IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Katherine Patricio
July 2023 – Sep 2023

Outline

Checklist to confirm that the ‘asks were completed’

Disclaimer

Introduction

Section 1: Methodology

Section 2: Insights Draw from Exploratory Data Analysis

Section 3: Launch Site Proximities Analysis

Section 4: Plotly Dash Dashboard

Section 5: Predictive Analysis Classification

Conclusion

Checklist

- ✓ Uploaded the URL of your GitHub repository including all the completed notebooks and Python files (1 pt)
- ✓ Uploaded your completed presentation in PDF format (1 pt)
- ✓ Completed the required Executive Summary slide (1 pt)
- ✓ Completed the required Introduction slide (1 pt)
- ✓ Completed the required data collection and data wrangling methodology related slides (1 pt)
- ✓ Completed the required EDA and interactive visual analytics methodology related slides (3 pts)
- ✓ Completed the required predictive analysis methodology related slides (1 pt)
- ✓ Completed the required EDA with visualization results slides (6 pts)
- ✓ Completed the required EDA with SQL results slides (10 pts)
- ✓ Completed the required interactive map with Folium results slides (3 pts)
- ✓ Completed the required Plotly Dash dashboard results slides (3 pts)
- ✓ Completed the required predictive analysis (classification) results slides (6 pts)
- ✓ Completed the required Conclusion slide (1 pts)
- ✓ Applied your creativity to improve the presentation beyond the template (1 pts) – refer to the conclusion slide/s
- ✓ Displayed any innovative insights (1 pts) – refer to the conclusion slide/s

Disclaimer

The next slides have been primarily prepared based on information from the Edx IBM DS0720EN Data Science and Machine Learning Capstone Project. The content and analysis presented in this project are solely derived from publicly accessible sources, such as published reports, articles, and data that are widely available to the general public.

While utmost care has been taken to ensure the accuracy and reliability of the information presented, it is important to note that the findings, conclusions, and recommendations expressed in this capstone project are solely my own and do not represent the views or opinions of any company or its affiliates.

I hereby acknowledge that the limitations imposed on my ability to access and utilize proprietary information may impact the comprehensiveness and depth of this capstone project. However, I have made every effort to provide a thorough and insightful analysis within the constraints of the available information.

Furthermore, any references to specific companies, products, or services within this capstone project are for illustrative purposes only and should not be interpreted as an endorsement or promotion. The inclusion of such references does not imply any association or affiliation with the mentioned entities.

Readers are encouraged to exercise their own judgment and conduct further research if necessary. This capstone project should be interpreted as an academic exercise, and any decisions or actions taken based on its contents are done at the reader's own risk.

By accessing or utilizing this capstone project, you agree to release me from any liability or responsibility arising from the use or misuse of the information presented herein. You understand that this project is intended for educational and non-commercial purposes only.

KP

Introduction

Project background and context

SpaceX, founded by Elon Musk, has transformed the space industry with its Falcon 9 rocket, featuring a reusable first stage that significantly reduces launch costs. Unlike traditional rockets that are discarded after launch, SpaceX's ability to recover and reuse the first stage has allowed them to offer Falcon 9 launches at a fraction of the cost compared to other providers. With a price tag of $62 million per launch, SpaceX's cost advantage has attracted attention from alternate companies looking to compete in the rocket launch market.

In this capstone project, the focus is on predicting the success of Falcon 9 first stage landings. By accurately determining whether the first stage will land successfully, we can predict where to invest. To achieve this goal, the project aims to develop a predictive model by analyzing historical data and relevant factors such as weather conditions, payload characteristics, and mission parameters. By leveraging this model, alternate companies can assess the likelihood of a successful first stage landing, providing them with crucial insights to make strategic decisions when competing against SpaceX. Ultimately, the project's outcome holds the potential to reshape the space industry by shedding light on the cost-effectiveness of reusable rocket technology, driving sustainable and economically viable space exploration endeavors, and fostering healthy competition in the market.

What are we trying to find:

What is the most successful launch site. What are the interesting factors that could show the highest instances of a successful launch?

Section 1

Methodology

Methodology

Executive Summary

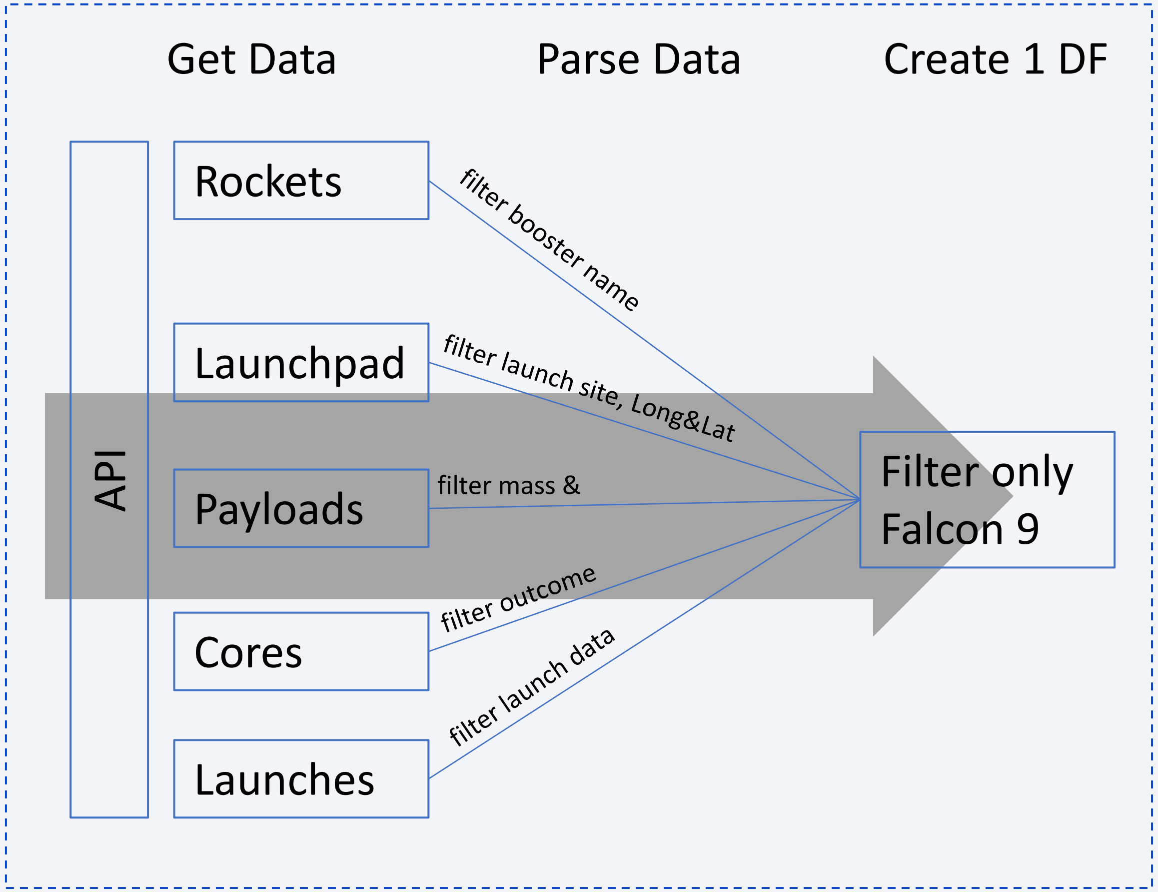
- Data collection methodology:
  - API data collection methodology involves using Application Programming Interfaces (APIs) to retrieve data from various sources, enabling automated and structured access to specific data endpoints for integration into applications or analysis purposes.
- Perform data wrangling
  - Data wrangling is the process of cleaning, transforming, and preparing raw data to make it suitable for analysis and modeling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Predictive analysis using classification models utilizes machine learning algorithms to classify new data instances based on patterns and relationships found in labeled historical data.

Data Collection

- Describe how data sets were collected.
  - Data was collected from
    - https://api.spacexdata.com/v4/rockets/
    - https://api.spacexdata.com/v4/launchpads/
    - https://api.spacexdata.com/v4/payloads/
    - https://api.spacexdata.com/v4/cores/
    - https://api.spacexdata.com/v4/launches/past
    - https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts
  - https://api.spacexdata.com/v4/rockets/
  - https://api.spacexdata.com/v4/launchpads/
  - https://api.spacexdata.com/v4/payloads/
  - https://api.spacexdata.com/v4/cores/
  - https://api.spacexdata.com/v4/launches/past
- Add the GitHub URL of the completed SpaceX API calls notebook as an external reference and peer-review purpose
  - https://github.com/keitpatricio/KP_GitHub_for_IBM_Python_Data_Science.git



Webscraping

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

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch W

First, let's perform an HTTP GET method to request the F

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

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

```
<Response [200]>
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup ob
soup = BeautifulSoup(response.text)
```

Print the page title to verify if the `BeautifulSoup` objec

```
# Use soup.title attribute
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 abo

```
# Use the find_all function in the BeautifulSoup object, with element type `tabl
# 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)
```

Outputs are collapsed ...

You should able to see the columns names embedded in the table header elements `<th`

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

```
column_names = []
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

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

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010
1	2	CCAFS	Dragon	0	LEO	NASA (COTS)\nNRO	Success	F9 v1.0B0004.1	Failure	8 December 2010
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013
...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX Capella Space and Tyvak	Success\n	F9 B5B1058.8	Success	15 May 2021
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA (CRS)	Success\n	F9 B5B1067.1	Success	3 June 2021
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021

121 rows × 11 columns

https://github.com/keitpatricio/KP_GitHub_for_IBM_Python_Data_Science.git

Data Wrangling

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()
```

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

TASK 3: Calculate the number and occurence of mission outcome of the orbits

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
```

```
Outcome
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: count, dtype: int64
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

df['landing_class'] = df['Outcome'].apply(lambda x: 0 if x == bad_outcomes else 1)

# a = filter(lambda x: x in bad_outcomes, df['Outcome'])

# landing_outcomes = df['Outcome'].map(lambda x: 0 if bad_outcomes else 1)

# df['new'] = lambda x: 0 if x in list bad_outcomes else 1
df
```

```
df.head(5)
```

