Applied Data Science Capstone

<Phyo Hein>

<07/23/2022>

OUTLINE

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

EXECUTIVE SUMMARY

- Learning topics
  - Data collection with API
  - Data collection with web scraping
  - Data wrangling
  - Exploratory data analysis with SQL
  - Exploratory data analysis with visualization
  - Interactive visual analytics and dashboards
  - Predictive analysis
- Results
  - Exploratory data analysis results
  - Interactive analytic screenshots
  - Predictive analysis results

INTRODUCTION

- Project

  Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- To determine:
  - If the rocket will land successfully
  - Interactions of various features
  - Operating conditions needed to ensure success rate of landings

METHODOLOGY

- Data collection methods through API and web scraping
- Data wrangling
- Exploratory data analysis (EDA) using SQL and visualization
- Interactive visual analytics using Folium and Plotly Dash
- Predictive analysis using classifier algorithms

Data Collection

- Data collection using get request to the SpaceX API.
- Decoded the response as a json and convert into pandas data frame using .json_normalize().
- Cleaned the data, checked for missing values and imputed using methods such as mean().
- Performed web scraping from Wikipedia for Falcon 9 lauch records with Beautiful Soup.
- To extract launch records as HTML table, parse it, and convert into a pandas data frame for analysis.

Data Collection – SpaceX API

- SpaceX API was used to collect data.
- The data was cleaned, wrangled, and formatted.
- The link to the complete notebook is:

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/Data%20Collection%20API%20Lab.ipynb>

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

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

Check the content of the response

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

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/3c/0e/T8iJcSN3_o.png","large":"https://images2.imgbox.com/40/e3/GypSkayF_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-launc

```
# 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

```
# Get the head of the dataframe
data.head(5)
```

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bl
								Successful first stage				

Data Collection – Web Scraping

- Web scraping was used to collect data.
- The data was cleaned, wrangled, and formatted.
- The link to the complete notebook is:

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/Data%20Collection%20with%20Web%20Scraping%20Lab.ipynb>

```
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 Wiki page from its URL

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

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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')
```

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

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

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

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

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

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

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

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

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

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
```

Data Wrangling

- The following exploratory data analyses and wrangling were performed:
  - Number of launches on each site.
  - Number and occurrence of each orbit.
  - Number and occurrence of mission outcome per orbit type.
  - A new landing outcome label from outcome column.
- The link to the complete notebook is:

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/Data%20Wrangling%20Lab.ipynb>

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-	Falcon 9	525.000000	LEO	CCAFS	None	1	False	False	False	NaN	1.0	0	B0005

```
df.isnull().sum()
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
```

```
df.isnull().sum()/df.count()*100
```

```
FlightNumber      0.000
Date              0.000
BoosterVersion    0.000
PayloadMass       0.000
Orbit             0.000
LaunchSite        0.000
```

```
df.dtypes
```

```
FlightNumber       int64
Date              object
BoosterVersion    object
PayloadMass      float64
Orbit             object
```

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

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

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
GTO     27
ISS     21
VLEO    14
PO       9
LEO      7
SSO      5
```

EDA with SQL

- SpaceX dataset is loaded to IBM db2 for EDA using SQL. The following were explored:
  - Names of unique launch sites.
  - Total payload mass carried by NASA (CRS) boosters.
  - Average payload mass carried by F9 V1.1 booster version.
  - First successful date of landing outcome in ground pad.
  - Names of boosters that have success in drone ship with 4000kg<payload<6000kg.
  - Total number of successful and failure mission outcomes.
  - Names of booster versions that carried the max payload mass, etc.
- The link to the complete notebook is:

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/EDA%20with%20SQL%20Lab.ipynb>

Task 1

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

```
%sql select Unique(LAUNCH_SITE) from SPACEXTBL;
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

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

Task 2

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

```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Task 3

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

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

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

totalpayloadmass
45596

Task 4

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

```
%sql select AVG(PAYLOAD_MASS__KG_) as averagepayloadmass from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'
```

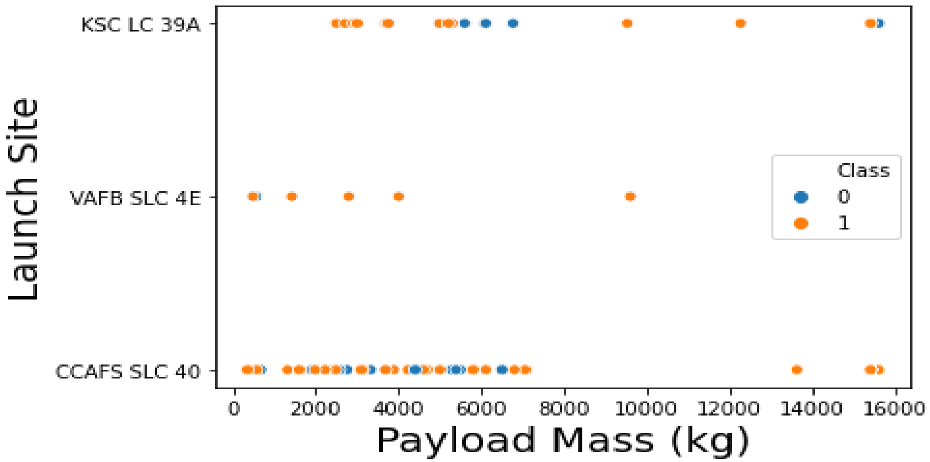
```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

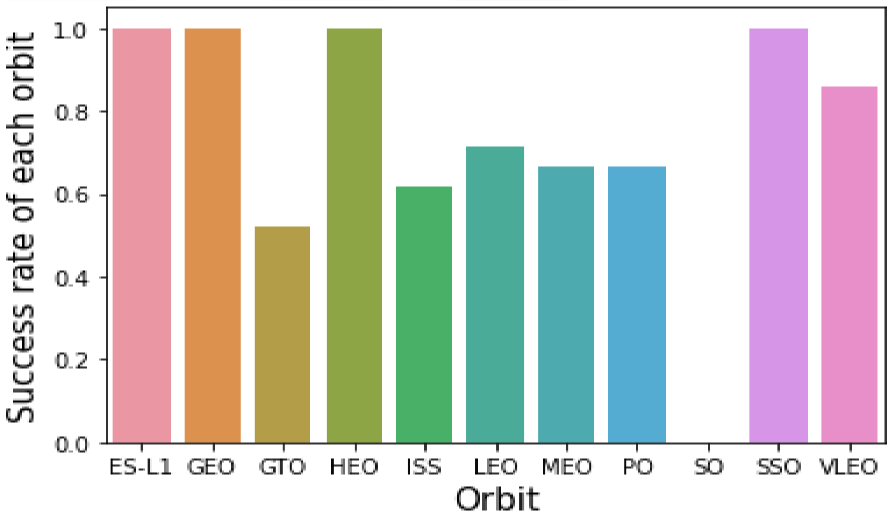
averagepayloadmass
2928

EDA with Visualization

- The following were visualized for EDA:
  - Relationship between flight number and launch site
  - Relationship between payload and launch site,
  - Success rate of each orbit type,
  - Relationship between flight number and orbit type,
  - Relationship between payload and orbit type,
  - Yearly trend of launch success.
- Feature engineering was done by:
  - Creating dummy variables to categorical columns
  - Changing numeric columns to float64
- The link to the complete notebook is:

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/EDA%20with%20Visualization%20Lab.ipynb>





