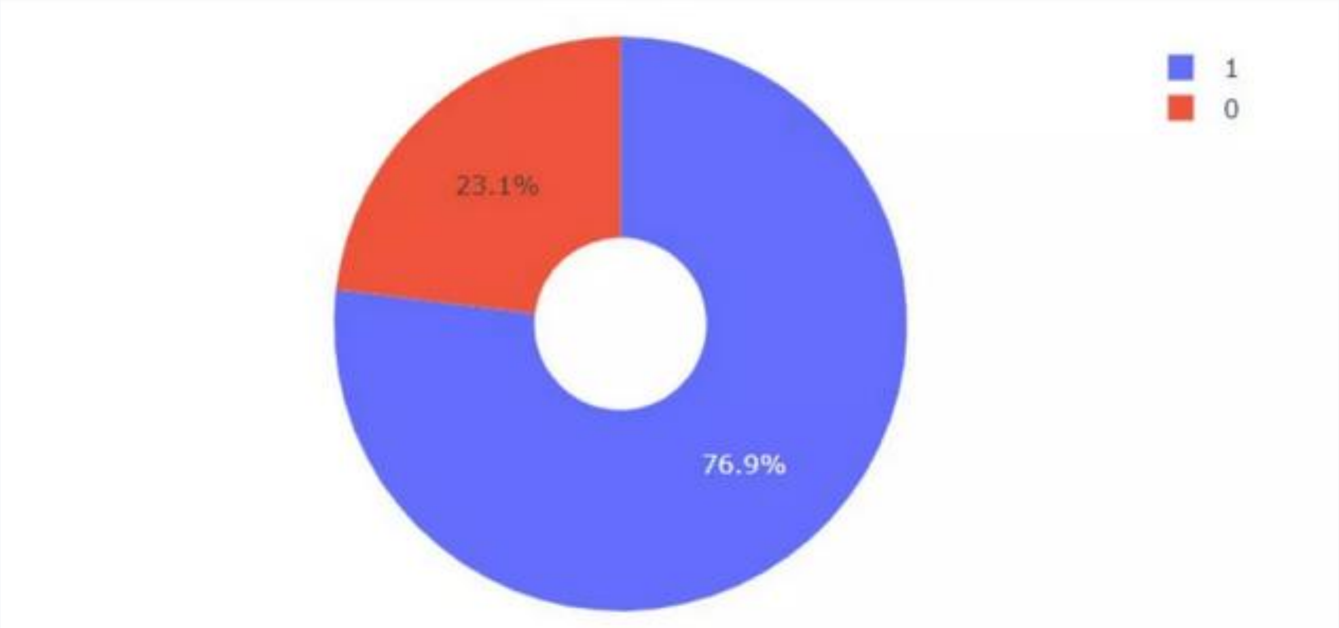


Total success by each launch sites



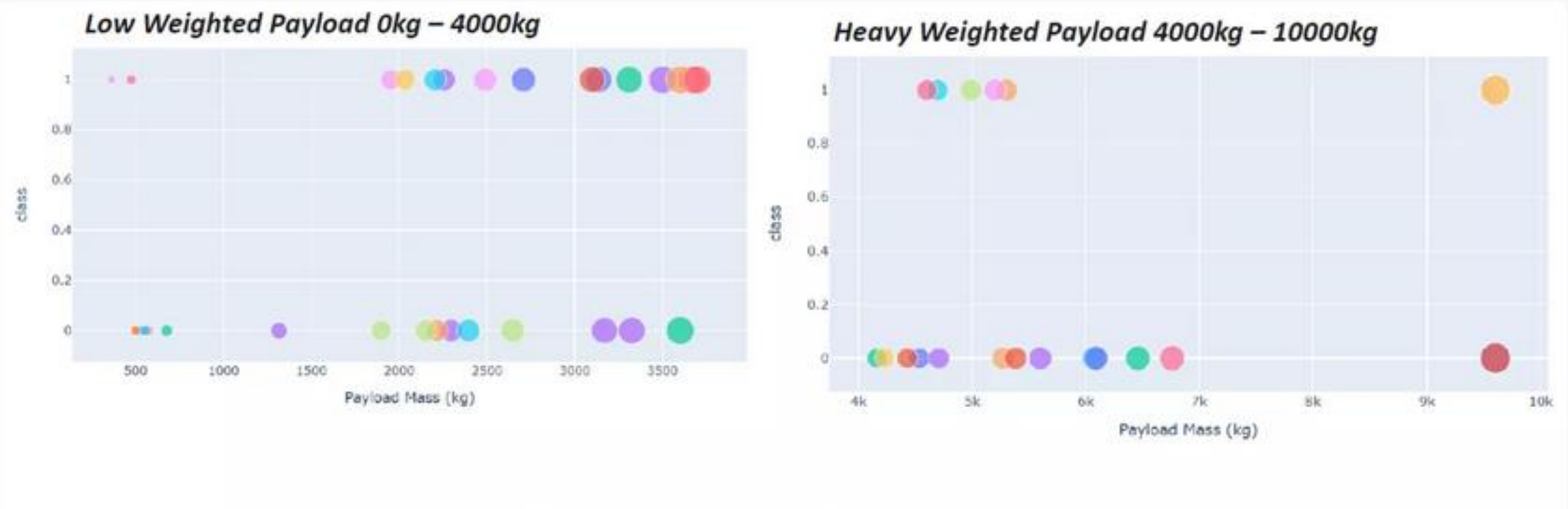
- KSC LC-39A had the most successful launches among all the sites

