IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Juan Camilo Pérez Torres
12/08/2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction

- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

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.

- Problems you want to find answers

  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

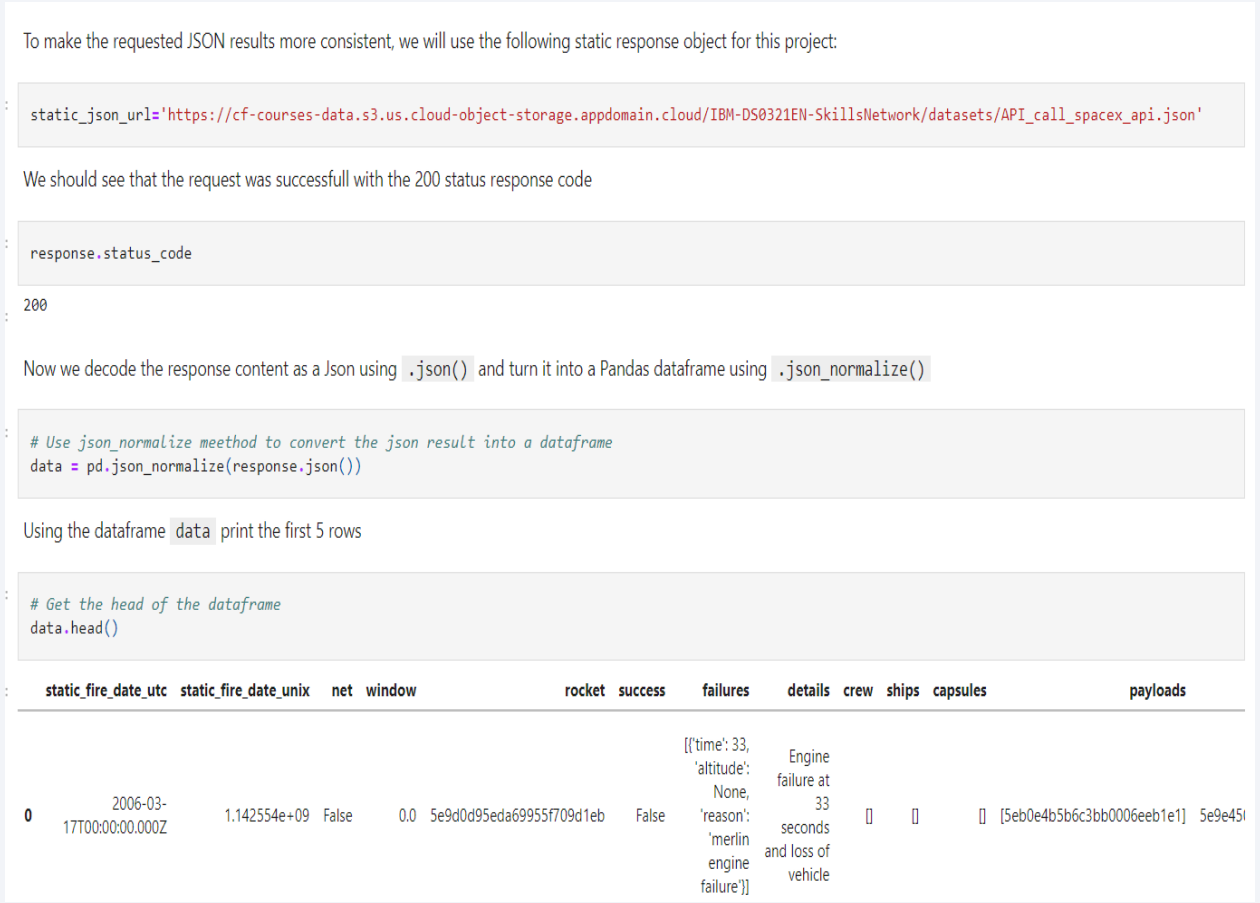
Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.

- The link to the notebook is https://github.com/Jperez-lab/testrepo/blob/072e24b36282a29ff26d8c2ab22cf8504df297e8/1_jupyter-labs-spacex-data-collection-api.ipynb



To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
response.status_code
```

```
200
```

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

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

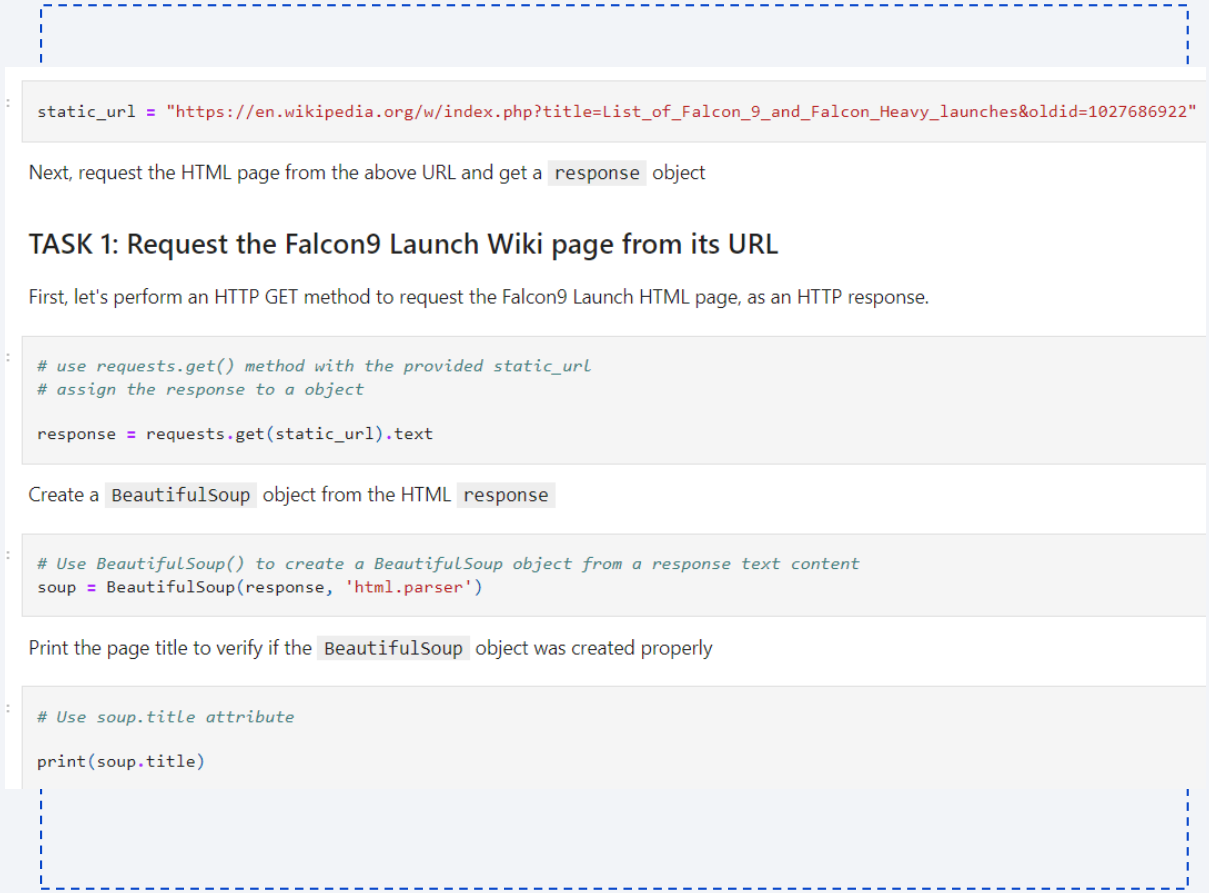
Using the dataframe `data` print the first 5 rows

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

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads	
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	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e45

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/Jperez-lab/testrepo/blob/072e24b36282a29ff26d8c2ab22cf8504df297e8/2_jupyter-labs-webscraping.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

response = 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(response, 'html.parser')
```

