IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Misyura Sergey Alexandrovich
10.11.2022

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 Visualization
  - Interactive Visual Analytics with Folium
  - Interactive Visual Analytics and Dashboard
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive Visual Analytics results
  - Predictive Analytics results

Introduction

- Project background and context
  - SpaceX 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 SpaceX 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 SpaceX for a rocket launch. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch.
- Problems you want to find answers
  - Determine the success of the landing of the 1st stage
  - Determine what features affect the success
  - The relationship of features that determine landing success

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data Collection with Web Scraping from Wikipedia and SpaceX API
- Perform data wrangling:
  - Categorical features transformed with one-hot encoding, cast all numeric features to float64
- 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

- Data sets collected by using request to the SpaceX API:
  - Request and parse the SpaceX launch data using the GET request
  - Filter the dataframe to only include Falcon 9 launches
  - Dealing with Missing Values
- Also data collected by using web scraping from Wikipedia for Falcon 9 with BeautifulSoup:
  - Request the Falcon9 Launch Wiki page from its URL
  - Extracted all column/variable names from the HTML table header
  - Created a data frame by parsing the launch HTML tables

Data Collection – SpaceX API

- To collect this data:
  - used get request to the SpaceX API;
  - data filtered out to only include Falcon 9 launches;
  - Nan-values changed to mean values.

- Link to notebook.

Task 1: Request and parse the SpaceX launch data using the GET request

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

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_
```

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 using `.json_normalize()`

In [13]:
```
data = response.json()
data = pd.json_normalize(data)
```

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

In [35]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

In [36]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [40]:
```
# Calculate the mean value of PayloadMass column
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(data_falcon9['PayloadMass'].mean(), inplace=True)
data_falcon9.isnull().sum()
```

Data Collection - Scraping

- Collect data with scraping:
  - used get request to the Falcon9 page, created a BeautifulSoup object;
  - extracted all column/variable names;
  - created a data frame from HTML tables.
- [Link to notebook]()

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

Out[5]: 200

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

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

In [11]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
th_elements = first_launch_table.find_all('th')

# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for th in th_elements:
    name = extract_column_from_header(th)

# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
    if name is not None and len(name) > 0:
        column_names.append(name)
```

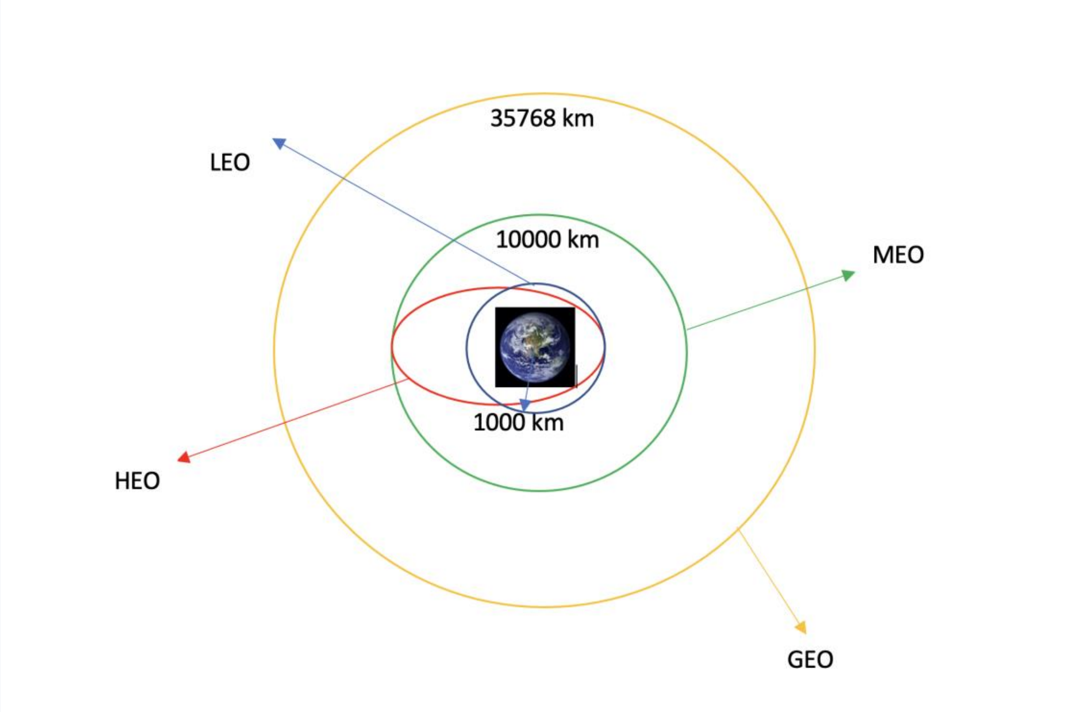
TASK 3: Create a data frame by parsing the launch HTML tables

In [51]:
```
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']=[]
```

Data Wrangling



- Calculated the number of launches on each site
- Calculated the number and occurrence of each orbit
- Calculated the number and occurrence of mission outcome per orbit type
- Created a landing outcome label