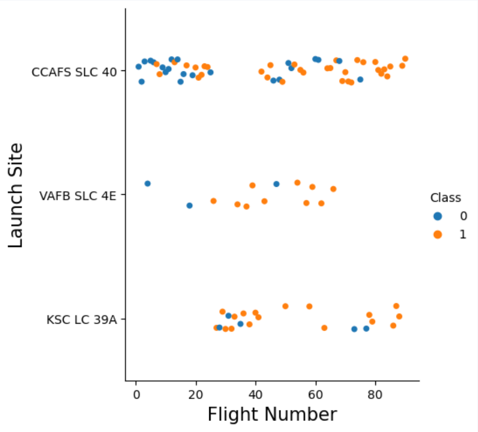
Python

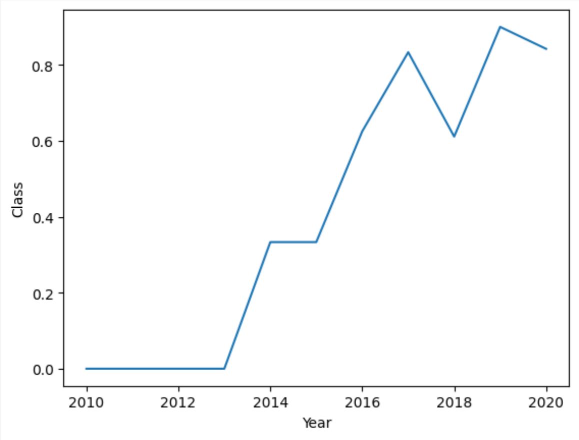
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
2	3	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

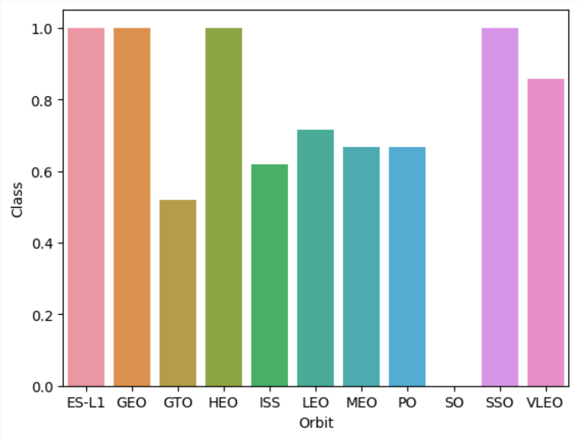
https://github.com/keitpatricio/KP_GitHub_for_IBM_Python_Data_Science.git

EDA with Data Visualization

- Scatter, line and bar charts were used to compare and see trends







https://github.com/keitpatricio/KP_GitHub_for_IBM_Python_Data_Science.git

KP_jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
  - %%sql
    - select * from SPACEXTBL
    - select Launch_Site from SPACEXTBL where Launch_Site like 'KSC%'
    - select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_version like '%F9 v1.1%'
    - select * from SPACEXTBL where lower(Landing_Outcome) like '%drone%' and lower(Landing_Outcome) like '%success%' order by 'Date' limit 1
    - select * from SPACEXTBL where lower(Landing_Outcome) like '%ground%' and lower(Landing_Outcome) like '%success%' and PAYLOAD_MASS__KG_ between 4000 and 6000
    - select case when lower(Mission_Outcome) like "%success%" then 'Success' else 'Failure' end as mission, count(*) as countM from SPACEXTBL group by case when lower(Mission_Outcome) like "%success%" then 'Success' else 'Failure' end
    - etc

```
[59]: %%sql select
substr(Date,4,2) as month
, Landing_Outcome
, Booster_Version
, Launch_Site
from SPACEXTBL
where substr(Date,7,4)='2017'
and lower(Landing_Outcome) like '%success%'
and lower(Landing_Outcome) like '%ground%'

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

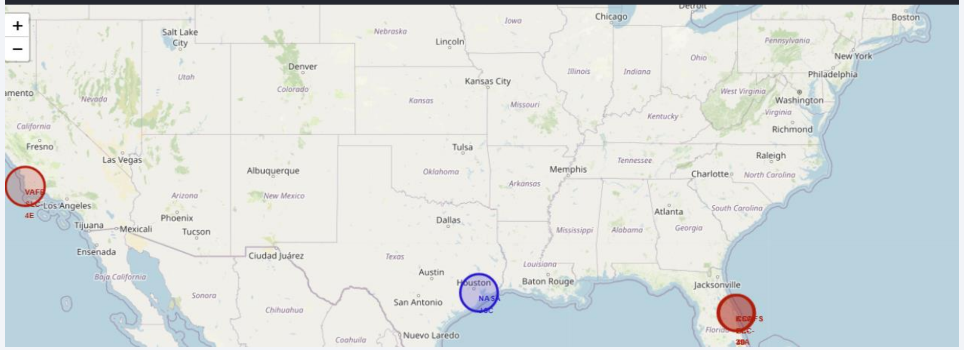
[59]:

month	Landing_Outcome	Booster_Version	Launch_Site
02	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
01	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
03	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
08	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
07	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

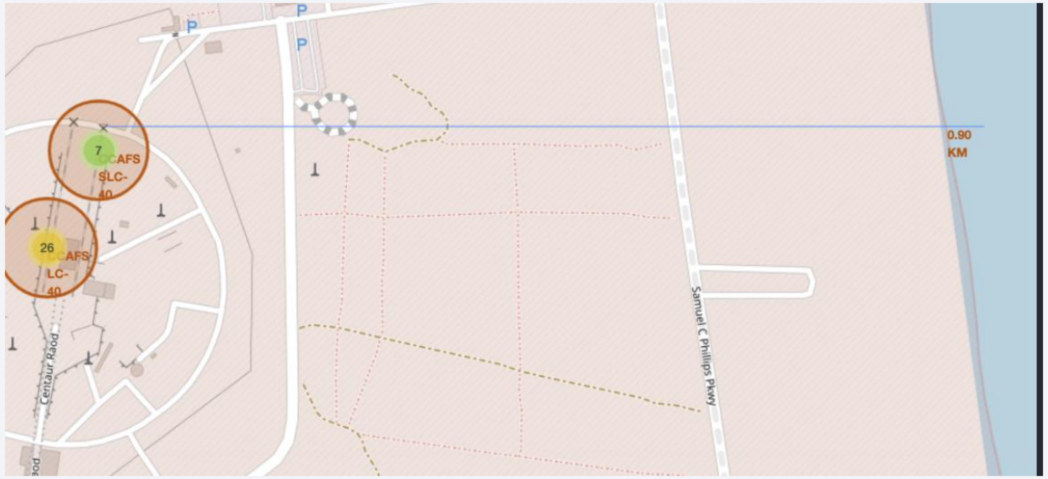
- Add the GitHub URL
  - https://github.com/keitpatricio/KP_GitHub_for_IBM_Python_Data_Science.git
  - KP_jupyter-labs-eda-sql-edx_sqllite.ipynb

Build an Interactive Map with Folium

```python
# Launch site 1 CCAFS LC-40
LS1_ll = [28.562302, -80.577356]
LS1_circle = folium.Circle(LS1_ll, radius=100000, color='#d30000', fill=True).add_child(folium.Popup('LC-40'))
LS1_marker = folium.map.Marker(LS1_ll, icon=DivIcon(icon_size=(10,10),icon_anchor=(0,0), html='<div style="font-size: 12;
site_map.add_child(LS1_circle)
site_map.add_child(LS1_marker)

# Launch site 2 CCAFS SLC-40
LS2_ll = [28.563197,    -80.576820]
LS2_circle = folium.Circle(LS2_ll, radius=100000, color='#d30000', fill=True).add_child(folium.Popup('SLC-40'))
LS2_marker = folium.map.Marker(LS2_ll, icon=DivIcon(icon_size=(10,10),icon_anchor=(0,0), html='<div style="font-size: 12;
site_map.add_child(LS2_circle)
site_map.add_child(LS2_marker)

# Launch site 3 KSC LC-39A
LS3_ll = [28.573255,    -80.646895]
LS3_circle = folium.Circle(LS3_ll, radius=100000, color='#d30000', fill=True).add_child(folium.Popup('LC-39A'))
LS3_marker = folium.map.Marker(LS3_ll, icon=DivIcon(icon_size=(10,10),icon_anchor=(0,0), html='<div style="font-size: 12;
site_map.add_child(LS3_circle)
site_map.add_child(LS3_marker)