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

```
# Use soup.title attribute

print(soup.title)
```

Data Wrangling

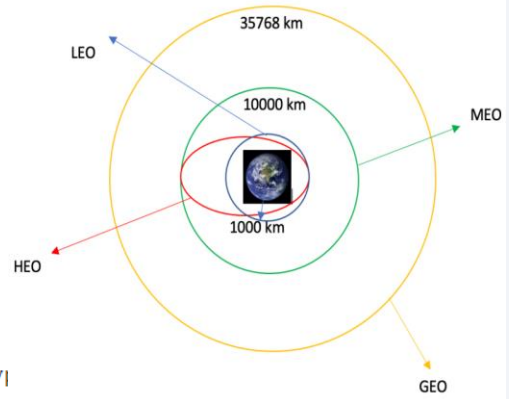
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/Jperez-lab/testrepo/blob/072e24b36282a29ff26d8c2ab22cf8504df297e8/3-spacex-Data%20wrangling.ipynb

```
# 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
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dty
```

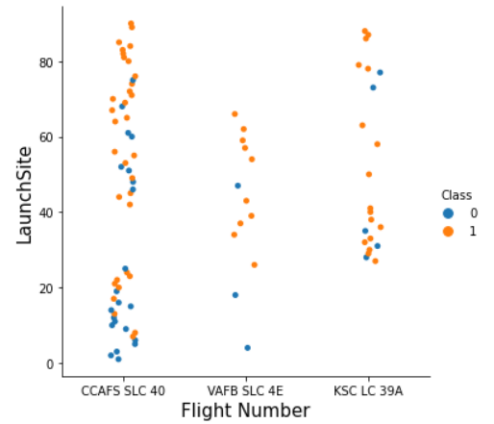



```
df['Class']=landing_class
df[['Class']].head(8)
```

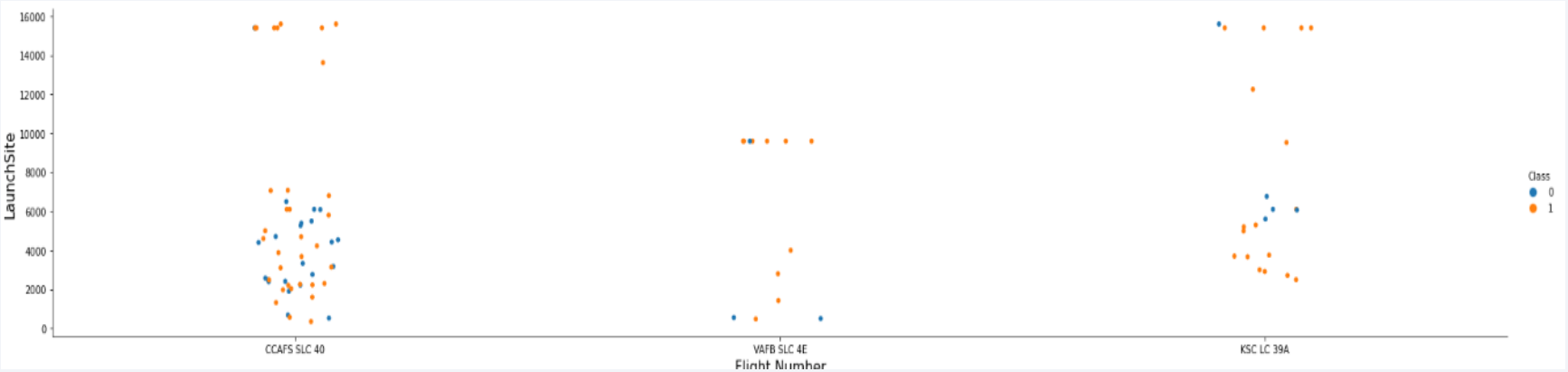
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

EDA with Data Visualization

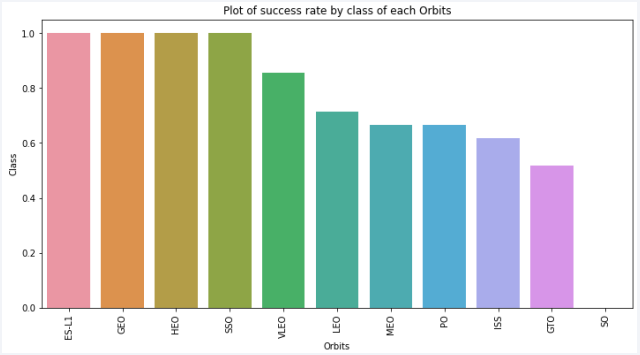
Relationship between Flight Number and Launch Site



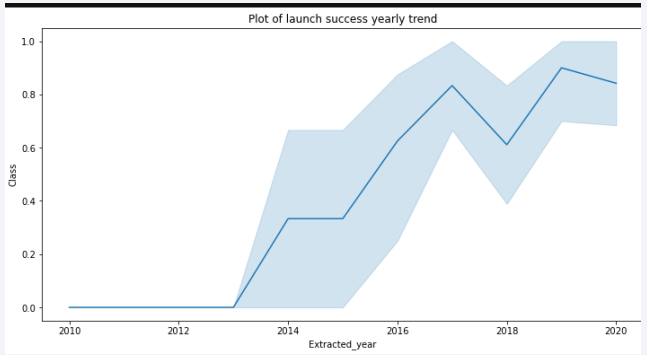
Relationship between Payload and Launch Site



