IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

David Morpeth
31 July 2022

Outline

Executive Summary

Introduction

Methodology

Results

Conclusion

Appendix

Executive Summary

Summary of methodologies

- Data Collection through API and Web Scraping
- Data Transformation
- Exploratory Data Analysis with SQL and Data Visualisation
- Interactive Maps with Folium
- Predictive Analytics and Machine Learning

Summary of all results

- Data analysis with visualisations
- Best model for Predictive Analytics

Introduction

Project background and context

- Space X is a company that builds and launches rockets with the ability to land\re-use the first stage. The ability to re-use their rockets is a cost saving close to $100m, $165m being the normal cost but reduced to $62m for SpaceX. Determining the success of landing the first stage would give a company significant advantage in bidding for contracts and as such the purpose of this project is to create a machine learning\predictive model that will enable a company to compete directly against SpaceX.

Problems you want to find answers

- What is the probability of a successful launch and landing
- What conditions a conducive to a successful launch.

ction 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

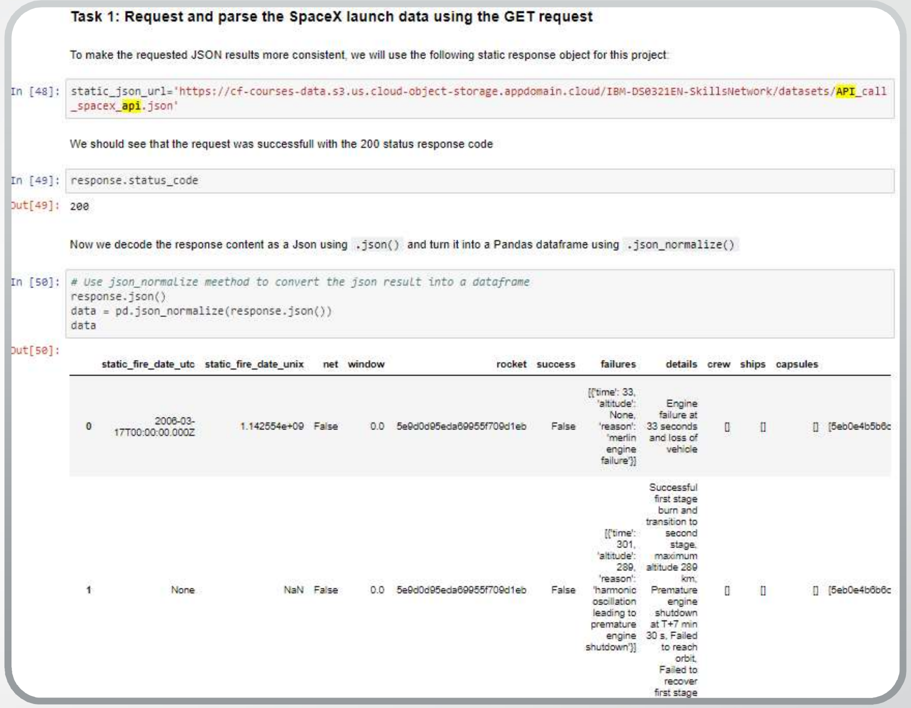
- How to build, tune, evaluate classification models

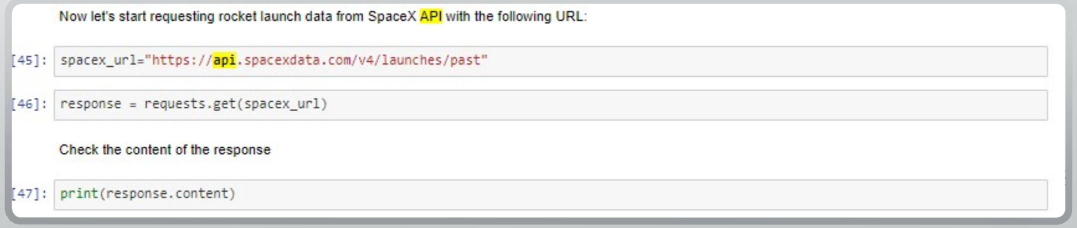
Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Using a get request we used .json() to load into a pandas dataframe and normalize the data
  - We checked the quality of data, resolving empty or null fields, remove duplicates and replaced invalid or missing data.
  - 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

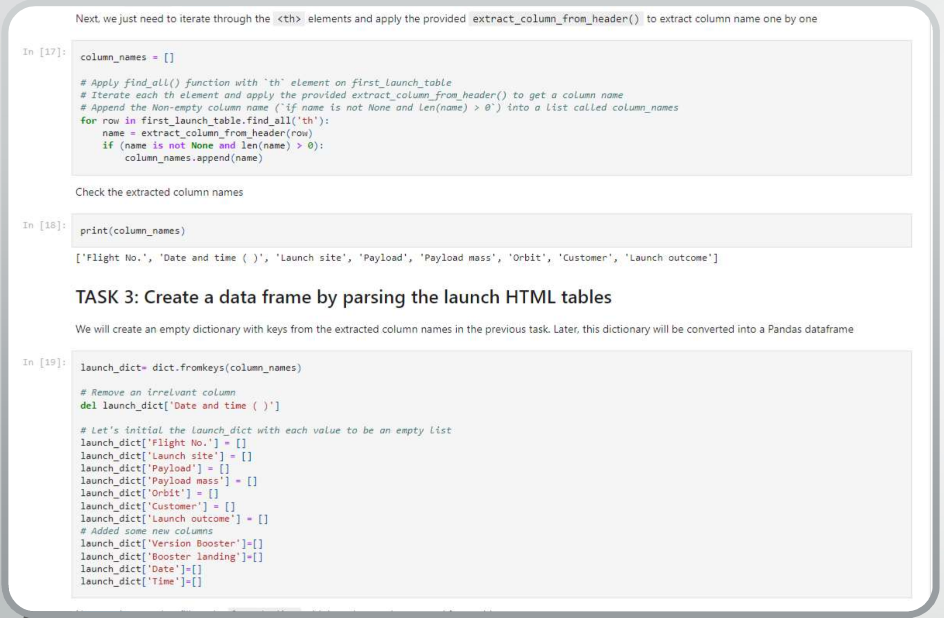
- We used the SpaceX API to extract the data
- Link to project
- https://github.com/DavidMorpeth/DataScienceCapstoneProject/blob/main/SpaceX%20Notebook.ipynb





