IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Vicenso Drushku

31/10/2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - 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 upwards 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. The 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 whether a rocket will land successfully
  - The correlation among different features that determine the success rate of rocket landing
  - What operating conditions need to be placed for a successful landing

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

- Various methods were used to collect data.
  - ➢ 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 clean 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

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

- Notebook link is https://github.com/ViD4314/Coursera-exercise/blob/2982c094e8074f2002502fd77c87e17af2e97c57/jupyter-labs-spacex-data-collection-api.ipynb

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clo
```

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

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

Out[10]: 200

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

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
response= response.json()
# apply json_normalize
data = pd.json_normalize(response)
```

Using the dataframe `data` print the first 5 rows

In [12]:
```
# Get the head of the dataframe
data.head()
```

Out[12]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False

Data Collection - Scraping

- BeautifulSoup was used for Web scraping to retrieve Falcon 9 records
- Table was parsed and converted into a pandas dataframe
- Notebook link https://github.com/ViD4314/Coursera-exercise/blob/7aed0be0609a73ad5e30250f279ceeaf733d34fc/jupyter-labs-webscraping.ipynb

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

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

```
[5]: 200
```

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

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

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

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

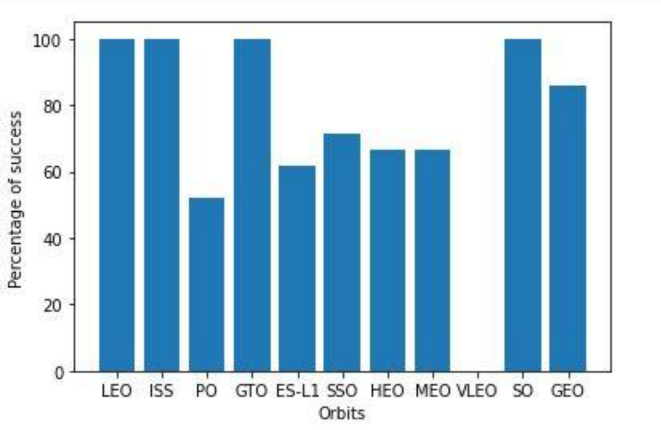
```
[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

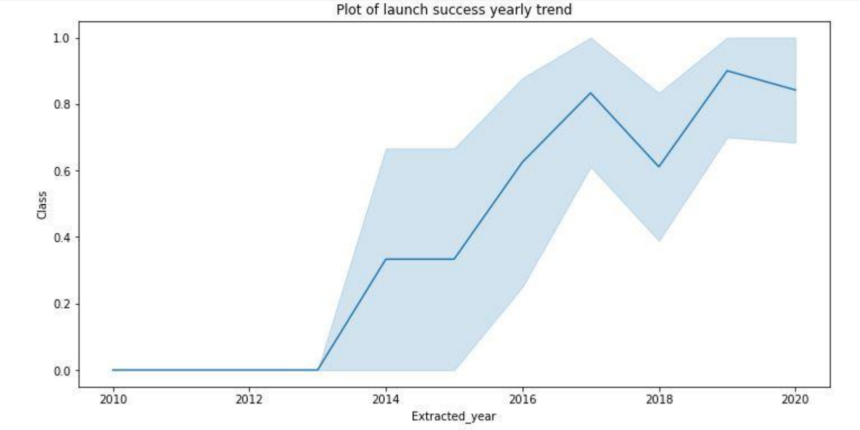
Data Wrangling

- Calculated the number of launches at each site, and the number and occurrence of each orbit
- Created landing outcome label from the outcome column and exported the results to a csv file.
- Notebook link https://github.com/ViD4314/Coursera-exercise/blob/5c77d3d4f6e6207998e075935e7875aeec33335a/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We used visualization to explore the data and see the relationship between flight number and Launch Site, Payload and launch Site, success rate of each orbit type, the launch success yearly trend.
- The notebook link https://github.com/ViD4314/Coursera-exercise/blob/481684f3b0e81d6506eac6ae19488de87d26e9ab/jupyter-labs-eda-dataviz.ipynb





EDA with SQL

- The SpaceX dataset was loaded into a PostgreSQL database
- W applied EDA with SQL to get insights from the data. We wrote queries to find out for instance:

✓ Unique launch site names

✓ The total payload mass carried by boosters launched by NASA (CSR)

✓ 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

- GitHub URL:https://github.com/ViD4314/Coursera-exercise/blob/d514c5cdb821462bcd62d70ce40526fc573a84a0/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- 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.
- Assigned the feature launch outcome to calss 0 and 1.
- Using the color-labeled marker clusters, we identified which launch sites have a relatively high success rate.
- Calculated the distance between a launch site to its proximities.
- GitHub URL: https://github.com/ViD4314/Coursera-exercise/blob/b2b6835158da66d15353a4f630da4dd1b6251d1d/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Created an interactive dashboard with Plotly dash
- Created pie charts that showed the total launches by a certain sites
- Plotted scatter graph showing relationship with Outcome and Payload Mass for the different booster version.
- GitHub URL: https://github.com/ViD4314/Coursera-exercise/blob/e005d55149d2e9fe6e792a65d0d9fab9ad1ae06a/Dash_capstone_project.ipynb

Predictive Analysis (Classification)

- Using Pandas and Numpy libraries we loaded, transformed and, split our data into training and testing sets.
- Built different machine learning models and tried different hyperparameters using GridSearchCV.
- Improved the model using algorithm tuning and figured out the best classification model.
- GitHub URL: https://github.com/ViD4314/Coursera-exercise/blob/847b9c3de3b829961167f287dde701fed411b0eb/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