Relationship between success rate of each orbit type



Launch success yearly trend



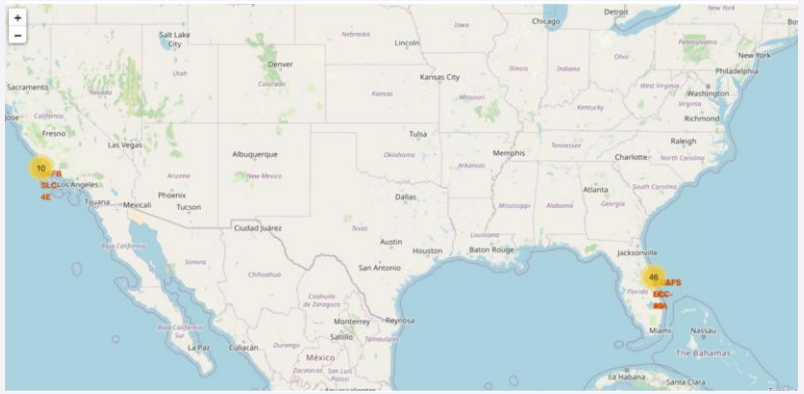
The link to the notebook is
https://github.com/Jperez-lab/testrepo/blob/effb365725ef68e8af31b4b247f5d303478e9e62/7-jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is
- https://github.com/jperez-lab/testrepo/blob/effb365725ef68e8af31b4b247f5d303478e9e62/4-jupyter-labs-eda-sql-edx_sqllite.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities.



The link to the notebook is https://github.com/Jperez -lab/testrepo/blob/effb365725ef68e8af31b4b247f5d303478e9e62/8-Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

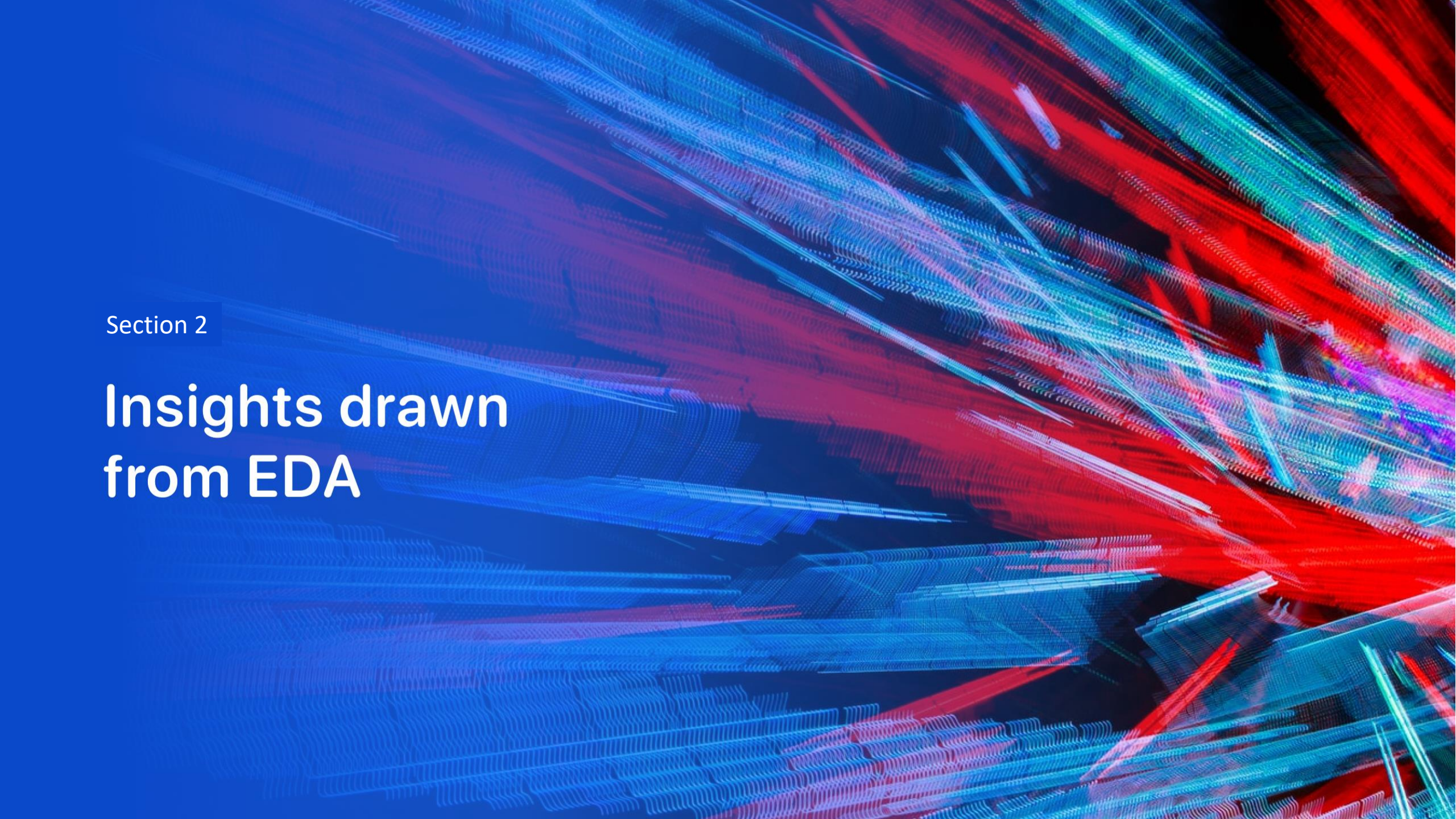
- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/Jperez-lab/testrepo/blob/029bd8d72db4724a43fa83d874f8696dde7861c4/6_interactive_map.ipynb

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/Jperez-lab/testrepo/blob/029bd8d72db4724a43fa83d874f8696dde7861c4/9_%20Assignment%20Machine%20Learning%20Prediction.ipynb