Data Collection - Scraping

- Beautiful Soup was used to webscrape the data and present the findings. Data was converted into a panda dataframe to allow further analysis.
- Link to book
- https://github.com/DavidMorpeth/DataScienceCapstoneProject/blob/main/SpaceX%20Notebook%20-%20Webscraping.ipynb


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

In [17]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name is not None and len(name) > 0):
        column_names.append(name)
```

Check the extracted column names

In [18]:
```
print(column_names)
```
```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

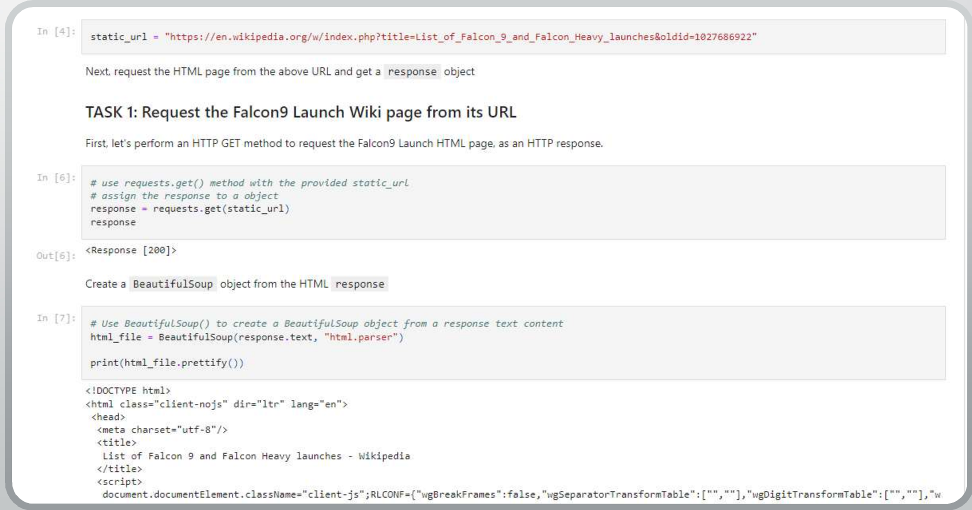
TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

In [19]:
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```


In [4]:
```
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.

In [6]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
response
```
Out[6]:
```
<Response [200]>
```

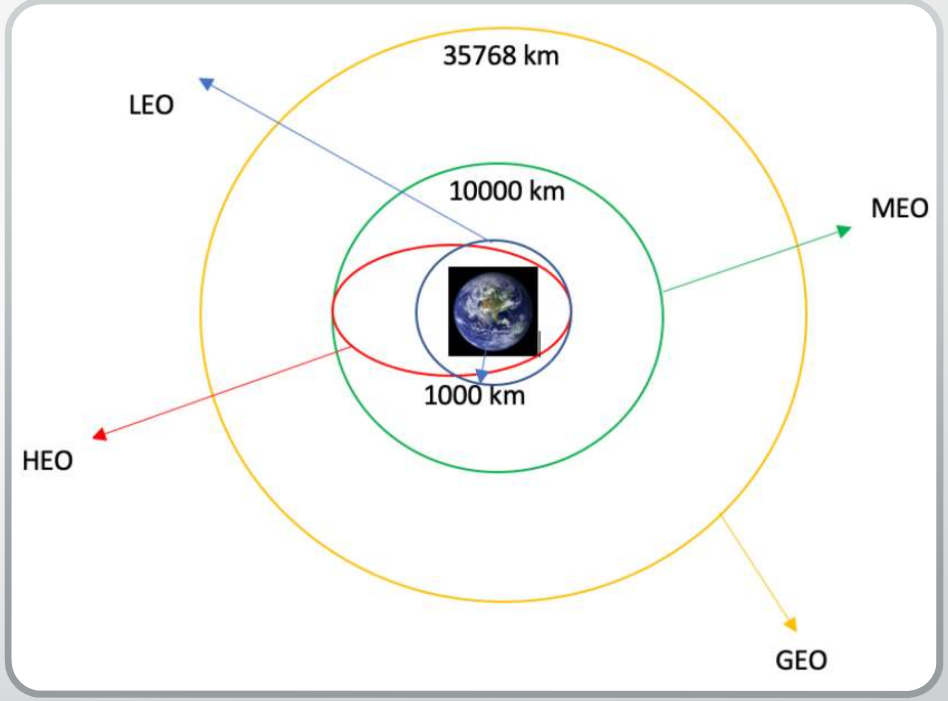
Create a `BeautifulSoup` object from the HTML `response`