- Link to notebook

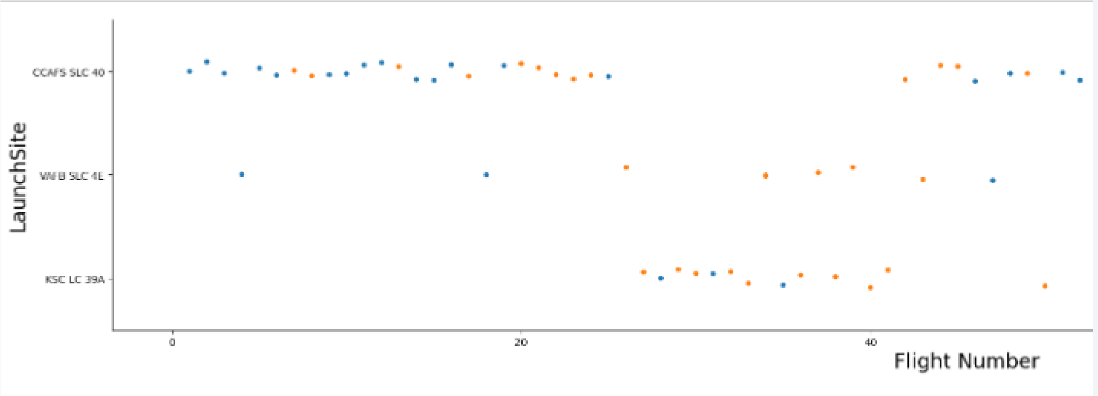
EDA with SQL

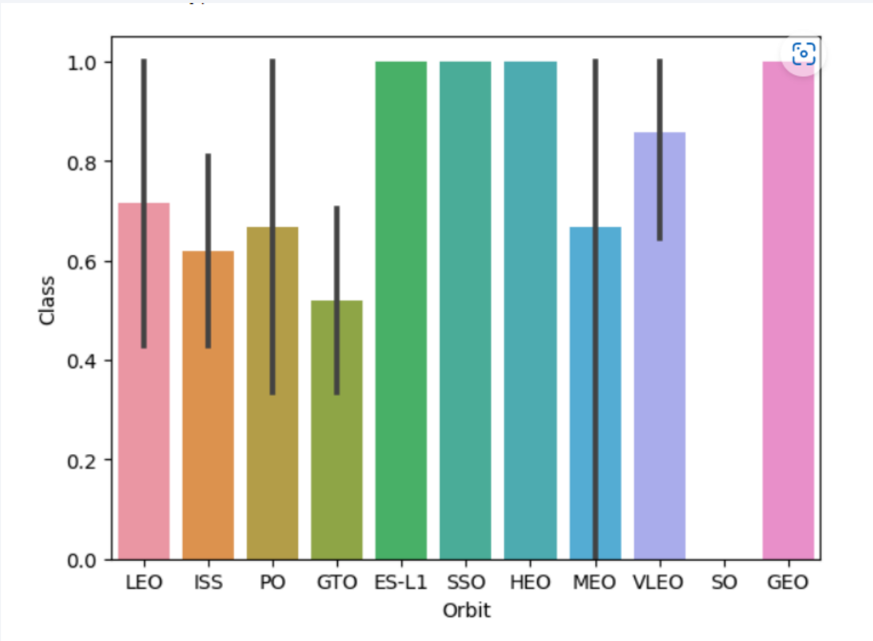
- SQL queries helped to find the following insights working directly with db2 via Jupyter:
  - The names of the unique launch sites.
  - The total payload mass carried by boosters launched by NASA (CRS).
  - The average payload mass carried by booster version F9 v1.1.
  - The date where the first successful landing outcome in drone ship was achieved.
  - The names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000.
  - The total number of successful and failure mission outcomes.
  - The names of the booster_versions which have carried the maximum payload mass.
- Link to notebook

EDA with Data Visualization

- To visualize the relationship between Class, Launch Site, Payload Mass, Flight Number scatter plots were built, reflecting every launch. The bar chart is built because grouping is needed to show the relationship between the probability of success and the type of orbit. Also built a line chart

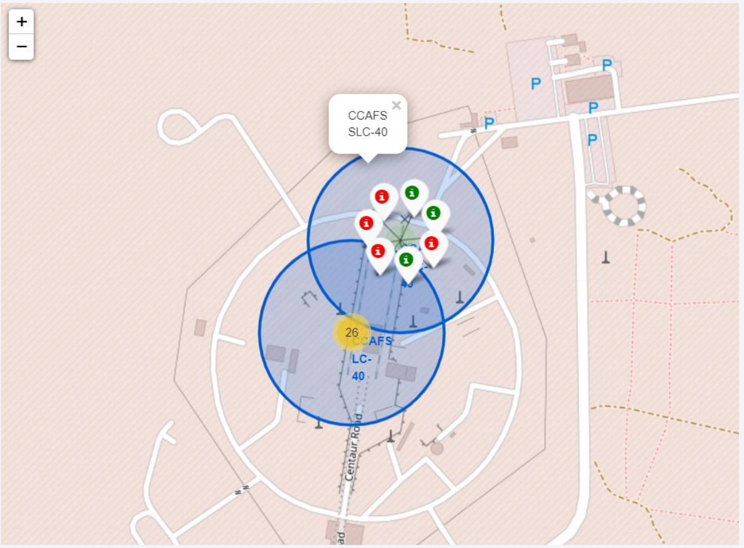
- [Link to notebook]()





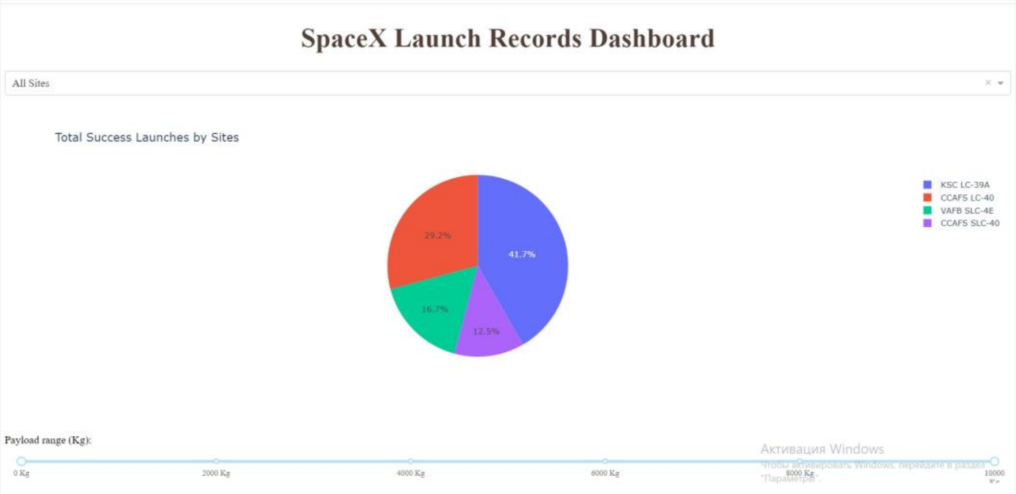
Build an Interactive Map with Folium

- On the interactive map have been created markers, circles, clusters and lines.
- Markers and circles added to indicate launch locations. Clusters added to show launches and their success, lines added to indicate the distance to the nearest infrastructure points.
- [Link to notebook]



Build a Dashboard with Plotly Dash

- To the dashboard were added:
  - Dropdown list to enable Launch Site selection.
  - Pie chart to show the total successful launches count for all sites.
  - Slider to select payload range.
  - Scatter chart to show the correlation between payload and launch success.
- [Link to dash app]()



Predictive Analysis (Classification)

- Prepared data has been loaded using numpy and pandas, standardized using preprocessing, split into training and test data.
- For data classification were used models: the logistic regression, support vector machine, decision tree classifier, k nearest neighbors. Hyperparameter fitting using GridSearchCV.
- The evaluation criterion was the R2-score parameter, as well as the error matrix. Best models – Logistic regression and KNN with score ~0.815.