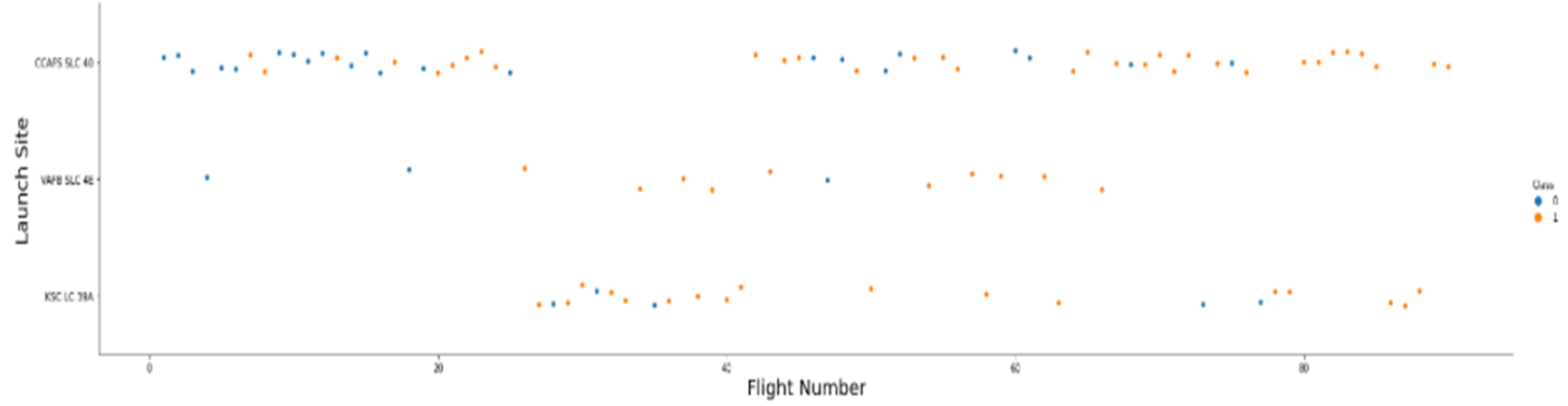
Results

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

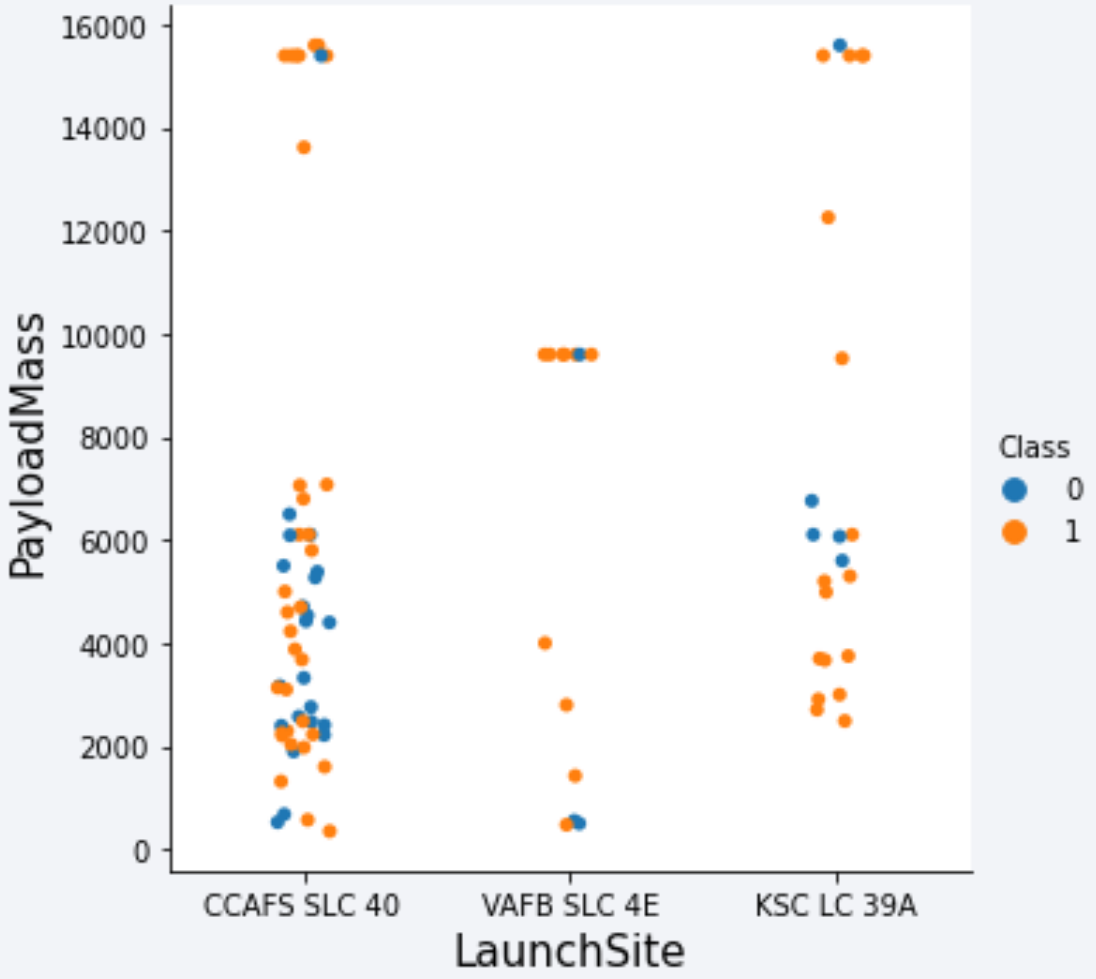

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

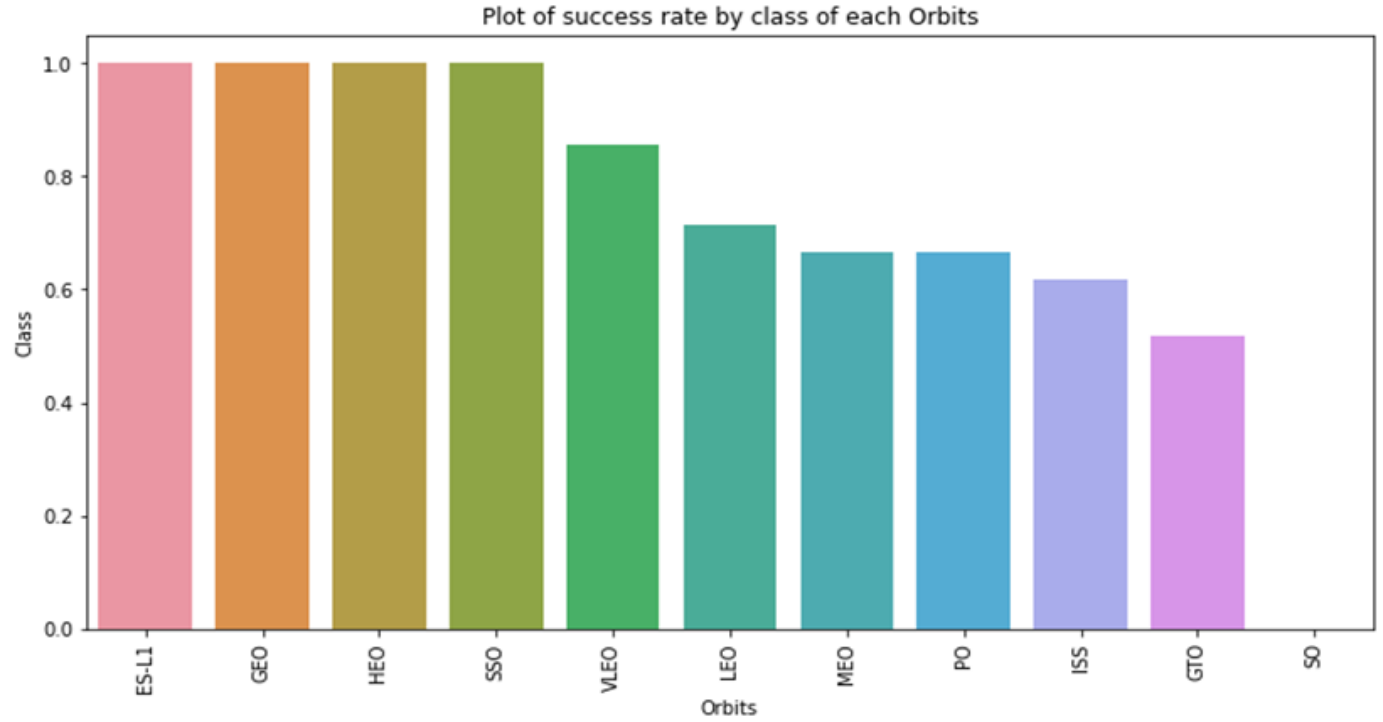


Payload vs. Launch Site



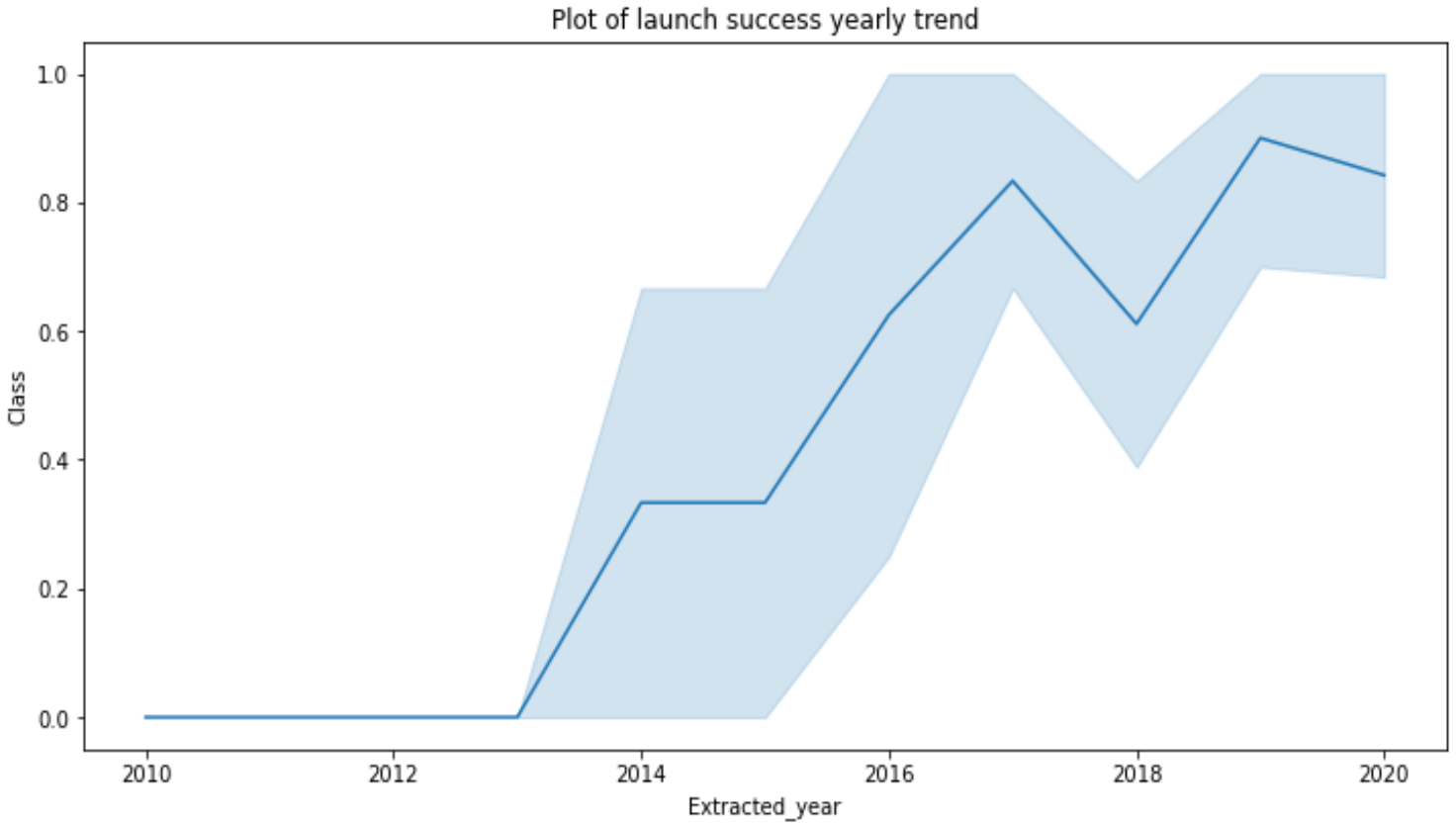
- In the graph of te PlayloadMass Vs LuchSite, we can see that the best class is in CCAFS SLC 40 whit Mass between 8000 and 2000 kg.
- the worst classification is in VAFB SLC 4E lunch site.

Success Rate vs. Orbit Type



- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

Launch Success Yearly Trend



- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.

All Launch Site Names

- We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'KSC'

```
In [8]: %sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'KSC%' LIMIT 5;
 * ibm_db_sa://syr79643:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/BLUDB
Done.
```

Out[8]:

launch_site
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A

- We used the query above to display 5 records where launch sites begin with

https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/4-jupyter-labs-eda-sql-edx_sqllite.ipynb

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 412029 Kg using the query below

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

```
 * ibm_db_sa://syr79643:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/BLUDB
Done.
```

payloadmass
412029

https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/4-jupyter-labs-eda-sql-edx_sqllite.ipynb

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 6437 kg

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

```
 * ibm_db_sa://syr79643:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/BLUDB
Done.
```

payloadmass
6437

https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/4-jupyter-labs-eda-sql-edx_sqllite.ipynb

First Successful Ground Landing Date

```
%sql select min(DATE) from SPACEXTBL;
```

```
 * ibm_db_sa://syr79643:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/BLUDB
Done.
```

1
2012-05-22

- We observed that the dates of the first successful landing outcome on ground pad was 2012-05-22

https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/4-jupyter-labs-eda-sql-edx_sqllite.ipynb

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of the boosters which have success in ground pad 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_ BETWEEN 4000 and 6000
```

```
 * ibm_db_sa://syr79643:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/BLUDB
Done.
```

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

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000.

https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/4-jupyter-labs-eda-sql-edx_sqllite.ipynb

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

```
%sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://syr79643:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/BLUDB
Done.
```

missionoutcomes
1
63

https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/4-jupyter-labs-eda-sql-edx_sqllite.ipynb

2017 Launch Records

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

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

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2017' for year.