Interactive Visual Analytics: Folium

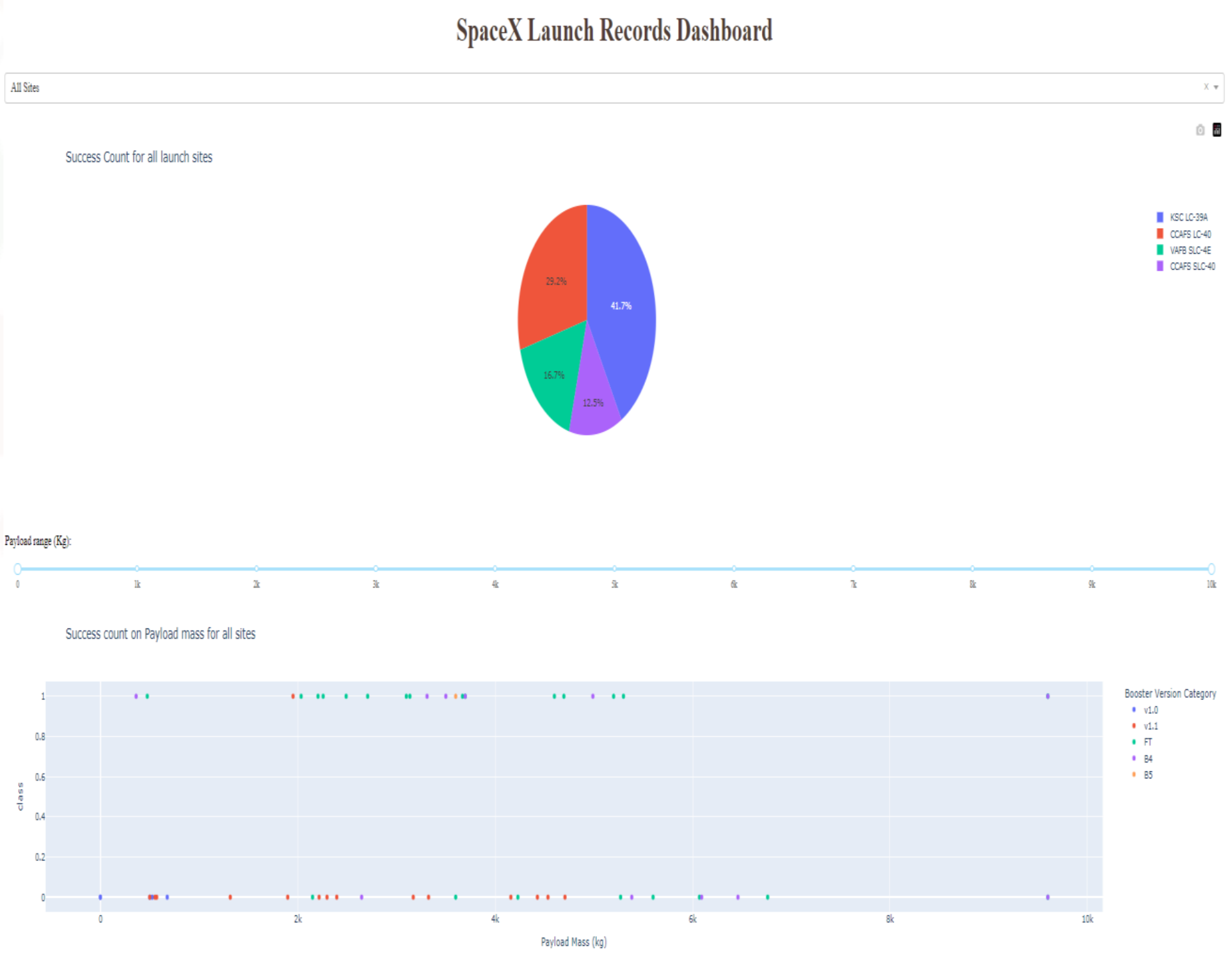
- All launch sites were marked and added with the following map objects for each site on the Folium map.
  - Markers
  - Circles
  - Lines
- The launch outcomes (failure/success) were assigned class 0/1.
- Through colored marker clusters, launch sites and success rates are identified.
- The distances from a launch site to its proximities (railways, highways, coastlines, cities) were calculated,
- The link to the complete notebook is:

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/Interactive%20Visual%20Analytics%20with%20Folium%20Lab.ipynb>

Interactive Visual Analytics: Plotly Dash

- An interactive dashboard with Plotly Dash was built.
- A pie chart was plotted showing the total launches by sites.
- A scatter graph was plotted showing the relationship between outcome and payload mass in kg for different boosters.
- The link to the file is:

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/SpaceX_PlotlyDash_App/spacex_dash_app.py>



Predictive Analysis: Classification

- Different machine learning models were built using multiple classification algorithms.
- The models were tuned for a set of respective hyperparameters using GridSearchCV.
- Accuracy metric was used for model performance measure.
- The best performing classification model was identified.
- The link to the complete note book is:

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb>

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
grid_search = GridSearchCV(svm, parameters, cv=10)
svm_cv = grid_search.fit(X_train, Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8472222222222222
```

TASK 7

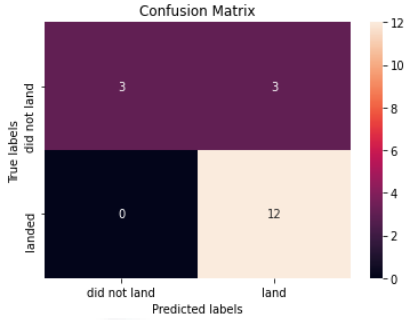
Calculate the accuracy on the test data using the method `score` :

```
svm_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



RESULTS & DISCUSSION

- Exploratory data analyses results.
- Interactive analytics screenshots
- Predictive analysis results

EDA Results: with SQL

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/EDA%20with%20SQL%20Lab.ipynb>

- The following slides show the screenshots of exploratory data analysis using SQL.

Task 1 & 2

Task 1

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

```
%sql select Unique(LAUNCH_SITE) from SPACEXTBL;
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

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

- Names of unique launch sites

Task 2

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

```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

- Launch sites starting with 'CCA'

Task 3 & 4

Task 3

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

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

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

totalpayloadmass
45596

- Total payload mass by NASA (CRS)

Task 4

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

```
%sql select AVG(PAYLOAD_MASS__KG_) as averagepayloadmass from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

averagepayloadmass
2928

- Average payload mass by F9 V1.1

Task 5 & 6

Task 5

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

Hint:Use min function

```
%sql select MIN(DATE) from SPACEXTBL where LANDING__OUTCOME = 'Success (ground pad)'
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

1
2016-04-08

- Date of the first successful landing outcome in ground pad

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING__OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYL
OAD_MASS__KG_ < 6000
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

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

- Names of the boosters succeeded in drone shop with 4000 kg < payload < 6000kg

Task 7 & 8

Task 7

List the total number of successful and failure mission outcomes

```
%sql select count(case when MISSION_OUTCOME like '%Success%' then 1 end) as successcounts, count(case when MISSION_OUTCOME like
'%Failure%' then 1 end) as failurecounts from SPACEXTBL
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

successcounts	failurecounts
100	1

- Total number of successful and failure outcomes

Task 8

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

```
%sql select distinct BOOSTER_VERSION, max(PAYLOAD_MASS__KG_) as boosterwithmaxpayloadmass from SPACEXTBL group by BOOSTER_VERSIO
N order by boosterwithmaxpayloadmass desc
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

booster_version	boosterwithmaxpayloadmass
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600
F9 B5 B1049.6	15440
F9 B5 B1059.3	15410
F9 B5 B1051.5	14932
F9 B5 B1049.3	13620

- Names of the booster versions that carried max payload mass

Task 9 & 10

Task 9

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

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

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

booster_version	launch_site	landing__outcome
F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

