IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Thong Yee Moon>
< 26 March 2024 >

Outline

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

Executive Summary

- Summary of methodologies
  - - SpaceX Data Collection using SpaceX API
  - - SpaceX Data Collection with Web Scraping
  - - SpaceX Data Wrangling
  - - SpaceX Exploratory Data Analysis using SQL
  - - Space-X EDA DataViz Using Python Pandas and Matplotlib
  - - Space-X Launch Sites Analysis with Folium-Interactive Visual Analytics and Ploty Dash
  - - SpaceX Machine Learning Landing Prediction
- Summary of all results
  - EDA graph
  - Dashboard
  - Model for Predictive Analysis

Introduction

- Project background and context
  - Most unsuccessful landings are planned. Space X performs a controlled landing in the oceans.
  - 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
  - In this project , we will predict if the Falcon 9 first stage will land successfully

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Request to the SpaceX API
  - web scraping [https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches]
- Perform data wrangling
  - Exploratory Data Analysis and Determine Training Labels
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Standardize the data & Split into training data and test data
  - -Find best Hyperparameter for SVM, Classification Trees and Logistic Regression

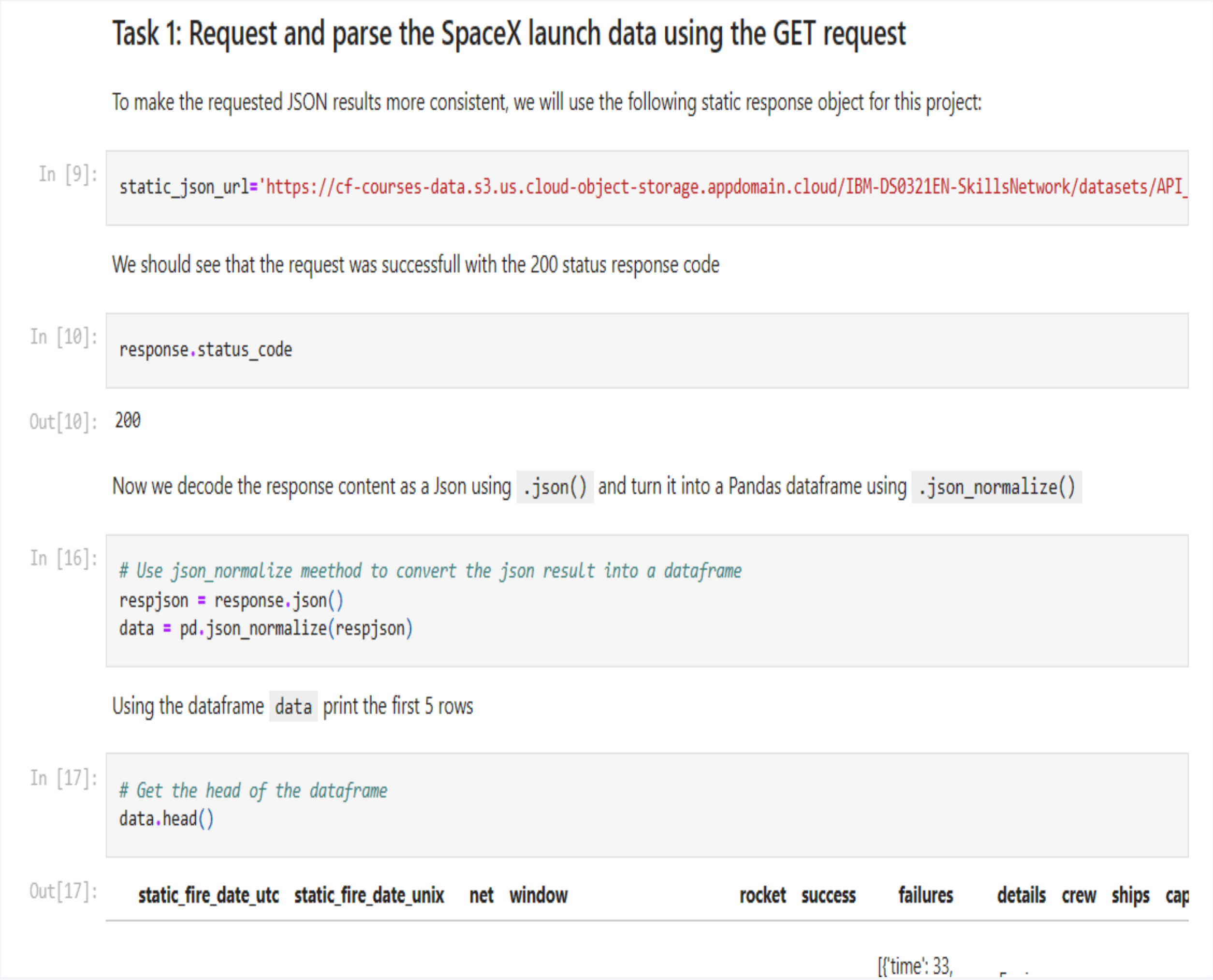
Data Collection

- Data was first collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API.
    - Used of the API to extract information using identification numbers in the launch data
    - Requested rocket launch data from the SpaceX API url.
    - Applied GET request and then decoded the response content as a Json format.
    - Converted into a Pandas data frame
- Web scraping [https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches] to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches of the launch records are stored in a HTML.
    - Used BeautifulSoup and request Libraries to extract the Falcon 9 launch HTML table
    - Parsed the table
    - Converted it into a Pandas data frame

Data Collection – SpaceX API

- SpaceX API
  - Used of the API to extract information using identification numbers in the launch data
  - Requested rocket launch data from the SpaceX API url.
  - Applied GET request and then decoded the response content as a Json format.
  - Converted into a Pandas data frame

- GitHub URL of the completed SpaceX API calls notebook
  - https://github.com/yeemoonthong/IBM-Data-Science/blob/500a96379f41f85607ef836b5986739307869e74/Capstone/1.%20jupyter-labs-spacex-data-collection-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_
```

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 [16]:
```
# Use json_normalize meethod to convert the json result into a dataframe
respjson = response.json()
data = pd.json_normalize(respjson)
```

Using the dataframe `data` print the first 5 rows

In [17]:
```
# Get the head of the dataframe
data.head()
```

Out[17]:

static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	cap
						[{'time': 33,				

Data Collection - Scraping

- Web scraping
  - Used BeautifulSoup and request Libraries to extract the Falcon 9 launch HTML table
  - Parsed the table
  - Converted it into a Pandas data frame
- GitHub URL
  - https://github.com/yeemoonthong/IBM-Data-Science/blob/500a96379f41f85607ef836b5986739307869e74/Capstone/2.%20jupyter-labs-webscraping.ipynb

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.

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, 'html.parser')
```