- [Link to notebook]

```
print("Logistic regression accuracy :",logreg_cv.score(X_test, Y_test))
print("SVM accuracy :", svm_cv.score(X_test, Y_test))
print("Decision tree accuracy :",tree_cv.score(X_test, Y_test))
print("KNN accuracy :",knn_cv.score(X_test, Y_test))
```

```
Logistic regression accuracy : 0.8148148148148148
SVM accuracy : 0.7777777777777778
Decision tree accuracy : 0.7407407407407407
KNN accuracy : 0.8148148148148148
```

Results

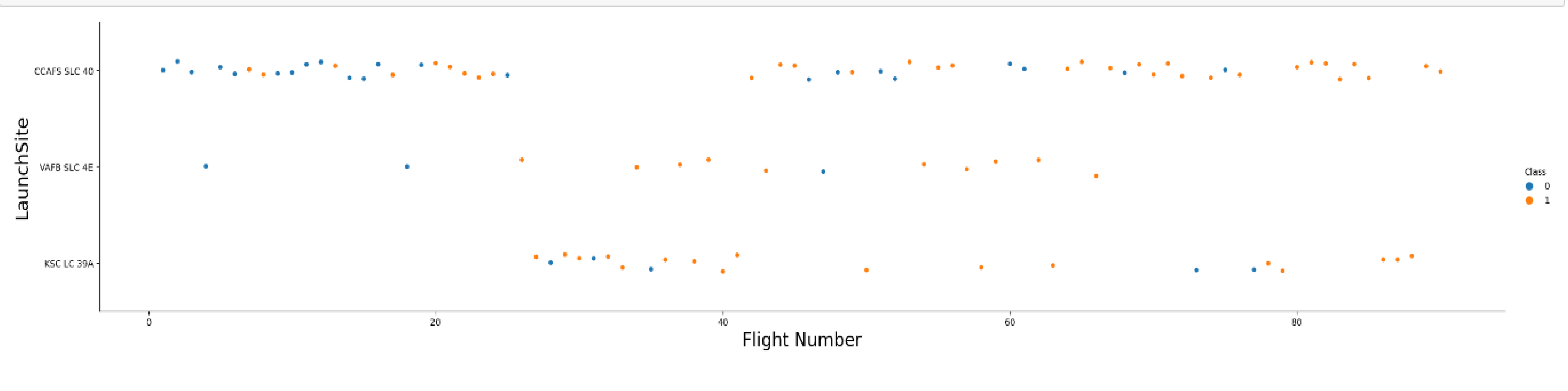
- Exploratory data analysis results (Section 2)
- Interactive analytics demo in screenshots (Section 3 and 4)
- Predictive analysis results (Section 5)

Section 2

Insights drawn from EDA

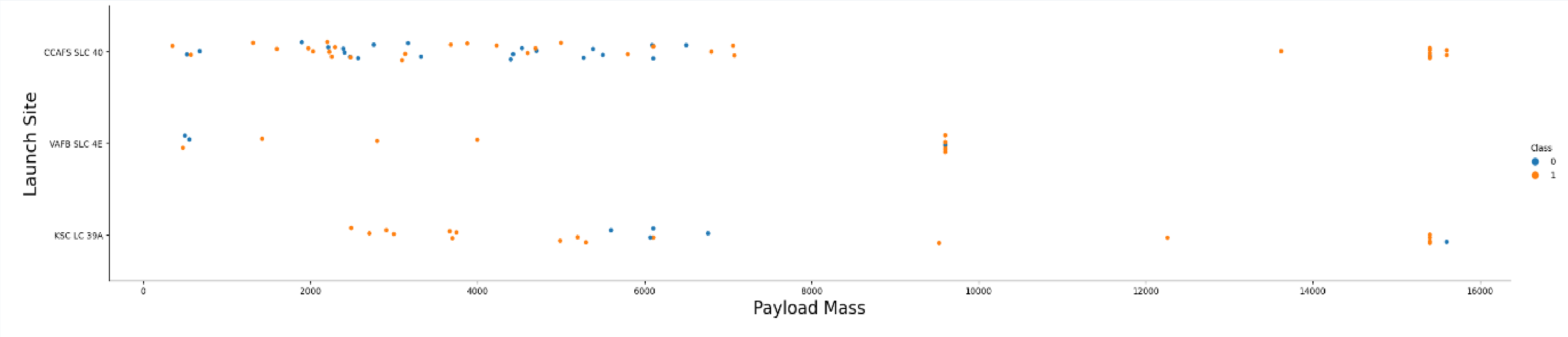
Flight Number vs. Launch Site

- From the graph we can see that with the increase in the number of launches from each platform, the percentage of successful landings for each platform increases.