- Failed landing outcomes in drone ship with their boosters and launch sites

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

```
%sql select LANDING__OUTCOME, count(*) as count from SPACEXTBL where (DATE between '2010-06-04' and '2017-03-20') group by LANDI
NG__OUTCOME order by count desc
```

```
 * ibm_db_sa://jqh80870:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

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

- Rank of the landing outcomes between 2010-06-04 and 2017-03-20

EDA Results: with Visualization

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/EDA%20with%20Visualization%20Lab.ipynb>

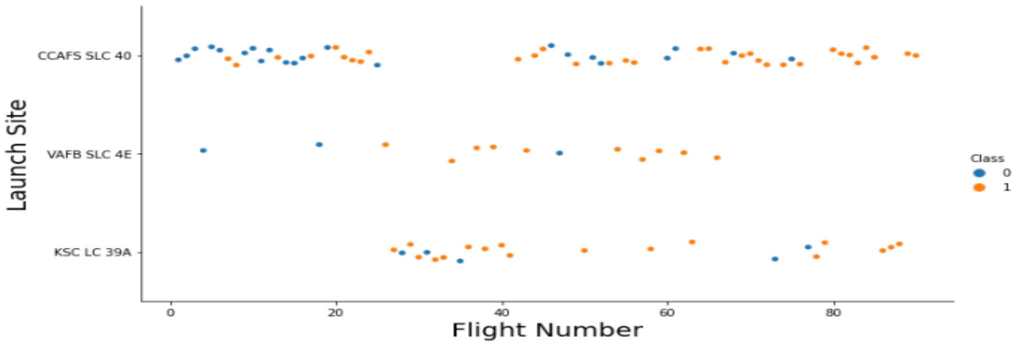
- The following slides show the screenshots of exploratory data analysis using visualizations.

Task 1 & 2

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y='LaunchSite',x='FlightNumber',hue='Class',data=df, aspect=2)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

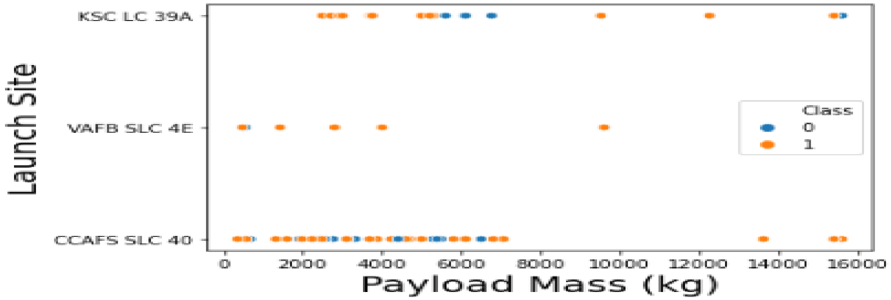


- Flight number & launch site

TASK 2: Visualize the relationship between Payload and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.scatterplot(x='PayloadMass',y='LaunchSite', hue='Class', data=df)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```



- Payload & launch site

Task 3

TASK 3: Visualize the relationship between success rate of each orbit type

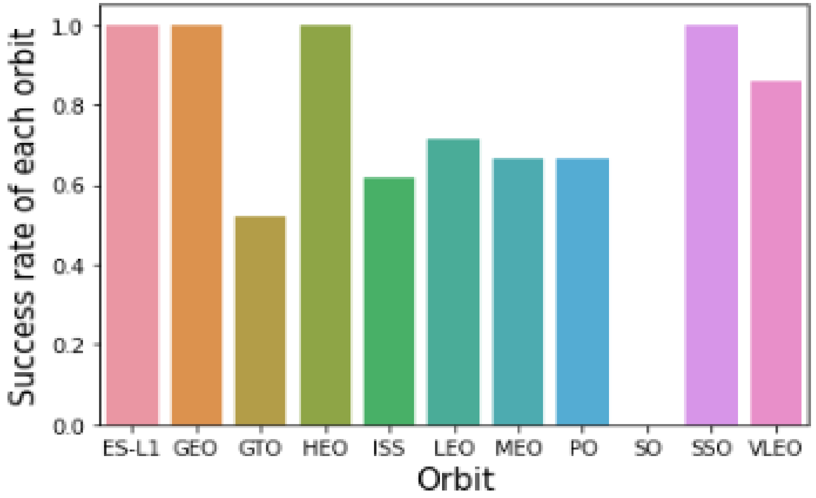
Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_success = df.groupby('Orbit')['Class'].mean().reset_index()
# orbit_success.plot(kind='bar',ylabel="Success rate of each orbit", xlabel="Orbit")
orbit_success
```

	Orbit	Class
0	ES-L1	1.000000
1	GEO	1.000000
2	GTO	0.518519
3	HEO	1.000000
4	ISS	0.619048
5	LEO	0.714286
6	MEO	0.666667
7	PO	0.666667
8	SO	0.000000
9	SSO	1.000000
10	VLEO	0.857143

```
sns.barplot(x="Orbit",y="Class", data=orbit_success)
plt.xlabel("Orbit",fontsize=15)
plt.ylabel("Success rate of each orbit", fontsize=15)
plt.show()
```



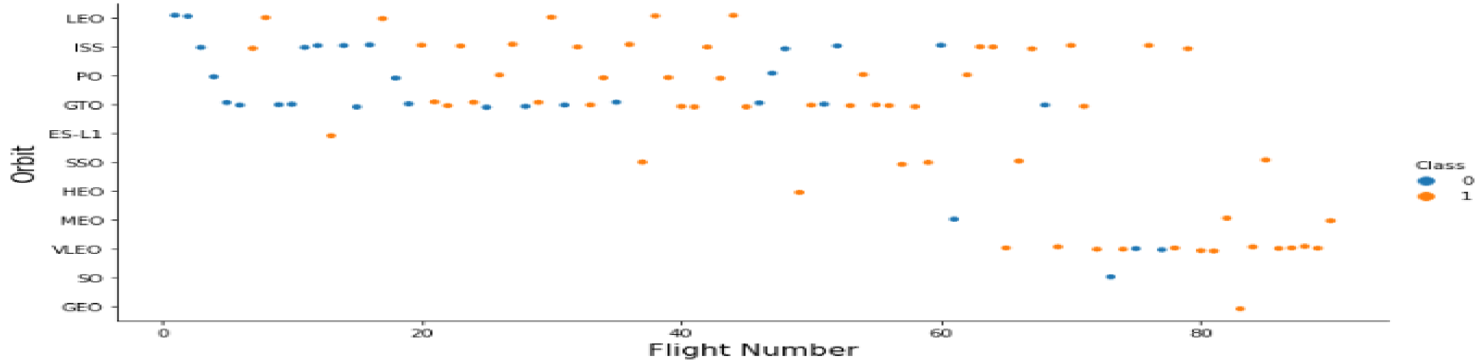
- Success rate of each orbit type

Task 4 & 5

TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="FlightNumber",y="Orbit",hue='Class' ,data=df,aspect=2)
plt.xlabel("Flight Number", fontsize=15)
plt.ylabel("Orbit", fontsize=15)
plt.show()
```



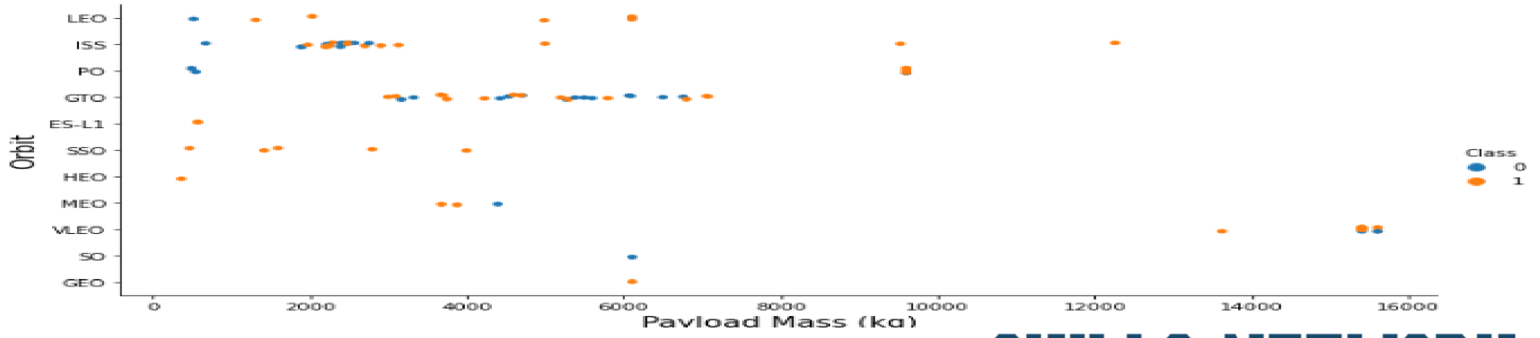
- Flight number & orbit type

TASK 5: Visualize the relationship between Payload and Orbit type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="PayloadMass",y="Orbit",hue='Class' ,data=df,aspect=2)
plt.xlabel("Payload Mass (kg)", fontsize=15)
plt.ylabel("Orbit", fontsize=15)
plt.show()
```