# Launch site 4 VAFB SLC-4E
LS4_ll = [34.632834,    -120.610745]
LS4_circle = folium.Circle(LS4_ll, radius=100000, color='#d30000', fill=True).add_child(folium.Popup('SLC-4E'))
LS4_marker = folium.map.Marker(LS4_ll, icon=DivIcon(icon_size=(10,10),icon_anchor=(0,0), html='<div style="font-size: 12;
site_map.add_child(LS4_circle)
site_map.add_child(LS4_marker)
```

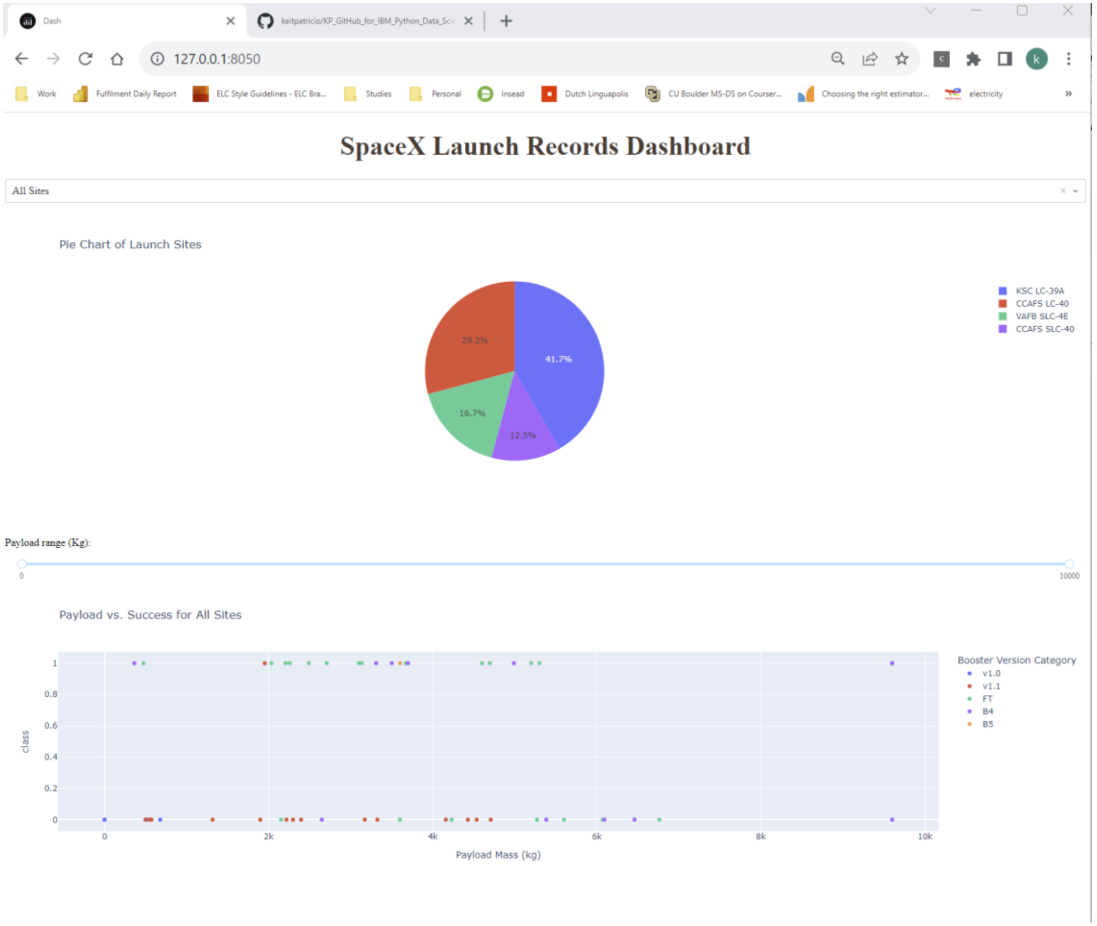


- Marked new circles in red
- Add the GitHub URL
  - https://github.com/keitpatricio/KP_GitHub_for_IBM_Python_Data_Science.git
  - KP_lab_jupyter_launch_site_location.jupyterlite.ipynb



Build a Dashboard with Plotly Dash

- Pie chart and scatterplot were added to plotlydash http://127.0.0.1:8050/
- Pie chart was used to show which has the most successful launch. This was identified with class 1 (successful). Note that class 0 != successful landing
- https://github.com/keitpatricio/KP_GitHub_for_IBM_Python_Data_Science/blob/0e22a7b08d5564d1a24f73342971975333c4401b/plotly_dash_interactivity_kp.py



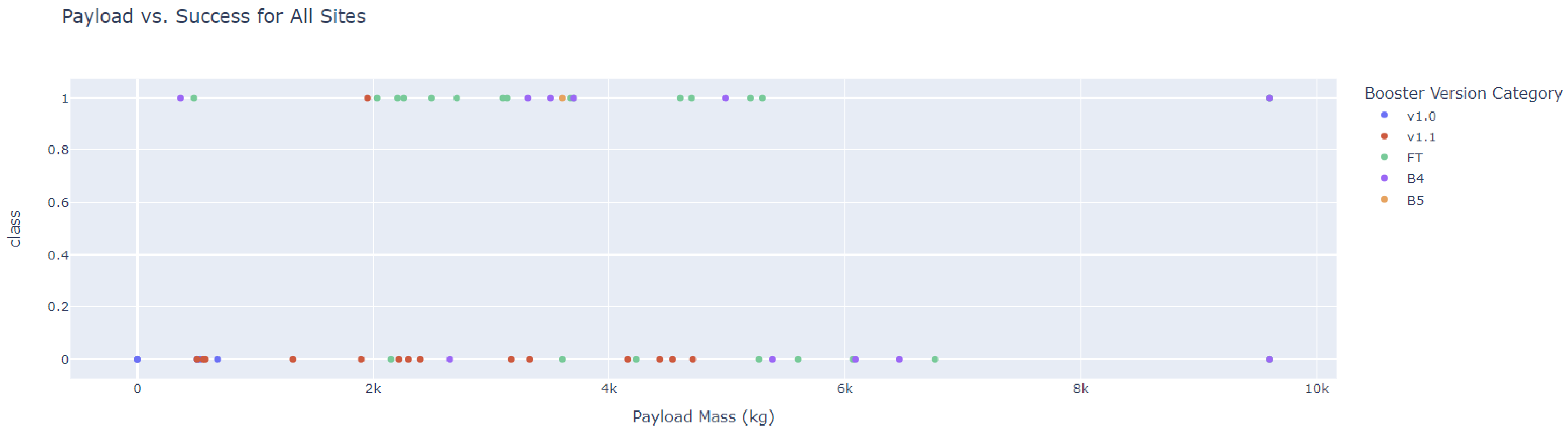
Build a Dashboard with Plotly Dash

- The site that has the largest successful launch is KSC LC-39A,
- The site that has the highest launch success rate is KSC LC-39A

Row Labels	Sum of class	Count of class2	Success Rate
CCAFS LC-40	7	26	27%
CCAFS SLC-40	3	7	43%
KSC LC-39A	10	13	77%
VAFB SLC-4E	4	10	40%
Grand Total	24	56	43%

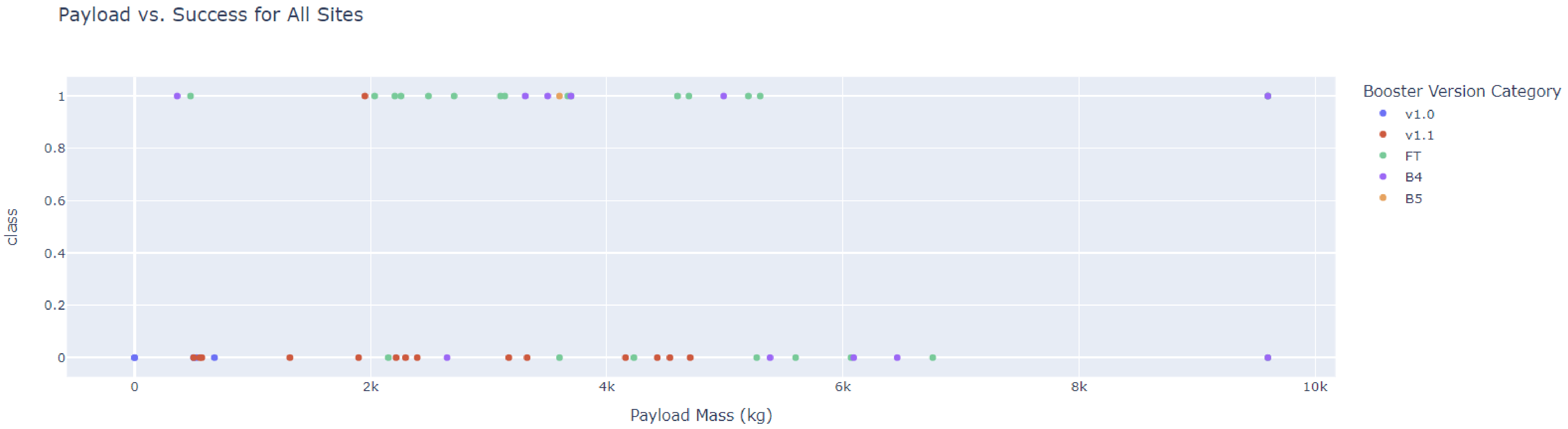
Build a Dashboard with Plotly Dash

- The F9 booster version that has the highest launch success rate is green FT



Build a Dashboard with Plotly Dash

- The payload range that has the highest launch success rates is 2k – 4k
- The payload range that has the lowest launch success rate is 6-8k



Predictive Analysis (Classification) - Process

Process Flow

Break into training data (to train the model) and testing (to check the accuracy of the model)

Break data into X independent & Y dependent data

Get 80% for the train data for classification modeling

Model the data based on each classification

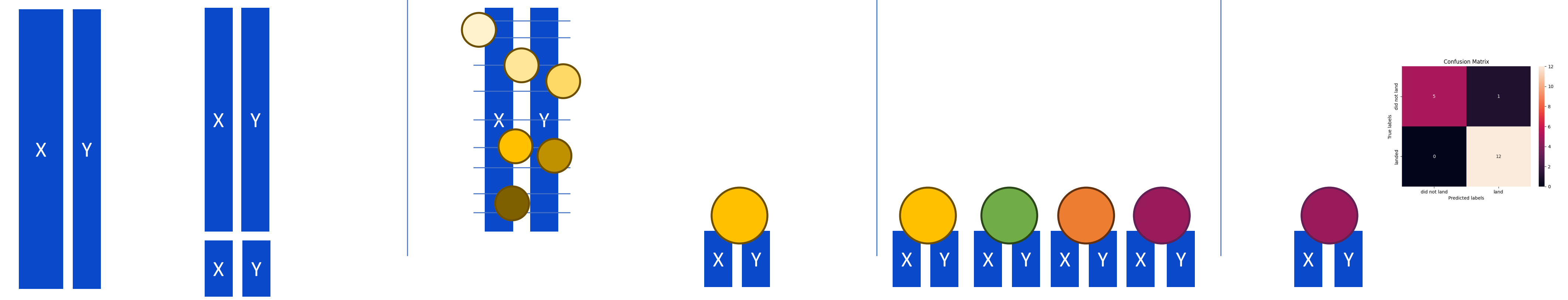
Split into 10 then fit the model to the training set, recommend best parameter combo

Apply the best model to the test data set and get the score

Analyze

Compare classification scores and confusion matrices

Analyze what are
- TP true positives
- TN true negatives
- FP false positives
- FN false negatives



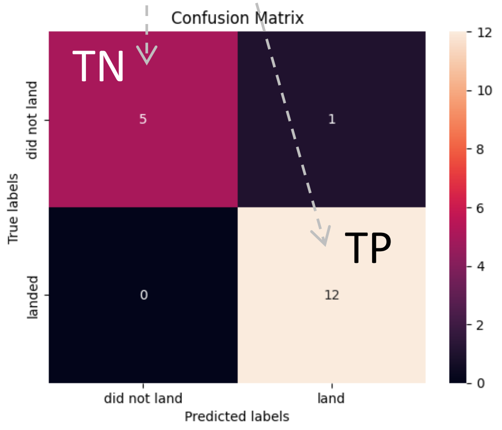
Git Hub URL:

https://github.com/keitpatricio/KP_GitHub_for_IBM_Python_Data_Science/blob/main/KP_train_test_split.ipynb

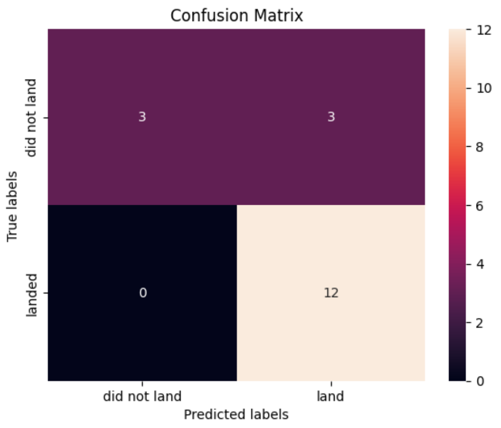
Predictive Analysis (Classification) - Result

- Decision Tree (DT) Classifier is the best at 94%
- TP True positive: DT can positively predict if will land
- TN True negative: DT can positively predict if will not land

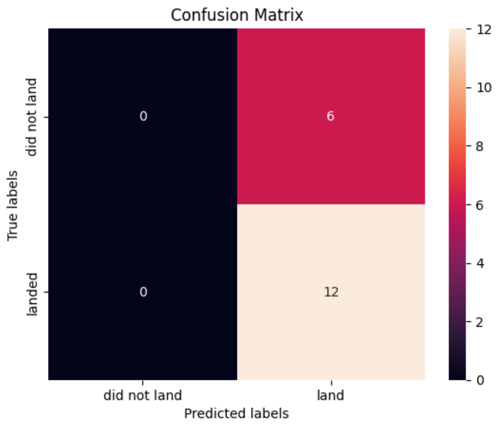
Decision Tree Classifier: 94%



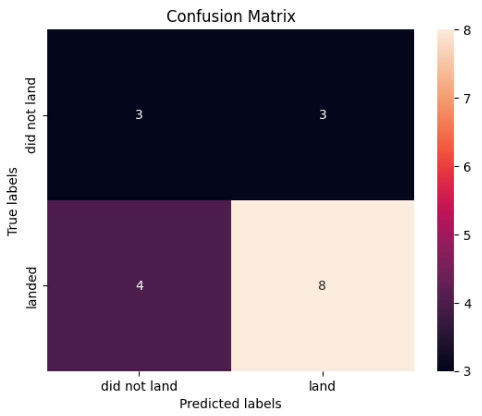
Logistics Regression: 83%



Support vector machine: 67%



K Nearest Neighbors: 61%



Git Hub URL:

https://github.com/keitpatricio/KP_GitHub_for_IBM_Python_Data_Science/blob/main/KP_train_test_split.ipynb

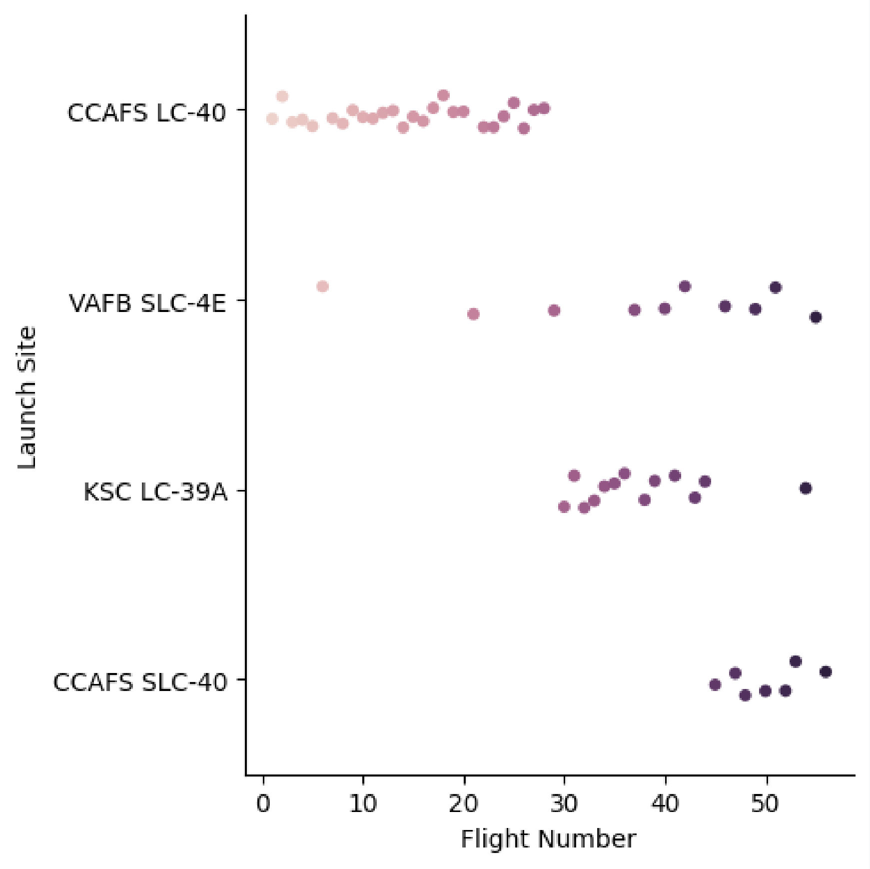
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Scatter plot of Flight Number vs. Launch Site

- The flights started with CCAFS LC40 launch site. The later flights were from CCAFS SLC40.

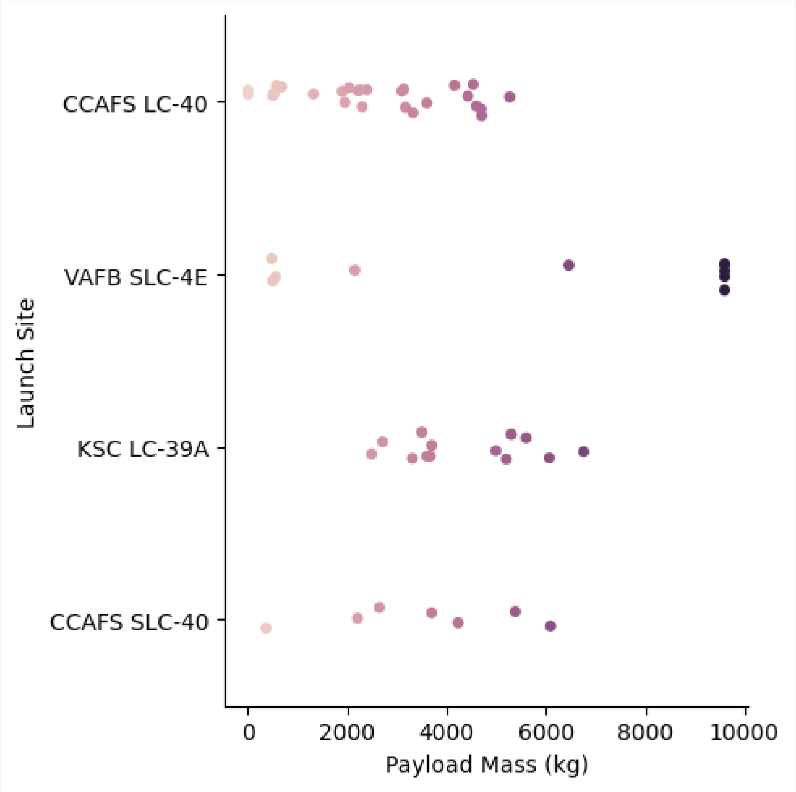