```
%sql SELECT MONTH(DATE),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where EXTRACT(YEAR FROM DATE)='2017';
```

```
 * ibm_db_sa://syr79643:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/BLUDB
Done.
```

1	mission_outcome	booster_version	launch_site
1	Success	F9 FT B1029.1	VAFB SLC-4E
2	Success	F9 FT B1031.1	KSC LC-39A
3	Success	F9 FT B1030	KSC LC-39A
3	Success	F9 FT B1021.2	KSC LC-39A
5	Success	F9 FT B1034	KSC LC-39A
6	Success	F9 FT B1029.2	KSC LC-39A
6	Success	F9 FT B1036.1	VAFB SLC-4E
8	Success	F9 B4 B1039.1	KSC LC-39A
8	Success	F9 FT B1038.1	VAFB SLC-4E
10	Success	F9 FT B1031.2	KSC LC-39A
10	Success	F9 B4 B1042.1	KSC LC-39A
12	Success	F9 FT B1035.2	CCAFS SLC-40
12	Success	F9 FT B1036.2	VAFB SLC-4E

https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/4-jupyter-labs-eda-sql-edx_sqllite.ipynb

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

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql SELECT LANDING__OUTCOME FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;
```

```
 * ibm_db_sa://syr79643:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/BLUDB
Done.
```

landing__outcome
No attempt
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Failure (drone ship)
Success (ground pad)
Precluded (drone ship)
No attempt
Failure (drone ship)
Controlled (ocean)
Failure (drone ship)
Uncontrolled (ocean)
Controlled (ocean)
Controlled (ocean)
Uncontrolled (ocean)
No attempt

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Section 4

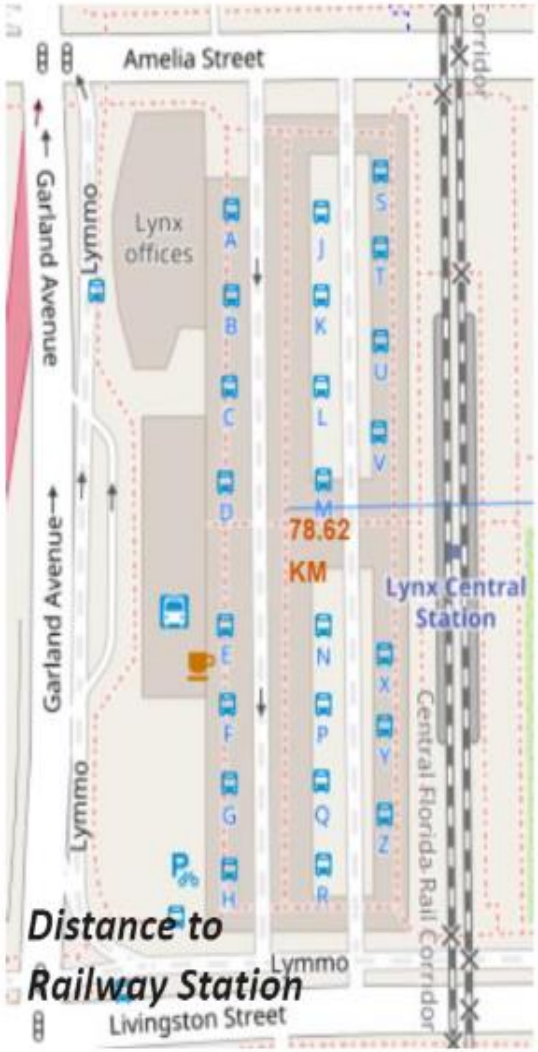
Launch Sites Proximities Analysis

All launch sites global map markers

We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Markers showing launch sites with color labels