Succes rate by sites KSC LC-39A



- KSC LC-39A success rate was 76.9 %

Payload vs Launch outcome for all sites, ranging different payload masses



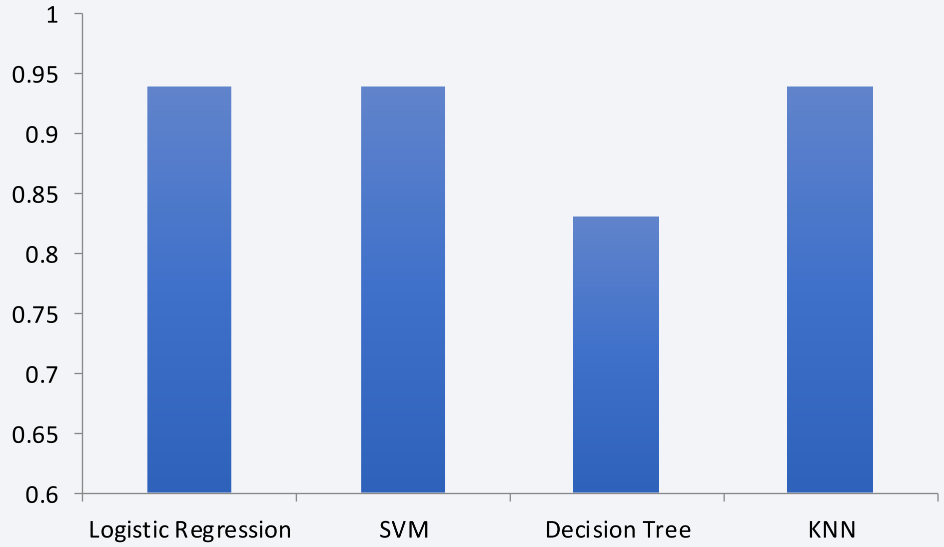
- Lighter payload performed better, compared to heavier payload mass

Section 5

Predictive Analysis (Classification)