```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_dash.csv")
sns.catplot(data=df,x='Flight Number',y='Launch Site', hue='Flight Number')
```

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

- The heaviest payloads were from launch site VAFB SLC4E.

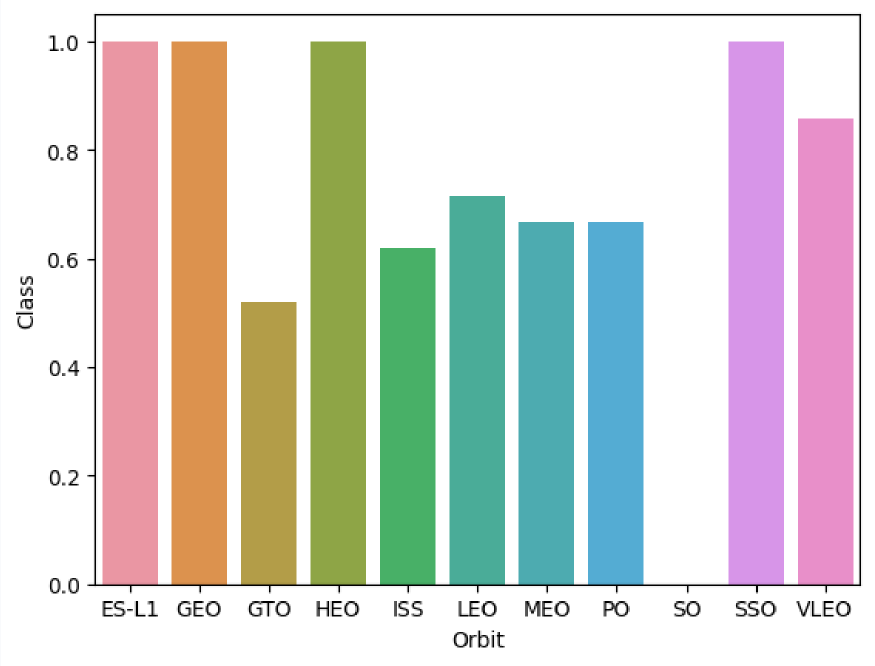