In [7]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
html_file = BeautifulSoup(response.text, "html.parser")

print(html_file.prettify())
```
```
<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   List of Falcon 9 and Falcon Heavy launches - Wikipedia
  </title>
  <script>
   document.documentElement.className="client-js";RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"w
```

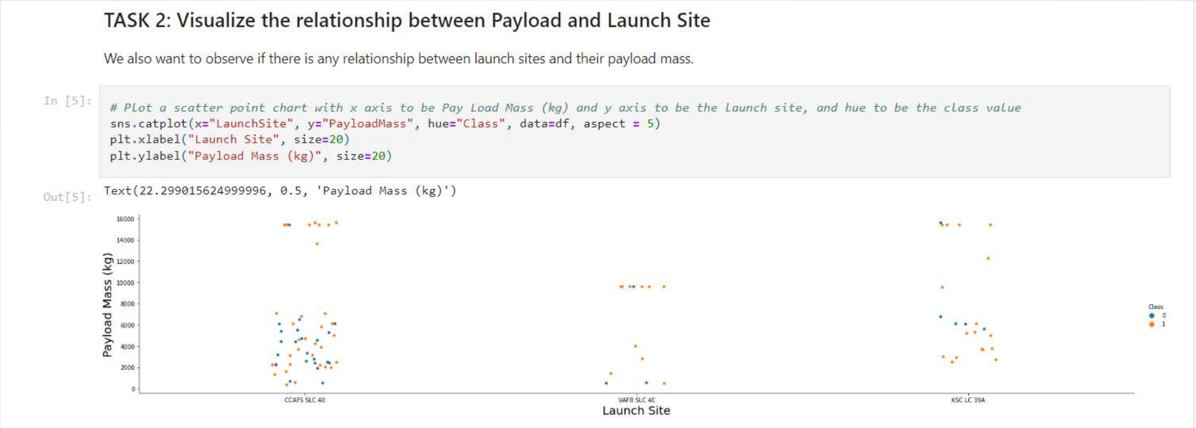
Data Wrangling

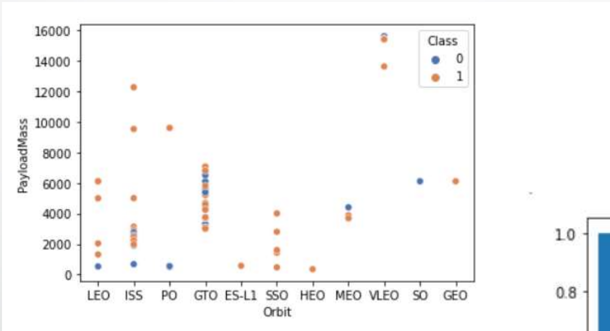
- We performed exploratory data analysis to determine the training labels.
- We calculated the number and occurrence of each orbit, created landing outcome label and exported the results for later analysis.
- The link to the notebook is
- https://github.com/DavidMorpeth/DataScienceCapstoneProject/blob/main/SpaceX%20Notebook%20-%20Data%20Wrangling.ipynb

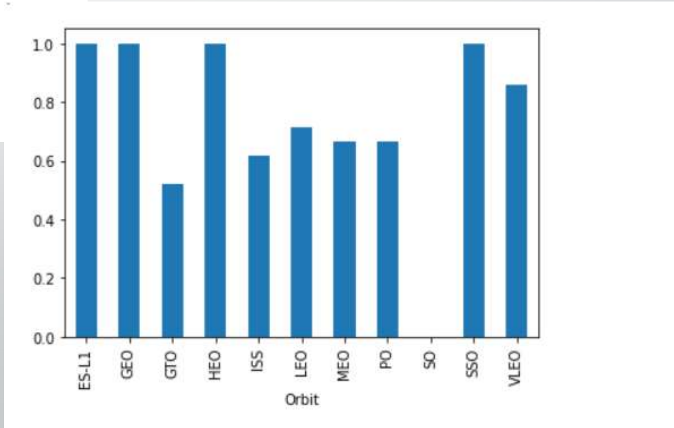


EDA with Visualisation

- Using libraries for Pandas and Matplotlib we used a number visualisations to gain insights into the launches.
  - Using catplot graphs we saw relationships between flight number and launch sites, payload mass and launch sites.
  - Using scatterplots we saw relationships between flight number and orbit type and payload mass.
  - Using bar charts we saw the relationship between success rate and orbit type.
- The link to the notebook is
- https://github.com/DavidMorpeth/DataScienceCapstoneProject/blob/main/EDA%20with%20Visualization%20lab.ipynb







EDA with SQL

Using IBM Cloud Paks we loaded the SpaceX dataset into a DB2 database

SQLalchemy libraries allowed connection to this dataset to obtain key statistics.

Sum of payload mass carried by boosters launched by NASA (CRS)

Average payload mass carried by booster version F9 v1.1

First successful landing outcome.

Count of successful and failed missions

The link to the notebook is

https://github.com/DavidMorpeth/DataScienceCapstoneProject/blob/main/SpaceX%20Notebook%20-%20EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

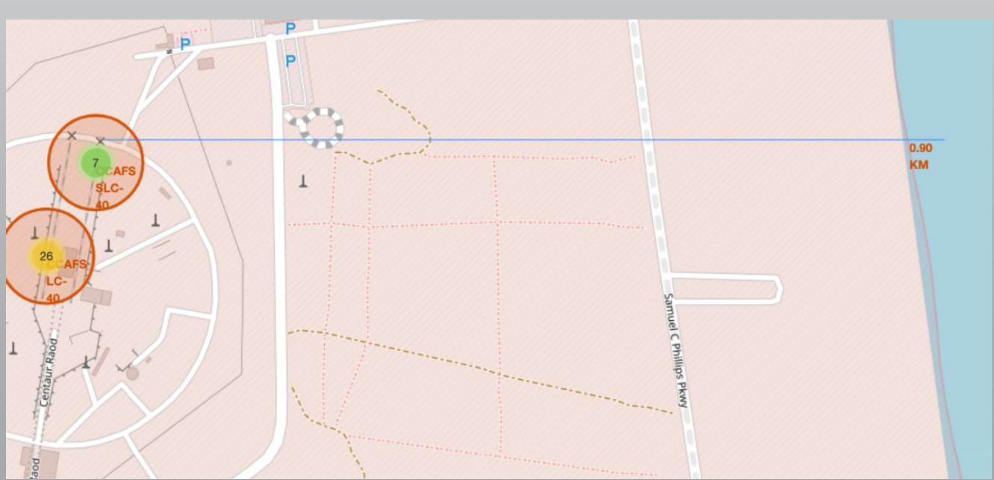
We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.

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. We answered some question for instance:

- Are launch sites near railways, highways and coastlines.
- Do launch sites keep certain distance away from cities.



Build a Dashboard with Plotly Dash

WE BUILT AN INTERACTIVE DASHBOARD WITH PLOTLY DASH

WE PLOTTED PIE CHARTS SHOWING THE TOTAL LAUNCHES BY SITE

WE PLOTTED DATA SHOWING THE RELATIONSHIP BETWEEN OUTCOME AND PAYLOAD MASS (KG) FOR THE DIFFERENT BOOSTER VERSION.

THE LINK TO THE NOTEBOOK IS

HTTPS://GITHUB.COM/DAVIDMORPETH/DATASCIENCECAPSTONEPROJECT/BLOB/MAIN/SPACEX_DASH_APP.PY

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas and split our data into training and test sets.
- Using logistic regression and GridSearchCV we obtained the best parameters and accuracy on validation data.
- 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.

TASK 12

Find the method performs best:

```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.7777777777777778
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```

- The link to the notebook is

https://github.com/DavidMorpeth/DataScienceCapstoneProject/blob/main/SpaceX%20Predictive.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

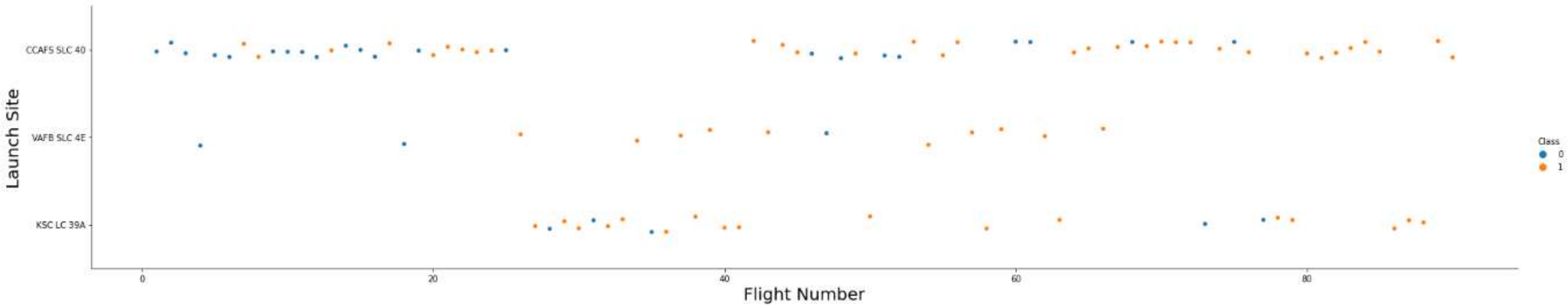
From the plot, we found that as the flight number increases the greater success and that there were far more launches at site CCAFS SLC 40

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'`

In [4]:
```
# 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(x="FlightNumber", y="LaunchSite", data=df, hue="Class", aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
```

Out[4]:
```
Text(23.199484374999997, 0.5, 'Launch Site')
```



Payload vs. Launch Site

For site CCAFS SLC 40 the greater the payload mass the higher the success.

Site VAFB SLC 4E there were no launches where payload mass exceeded 10000 kgs.

TASK 2: Visualize the relationship between Payload and Launch Site

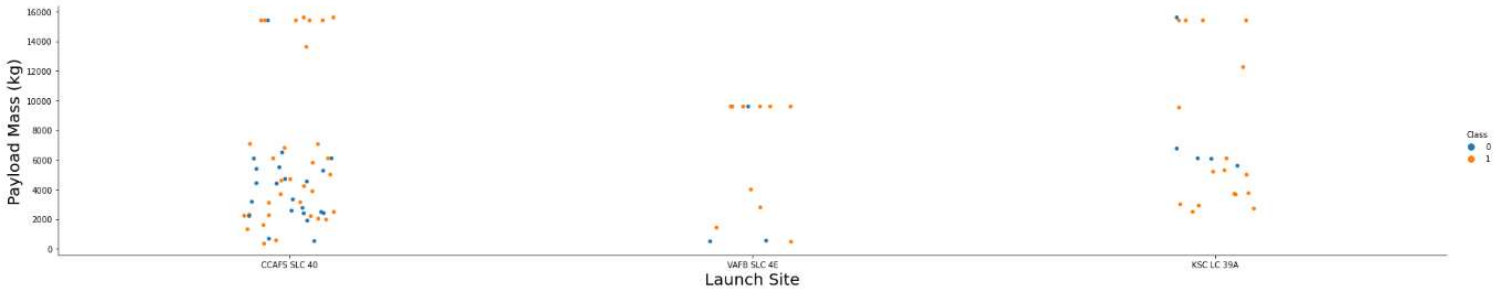
We also want to observe if there is any relationship between launch sites and their payload mass.