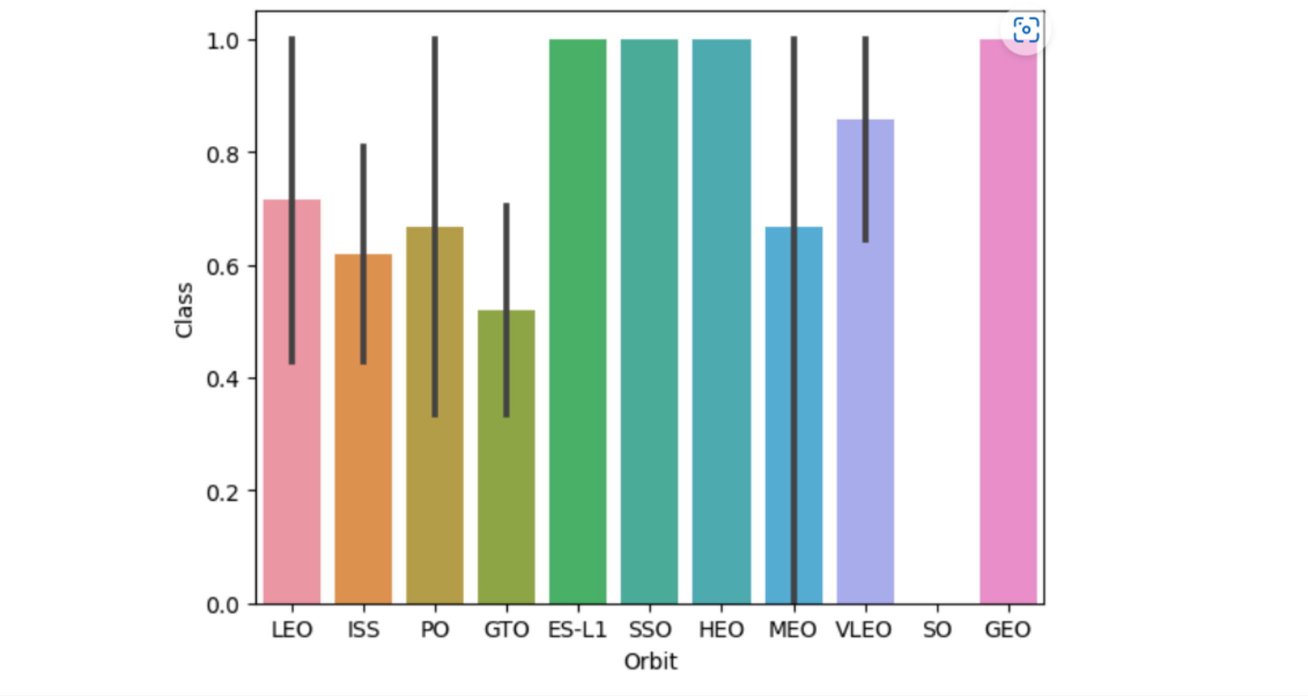


Payload vs. Launch Site

- From the graph we can see that for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000). Also launches from 9600 payload mass have a high success rate landing.



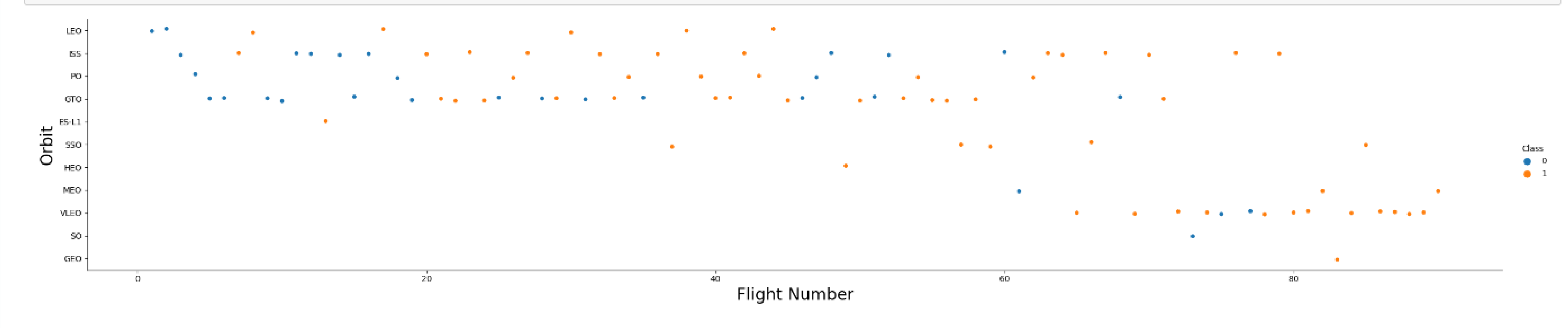
Success Rate vs. Orbit Type



- The diagram shows ES-L1, SSO, HEO, GEO orbits have high success rate.