```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_dash.csv")
sns.catplot(data=df,x='Payload Mass (kg)',y='Launch Site', hue='Payload Mass (kg)')
```

Success Rate vs. Orbit Type

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

- The bar chart shows the mean (average) of the success rate per orbit. The bars that reach 1 have a success rate of 100%

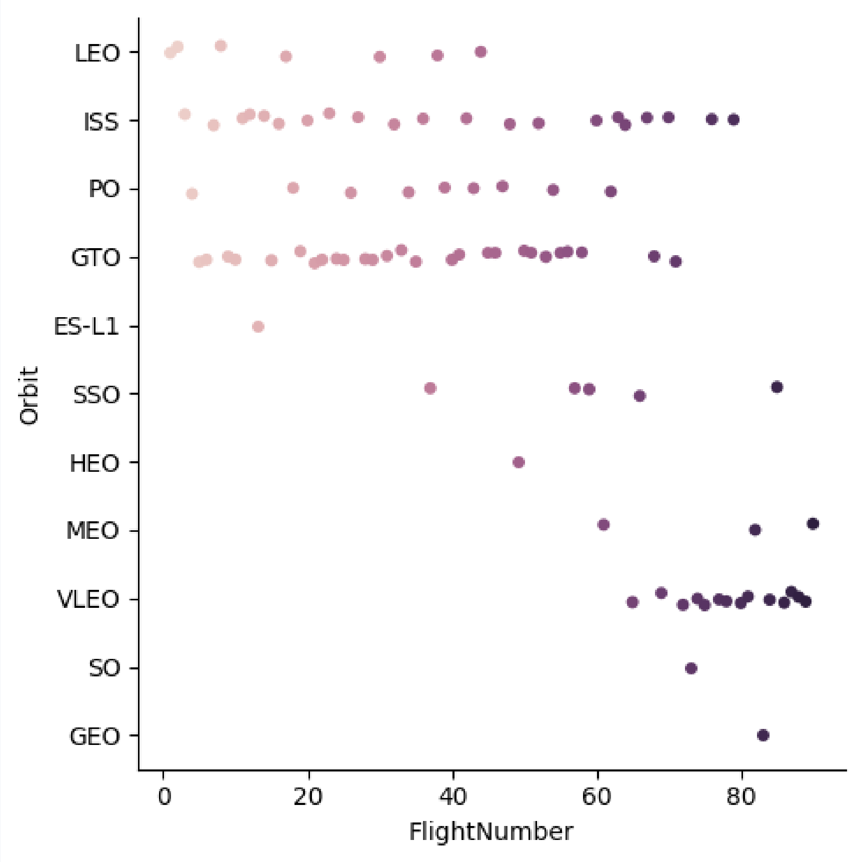



```
df2 = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_part_2.csv')
orbit_success = df2[['Orbit','Class']].groupby('Orbit').mean()
orbit_success.reset_index(inplace=True)
sns.barplot(data=orbit_success,x='Orbit',y='Class')
```

Flight Number vs. Orbit Type

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

- The earliest flights were orbit LEO, ISS PO etc. The last flights were orbit SSO, MEO, VLEO, SO, GOE.

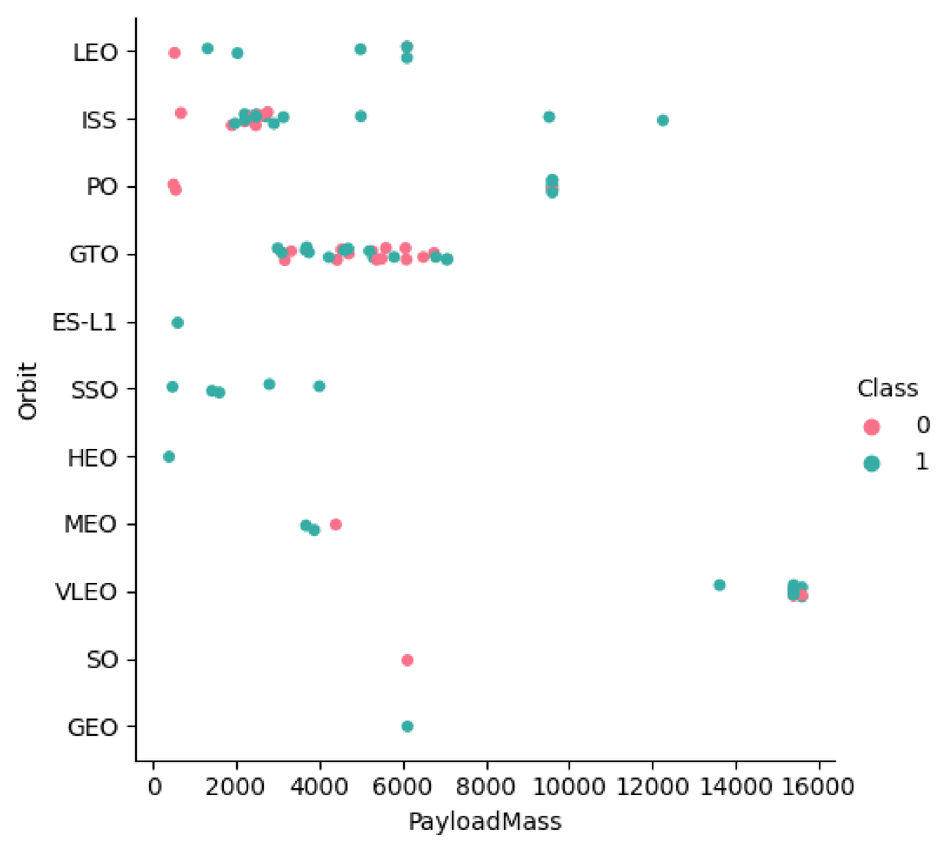



```
sns.catplot(data=df2,x='FlightNumber',y='Orbit', hue='FlightNumber')
```

Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

- Class 1 is are successful launches. It's interesting to see that ES-L1, SSO and HEO have low payload mass and were successful. But not all low payload masses were successful (like LEO, ISS, and PO)



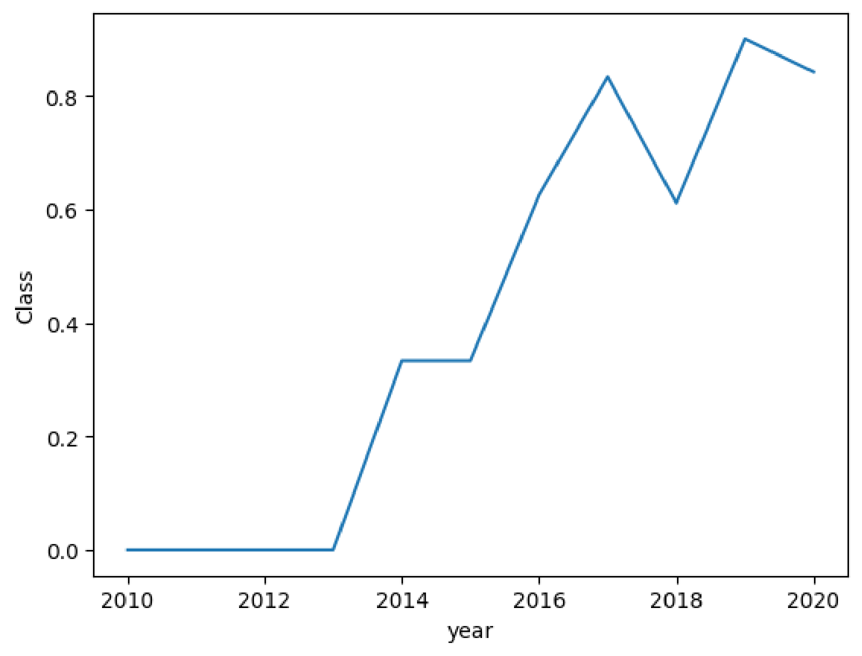

```
sns.catplot(data=df2,x='PayloadMass',y='Orbit', hue='Class', palette='husl')
```

Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- The success rate improved through the years. Highest point is 2019.

```
df2['year'] = pd.to_datetime(df2['Date']).dt.year
df3 = df2[['year','Class']]
df3 = df3.groupby('year').mean()
df3.reset_index(inplace=True)
sns.lineplot(data=df3, x='year', y='Class')
```



All Launch Site Names

- Find the names of the unique launch sites
  - df2['LaunchSite'].unique()
  - ['CCAFS SLC 40', 'VAFB SLC 4E', 'KSC LC 39A']
- Present your query result with a short explanation here
  - There are 3 launch sites

Launch Site Names Begin with 'KSC'

- Find 5 records where launch sites' names start with `KSC`
  - `df2[df2['LaunchSite'].str.startswith('KSC')].head(5)`
- Present your query result with a short explanation here
  - The first 5 flight numbers from a launch site that starts with KSC is listed below. I used str.starts with, then head 5 to select 5.

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount
26	27	2017-02-19	Falcon 9	2490.000000	ISS	KSC LC 39A	True RTLS	1	True	False	True	5e9e3032383ecb267a34e7c7	3.0	1
27	28	2017-03-16	Falcon 9	5600.000000	GTO	KSC LC 39A	None None	1	False	False	False	NaN	3.0	0
28	29	2017-03-30	Falcon 9	5300.000000	GTO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	2.0	1
29	30	2017-05-01	Falcon 9	6104.959412	LEO	KSC LC 39A	True RTLS	1	True	False	True	5e9e3032383ecb267a34e7c7	3.0	1
30	31	2017-05-15	Falcon 9	6070.000000	GTO	KSC LC 39A	None None	1	False	False	False	NaN	3.0	0

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here
    - Filtered the table, then summed the mass

```
URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'
df8 = pd.read_csv(URL)
df8.query("Customer.str.contains('NASA')")['Payload Mass (kg)'].sum()
```

✓ 0.4s

```
39157.0
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
  - `df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_dash.csv")`
  - `df[df['Booster Version'].str.startswith('F9 v1.1')][['Booster Version','Payload Mass (kg)']].groupby('Booster Version').mean()`
- Present your query result with a short explanation here
  - The average mass is per booster version listed attached. I continued the code (method chaining) since python is objected oriented. I also loaded a new df.