In [5]:

```
# 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.catplot(x="LaunchSite", y="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Launch Site", size=20)
plt.ylabel("Payload Mass (kg)", size=20)
```

Out[5]:

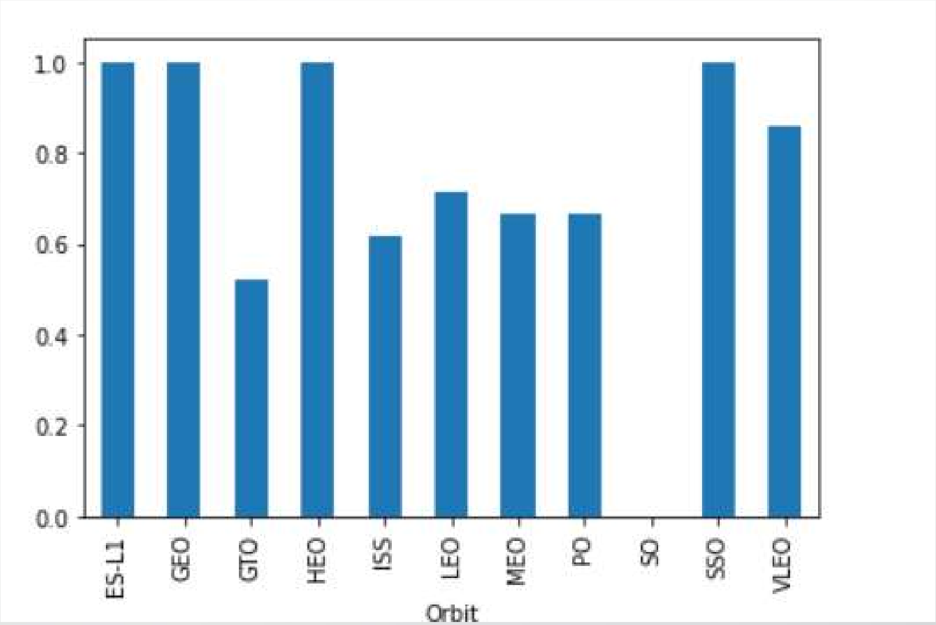
```
Text(22.299015624999996, 0.5, 'Payload Mass (kg)')
```



Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

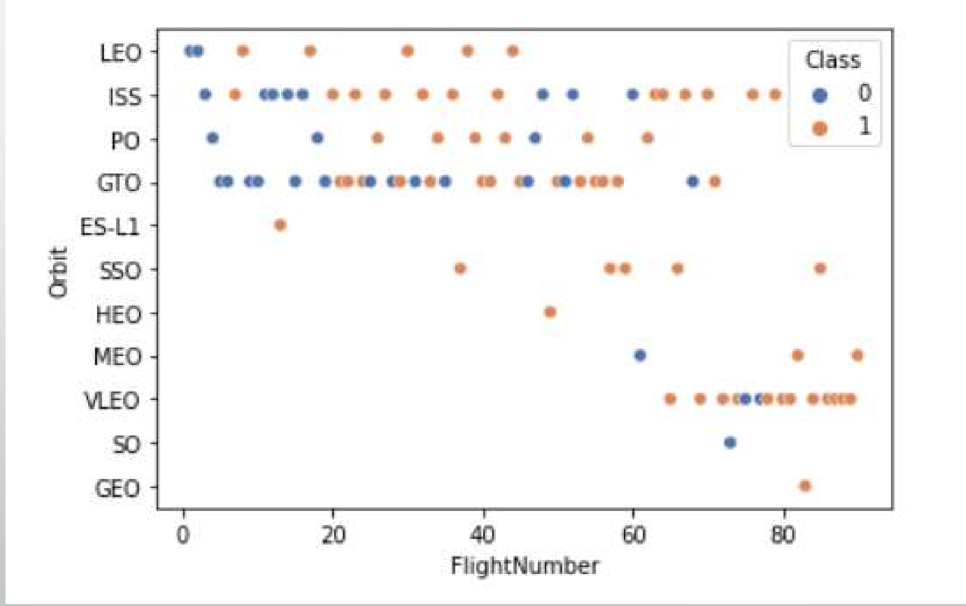
Success Rate vs. Orbit Type

- Sites ES-L1, GEO, HEO, SSO, VLEO had the most success rate with GTO having the lowest success rate.



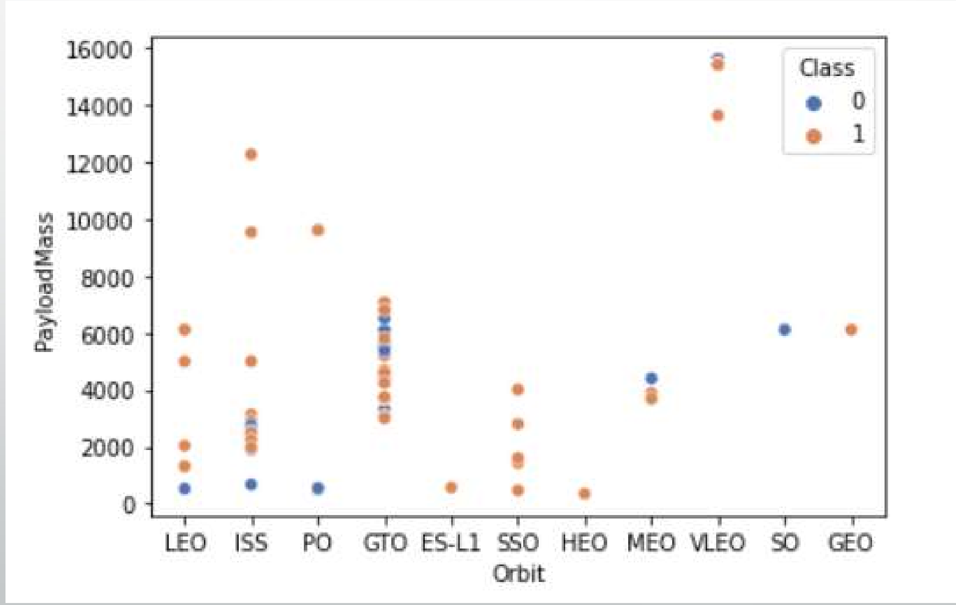
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type and their relationships. VLEO has a high success rate the higher the flight number where as LEO has low success the lower the flight number.
- GTO there appears to be no relationship.