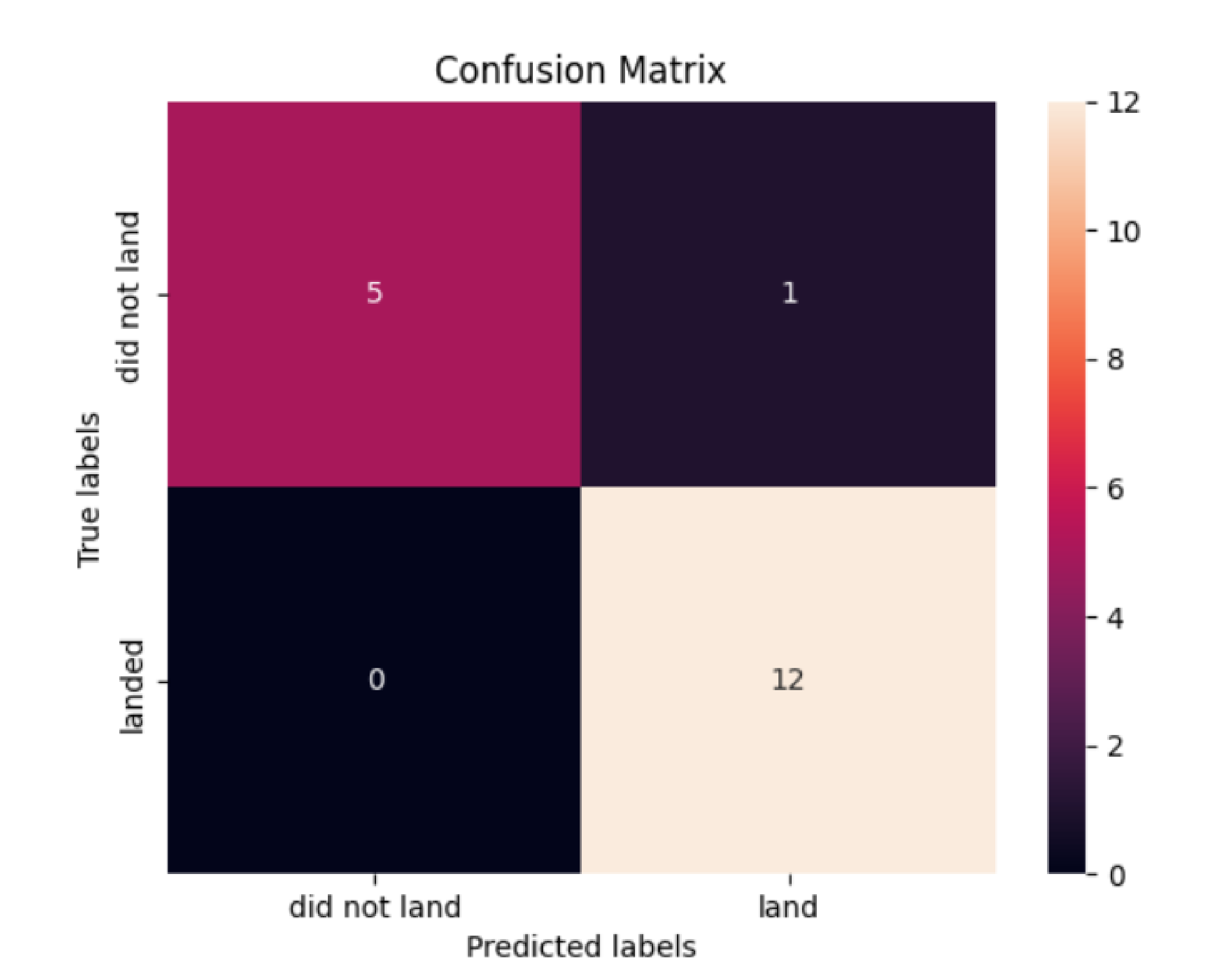
Classification Accuracy

- Used GridSearch to identify best parameters
- Accuracy score was calculated for the test data
- Logistc regression, SVM and KNN models performed better, compared to Decision Tree model



Confusion Matrix

- Logistic regression distinguished between the different classes
- Major problem is false positives (i.e. unsuccessful landings marked as successful)



Conclusions

- Logistic regression, SVM and KNN models were the best in terms of accuracy
- Lower payload mass performed better, compared to heavier payload
- The success of SpaceX launch has been increasing since 2013
- Orbit GEP, HEO, SSO, ED L2 gave best success rate.
- KSL LC 39A was the highest success rate

Appendix- list of Github links for Juoyter notebooks

Data collection – REST API	https://github.com/PomuPomu8/Testrepo/blob/master/REST_API.ipynb
Data collection – Web scraping	https://github.com/PomuPomu8/Testrepo/blob/master/jupyter-labs-webscraping(2).ipynb
Data wrangling	https://github.com/PomuPomu8/Testrepo/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb
Exploratory Data Analysis - SQL	https://github.com/PomuPomu8/Testrepo/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb
Exploratory Data Analysis - Matplotlib	https://github.com/PomuPomu8/Testrepo/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb
Data visualization – Plotly Dash	https://github.com/PomuPomu8/Testrepo/blob/master/spacex_dash_app.py
Data visualization - Folium	https://github.com/PomuPomu8/Testrepo/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb
Predictive analysis – classification model	https://github.com/PomuPomu8/Testrepo/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Thank you!