	Payload Mass (kg)
Booster Version	
F9 v1.1	2928.4
F9 v1.1 B1003	500.0
F9 v1.1 B1010	2216.0
F9 v1.1 B1011	4428.0
F9 v1.1 B1012	2395.0
F9 v1.1 B1013	570.0
F9 v1.1 B1014	4159.0
F9 v1.1 B1015	1898.0
F9 v1.1 B1016	4707.0
F9 v1.1 B1017	553.0
F9 v1.1 B1018	1952.0

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on drone ship.Present your query result with a short explanation here
  - Answer: 2016-04-08
  - I loaded a new spacex_df with the information needed. I then checked the columns and noticed that there are two versions of Success drone (one with more spaces). I then wrote df query that contains success and contains drone to get everything. Then I proceeded to sort by date and select the earliest date.
  - URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'
  - spacex_df=pd.read_csv(URL)
  - spacex_df['Landing_Outcome'] = spacex_df['Landing Outcome']
  - spacex_df['Landing_Outcome'].unique()
  - spacex_df.query("Landing_Outcome.str.contains('Success') and Landing_Outcome.str.contains('drone')").sort_values('Date').head(1)['Date']

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
  - 'F9 FT B1022', 'F9 FT B1026', 'F9 FT B1021.2', 'F9 FT B1031.2'
- Present your query result with a short explanation here
  - **Code notes. I first changed the column heads and replaced the spaces and parenthesis with _ . Then I filtered the data, sliced the booster, and did unique to get the list.**
  - `df1 = spacex_df.query("Landing_Outcome.str.contains('Success') and Landing_Outcome.str.contains('drone')")`
  - `df1.columns = df1.columns.str.replace(" ","_")`
  - `df1.columns = df1.columns.str.replace("(","")`
  - `df1.columns = df1.columns.str.replace(")","")`
  - `df1.query("4000 < Payload_Mass_kg < 6000")['Booster_Version'].unique()`

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

class	Landing Outcome	Total	%
0	32	56	57
1	24	56	42

- Present your query result with a short explanation here
  - Notes: I added two more columns to get the percentage of successful landing outcomes. There was a 42% success rate (57% failed).

```
URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'
spacex_df=pd.read_csv(URL)
spacex_df.groupby('class').count()['Landing Outcome']
col1 = spacex_df.groupby('class').count()['Landing Outcome']
col2 = spacex_df.groupby('class').count()['Landing Outcome'].sum()
df5 = pd.DataFrame(col1)
df5['Total'] = col2
df5['%'] = (df5['Landing Outcome'] / df5['Total'] * 100).astype(int)
df5
```

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
    - F9 FT B1029.1', 'F9 FT B1036.1', 'F9 B4 B1041.1', 'F9 FT B1036.2', 'F9 B4 B1041.2
- Present your query result with a short explanation here
    - Code Notes: df filtered by max payload
    - `spacex_df.sort_values('Payload Mass (kg)', ascending=False)`
    - `spacex_df[spacex_df['Payload Mass (kg)'] == spacex_df['Payload Mass (kg)'].max()]['Booster Version'].unique()`

	Flight Number	Date	Time (UTC)	Booster Version	Launch Site	Payload	Payload Mass (kg)	Orbit	Customer	Landing Outcome	class	Lat	Long
28	29	2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600.0	Polar LEO	Iridium Communications	Success (drone ship)	1	34.632834	-120.610745
29	37	2017-06-25	20:25:00	F9 FT B1036.1	VAFB SLC-4E	Iridium NEXT 2	9600.0	LEO	Iridium Communications	Success (drone ship)	1	34.632834	-120.610745
31	42	2017-10-09	12:37:00	F9 B4 B1041.1	VAFB SLC-4E	Iridium NEXT 3	9600.0	Polar LEO	Iridium Communications	Success (drone ship)	1	34.632834	-120.610745
32	46	2017-12-23	1:27:00	F9 FT B1036.2	VAFB SLC-4E	Iridium NEXT 4	9600.0	Polar LEO	Iridium Communications	Controlled (ocean)	0	34.632834	-120.610745
34	51	2018-03-30	14:14:00	F9 B4 B1041.2	VAFB SLC-4E	Iridium NEXT 5	9600.0	Polar LEO	Iridium Communications	No attempt	0	34.632834	-120.610745

2015 Launch Records (selected 2017)

- List the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

Date	Landing Outcome	Booster Version	Launch Site
December	Success (ground pad)	F9 FT B1019	CCAFS LC-40

- Present your query result with a short explanation here
  - First created a list of the months in 2017. Then used this list to filter the table.

```
list_of_month_in_2017 = spacex_df[spacex_df['Date'].dt.year == 2017]['Date'].dt.month.unique()
```

```
df6 = spacex_df[
    (spacex_df['Date'].dt.month.isin(list_of_month_in_2017)) &
    (spacex_df['Date'].dt.year == 2015) &
    (spacex_df['Landing Outcome'] == "Success  (ground pad)")
]

df7 = df6[['Date','Landing Outcome','Booster Version','Launch Site']]
df7['Date'] = df7['Date'].dt.month_name()
df7
```

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

- Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order
- Present your query result with a short explanation here
    - Filtered the df based on the parameters, and then sorted the df

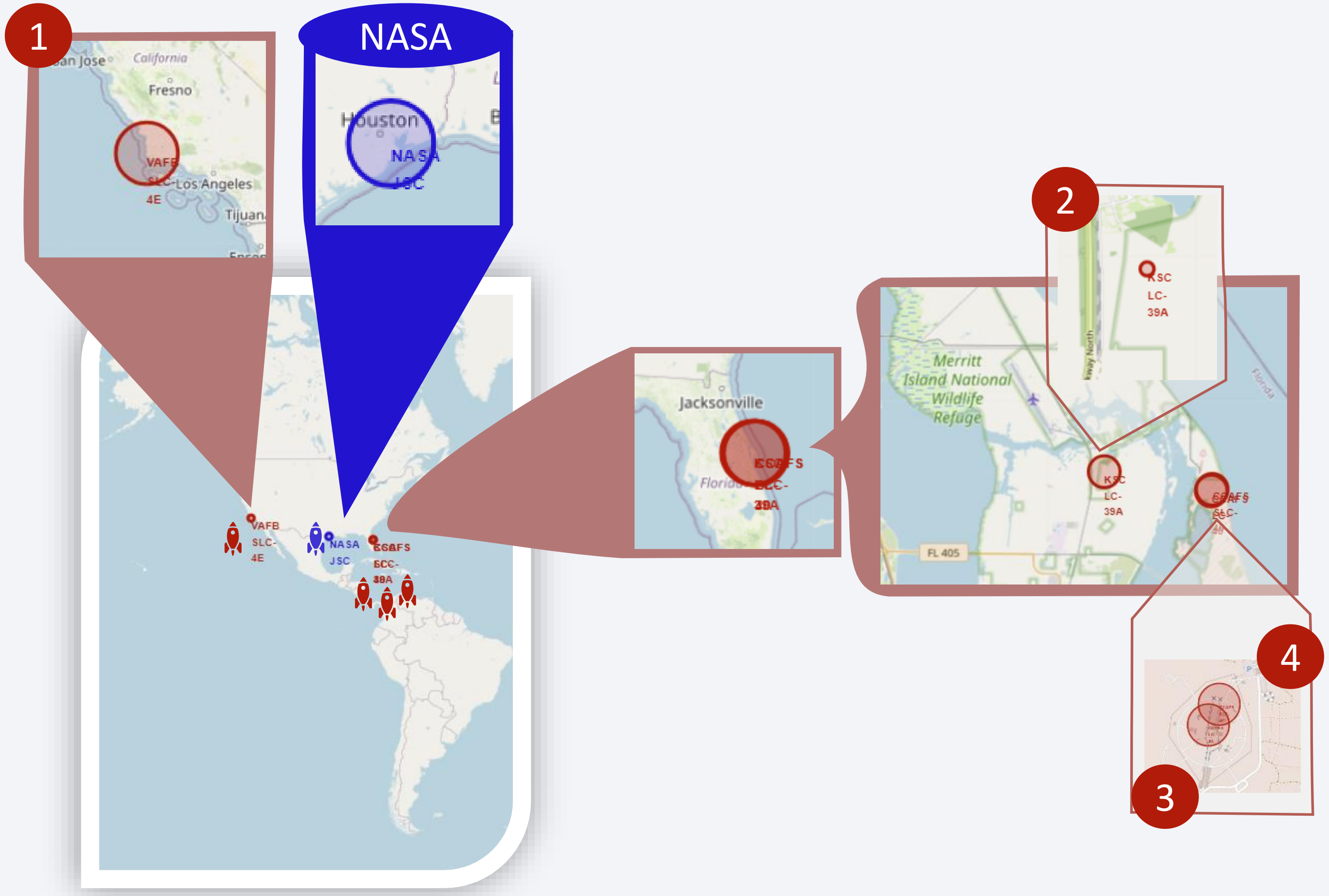
```
URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv'
spacex_df1=pd.read_csv(URL)
spacex_df1.columns = spacex_df1.columns.str.replace(" ","_")
✓ 0.5s

outcomes = spacex_df1.query("(Landing_Outcome.str.contains('Success ') & ('2010-06-04' < Date < '2017-03-20'))")
```

```
Landing_Outcome
Success  (drone ship)    3
Success (drone ship)     2
Success (ground pad)     2
Success  (ground pad)    1
Name: Date, dtype: int64
```

Section 3

Launch Sites Proximities Analysis

Location of Launch Sites



1 VAFB SLC-4E
2 KSC LC-39A
3 CCAFS SLC-40
4 CCAFS LC-40
NASA JSC

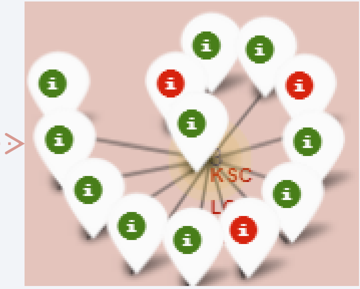
Successful Launches by Launch Site

KSC LC-39A has the highest launch success rate at

77%

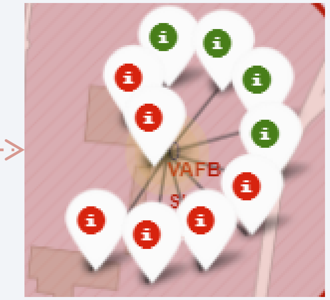
KSC LC-39A

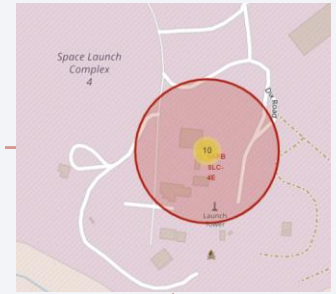
77% success rate
(10 green / 13 total)



VAFB SLC-4E

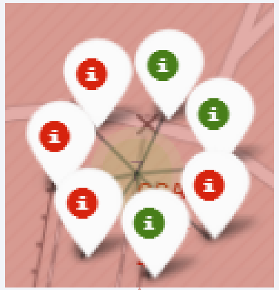
40% success rate
(4 green / 10 total)





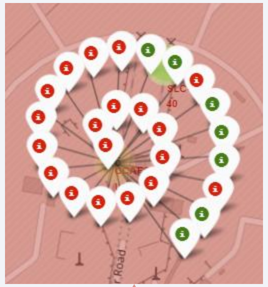
CCAFS SLC-40

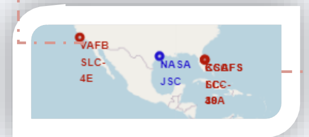
43% success rate
(3 green / 7 total)