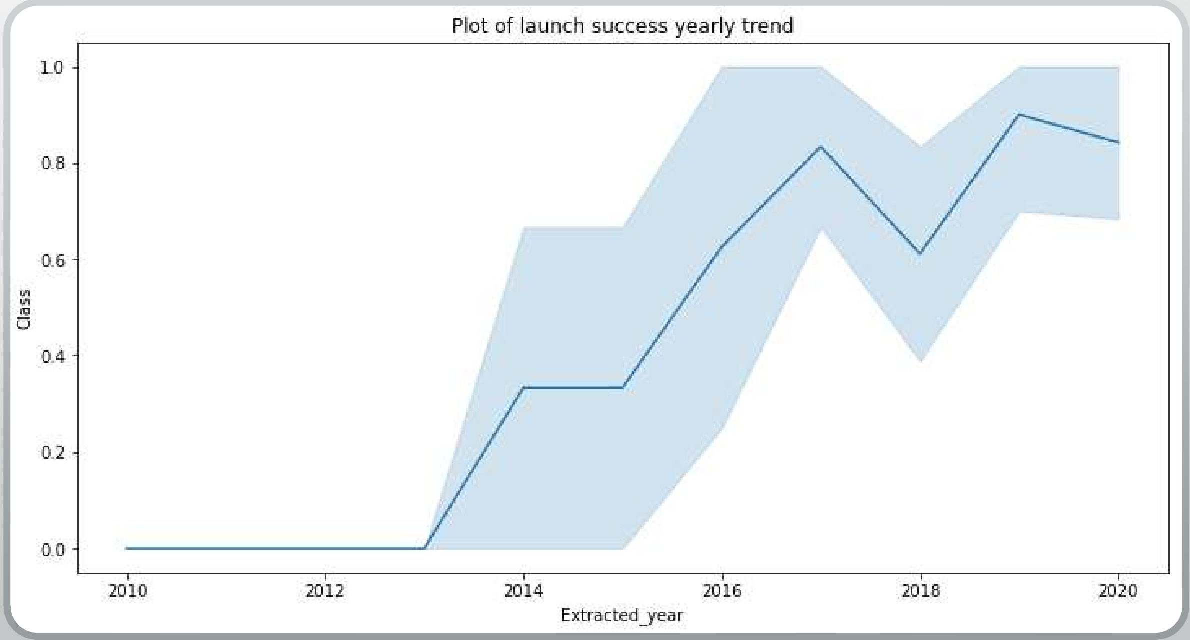
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- For GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there.



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.

Task 1

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

```
country = "Canada"
#%sql select * from INTERNATIONAL_STUDENT_TEST_SCORES where country = :country
%sql select distinct launch_site from spacex
```

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

- We used the query below to limit the results to 5 rows where the launch site begins with CCA.

Task 2

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

```
%sql select * from spacex where launch_site like 'CCA%' FETCH FIRST 5 ROWS ONLY
```

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We calculated the total payload mass in kgs to be 45596 where the customer was “NASA (CRS)”

Task 3

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

```
%sql select sum(payload_mass__kg_) as payload from spacex where customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

payload
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version “F9 v1.1” equal to 2928 kgs

Task 4

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

```
%sql select avg(payload_mass__kg_) as payload from spacex where booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

payload
2928

First Successful Ground Landing Date

- The first successful landing was on the 22 July 2018.

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 spacex where landing__outcome = 'Success'
```

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

1
2018-07-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- By filtering our dataset to payload mass between 4000 and 6000 kgs and successful landings we found there were only 4 booster versions.
- Using WHERE and < > OPERATORS we could apply this criteria to our dataset.

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 spacex where landing__outcome = 'Success (drone ship)' and payload_mass__kg_ > 4000 and payload_mass__kg_ <
```

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

- Using COUNT function and GROUP BY statement we are able to determine the total number of “Failure (in flight)”, “Success” and “Success (payload status unclear)” missions.
- Out of 101 mission outcomes 99 where a success.

Task 7

List the total number of successful and failure mission outcomes

```
%sql select mission_outcome, COUNT(*) from spacex GROUP BY mission_outcome
```

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- The below booster versions are those that carried the max payload mass in kgs.