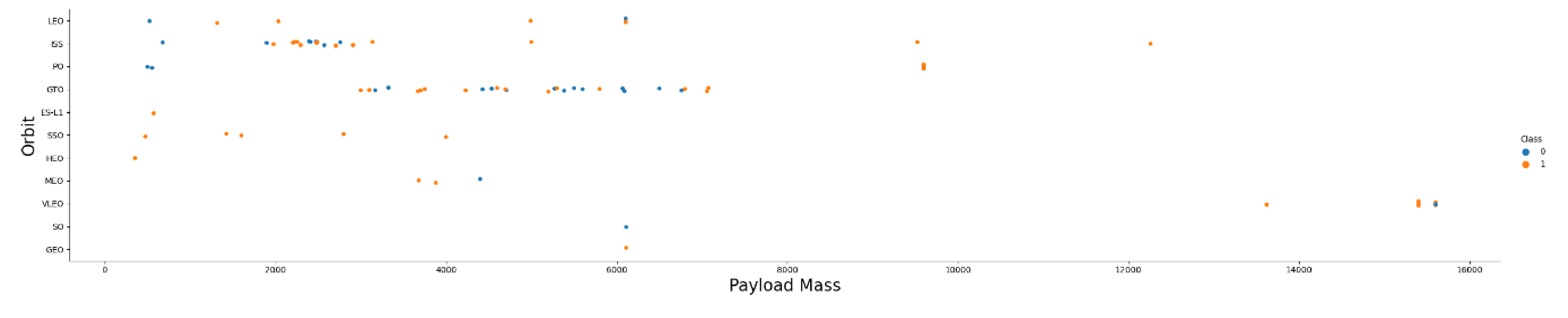
Flight Number vs. Orbit Type

- From the graph we can see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



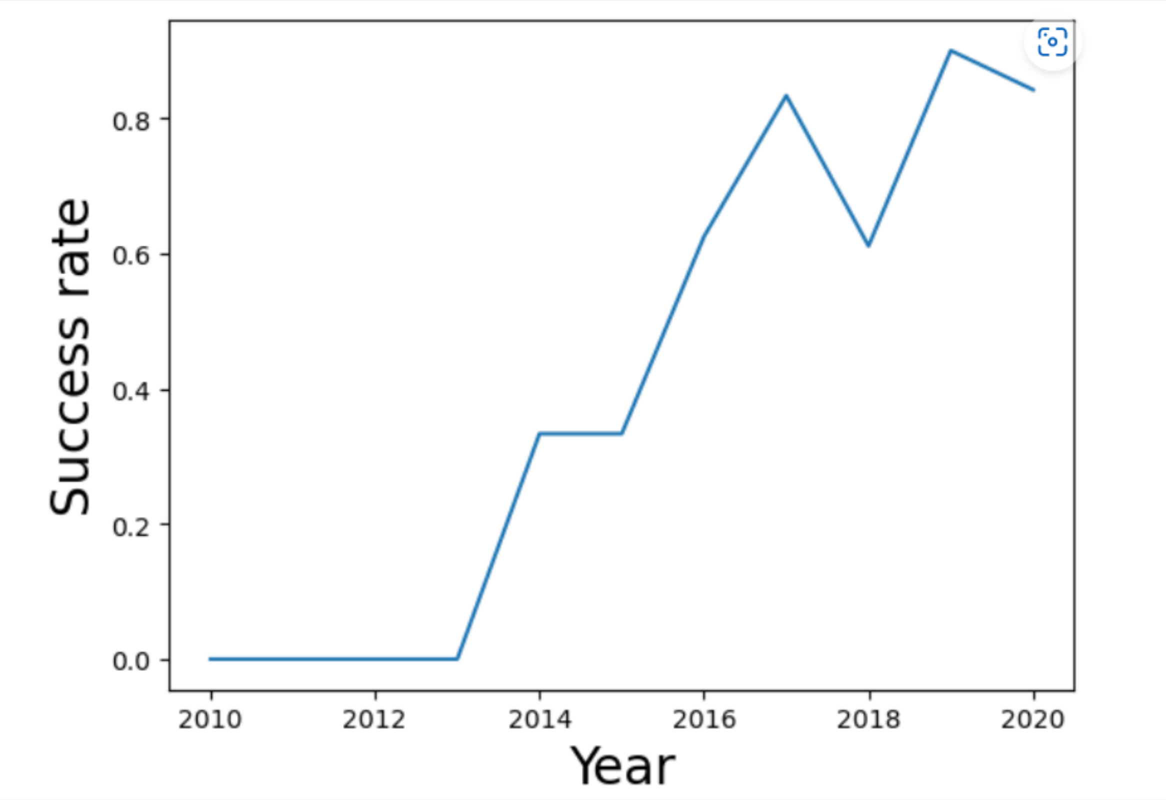
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However for GTO cannot distinguish this well as both positive landing rate and negative landing are both there here.



Launch Success Yearly Trend

- The line plot shows that the success rate since 2013 kept increasing till 2020



All Launch Site Names

- To find the names of the unique launch sites used keyword **DISTINCT**.

In [36]:
```
%sql select DISTINCT LAUNCH_SITE from SPACEXTBL;
```

Out[36]:

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

Launch Site Names Begin with 'KSC'

- To find 5 records where launch sites' names start with `KSC` used query with the condition **like** and **LIMIT**.

In [54]:
```
%sql select * from SPACEXTBL where LAUNCH_SITE like 'KSC%' LIMIT 5
```

Out[54]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass

- The total payload carried by boosters from NASA calculated with **SUM** keyword.

In [58]:
```
%sql select SUM(PAYLOAD_MASS__KG_) as Total_Payload_mass from SPACEXTBL where CUSTOMER like 'NASA%'
```

Out[58]:

total_payload_mass
99980

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 calculated with **AVG** keyword.

In [64]:
```
%sql select AVG(PAYLOAD_MASS__KG_) as average_payload_mass from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'
```

Out[64]:

average_payload_mass
2928

First Successful Drone ship Landing Date

- The date of the first successful landing outcome on drone ship received below.

In [66]:
```
%sql select MIN(DATE) as DATE from SPACEXTBL where LANDING__OUTCOME = 'Success (drone ship)'
```

Out[66]:

DATE
2016-04-08

Successful Ground Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on ground pad and had payload mass greater than 4000 but less than 6000 filtered with **WHERE** and **between** keywords.

In [70]:
```
%%sql select BOOSTER_VERSION, LANDING__OUTCOME, PAYLOAD_MASS__KG_ from SPACEXTBL
where LANDING__OUTCOME = 'Success (ground pad)' and PAYLOAD_MASS__KG_ between 4000 and 6000
```

Out[70]:

booster_version	landing__outcome	payload_mass__kg_
F9 FT B1032.1	Success (ground pad)	5300
F9 B4 B1040.1	Success (ground pad)	4990
F9 B4 B1043.1	Success (ground pad)	5000

Total Number of Successful and Failure Mission Outcomes

- To calculate the total number of successful and failure mission outcomes used **CROSS JOIN** query.

In [102]:
```
%%sql select * from
(select count(mission_outcome) as Success from SPACEXTBL where mission_outcome like 'Success%') CROSS JOIN
(select count(mission_outcome) as Failure from SPACEXTBL where mission_outcome like 'Failure%')
```

Out[102]:

success	failure
100	1

Boosters Carried Maximum Payload

- Subquery used to display the list the names of the booster which have carried the maximum payload mass.

```
In [112]: %%sql select BOOSTER_VERSION, PAYLOAD_MASS__KG_ from SPACEXTBL
          where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

Out[112]:

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- To display the list of records with month names, successful landing outcomes in ground pad, booster versions, launch site for the months in year 2017, data filtered and used **MONTHNAME** keyword.

In [135]:
```
%%sql select MONTHNAME(DATE), LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE from SPACEXTBL
where (LANDING__OUTCOME like 'Success (ground pad)') and (DATE LIKE '2017%')
```

Out[135]:

1	landing__outcome	booster_version	launch_site
February	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
May	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
June	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
August	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
September	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
December	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

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

- The count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 ranked in descending order with keywords **COUNT, WHERE, and, between, GROUP BY, ORDER BY, DESC**.

In [150]:
```
%%sql select LANDING__OUTCOME, COUNT(LANDING__OUTCOME) as Count from SPACEXTBL
where (LANDING__OUTCOME like 'Success%') and (DATE between '2010-06-04' and '2017-03-20 ')
group by LANDING__OUTCOME ORDER BY Count DESC
```

Out[150]:

landing__outcome	COUNT
Success (drone ship)	5
Success (ground pad)	3

Section 3

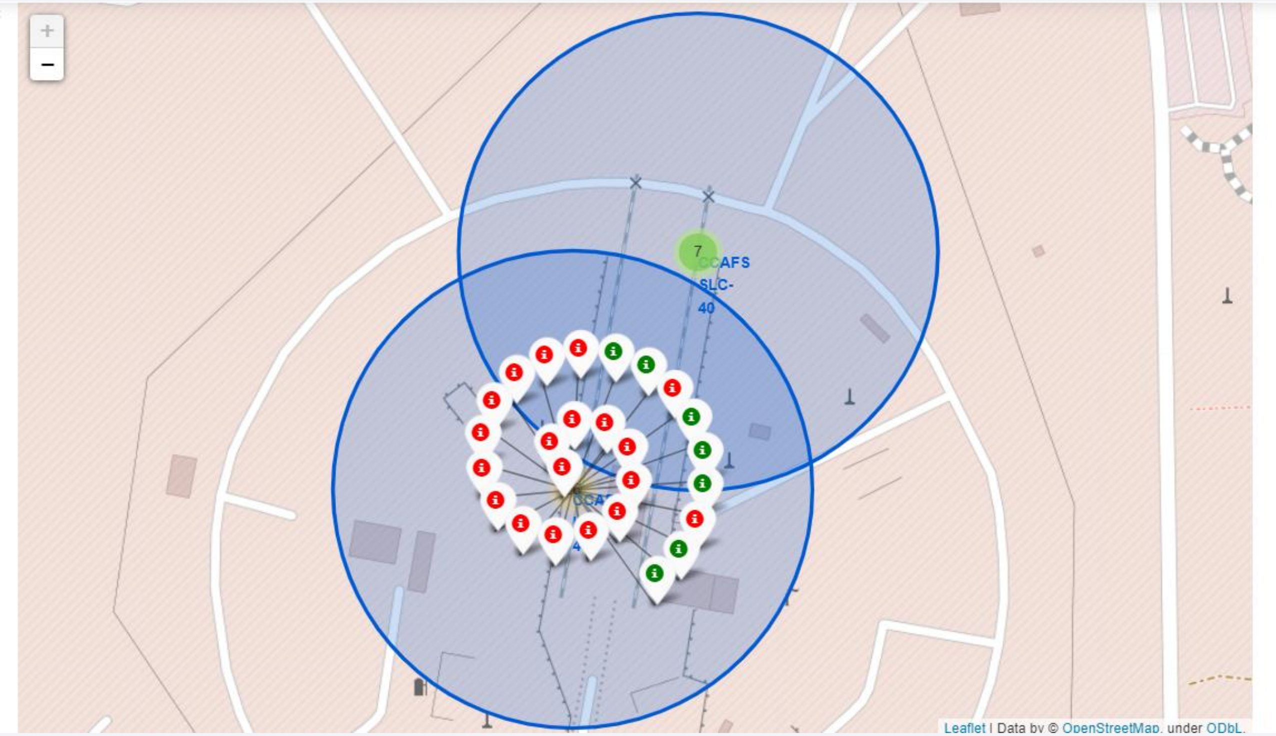
Launch Sites Proximities Analysis

All launch sites on the map

- Launch locations are on the west and east coasts.

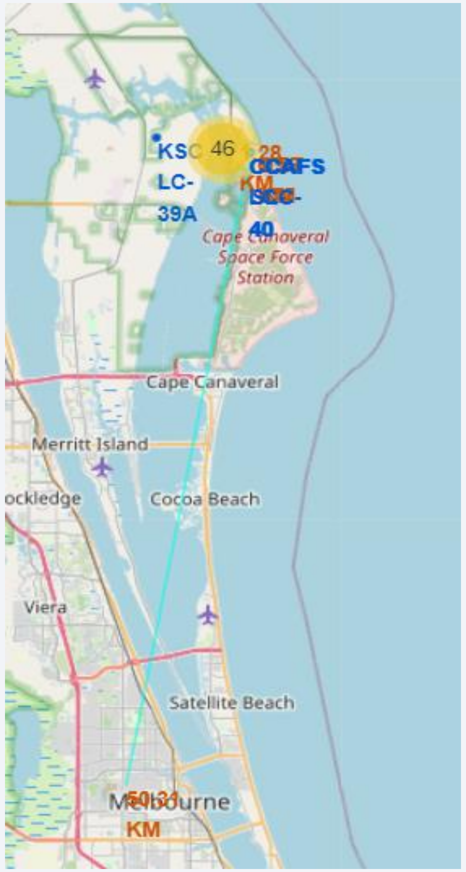
Marked launches

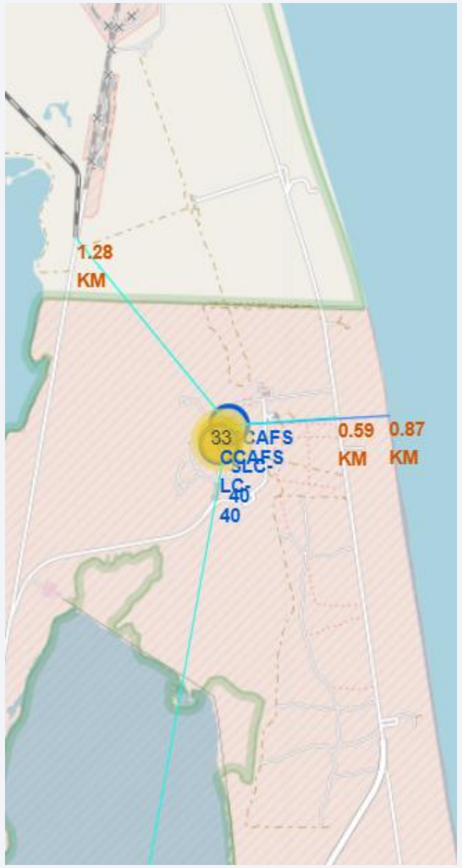
- All launches were marked on the map: green - successful, red - unsuccessful. Indicated the total number of launches for the platform



Launch site and proximities

- The lines show the distance from the platform to its proximities, such as city Melbourne, NASA Railroad, Samuel C Phillips Pkwy.



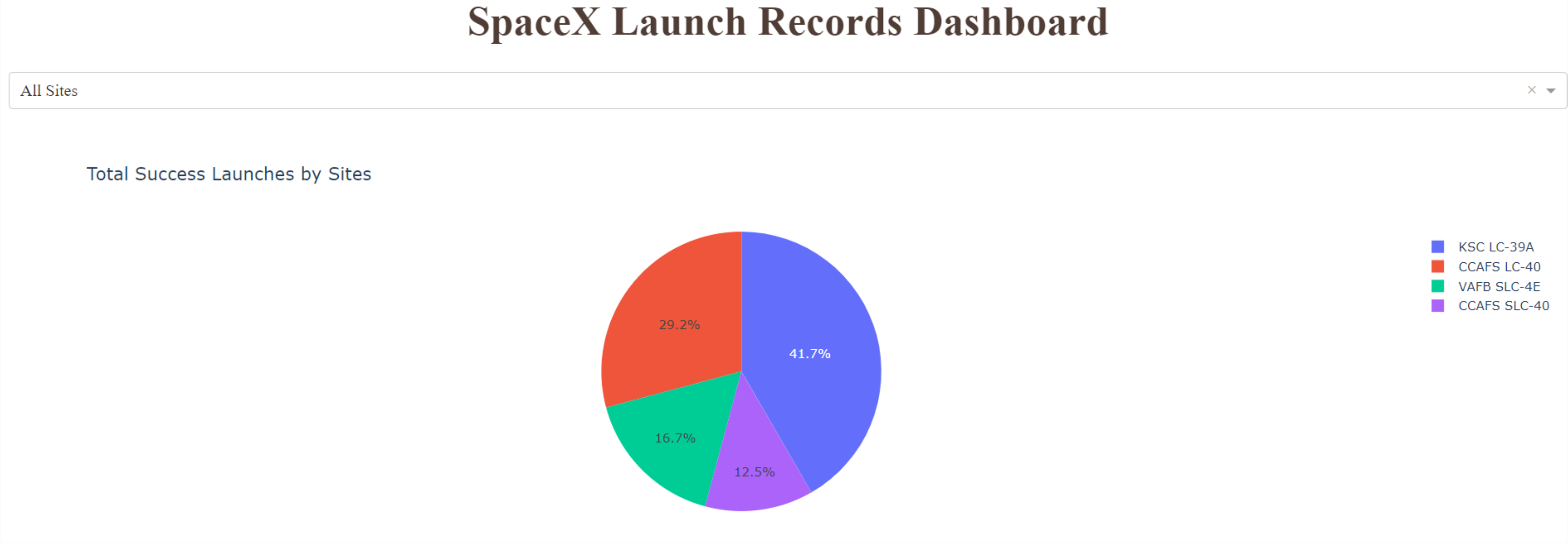


Section 4

Build a Dashboard with Plotly Dash

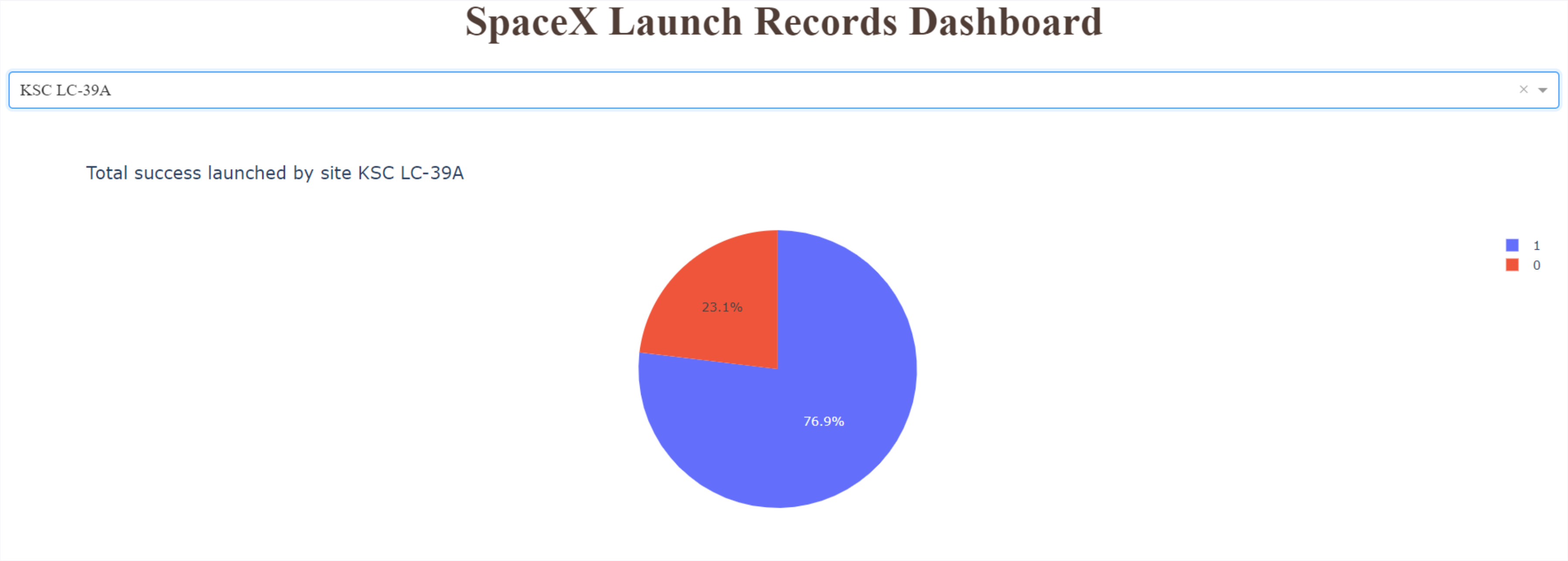
Pie chart of successful launches

- The chart shows that KSC LC-39A has the most successful launches.

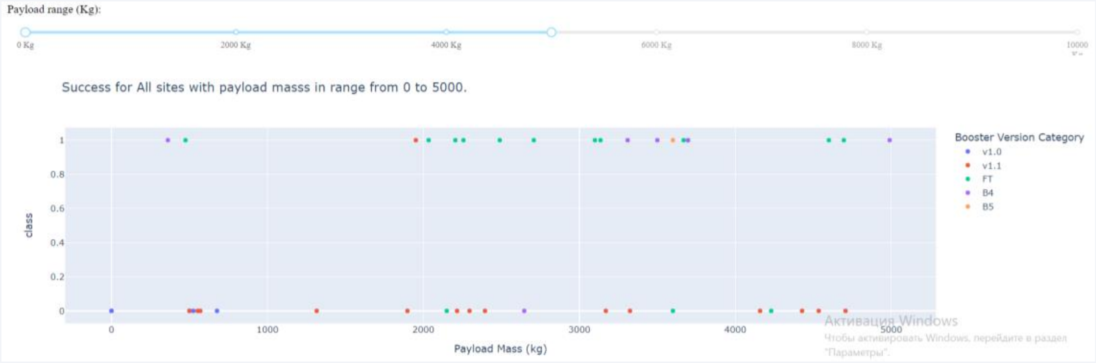


Success ratio

- Highest launch success ratio has KSC LC-39A.