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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<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

In [10]:
```
# 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.

In [11]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
```

Data Wrangling

1. Filtered data using the BoosterVersion column to only keep the Falcon 9 launches,
2. Handled with the missing data values by replaced the PayloadMass , using mean value of column.
3. Created a landing outcome label from Outcome column
4. Performed some Exploratory Data Analysis (EDA)

GitHub URL

- https://github.com/yeemoonthong/IBM-Data-Science/blob/500a96379f41f85607ef836b5986739307869e74/Capstone/3.%20labs-jupyter-spacex-Data%20wrangling.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 [17]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = [0 if x in bad_outcomes else 1 for x in df['Outcome']]
```

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 [18]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

Out[18]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

In [20]:
```
df['Class'].value_counts()
```

Out[20]:
```
1    60
0    30
Name: Class, dtype: int64
```

In [19]:
```
df.head(5)
```

Out[19]:

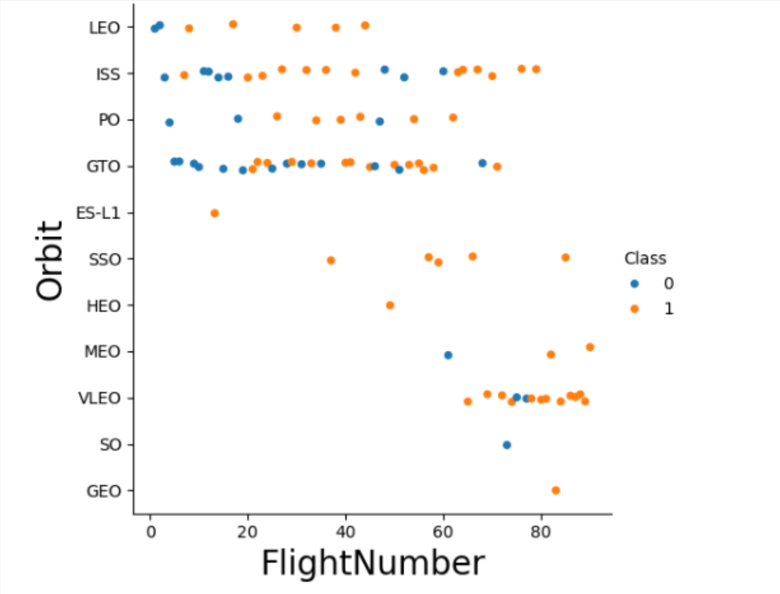
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block

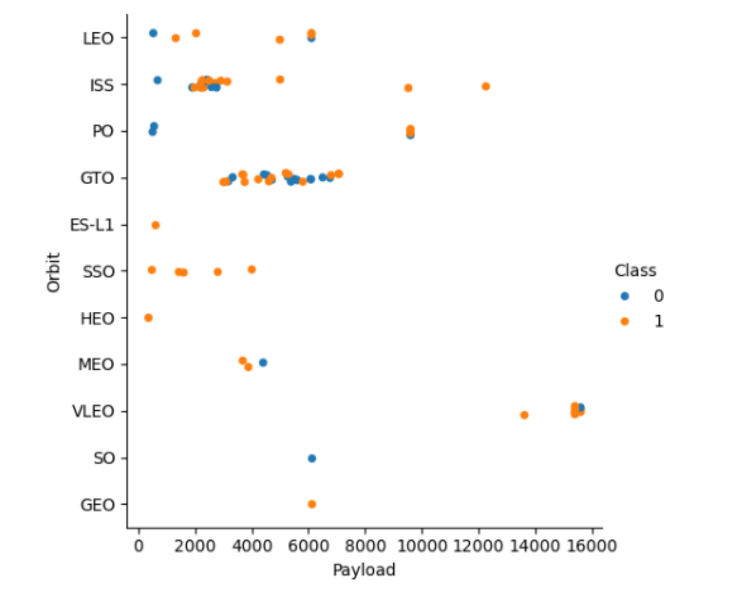
EDA with Data Visualization

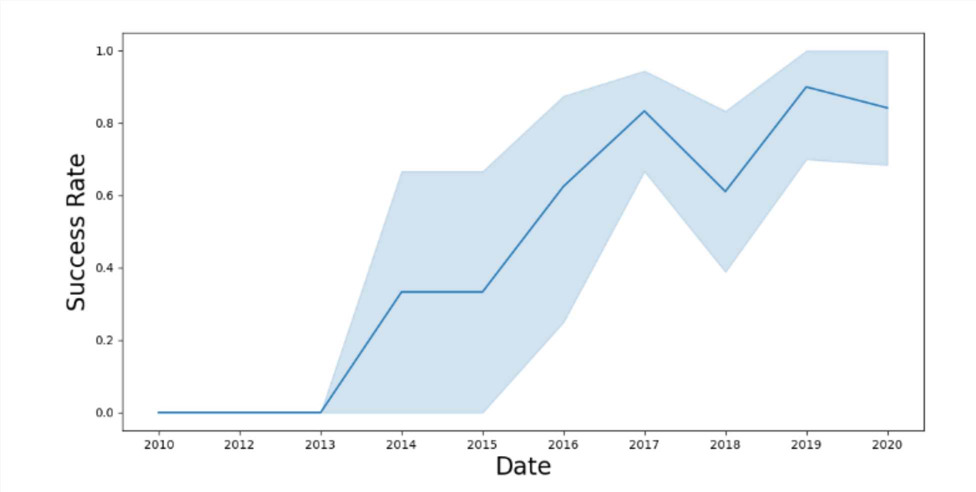
1. Exploratory Data Analysis
2. Preparing Data Feature Engineering
3. Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, FlightNumber and Orbit type, Payload and Orbit type.
4. Used Bar chart to Visualize the relationship between success rate of each orbit type
5. Line plot to Visualize the launch success yearly trend.
6. GitHub URL

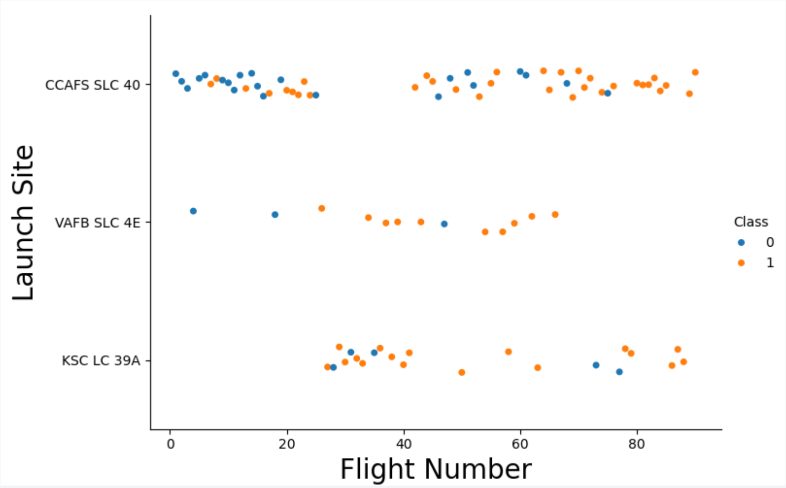
https://github.com/yeemoonthong/IBM-Data-Science/blob/500a96379f41f85607ef836b5986739307869e74/Capstone/5.%20jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

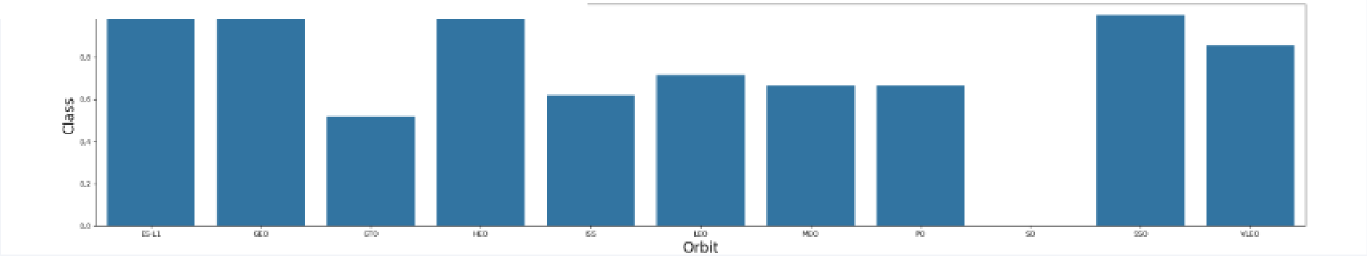
EDA with Data Visualization

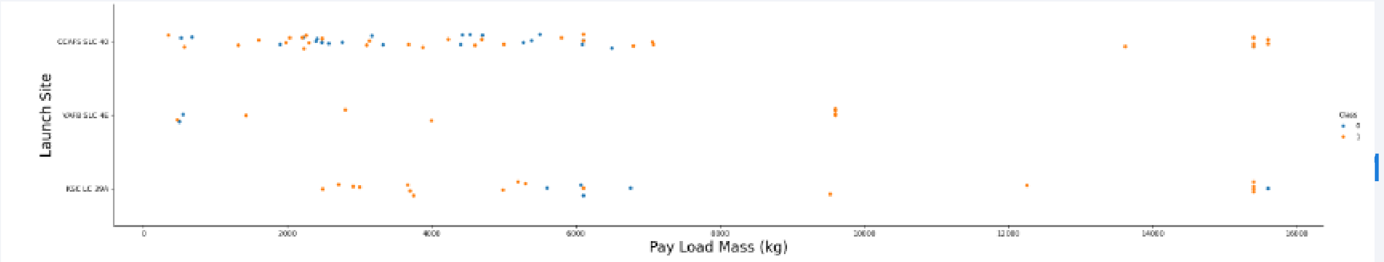












Feature Engineering

Now that our `features_one_hot` dataframe only contains numbers cast the entire dataframe to variable type `float64`

In [29]:

```
# HINT: use astype function
features_one_hot.astype(float)
```

Out[29]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	
1	2.0	525.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	
2	3.0	677.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	
3	4.0	500.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	
4	5.0	3170.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
85	86.0	15400.000000	2.0	1.0	1.0	1.0	5.0	2.0	0.0	0.0	...	0.0	
86	87.0	15400.000000	3.0	1.0	1.0	1.0	5.0	2.0	0.0	0.0	...	0.0	
87	88.0	15400.000000	6.0	1.0	1.0	1.0	5.0	5.0	0.0	0.0	...	0.0	
88	89.0	15400.000000	3.0	1.0	1.0	1.0	5.0	2.0	0.0	0.0	...	0.0	
89	90.0	3681.000000	1.0	1.0	0.0	1.0	5.0	0.0	0.0	0.0	...	0.0	

90 rows × 80 columns

EDA with SQL

- GitHub URL https://github.com/yeemoonthong/IBM-Data-Science/blob/500a96379f41f85607ef836b5986739307869e74/Capstone/4.%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA‘
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1`
5. List the date when the first successful landing outcome in ground pad was achieved
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcome
8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
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.
10. 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

EDA with SQL

- GitHub URL https://github.com/yeemoonthong/IBM-Data-Science/blob/500a96379f41f85607ef836b5986739307869e74/Capstone/4.%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

Example

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

In [12]:
```
%sql SELECT DISTINCT (Launch_Site) FROM SPACEXTBL
```

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

Out[12]:

Launch_Site
CCAFS LC-40

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

In [10]:
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE like 'CCA%' limit 5
```

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

Out[10]:

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)