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 spacex where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) FROM spacex)
```

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

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

2015 Launch Records

- Using WHERE clause and filtering using the YEAR function we are able to isolate those failed landings for the 2015 year.

Task 9

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

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

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

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

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

- Using COUNT and GROUP BY clause we can filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.

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(LANDING__OUTCOME) as count from spacex where date between '2010-06-04' and '2017-03-20' group by LANDING__OUTCOME
```

```
 * ibm_db_sa://lxz00614:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

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

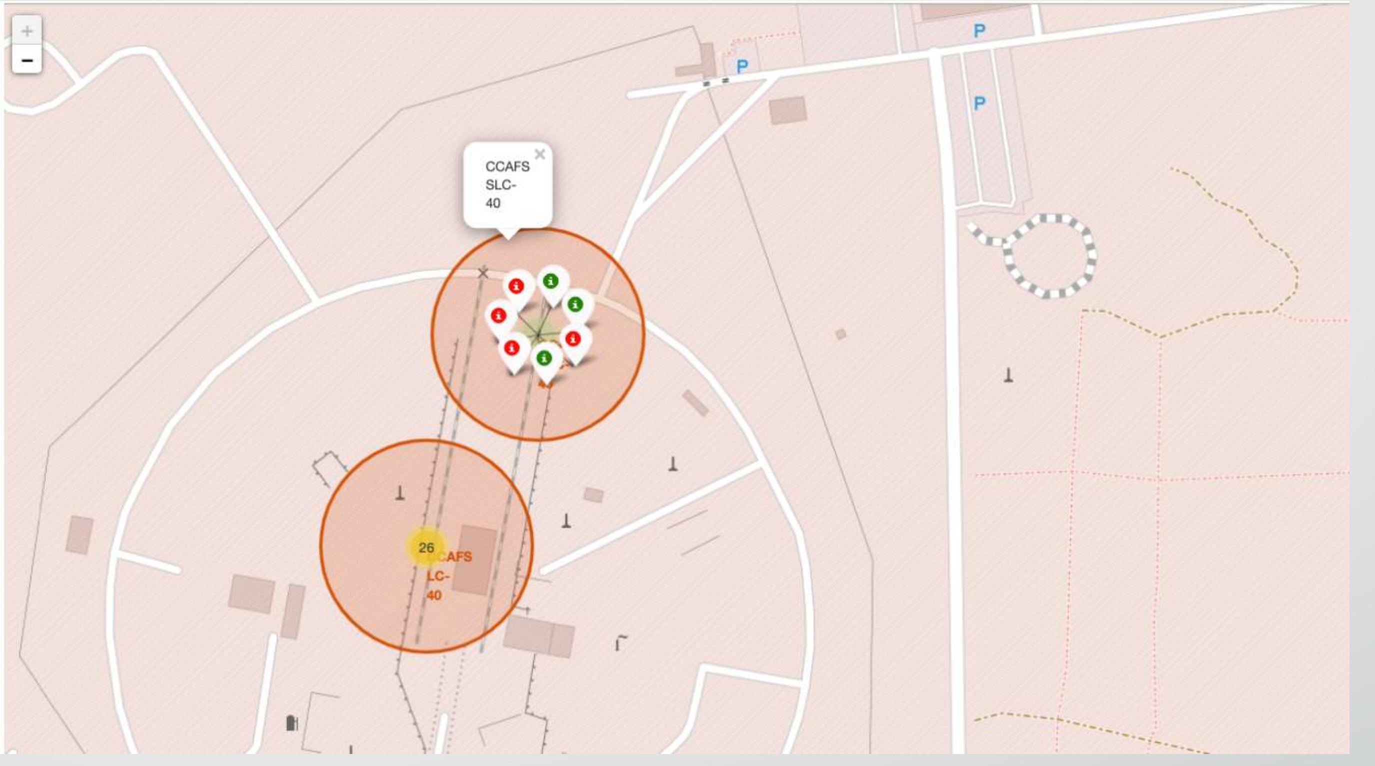
Section 4

Launch Sites Proximities Analysis

All launch sites global map markers

- All launch sites are concentrated on the coast in the United States of America

Markers showing launch sites with color labels