- Payload & orbit type



Task 6

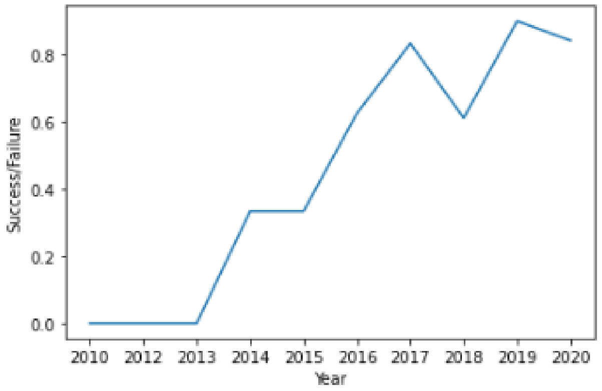
TASK 6: Visualize the launch success yearly trend

You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

```
# A function to Extract years from the date
year=[]
def Extract_year(date):
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year(1)
df["Year"]=year
average_by_year = df.groupby(by="Year").mean().reset_index()
```

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
plt.plot(average_by_year["Year"],average_by_year["Class"])
plt.xlabel("Year")
plt.ylabel("Success/Failure")
plt.show()
```



- Yearly trend of launch success

Task 7

Features Engineering

By now, you should obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block', 'ReusedCount', 'Serial']]
features.head()
```

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0007
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

TASK 7: Create dummy variables to categorical columns

Use the function `get_dummies` and `features` dataframe to apply OneHotEncoder to the column `Orbits`, `LaunchSite`, `LandingPad`, and `Serial`. Assign the value to the variable `features_one_hot`, display the results using the method head. Your result dataframe must include all features including the encoded ones.

```
# HINT: Use get_dummies() function on the categorical columns
features_one_hot=pd.get_dummies(features, columns=['Orbit','LaunchSite', 'LandingPad', 'Serial'])
features_one_hot.head()
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_B1058	Serial_B1059	S
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	

5 rows × 80 columns

- Feature engineering: create dummy variables to categorical columns

Task 8

TASK 8: Cast all numeric columns to `float64`

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

```
# HINT: use astype function
features_one_hot = features_one_hot.astype('float64')
features_one_hot.head()
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_B1058	Serial_B1059	S
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0	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	0.0	0.0	0.0	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	0.0	0.0	0.0	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 80 columns

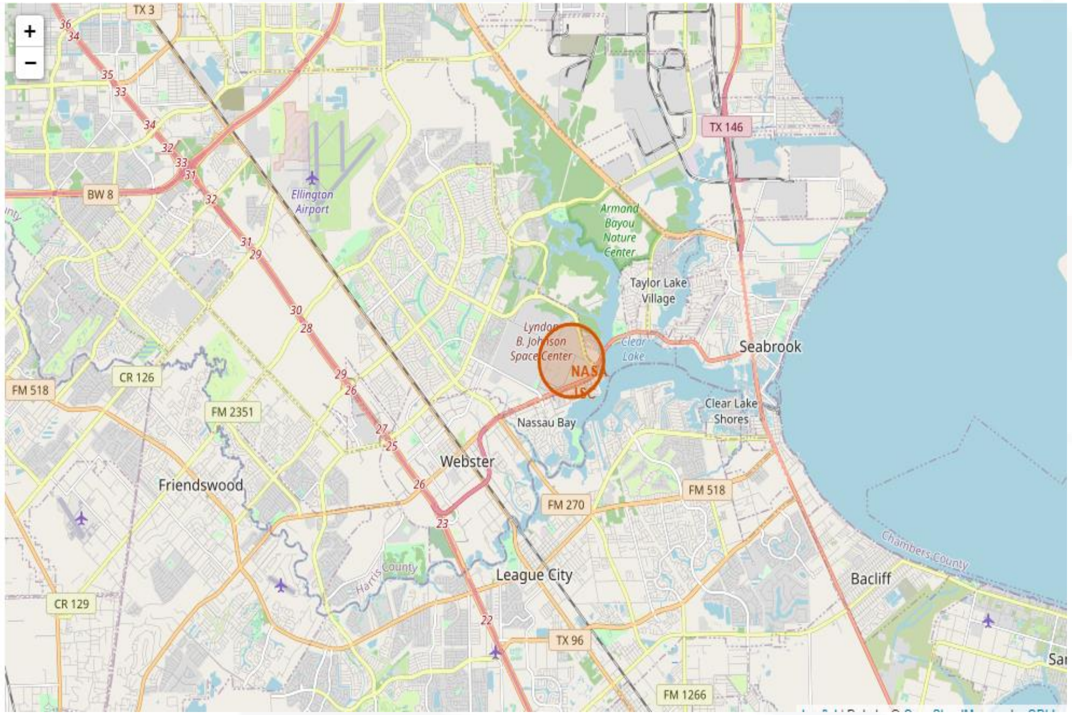
- Feature engineering: cast numeric columns to float64

Interactive Visual Analytics: Folium

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/Interactive%20Visual%20Analytics%20with%20Folium%20Lab.ipynb>

- The following slides show the screenshots of visual analytics using Folium.