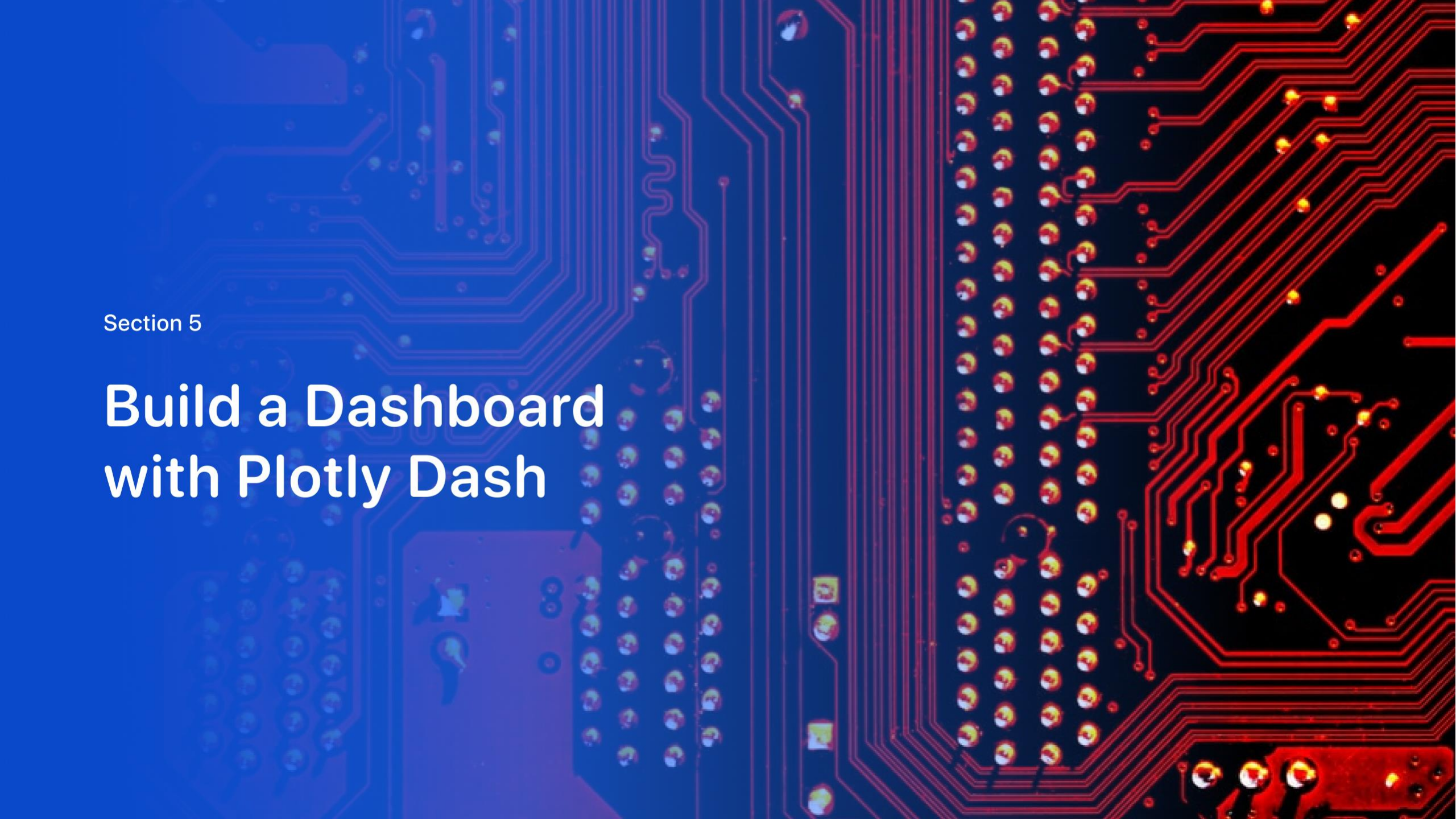
Launch Site distance to landmarks



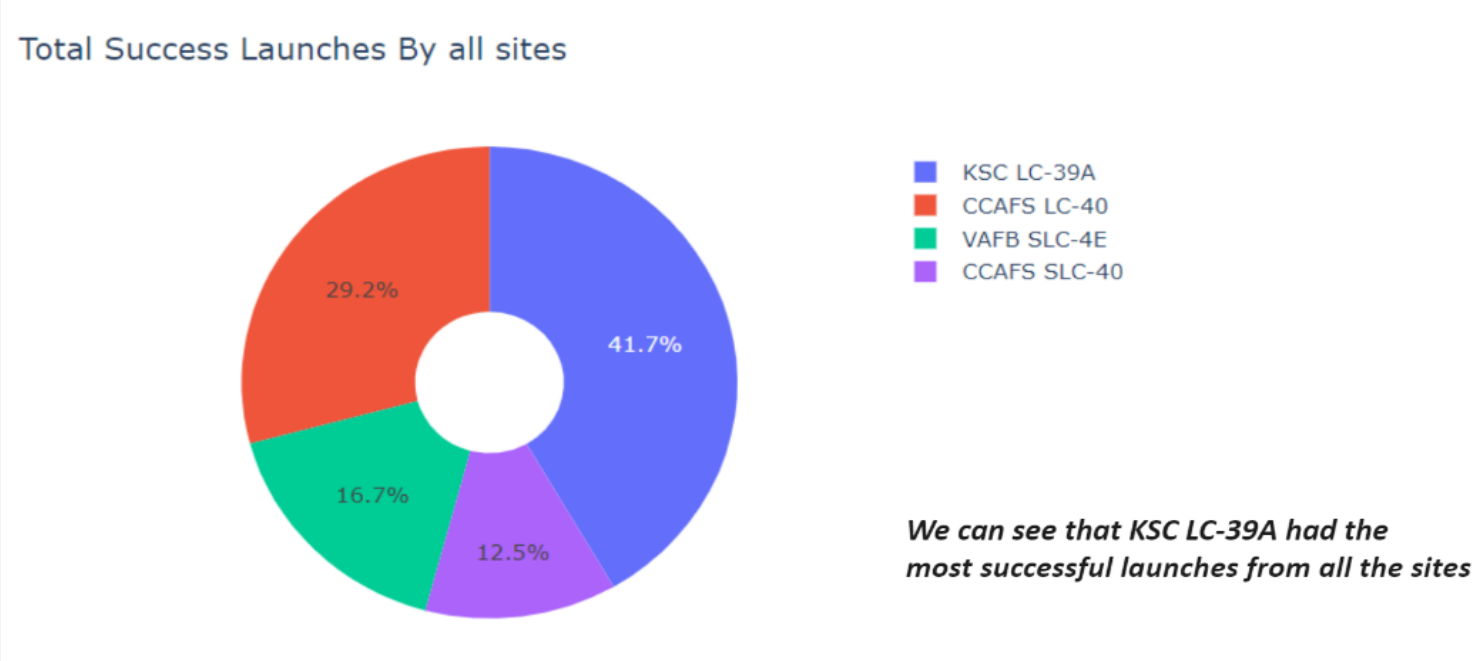
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 5

Build a Dashboard with Plotly Dash

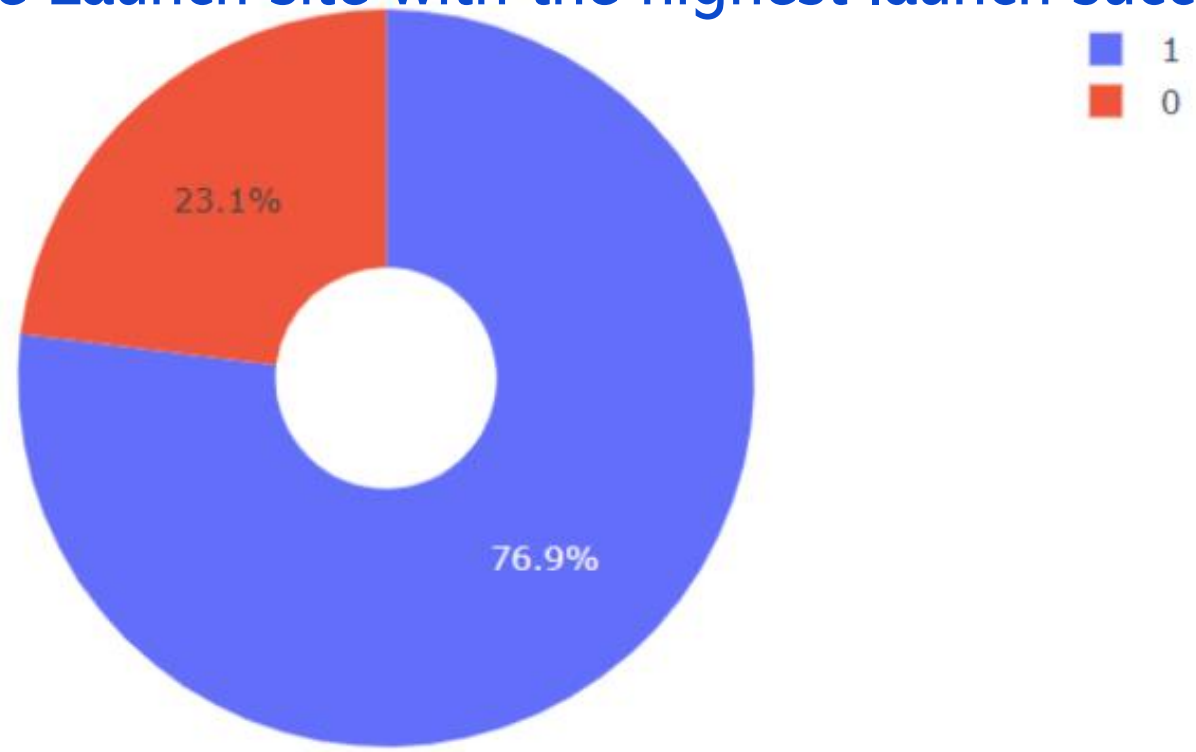


Pie chart showing the success percentage achieved by each launch site



https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/6_interactive_map.ipynb