CCAFS LC-40

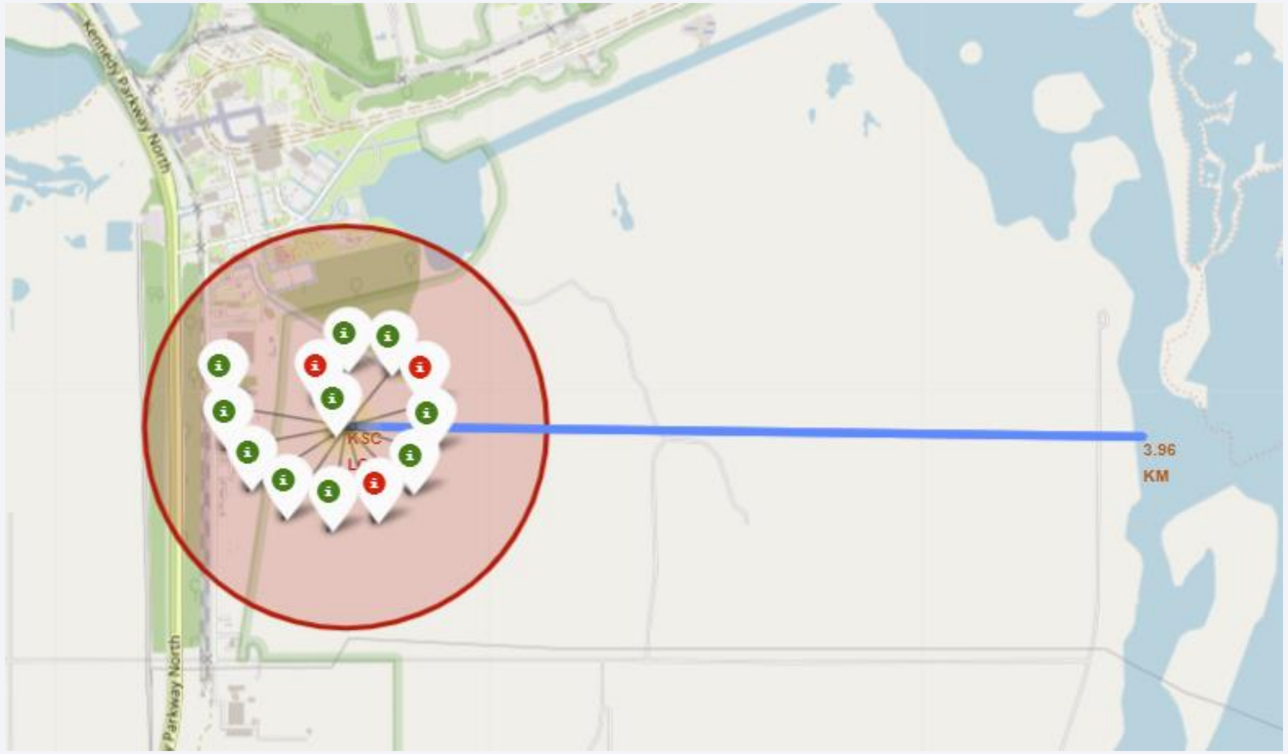
27% success rate
(3 green / 7 total)





Nearest coastline

The eastern coastline is 3,96km away from the launch site KSC LC-39A.

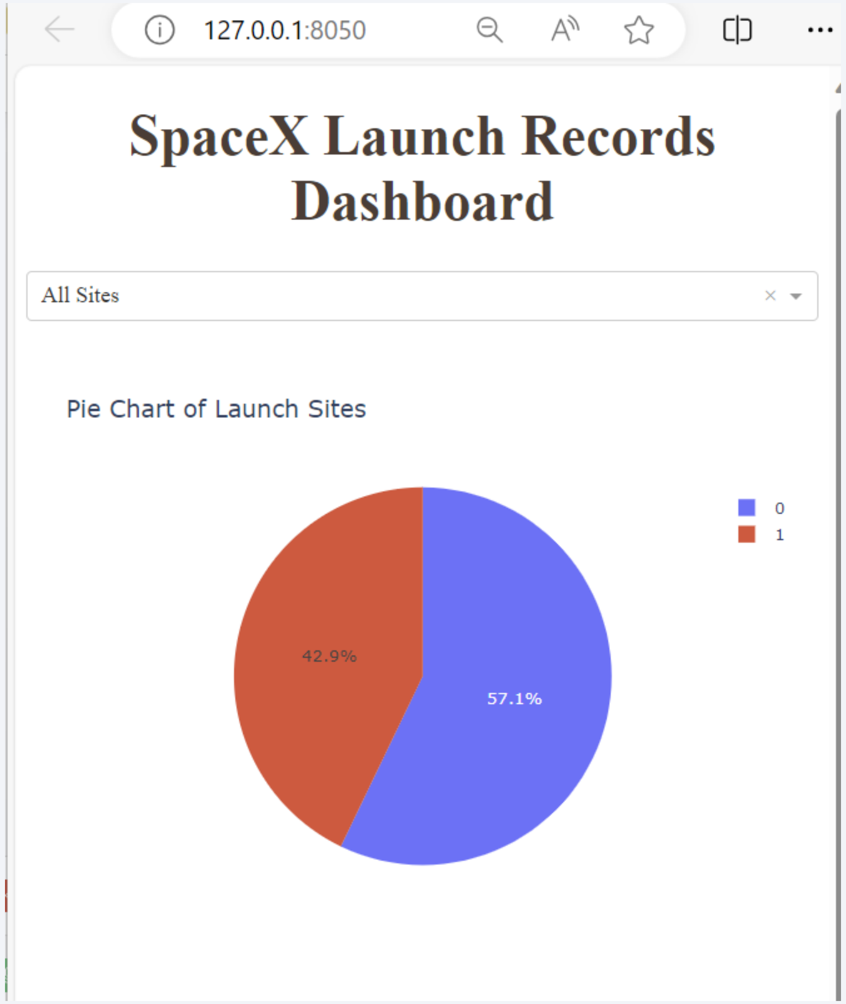


Section 4

Build a Dashboard with Plotly Dash

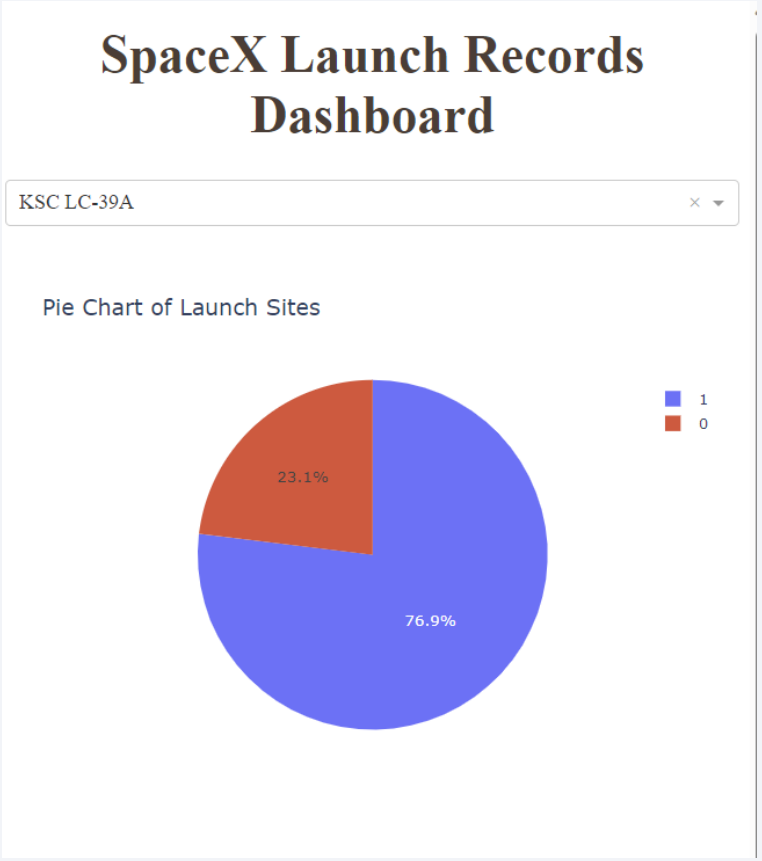
Total Successful Launches

43% of the launches were successful as marked by the color red (legend = 1)



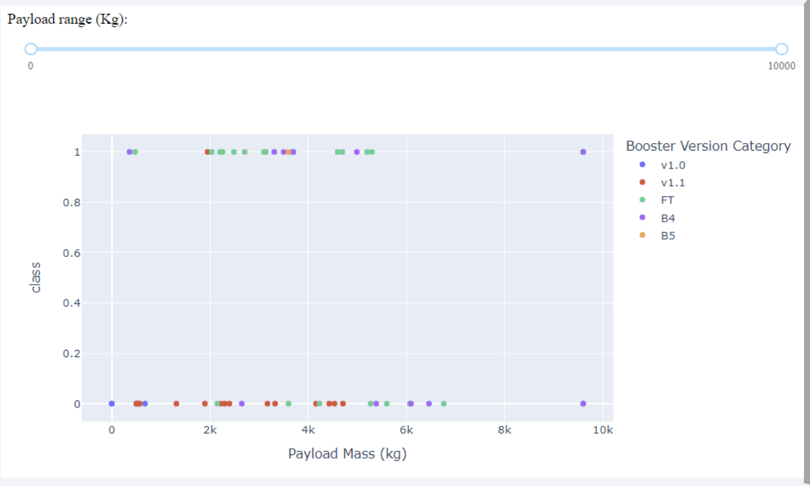
KSC LC-39A has the highest launch success