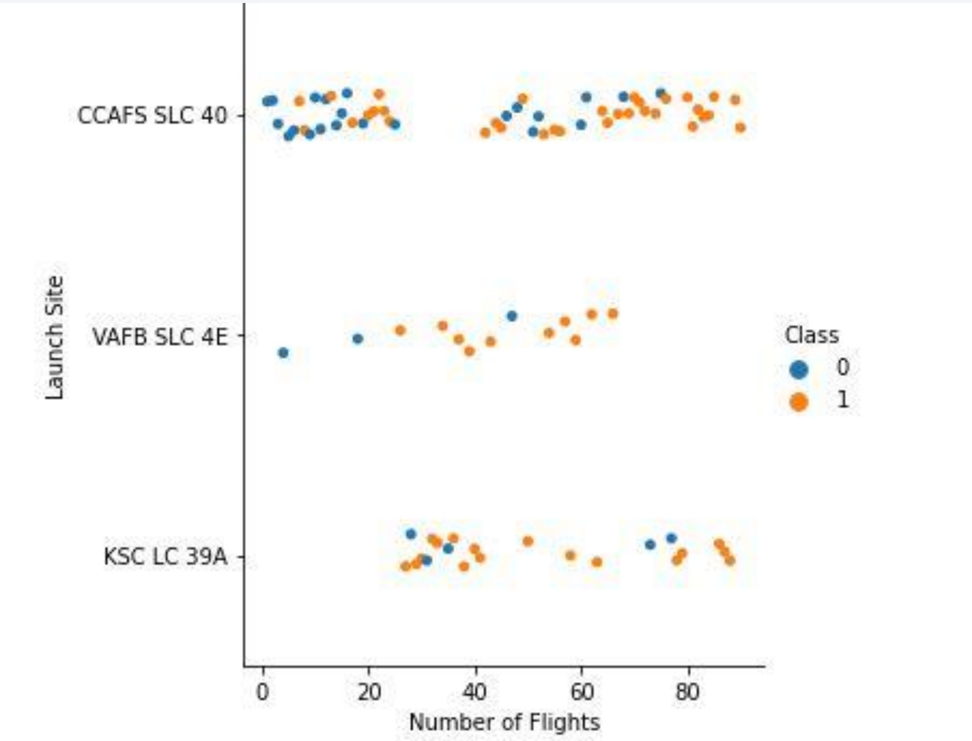
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

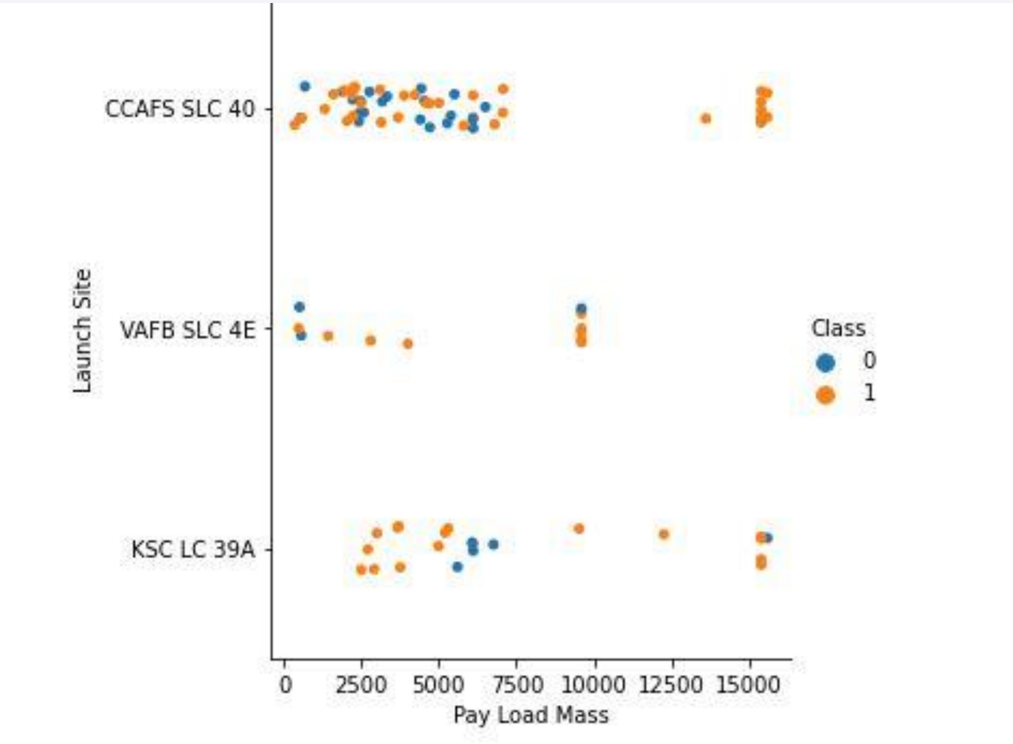
Flight Number vs. Launch Site

- The larger the flight amount from a launch site, the greater the success rate.



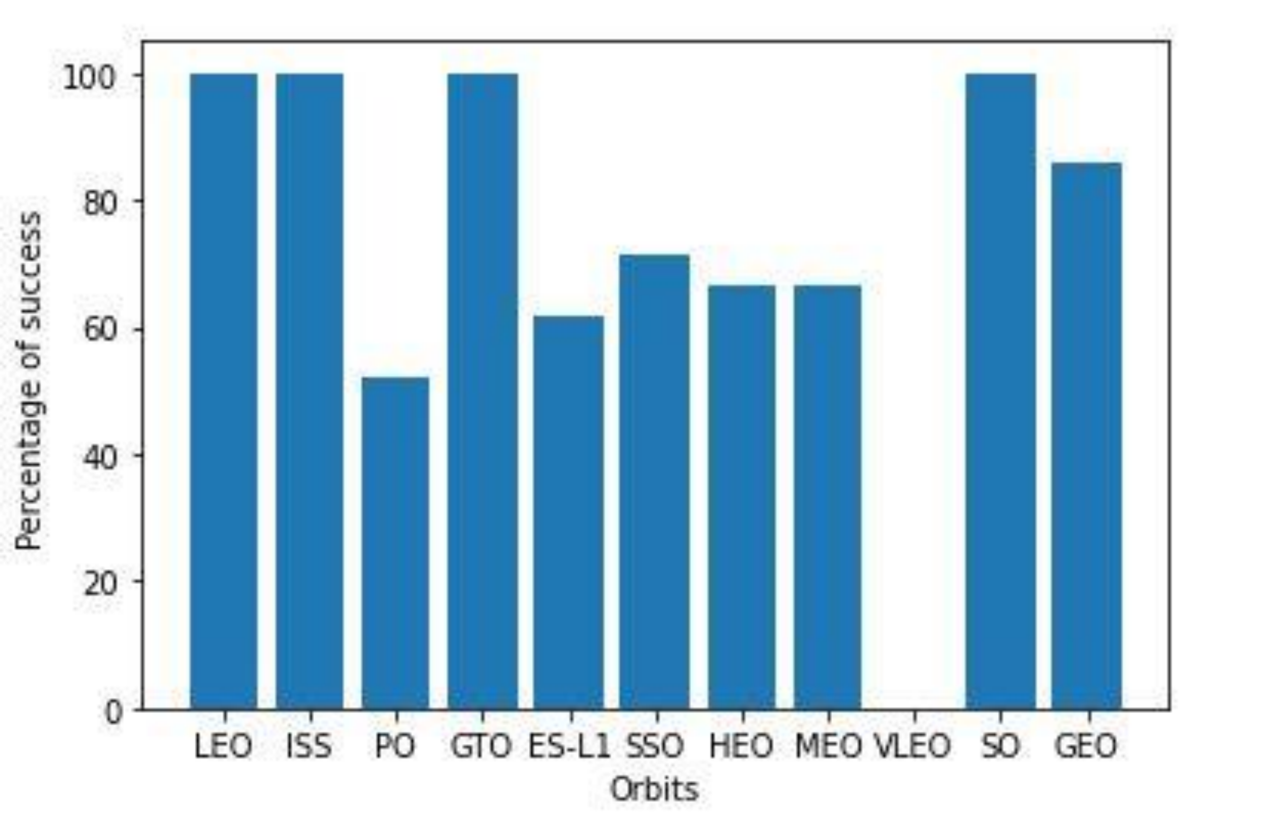
Payload vs. Launch Site

- The scatter plot shows that for the launch site CCAFS SLC 40 the greater the payload, the higher the success rate,



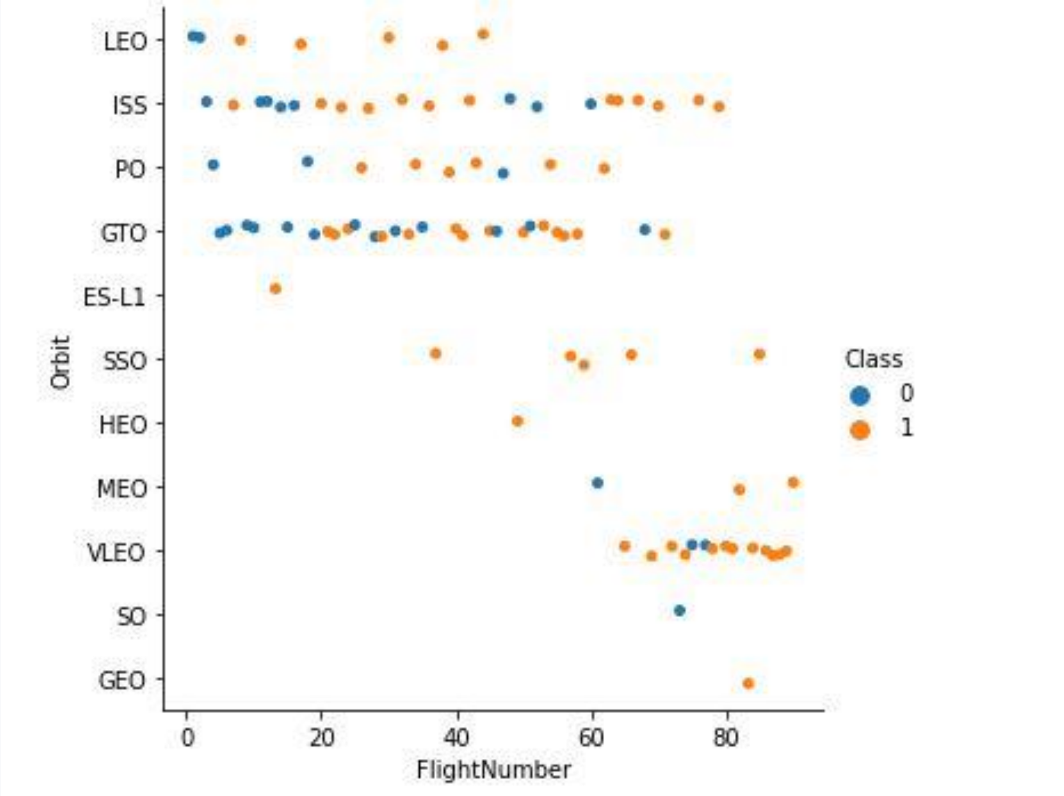
Success Rate vs. Orbit Type

- We see that LEO, ISS, GTO, SO and GEO have highest success rate



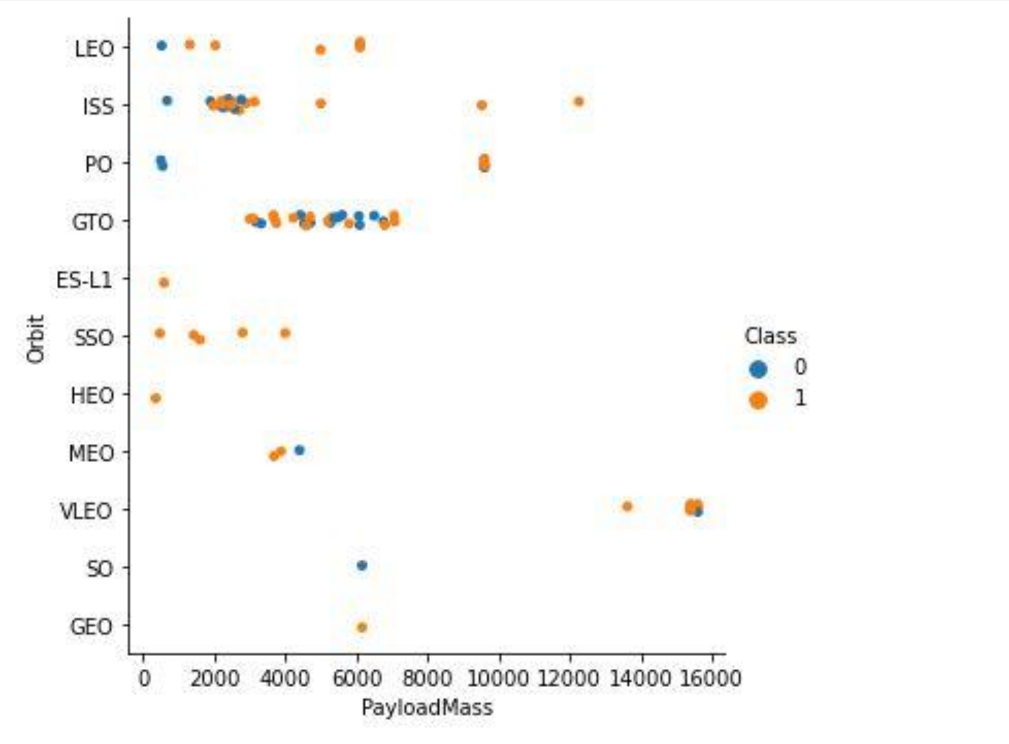
Flight Number vs. Orbit Type

- In the LEO orbit, the success is related to number of flights.
- GTO orbit, there is no relationship between flight number and orbit.



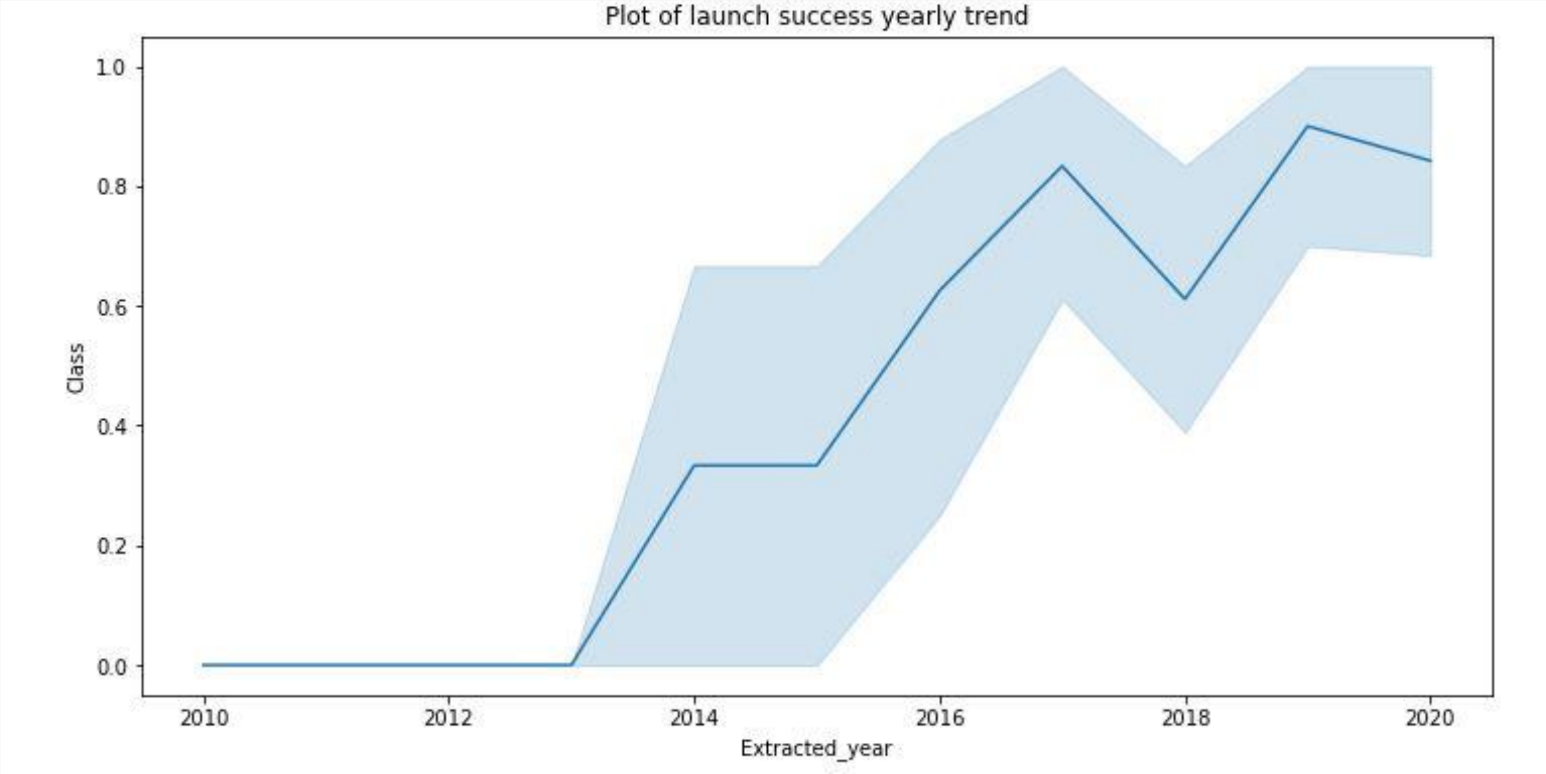
Payload vs. Orbit Type

- LEO, ISS and PO have a higher success rate with heavy loads.



Launch Success Yearly Trend

- We notice an increase starting from year 2013 up to 2020 with a small dip in 2018.



All Launch Site Names

- Using DISTINCT to only show unique launch site names

Task 1

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

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

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

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

Launch Site Names Begin with 'CCA'

- Here we use LIKE '...%' to find records starting with 'CCA'

Task 2

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