Task 3

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

In [16]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_), Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)'
```

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

Out[16]:

SUM(PAYLOAD_MASS__KG_)	Customer
45596	NASA (CRS)

Build an Interactive Map with Folium

- Created folium map to marked all the launch sites, and created map objects such as markers, circles, lines to mark the success or failure of launches for each launch site.
- Created a launch set outcomes (failure=0 or success=1)
- GitHub URL https://github.com/yeemoonthong/IBM-Data-Science/blob/500a96379f41f85607ef836b59867393 07869e74/Capstone/6.%20lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Adding a Launch Site Drop-down Input Component
- Adding a callback function to render success-pie-chart based on selected site dropdown
- Adding a Range Slider to Select Payload
- Adding a callback function to render the success-payload-scatter-chart scatter plot

Predictive Analysis (Classification)

- Method = SVM,Decision Trees, k nearest neighbours Logistic Regression

1. Standardized the feature dataset (x) by transforming it using preprocessing.StandardScaler()
2. Split into training and testing sets using the function train_test_split with
3. Created an object for each of the algorithms
4. Created a GridSearchCV object and assigned them a set of parameters for each model to find best Hyperparameter.
5. Displayed the best parameters using the data attribute best_params_and the accuracy on the validation data using the data attribute best_score_
6. Used the method score to calculate the accuracy on the test data for each model
7. Plotted a confussion matrix for each using the test and predicted outcomes

- GitHub URL https://github.com/yeemoonthong/IBM-Data-Science/blob/500a96379f41f85607ef836b5986739307869e74/Capstone/7.%20SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

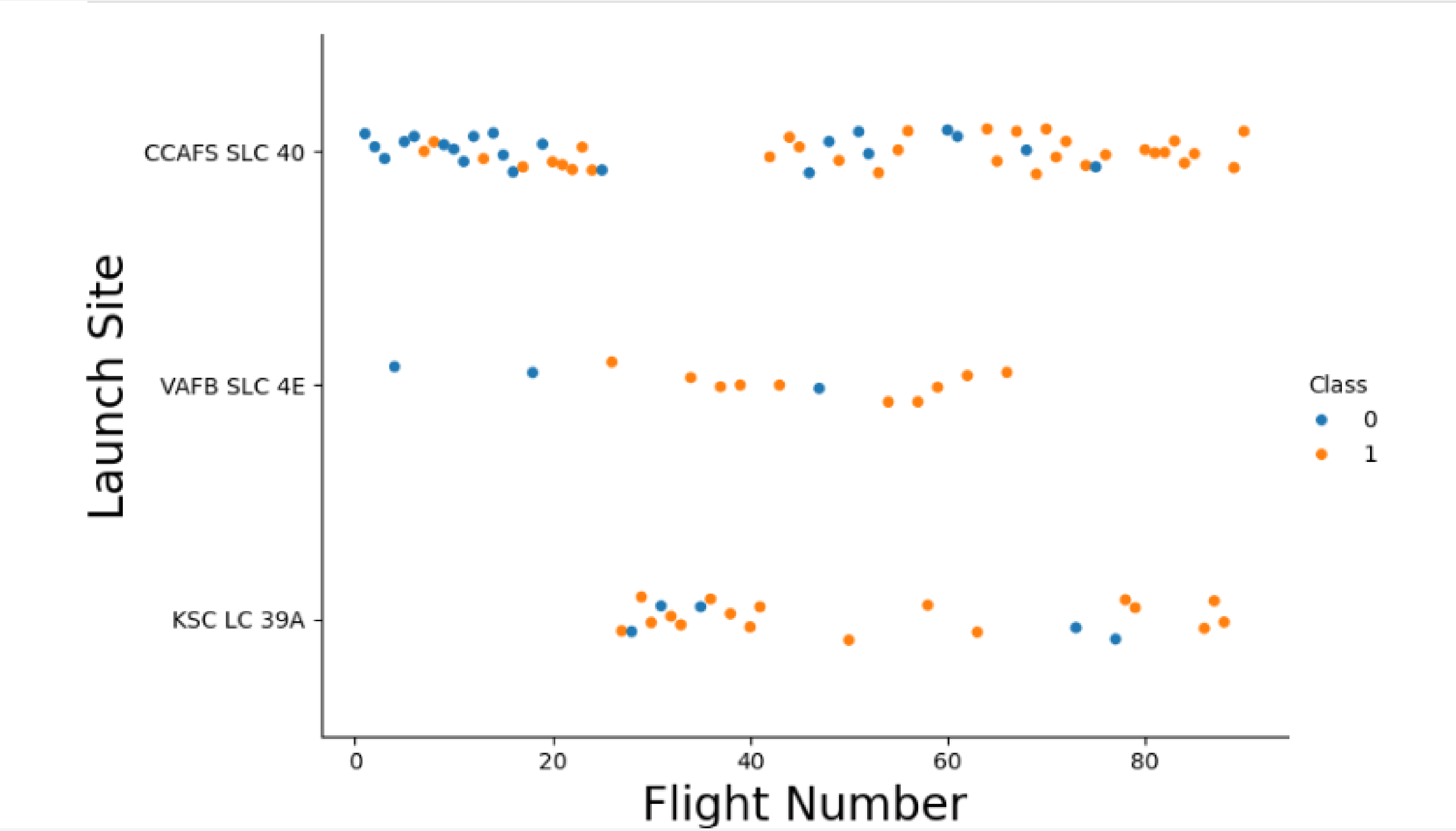
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site

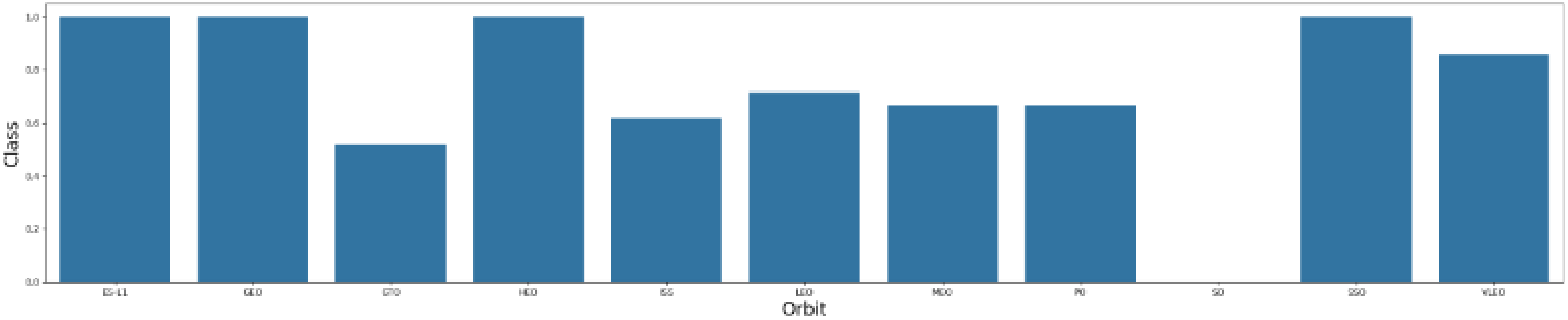


Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

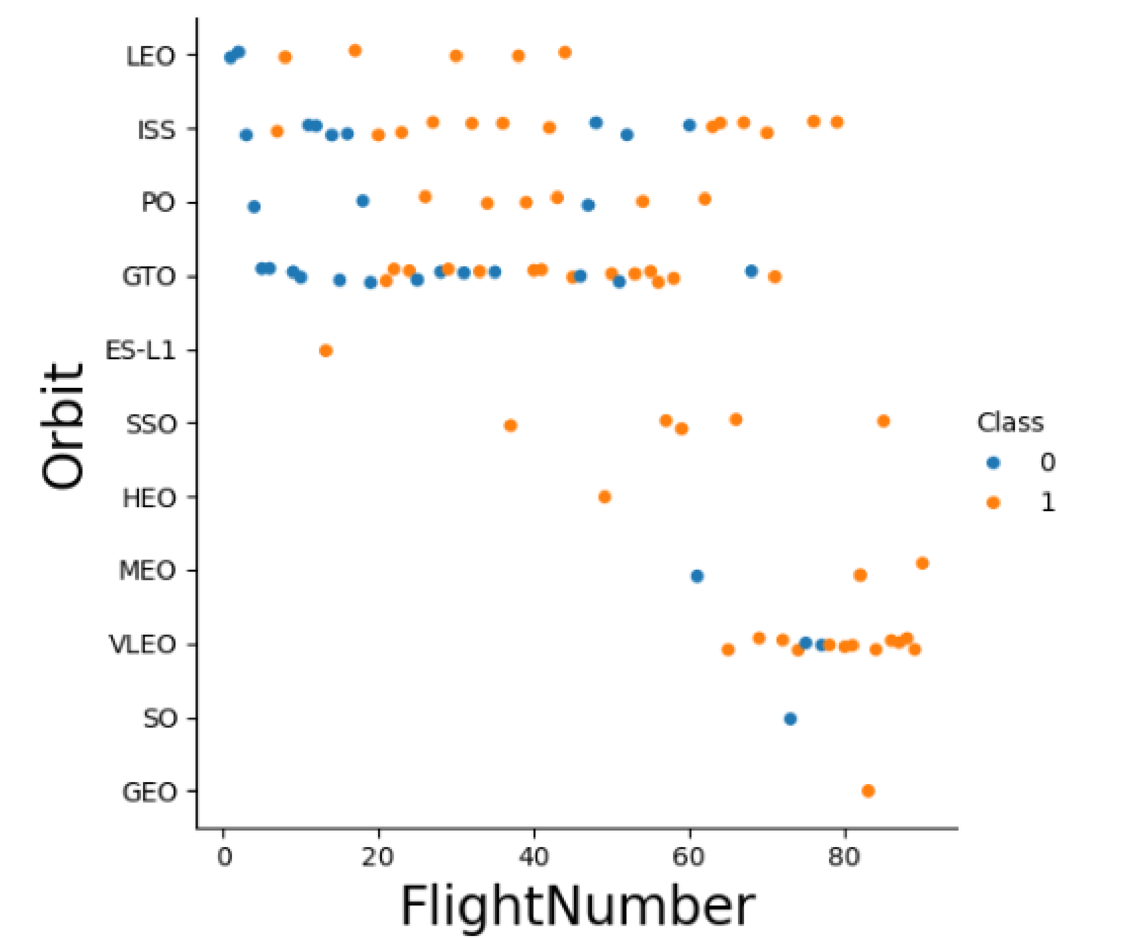
Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type



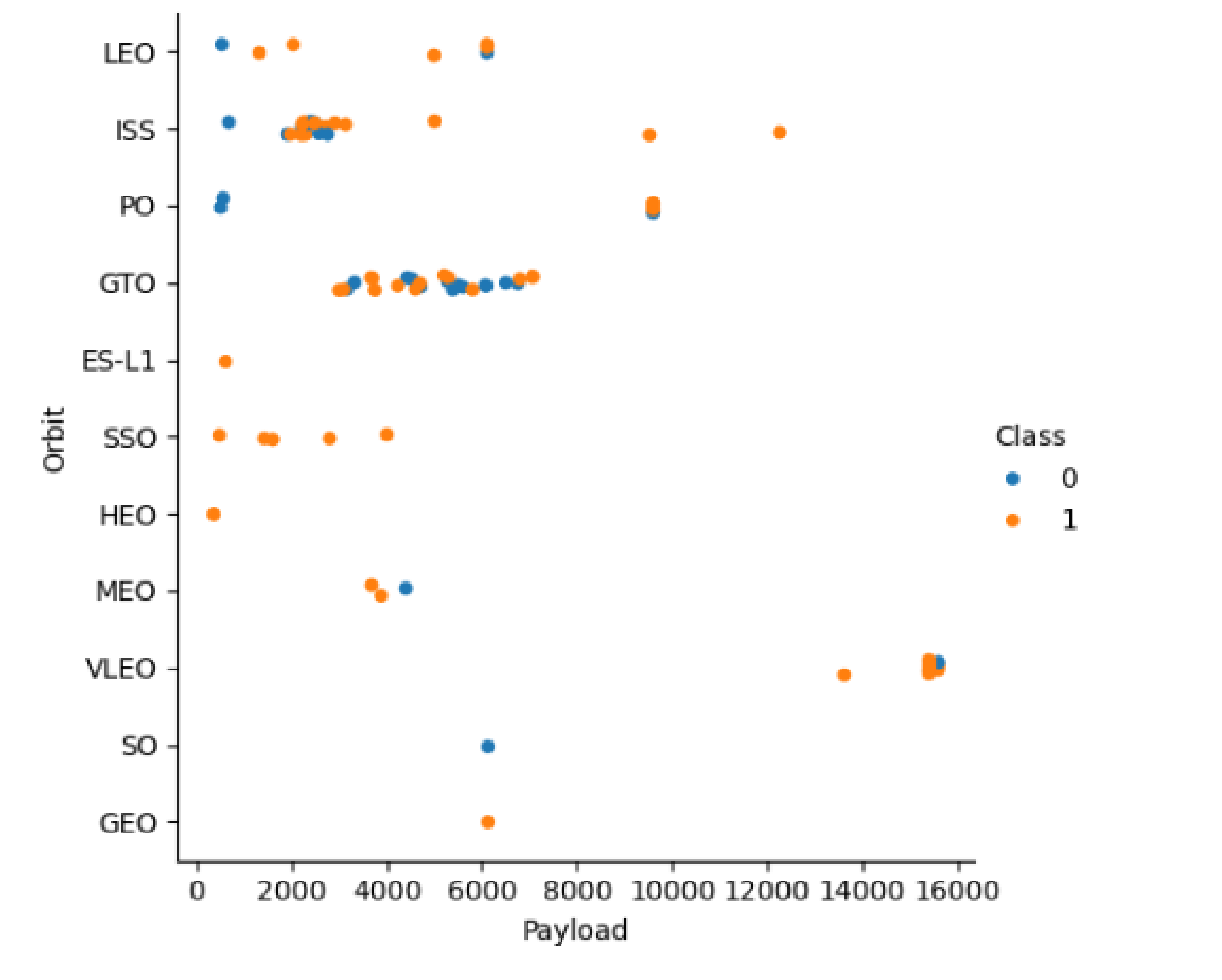
Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type



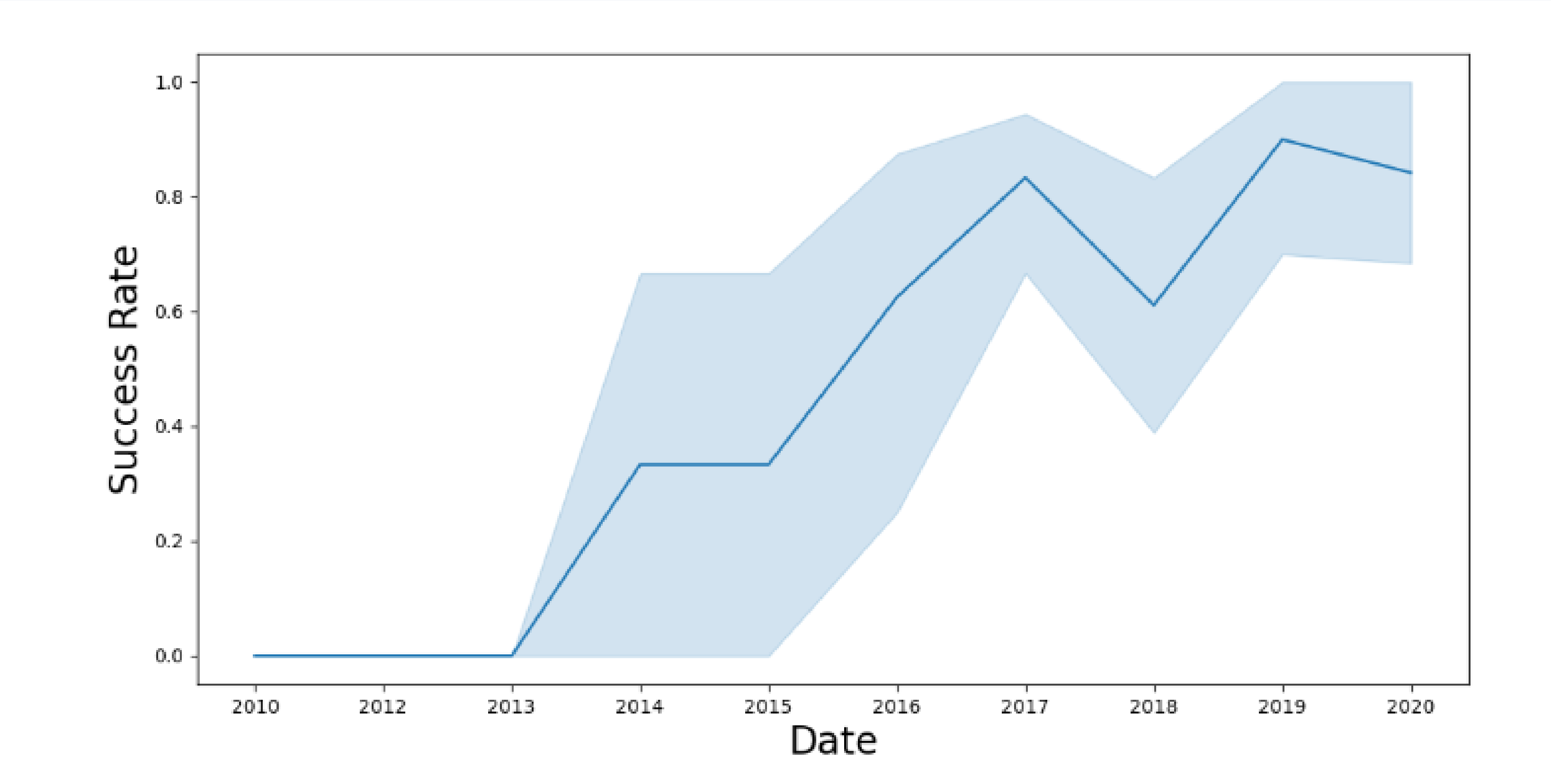
Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type



Launch Success Yearly Trend

- Show a line chart of yearly average success rate



All Launch Site Names

- Find the names of the unique launch sites

Task 1

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

In [12]:

```
%sql SELECT DISTINCT (Launch_Site) FROM SPACEXTBL
```

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

Out[12]:

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`

Task 2

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

In [10]:
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE like 'CCA%' limit 5
```

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

Out[10]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

- Calculate the total payload carried by boosters from NAS

Task 3

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

In [16]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_), Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)'
```

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

Out[16]:

SUM(PAYLOAD_MASS__KG_)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

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

Task 4

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

In [17]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_), Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version = 'F9 v1.1'
```

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

Out[17]:

AVG(PAYLOAD_MASS__KG_)	Booster_Version
2928.4	F9 v1.1

First Successful Ground Landing Date

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

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

Hint:Use min function

In [20]:

```
%sql SELECT min(Date) FROM SPACEXTBL WHERE Landing_Outcome = "Success (ground pad)"
```

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

Out[20]:

min(Date)
2015-12-22

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

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 [21]:
```
%sql select Booster_Version from SPACEXTBL \
WHERE Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
```

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

Out[21]:

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

Task 7

List the total number of successful and failure mission outcomes

In [23]:
```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") FROM SPACEXTBL\
GROUP BY "Mission_Outcome"
```

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

Out[23]:

Mission_Outcome	COUNT("Mission_Outcome")
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Task 8

Boosters Carried Maximum Payload

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

Task 8

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

In [27]:
```
%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.
```

Out[27]:

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

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [30]:

```
%sql SELECT substr(Date,0,5) as 'Year' , substr(Date, 6, 2) as 'Month', Booster_Version, Launch_Site, Payload, PAYLOAD_MASS_
FROM SPACEXTBL \
WHERE substr(Date,0,5)='2015' AND Landing_Outcome = 'Failure (drone ship)';
```

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

Out[30]:

Year	Month	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing_Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone ship)

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

Task 10

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.