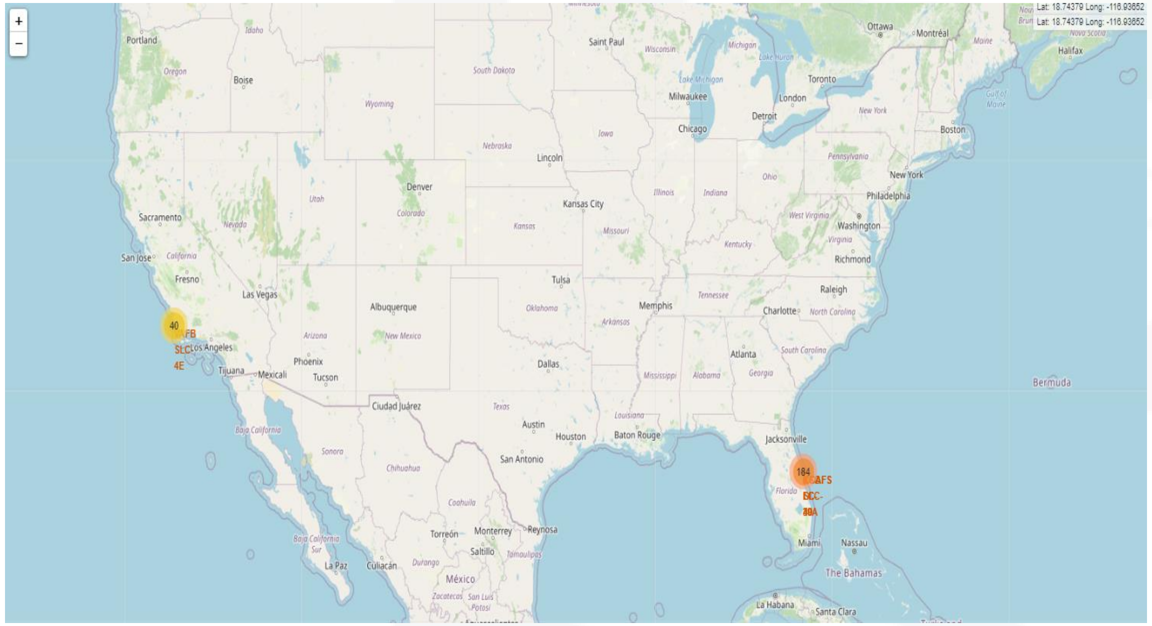
Launch Sites On a Map

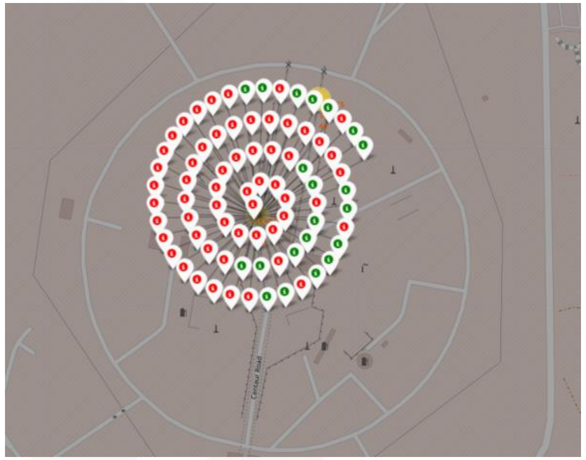


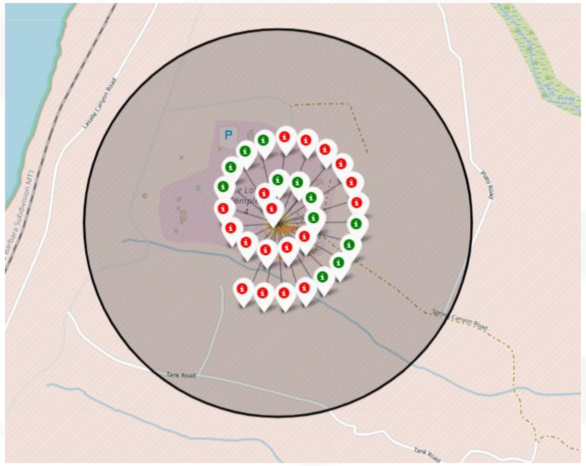
IBM Developer

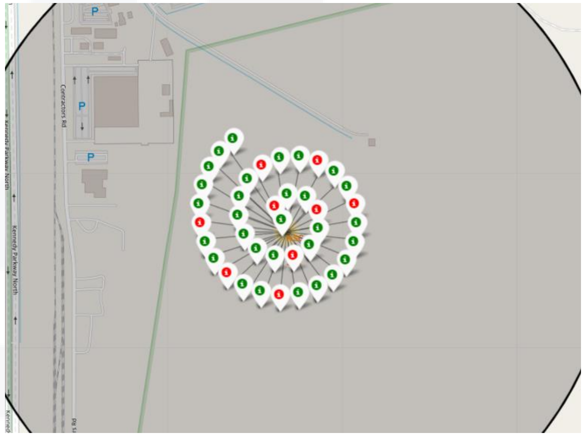
SKILLS NETWORK

Success/Failed Launches For Each Site





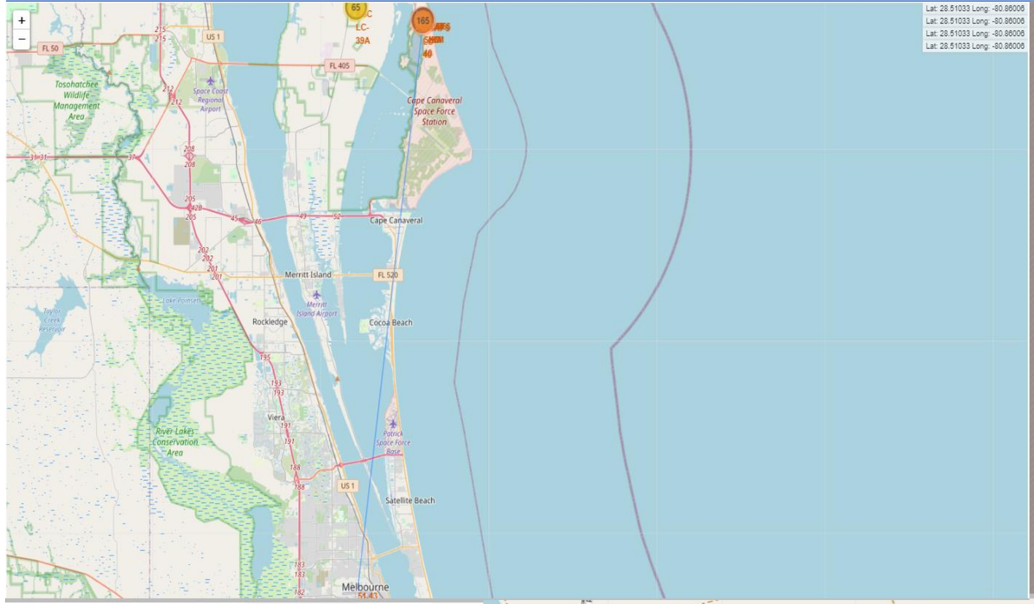


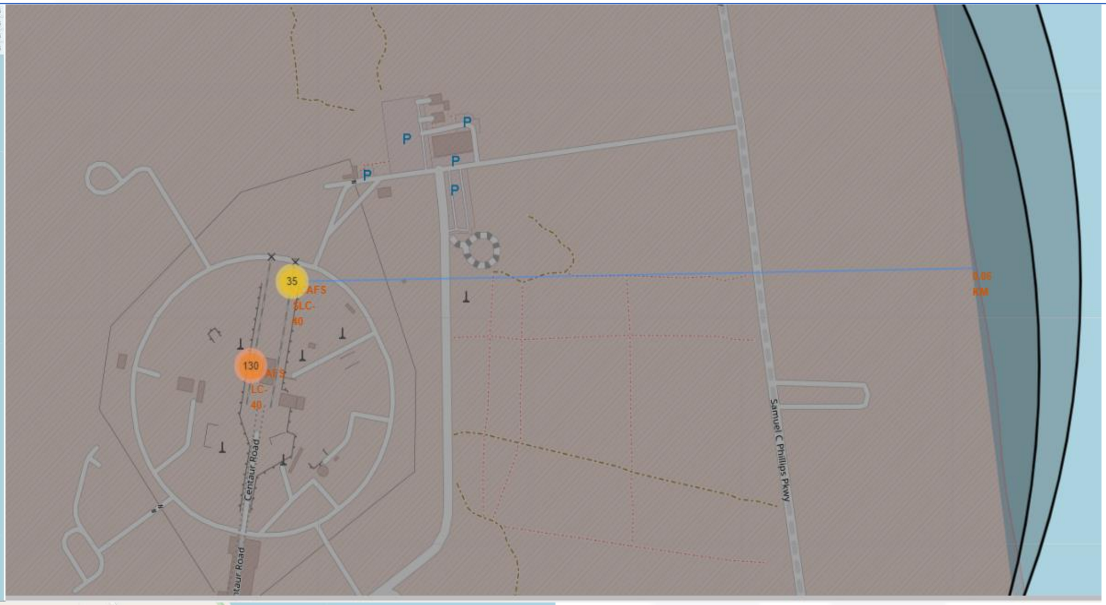


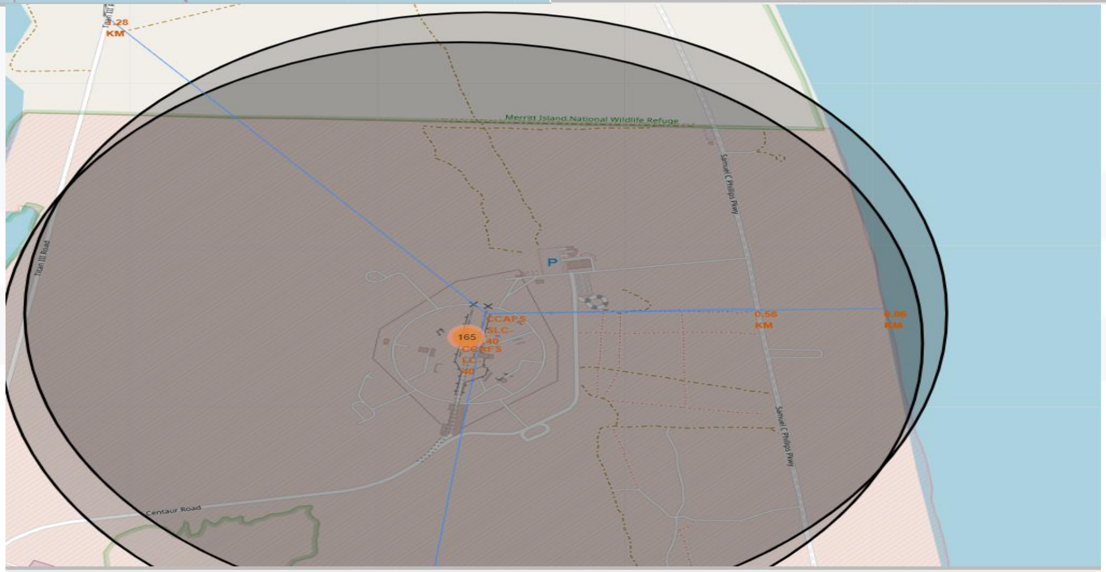
IBM Developer