```
%%sql
SELECT *
FROM SPACEXTBL
WHERE Launch_Site LIKE 'CCA%'
LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Here we use the SUM() function to calculate the sum of all values in the column

Task 3

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

[9]:

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer='NASA (CRS)'
```

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

t[9]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Here we use the avg() FUNCTION to find the average of all values in the column

Task 4

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

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.1%'
```

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

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

- Since the column was already ordered by date, we chose the first record that had a successful ground pad landing outcome

Task 5

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

Hint:Use min function

```
%%sql
SELECT "Date"
FROM SPACEXTBL
WHERE "Landing _Outcome"='Success (ground pad)'
LIMIT 1
```

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

Date
22-12-2015

Successful Drone Ship Landing with Payload between 4000 and 6000

- Here we added multiple conditions on the WHERE syntax and used the DISTINCT function to get the unique names

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 DISTINCT Booster_Version
FROM SPACEXTBL
WHERE "Landing _Outcome"='Success (drone ship)'
AND PAYLOAD_MASS__KG_>4000
AND PAYLOAD_MASS__KG_<6000
```

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

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

Total Number of Successful and Failure Mission Outcomes

- Here we simply used the GROUP BY method to group the number of records by mission outcome.

Task 7

List the total number of successful and failure mission outcomes

```
%%sql
SELECT Mission_Outcome, COUNT(DATE)
FROM SPACEXTBL
GROUP BY Mission_Outcome
```

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

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

Boosters Carried Maximum Payload

- Here we got the unique names of booster versions where the mass was equal to max mass which we got using a subquery.

Task 8

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

```
%%sql
SELECT DISTINCT Booster_Version
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_=(SELECT MAX(PAYLOAD_MASS__KG_)
                         FROM SPACEXTBL)
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- Here we used the substr() function to get the month and the year from the Date column

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

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

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

Month	Landing _Outcome	Booster_Version	Launch_Site	Year
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015

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

- Again we used GROUP BY to group the count of success landings and we ordered it descending by using ORDER BY and DESC

Task 10

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

```
%%sql
SELECT "Landing _Outcome", COUNT("Landing _Outcome") AS Success_number
FROM SPACEXTBL
WHERE "Landing _Outcome" LIKE '%Success%' AND "Date" BETWEEN '04-06-2010' AND '20-03-2017'
GROUP BY "Landing _Outcome"
ORDER BY Success_number DESC
```

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

Landing _Outcome	Success_number
Success	20
Success (drone ship)	8
Success (ground pad)	6