In [32]:

```
%sql SELECT * FROM SPACEXTBL \
WHERE Landing_Outcome LIKE 'Success%' AND (Date BETWEEN '2010-06-04' AND '2017-03-20') \
ORDER BY Date DESC;
```

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

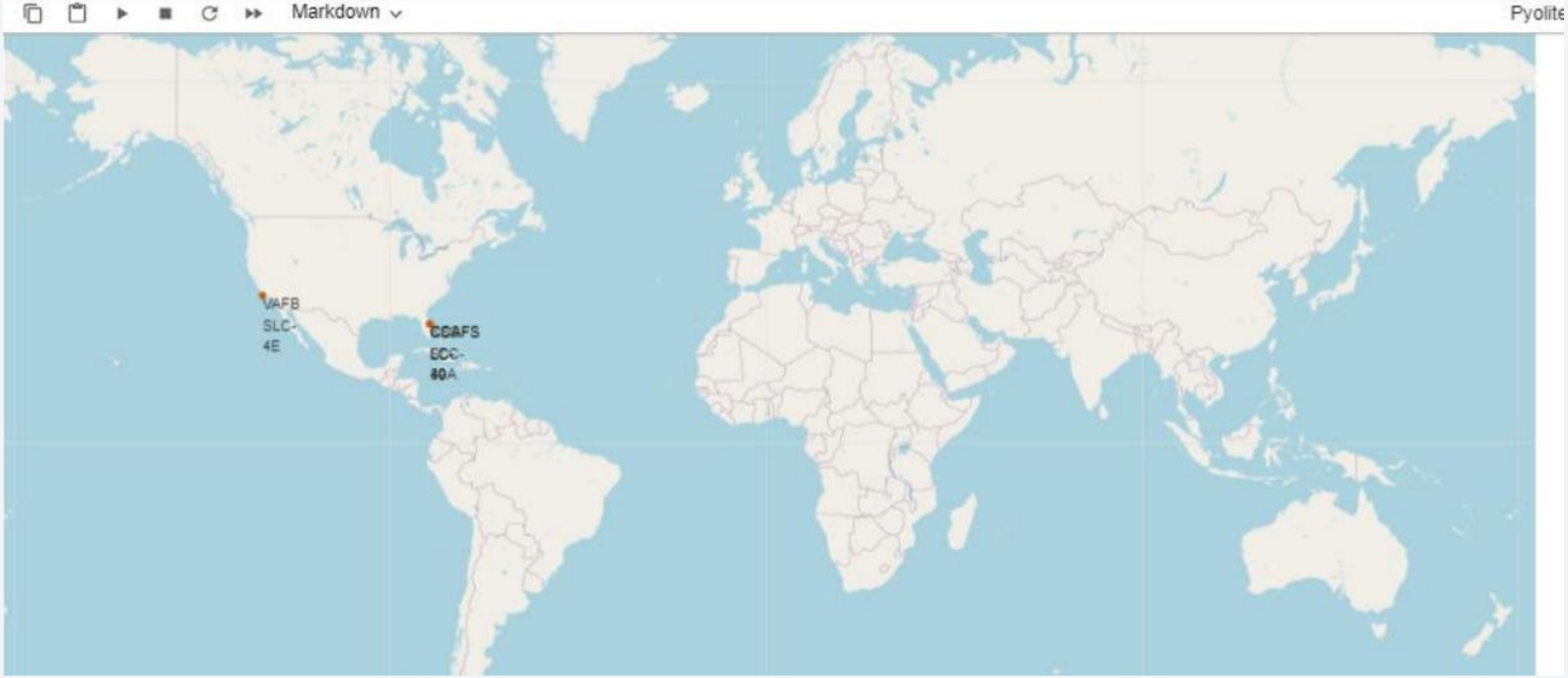
Out[32]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (gro
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600	Polar LEO	Iridium Communications	Success	Success (dr
2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (dr
2016-07-18	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (gro
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	Thaicom 8	3100	GTO	Thaicom	Success	Success (dr
2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (dr
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	Success (dr
2015-			CCAFS LC-	OG2 Mission 2 11					Success (gro

Section 3

Launch Sites Proximities Analysis

<Markers of all launch sites on global map>



<Launch outcomes for each site on the map With Color Markers>

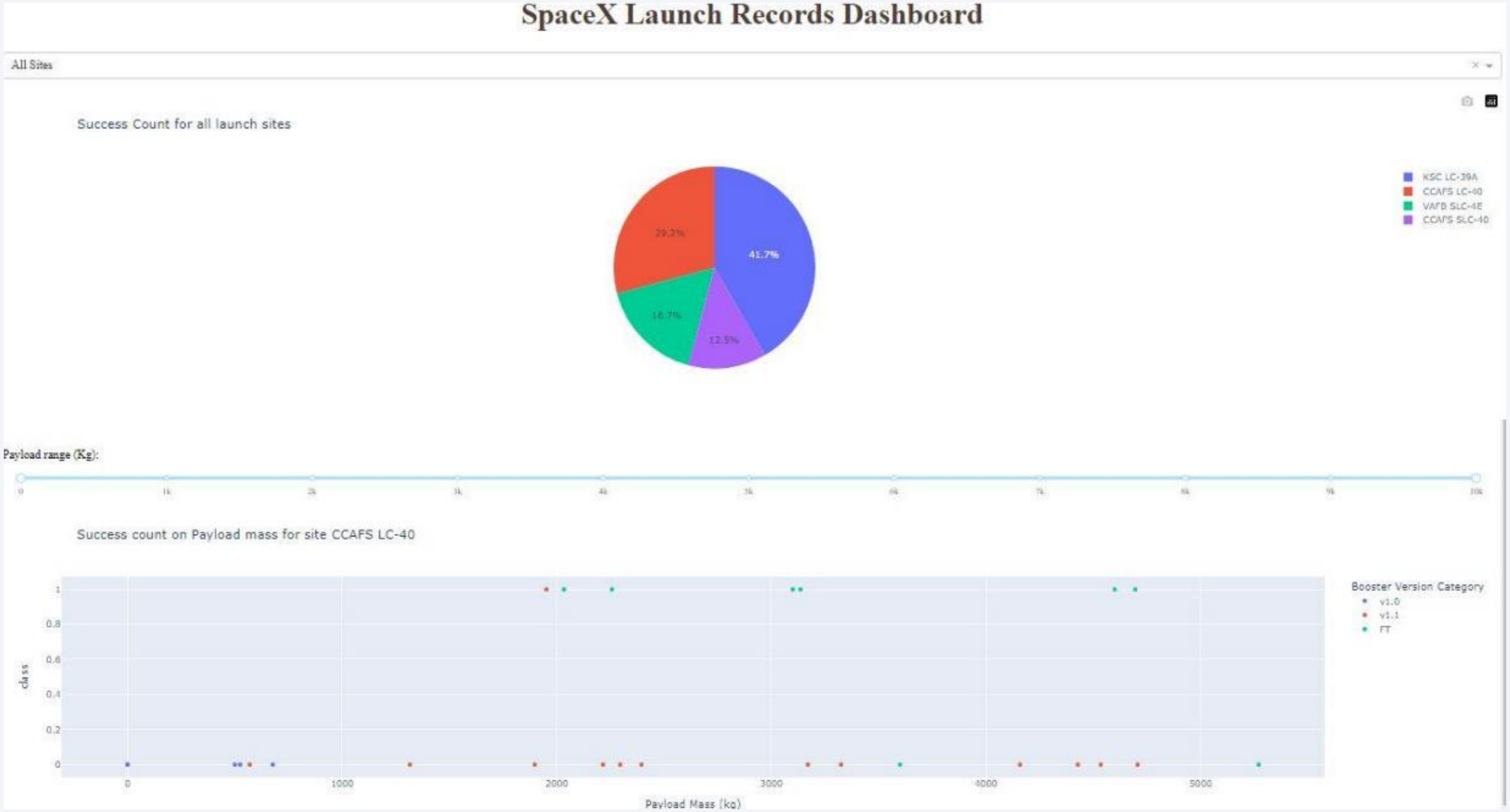
<Distances between a launch site to its proximities >