SKILLS NETWORK

Distances Between a Launch Site to its Proximities





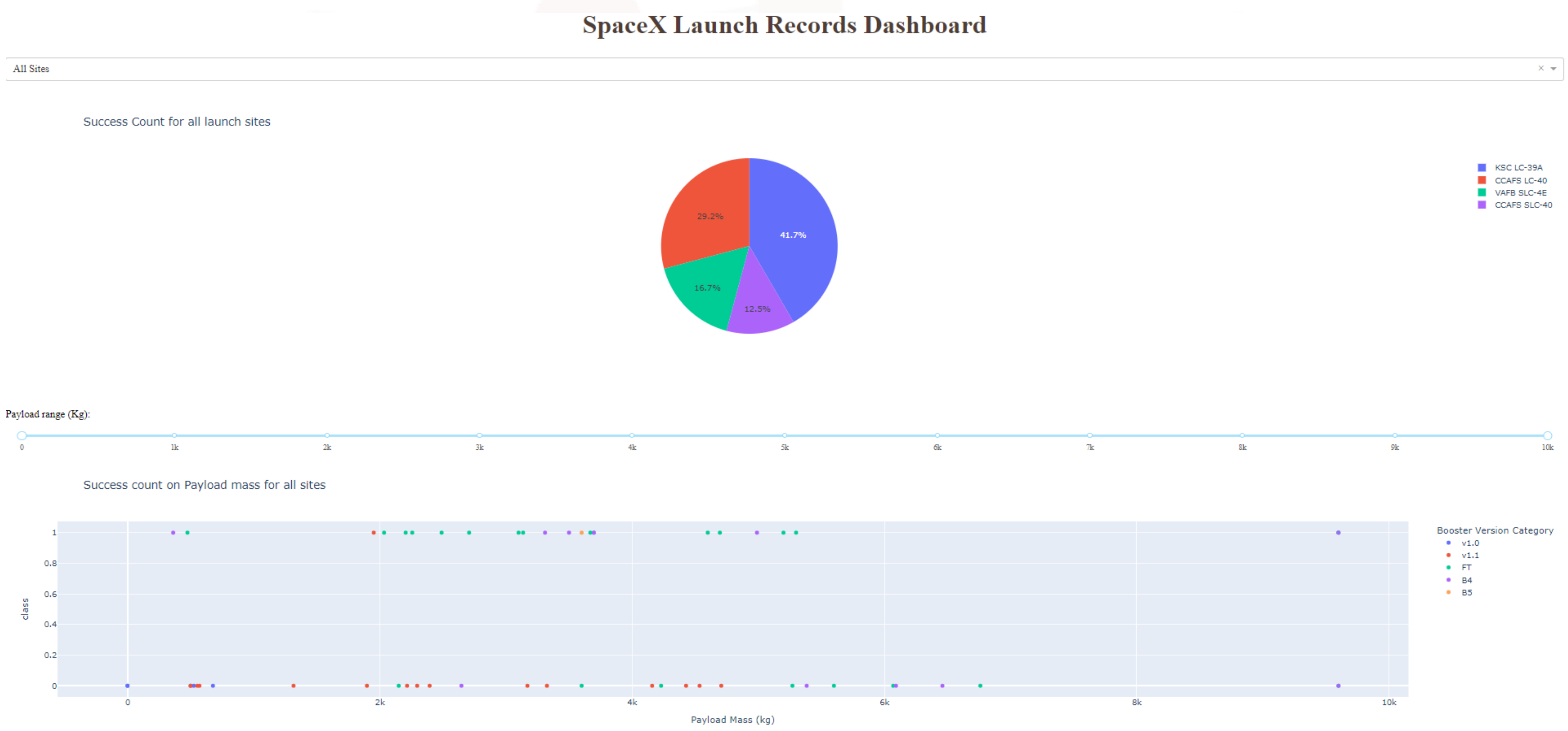


Interactive Visual Analytics: Plotly Dashboard

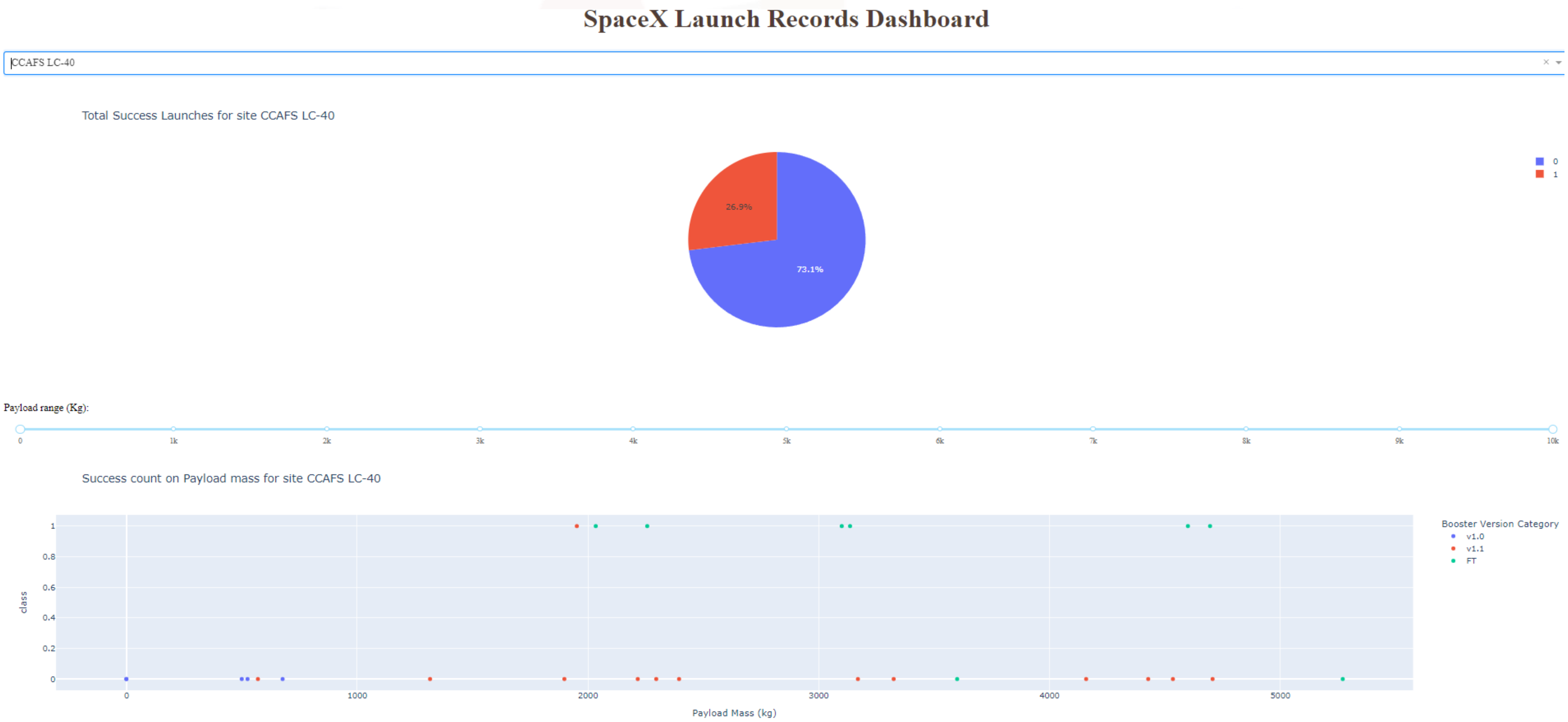
<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/SpaceX_PlotlyDash_App/spacex_dash_app.py>

- The following slides show the screenshots of SpaceX Launch Records Dashboard using Plotly Dash.