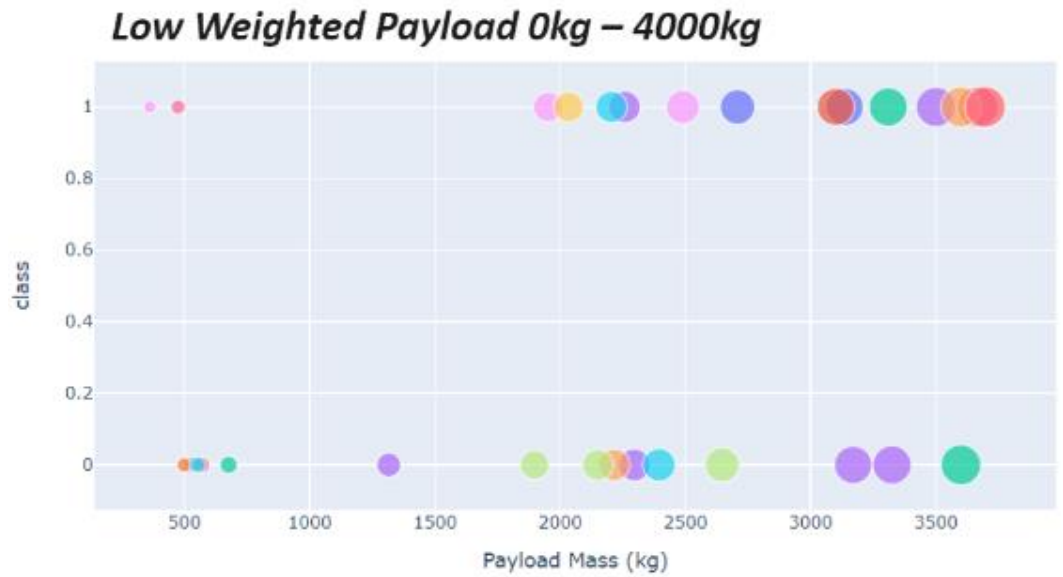
Pie chart showing the Launch site with the highest launch success ratio

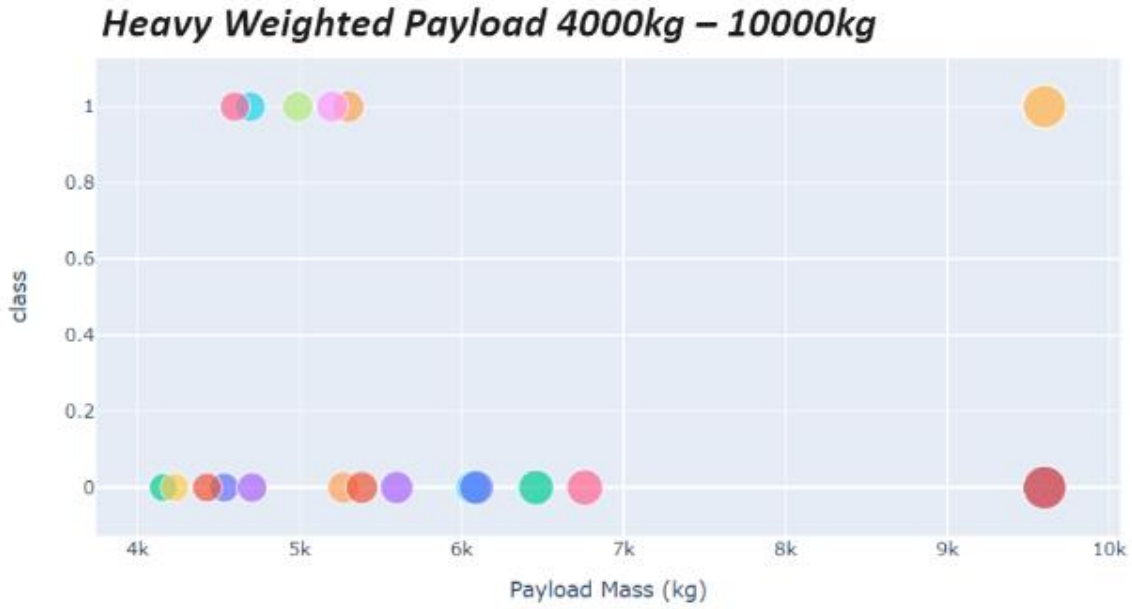


KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/6_interactive_map.ipynb

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

https://github.com/Jperez-lab/testrepo/blob/fa136f583700a7c19ba0834e08b17768a84ae4a7/6_interactive_map.ipynb

Section 6

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
parameters = {'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()
```

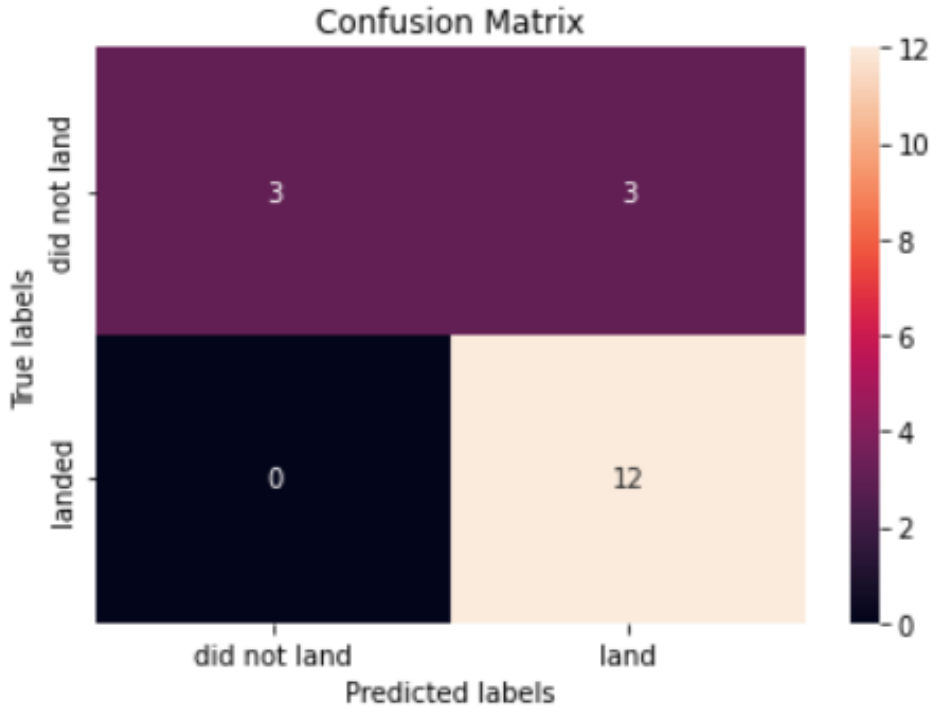
```
grid_search = GridSearchCV(tree, parameters, cv=10)
tree_cv = grid_search.fit(X_train, Y_train)
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10,
'splitter': 'random'}
accuracy : 0.875
```

Confusion Matrix

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



- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.

Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits 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 classifier is the best machine learning algorithm for this task.

Thank you!