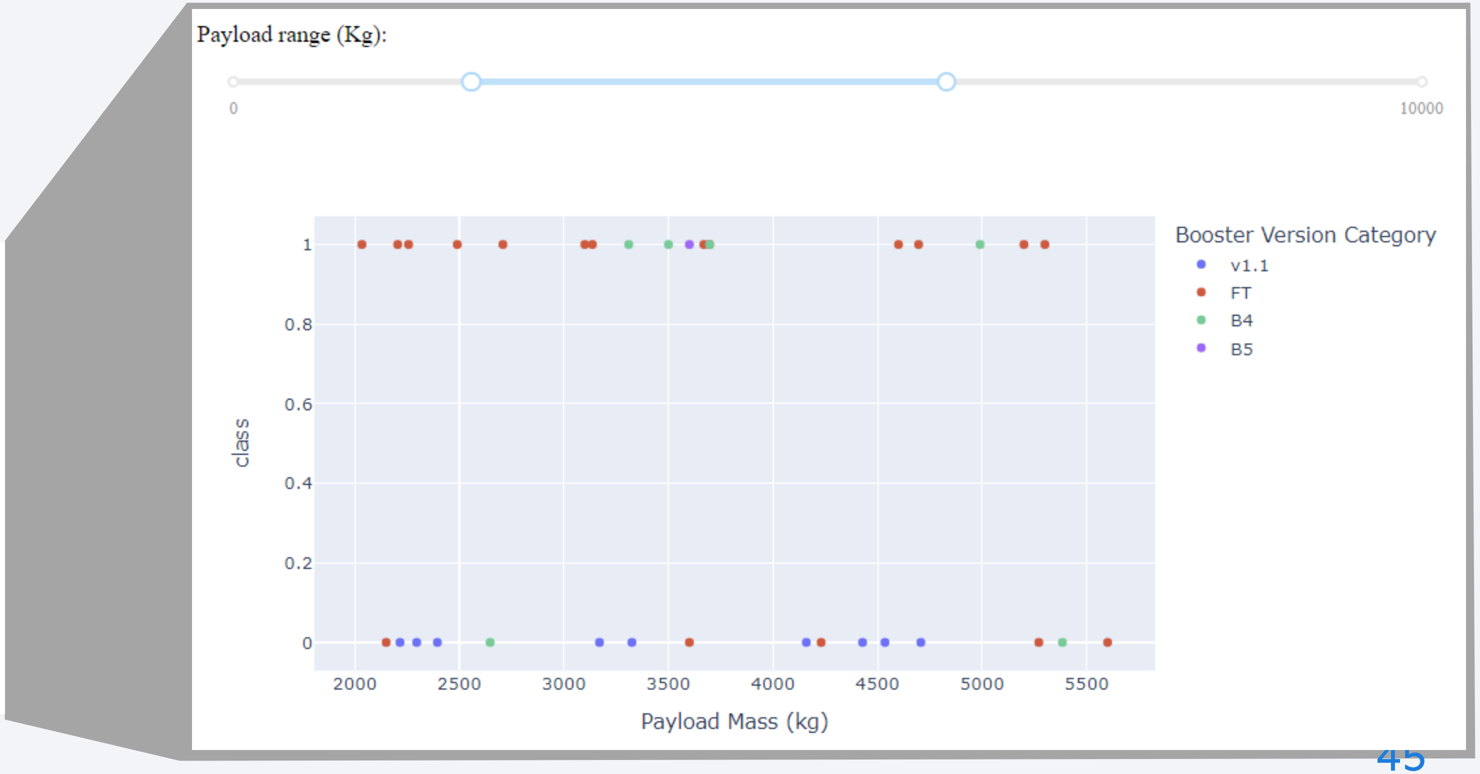
77% of the launches in KSC LC-39A were successful as marked by the color blue (legend = 1)



Success ratio filtered by payload range

Zooming into the payload range of 2000 – 6000, we see several successful (class 1) launches with a booster of booster FT





Section 5

Predictive Analysis (Classification)

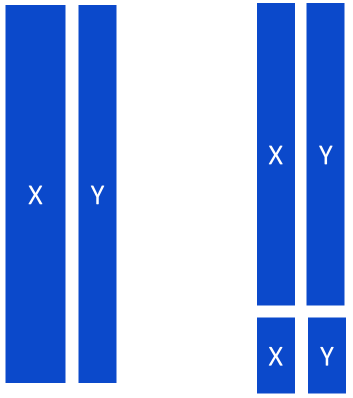
Classification Process

Process Flow

Break into training data (to train the model) and testing (to check the accuracy of the model)

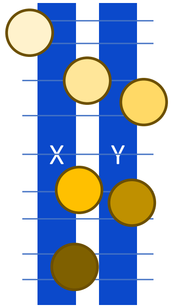
Break data into X independent & Y dependent data

Get 80% for the train data for classification modeling

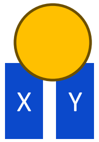


Model the data based on each classification

Split into 10 then fit the model to the training set, recommend best parameter combo

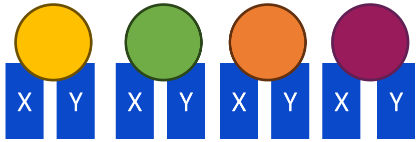


Apply the best model to the test data set and get the score

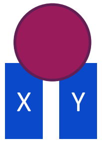


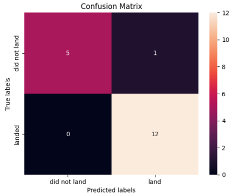
Analyze

Compare classification scores and confusion matrices



Analyze what are
- TP true positives
- TN true negatives
- FP false positives
- FN false negatives

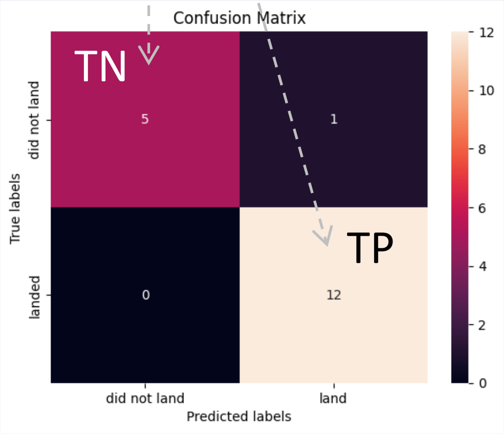




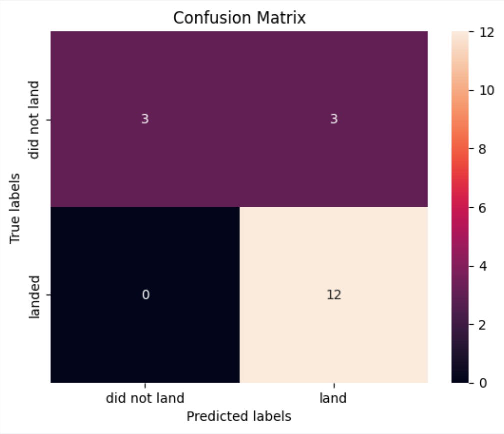
Confusion Matrix

- Decision Tree (DT) Classifier is the best at 94%
- TP True positive: DT can positively predict if will land
- TN True negative: DT can positively predict if will not land

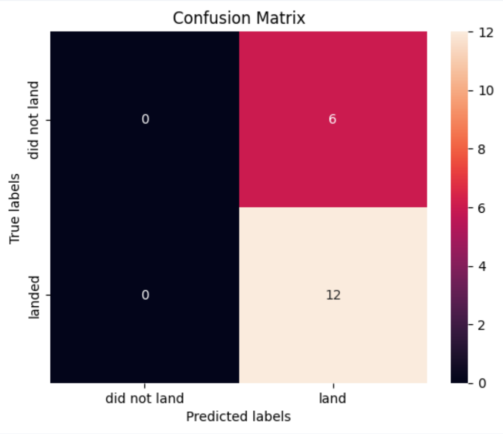
Decision Tree Classifier: 94%



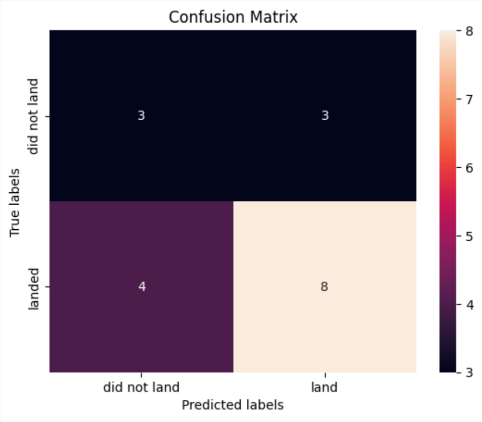
Logistics Regression: 83%



Support vector machine: 67%



K Nearest Neighbors: 61%



Conclusion

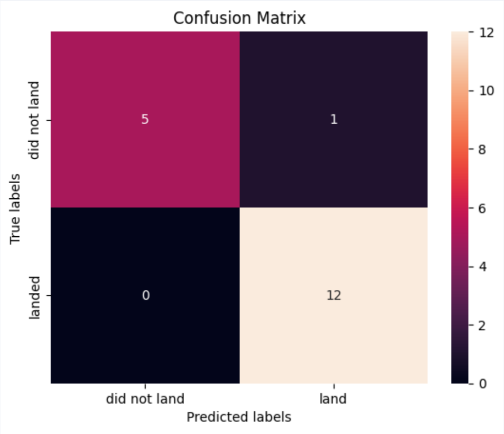
Rember what are we trying to solve:

What is the most successful launch site. What are the interesting factors that could show the highest instances of a successful launch?

Answers below:

Decision Tree

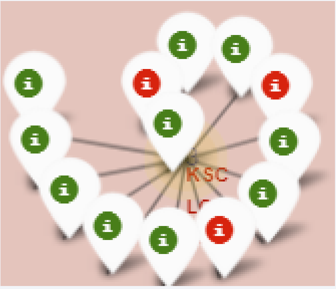
is the best classifier @ 94%



KSC LC-39A

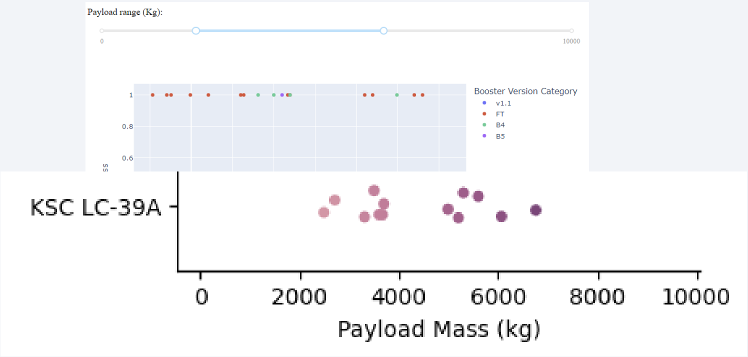
77% success rate

(10 green / 13 total)



2000-6000 mass

has the most instances of successful launches



Thank you!