DASHBOARD: all sites



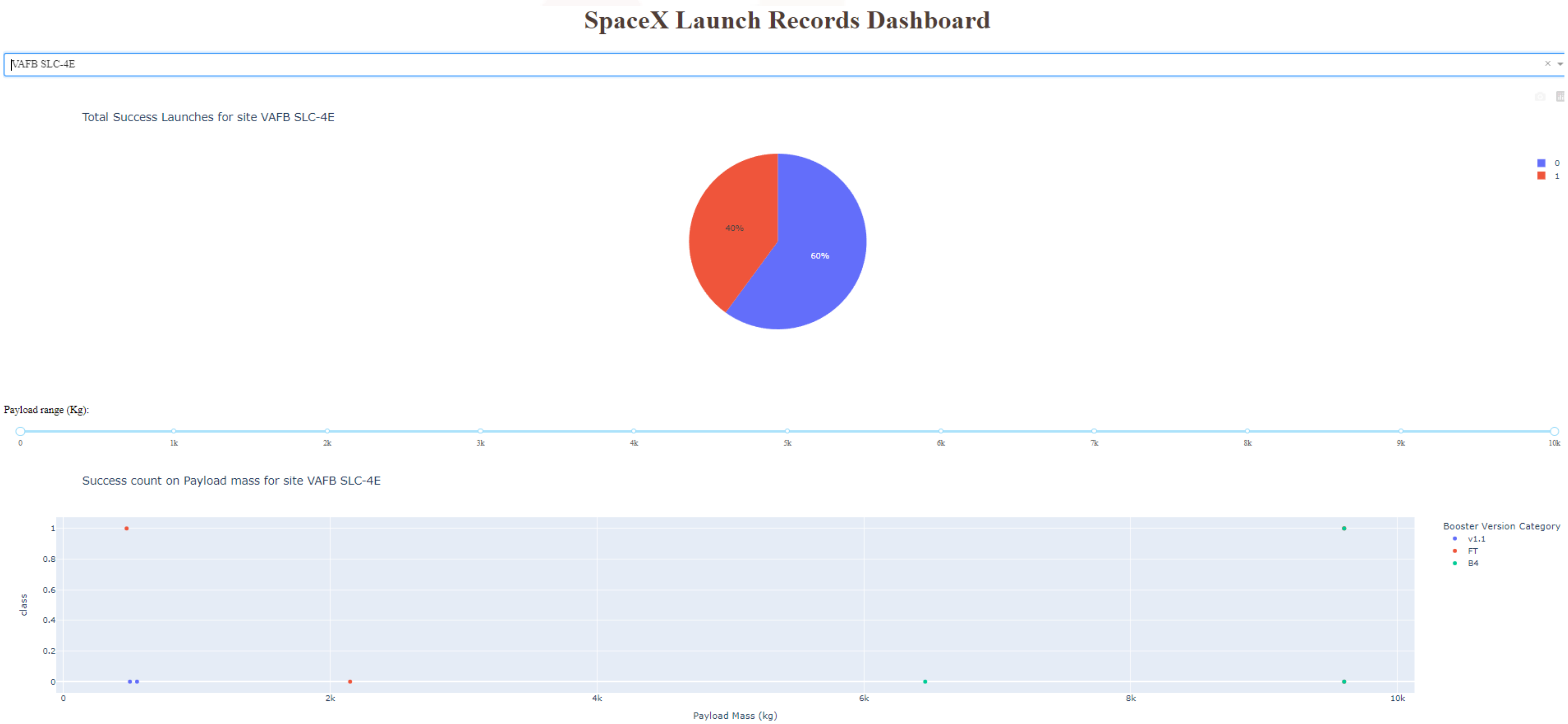
DASHBOARD: CCAFS LC-40



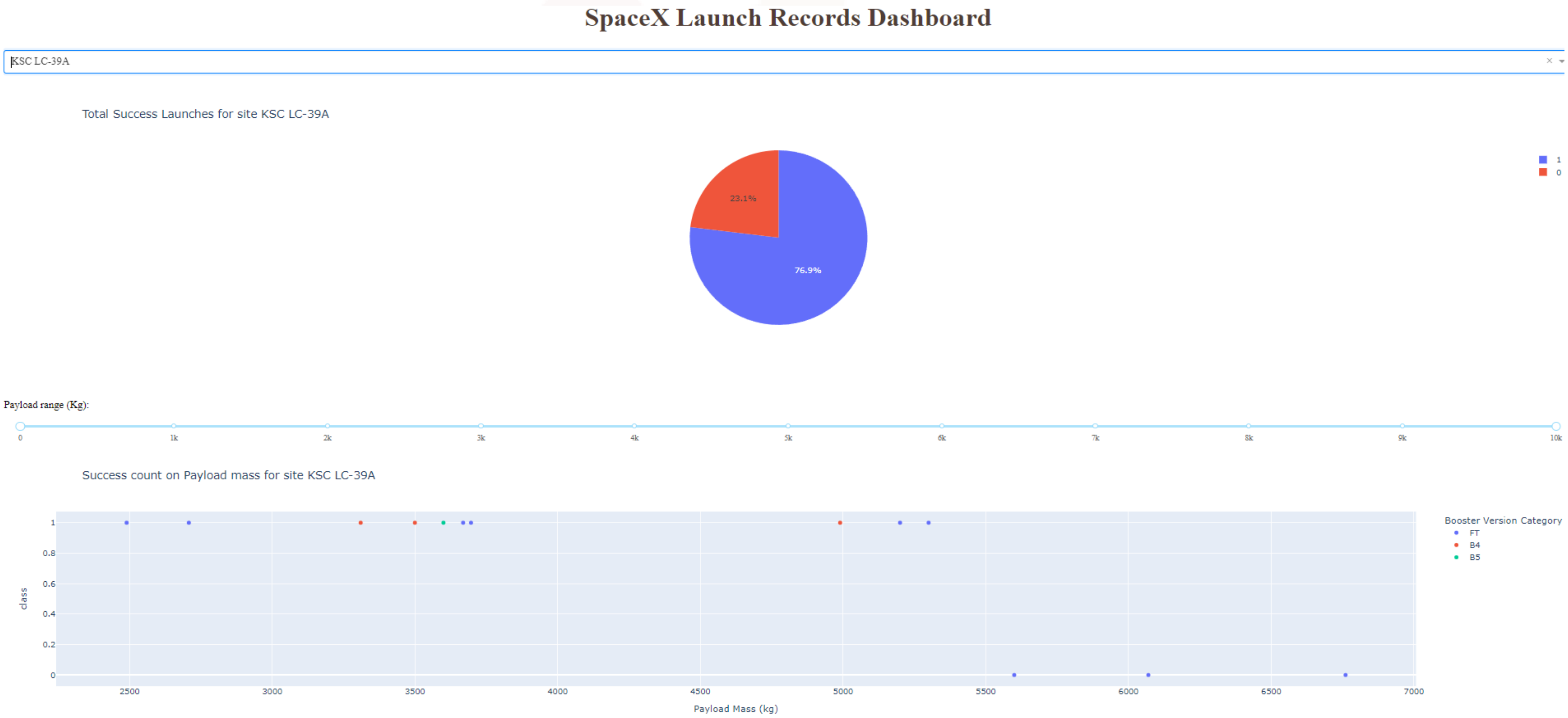
IBM Developer

SKILLS NETWORK

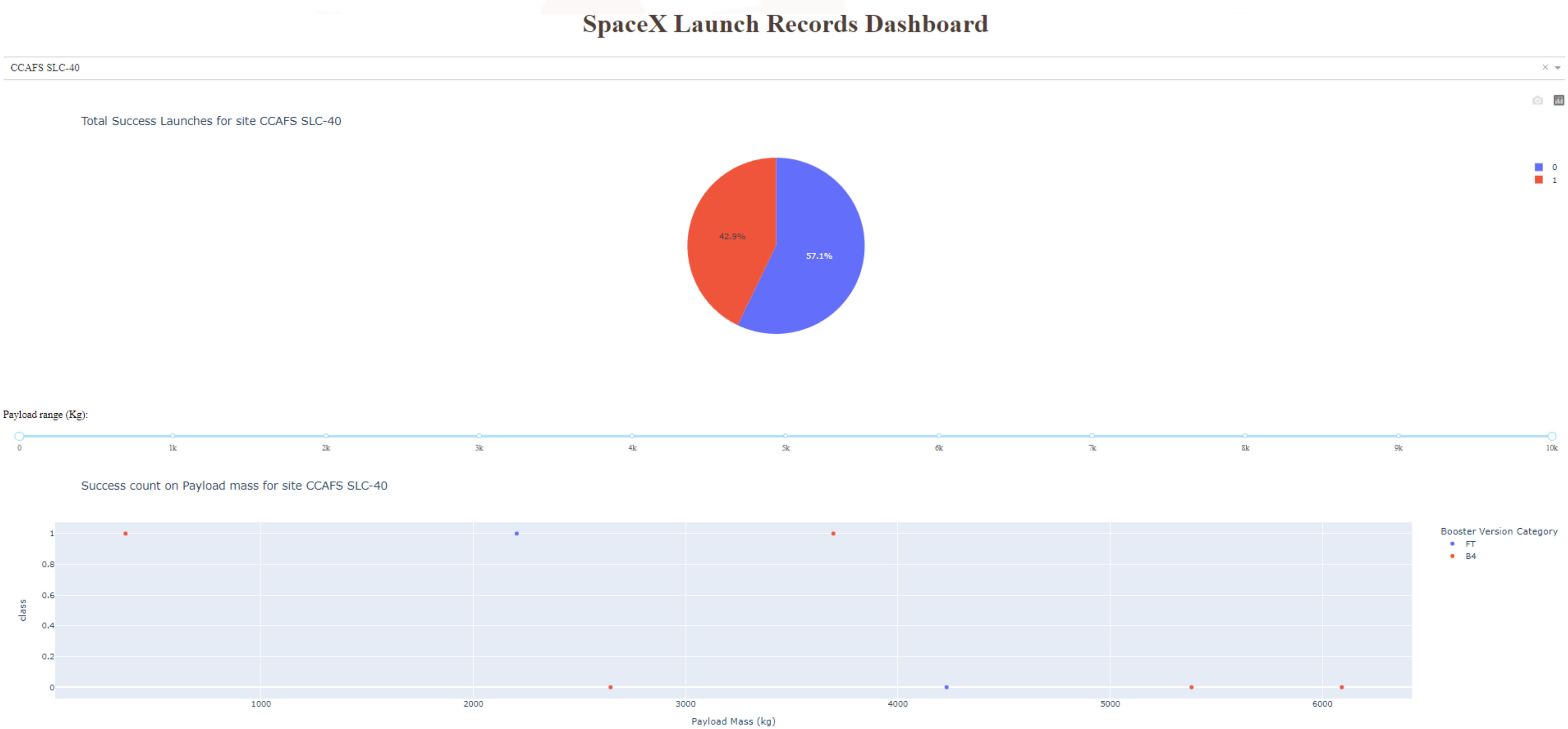
DASHBOARD: VAFB SLC-4E



DASHBOARD: KSC LC-39A



DASHBOARD: CCAFS SLC-40



Predictive Analysis: Classification

<https://github.com/phyohhein/Applied-Data-Science-IBM/blob/4379dc6cb1e4efd8c8376629dd55eb6ddd63166f/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb>

- The following slides show the screenshots of predictive analysis using classification algorithms.

LR & SVM

```
parameters_lr ={"C":[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}# l1 lasso l2 ridge

lr = LogisticRegression(random_state = 12345)
```

```
grid_search_lr = GridSearchCV(lr, parameters_lr, 'accuracy', cv = 10
)
logreg_cv = grid_search_lr.fit(X_train, Y_train)
```

We output the `GridSearchCV` object for logistic regression. We display the best parameter: `best_score_`.

```
print("tuned hyperparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :", logreg_cv.best_score_)
```

```
tuned hyperparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8472222222222222
```

- Logistic Regression

```
parameters_svm = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC(random_state = 12345)
```

```
grid_search_svm = GridSearchCV(svm, parameters_svm, 'accuracy', cv = 10)
svm_cv = grid_search_svm.fit(X_train,Y_train)
```

```