- Launch site CCAFS SLC-40 proximity to coastline is 0.86km

Section 4

Build a Dashboard with Plotly Dash

<Dashboard


SpaceX Launch Records Dashboard
All Sites
Success Count for all launch sites
41.7%
29.2%
16.7%
12.5%
KSC LC-39A
CCAFS LC-40
VAFB SLC-4E
CCAFS SLC-40
Payload range (Kg):
Success count on Payload mass for site CCAFS LC-40
Booster Version Category
v1.0
v1.1
FT
class
Payload Mass (kg)

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Out[41]:

	0	1
Method	Test Data Accuracy	Best Parameter
Logistic_Reg	0.833333	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
SVM	0.833333	{'C': 1.0, 'gamma': 0.03162277660168379, 'kern...
Decision Tree	0.666667	{'criterion': 'entropy', 'max_depth': 16, 'max...
KNN	0.833333	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}

- Highest Prediction Accuracy 83%

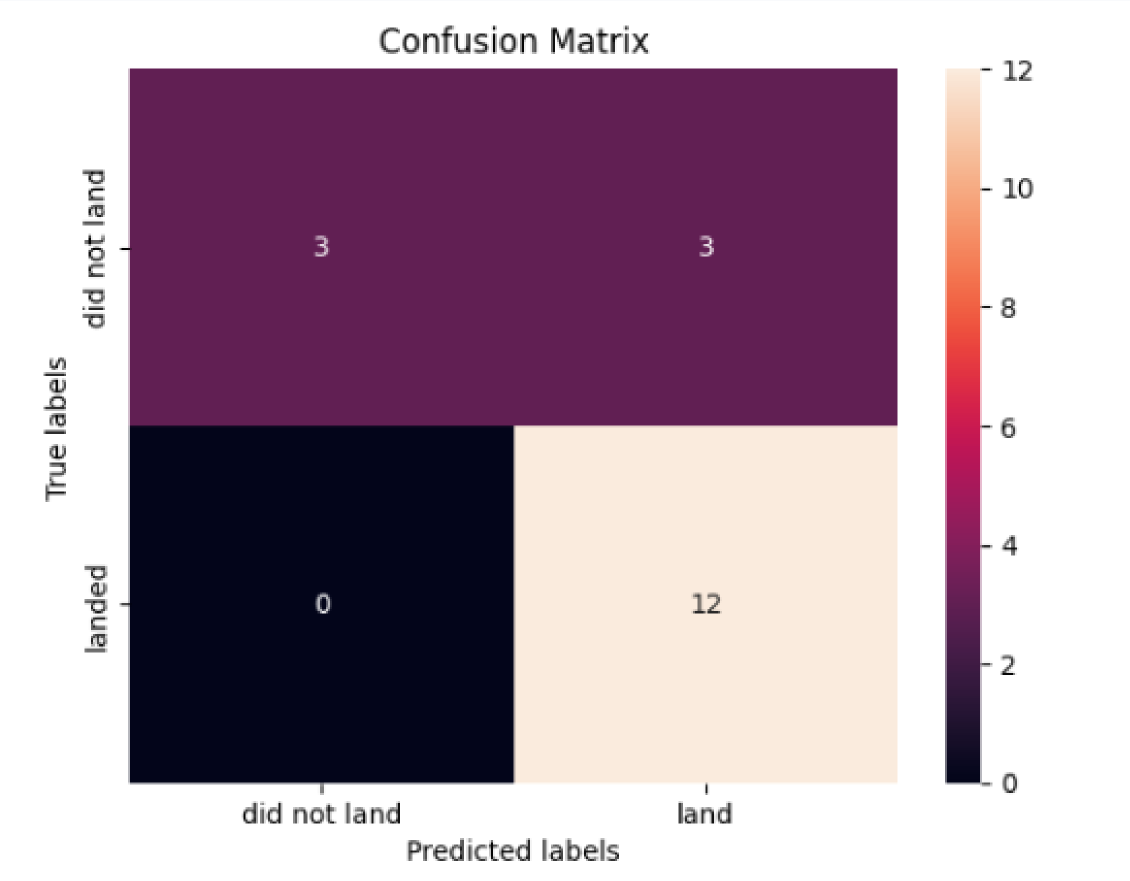
TASK 12

Find the method performs best:

In [41]:

```
Report = pd.DataFrame({'Method' : ['Test Data Accuracy','Best Parameter']})

knn_accuracy=knn_cv.score(X_test, Y_test)
Decision_tree_accuracy=tree_cv.score(X_test, Y_test)
SVM_accuracy=svm_cv.score(X_test, Y_test)
Logistic_Regression=logreg_cv.score(X_test, Y_test)

knn_param=knn_cv.best_params_
Decision_tree_param=tree_cv.best_params_
SVM_param=svm_cv.best_params_
Logreg_param =logreg_cv.best_params_


Report['Logistic_Reg'] = [Logistic_Regression,Logreg_param ]
Report['SVM'] = [SVM_accuracy,SVM_param]
Report['Decision Tree'] = [Decision_tree_accuracy,Decision_tree_param]
Report['KNN'] = [knn_accuracy,knn_param]

Report.transpose()
```

Confusion Matrix



Conclusions

- There is a correlation between launch site and success rate. Different launch sites have different success rates.
- Orbit SO has the least success rate but orbit ES-L1, GEO, HEO andSSO have the highest success rate.
- No relationship between flight number when in GTO orbit
- It can observe that success rate increased since 2013 kept increasing till 2020
- Logistic regression, KNN, and SVM used in prediction yielded the highest accuracy of 84% with best parameter using GridSearchCV.

Thank you!