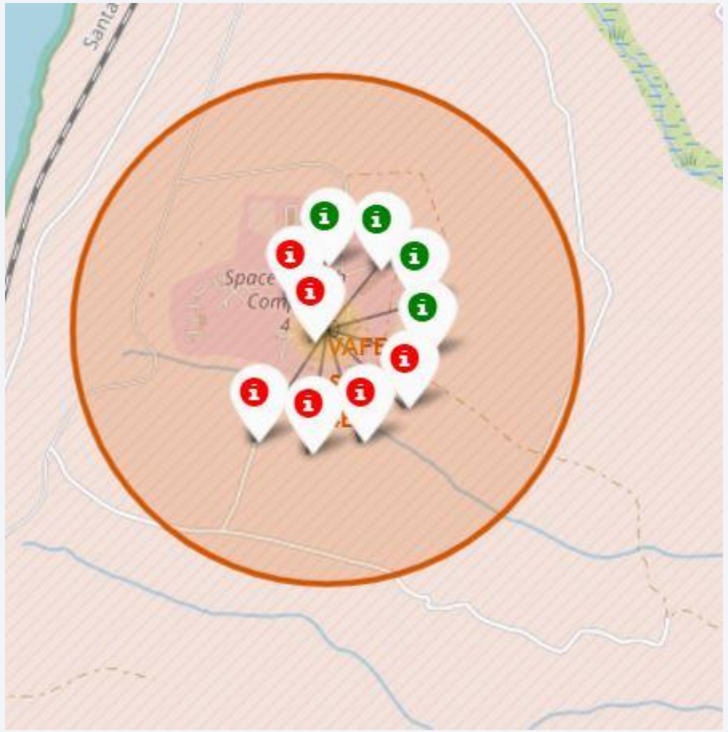
Section 3

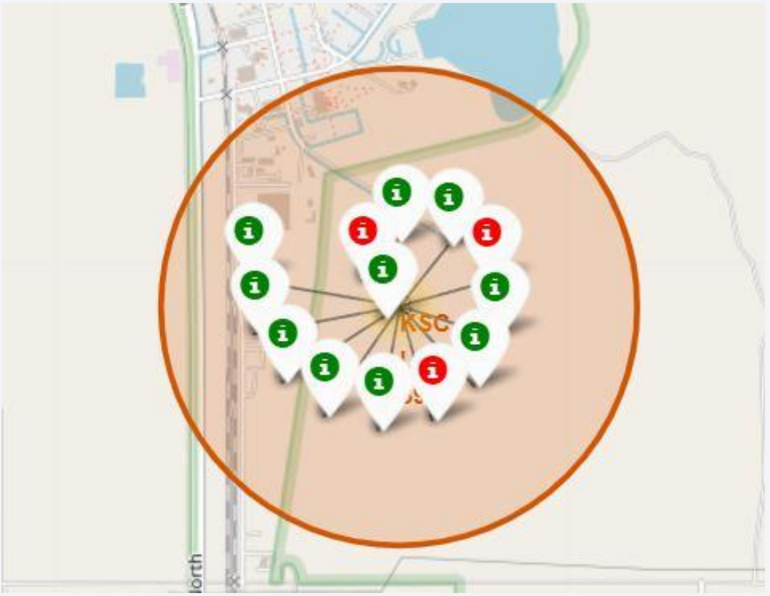
Launch Sites Proximities Analysis

Where are the launch sites?