print("tuned hyperparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hyperparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8472222222222222
```

- Support Vector Machine

Tree & KNN

```
parameters_tree = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier(random_state = 12345)
```

```
grid_search_tree = GridSearchCV(tree, parameters_tree, 'accuracy', cv = 10)
tree_cv = grid_search_tree.fit(X_train, Y_train)
```

```
print("tuned hyperparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hyperparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
accuracy : 0.875
```

- Decision Tree

```
parameters_knn = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

knn = KNeighborsClassifier()
```

```
grid_search_knn = GridSearchCV(knn, parameters_knn, 'accuracy', cv = 10)
knn_cv = grid_search_knn.fit(X_train, Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 9, 'p': 1}
accuracy : 0.8472222222222222
```

- K Nearest Neighbors

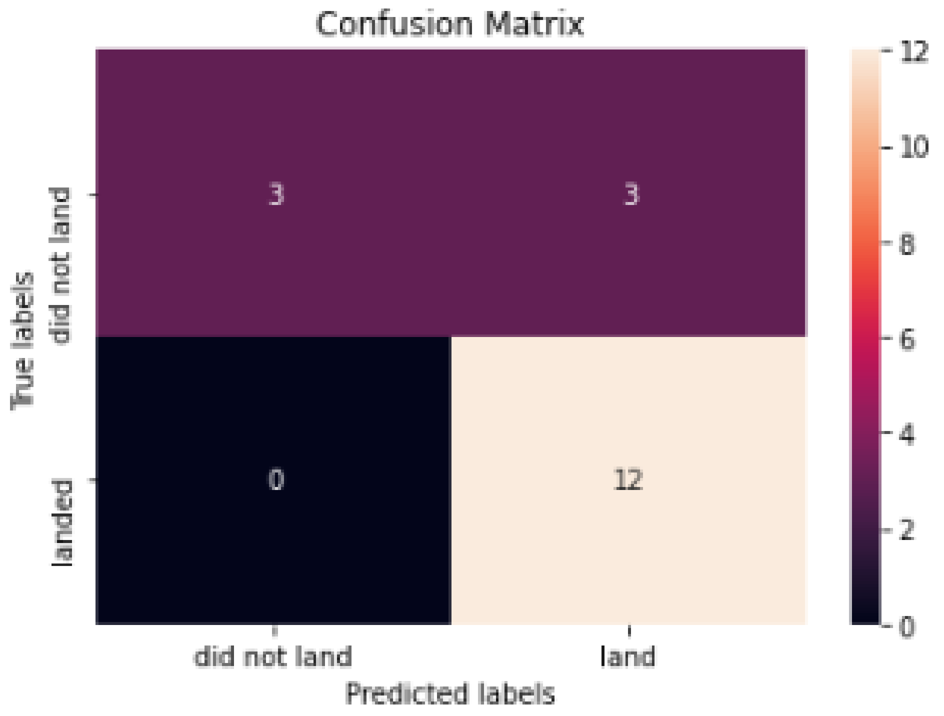
Best Model

```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
bestalgorithm
```

```
'DecisionTree'
```

- Decision Tree



CONCLUSION

The following conclusion can be drawn:

- Launch success rate started to increase in 2013.
- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The decision tree model classifier is the best algorithm for predicting landing outcomes.

APPENDIX

The repository link:

<https://github.com/phyohhein/Applied-Data-Science-IBM.git>