Launch Site distance to landmarks

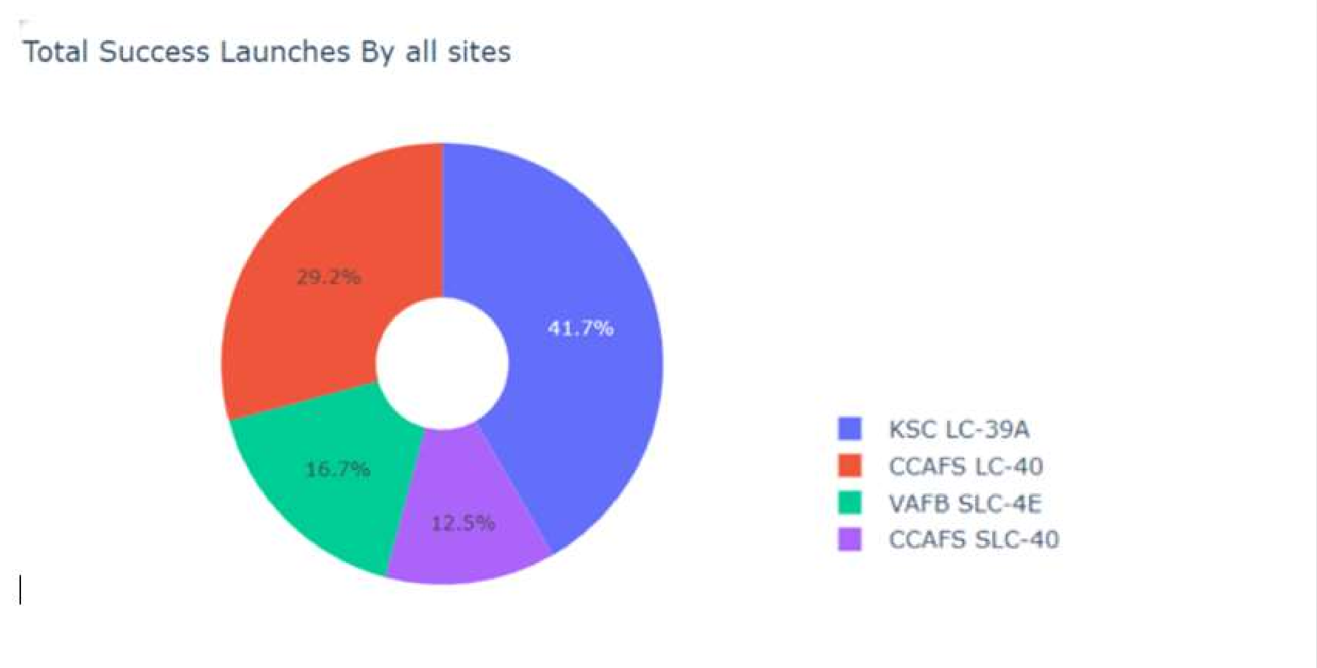
- Launch sites are close to coast.

Section 5

Build a Dashboard with Plotly Dash

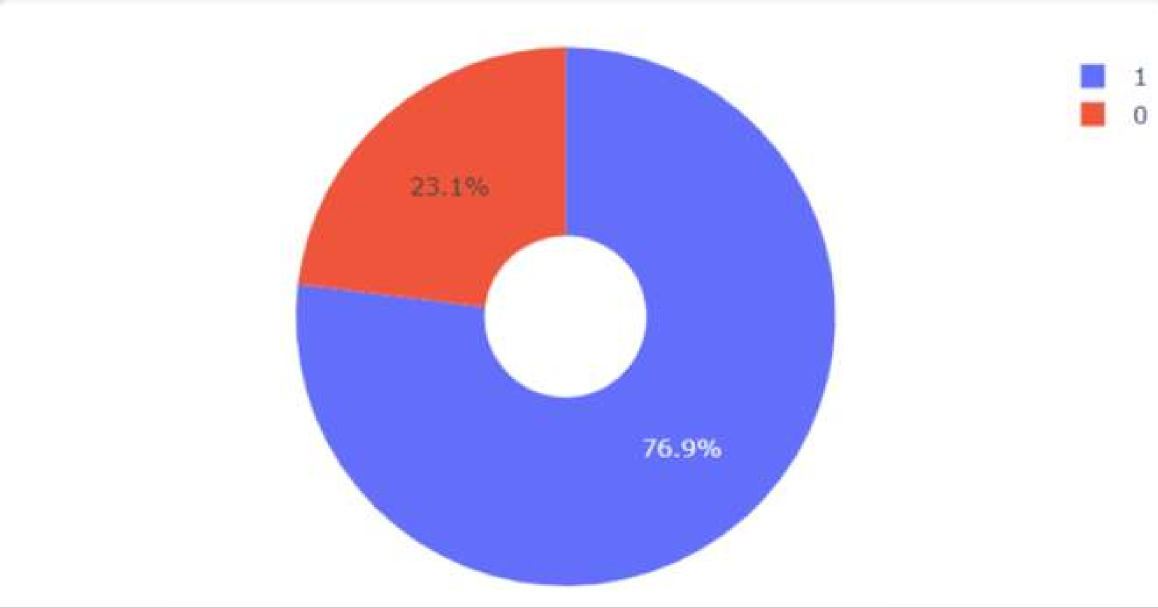
Pie chart showing the success percentage achieved by each launch site

- KSC LC-39A has the most successful launch sites.



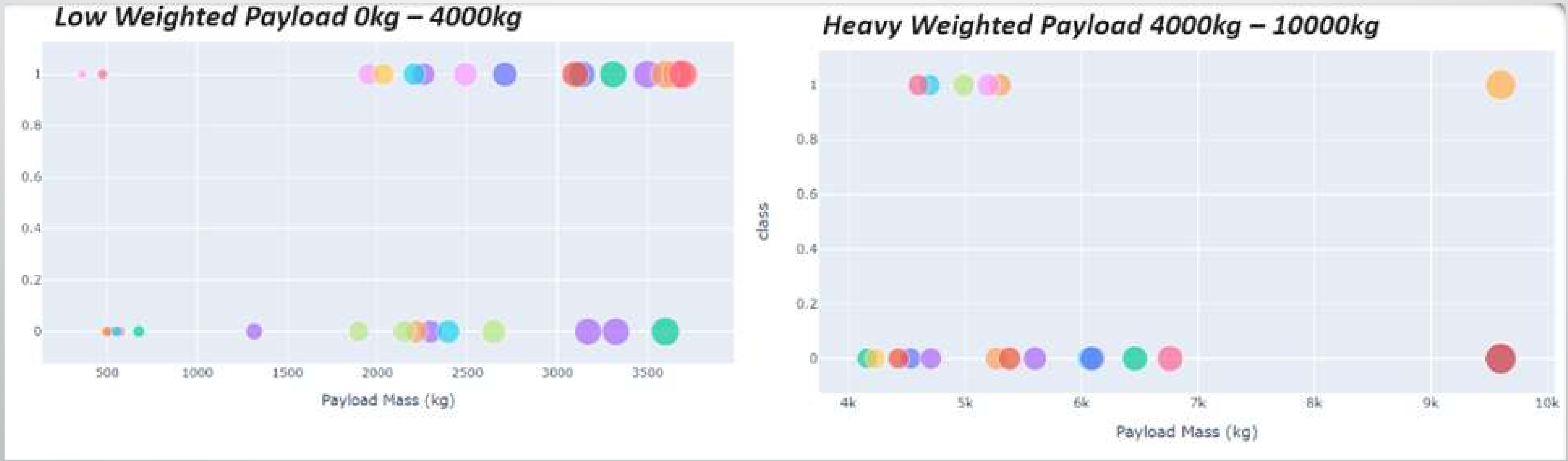
Pie chart showing the Launch site with the highest launch success ratio

- KSC LC-39A had a 76.9% success rate.



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

- Low weighted payloads had a higher success than Heave weighted.



Section 6

Predictive Analysis (Classification)

Classification Accuracy

- Decision Tree Classifier algorithm had the highest accuracy of ~.9036

TASK 8

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': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10,
'splitter': 'random'}
accuracy : 0.9035714285714287
```

Confusion Matrix

TASK 9

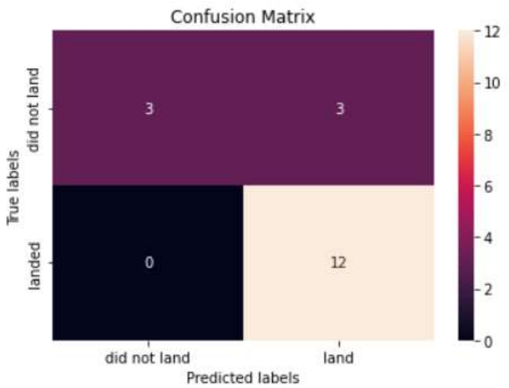
Calculate the accuracy of tree_cv on the test data using the method `score` :

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

```
0.7777777777777778
```

We can plot the 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.
- Issue with false positives predicted by the model.

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!