- All of them seem to be in the USA

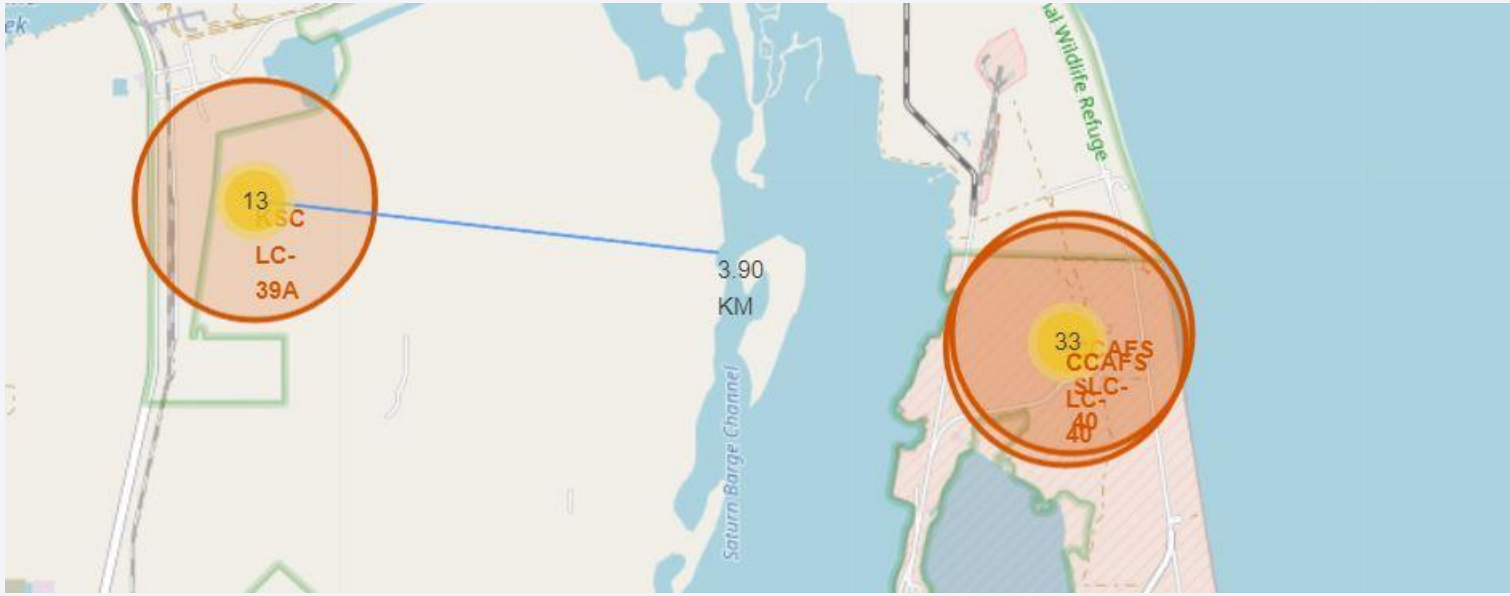
Launch sites with color labels





Green represent successful launch and red represents unsuccessful

Launch Site Proximity



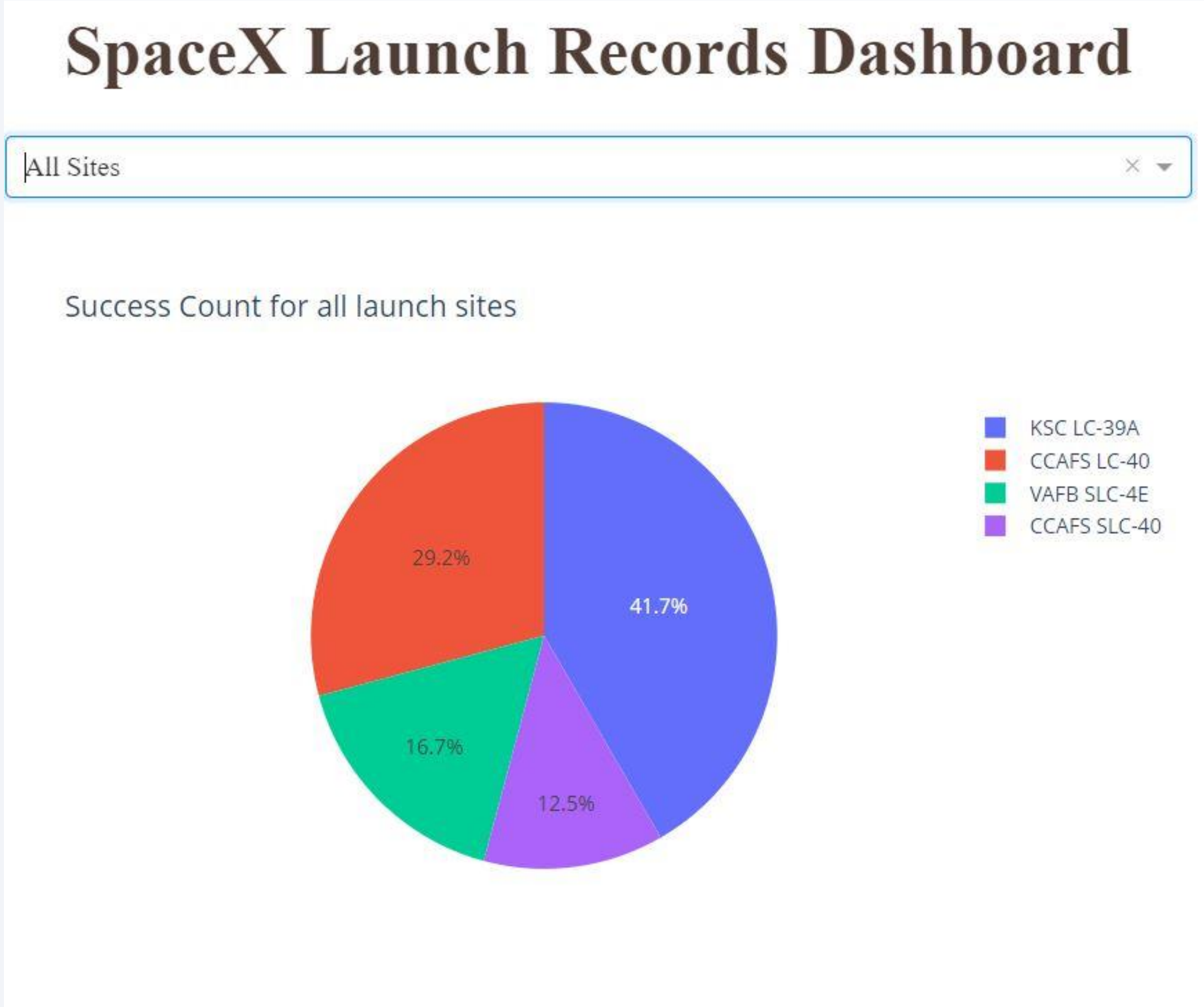
Distance to the coastline.

Section 4

Build a Dashboard with Plotly Dash

Success percentage by launch site

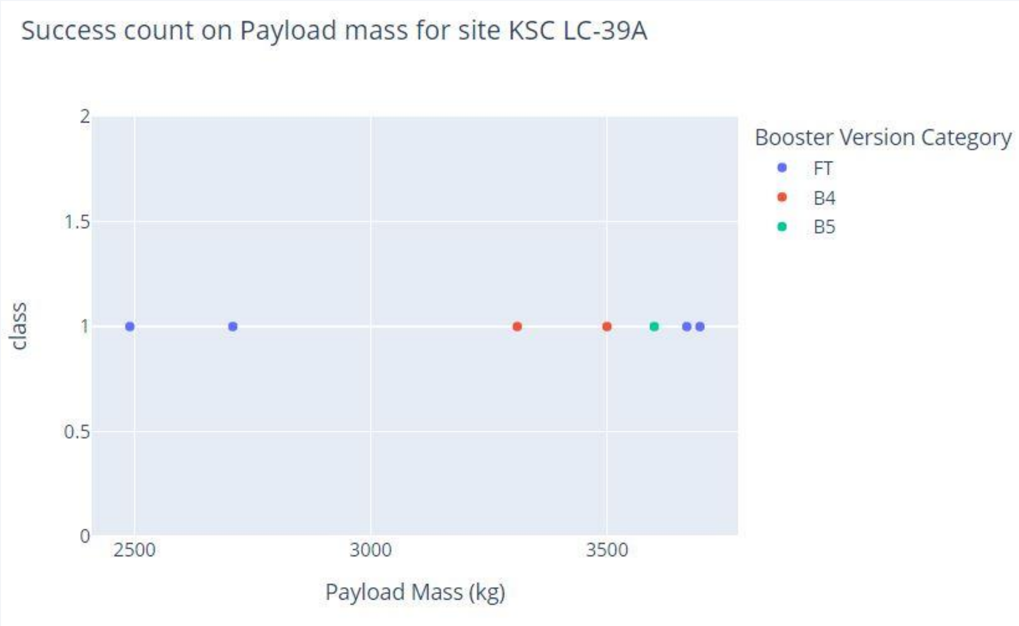
KSC LC-39A has the highest number of successful launches compared to all other launch sites.

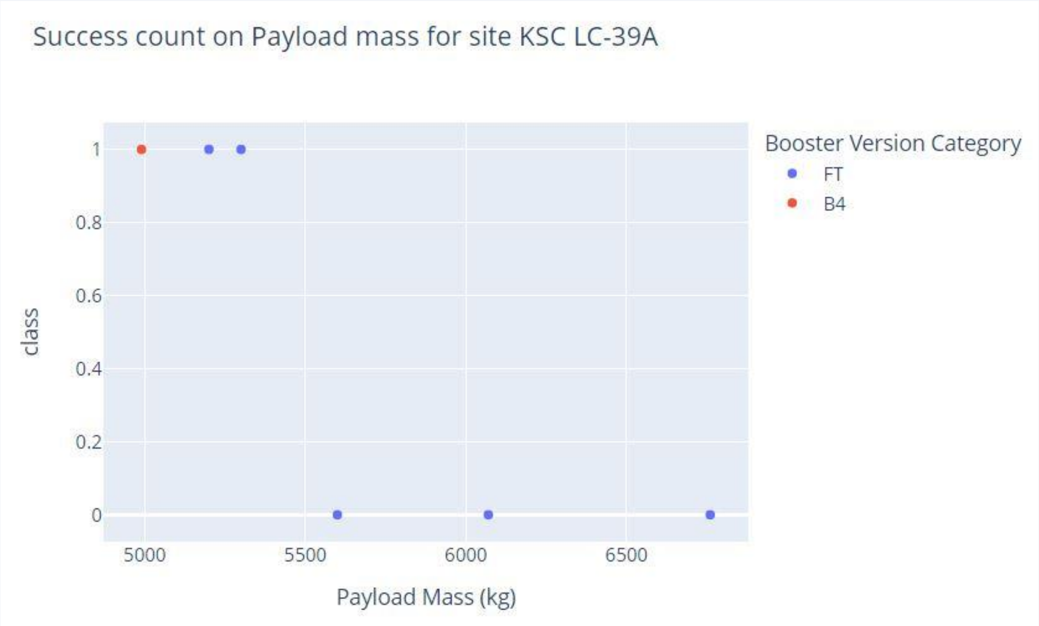


Pie chart showing success rate of the best launch site

KSC LC-39A has a success rate of 76.9% and a failure rate of 23.1%.

Payload vs Launch Outcome scatter plot





We can tell from the two ranges that the lower the mass the higher the chances for a successful outcome.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Best model seems to be the DecisionTree with a score of 0.875

TASK 12

Find the method performs best:

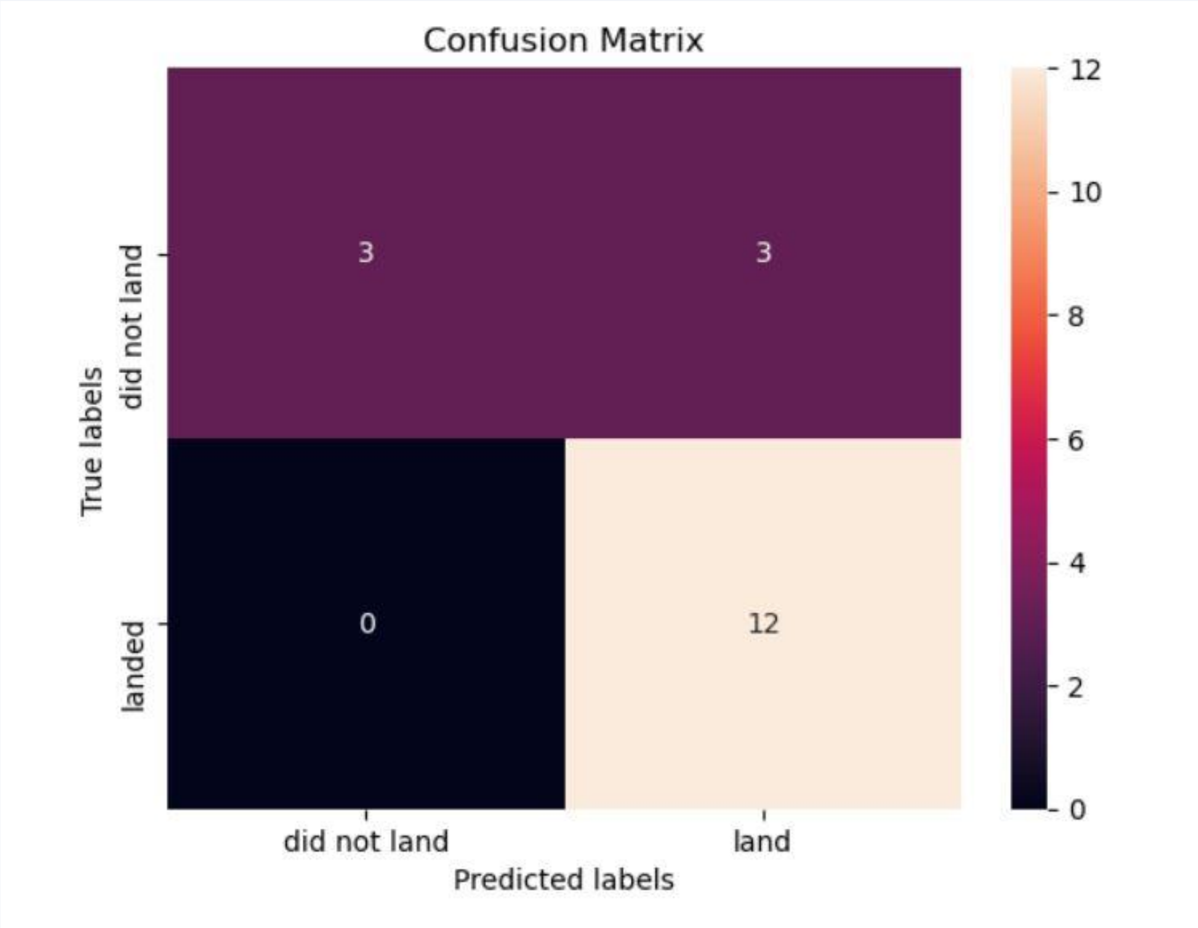
```
[50]: models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.875
Best params is : {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt',
```

Confusion Matrix

- From the confusion matrix our model can accurately calculate when a rocket lands but we seem to have a problem with false positives.



Conclusions

- The more flights at a launch site, the higher the success.
- Launch success started to increase from 2013 to 2020.
- Orbits LEO, ISS, GTO, SO and GEO have the highest success rate.
- KSC LC-39A has the highest number of successful launches.
- The Decision tree classifier is the best machine learning model for this case.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

- The Github repository for this project: https://github.com/ViD4314/Coursera-exercise.git

Thank you!