Scatter plot of Payload and Launch Outcome



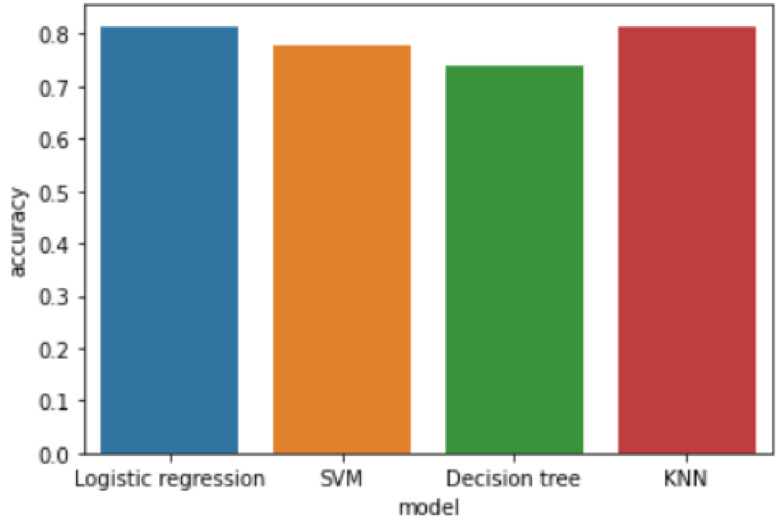
- Scatter plots of Payload and Launch Outcome for All sites in payload mass ranges 0kg – 5000kg and 5000kg – 10000kg.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

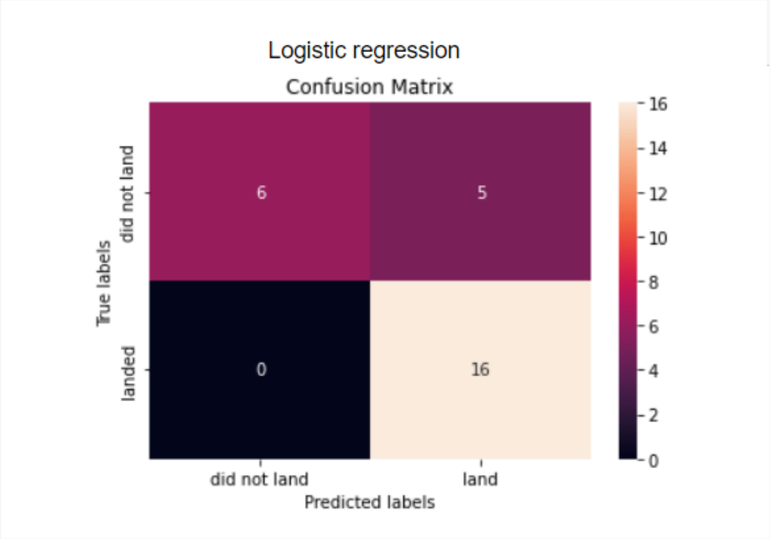
- The best models for landing prediction are Logistic regression and KNN with accuracy ~0,815.

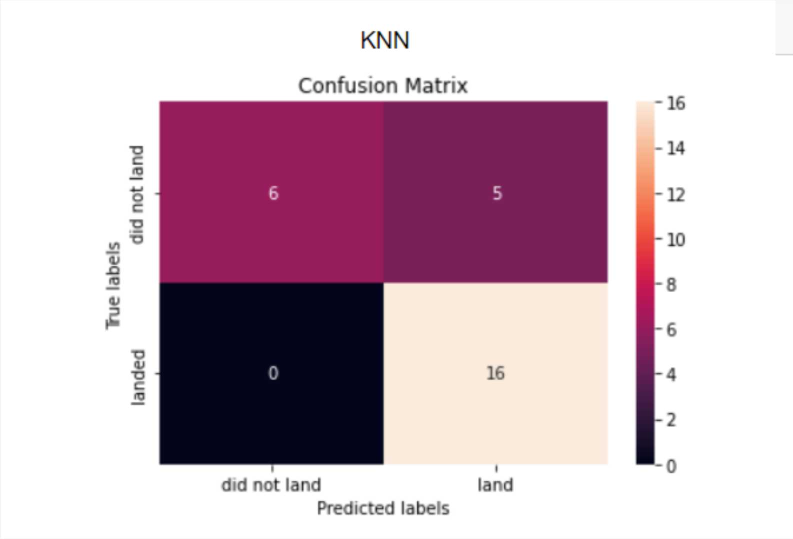



```
Logistic regression accuracy : 0.8148148148148148
SVM accuracy : 0.7777777777777778
Decision tree accuracy : 0.7407407407407407
KNN accuracy : 0.8148148148148148
```

Confusion Matrix

- The confusion matrices are identical for logistic regression and KNN. The major problem is false positives landings.





Conclusions

- With the increase in the number of launches from each platform, the percentage of successful landings for each platform increases.
- ES-L1, SSO, HEO, GEO orbits have high success rate.
- The success rate since 2013 kept increasing till 2020.
- KSC LC-39A has the most successful launches and highest launch success ratio.
- The best models for landing prediction are Logistic regression and KNN.

Appendix

- I find it convenient to use SQL magic commands %sql, %%sql in this project.
- All resources like Python code, SQL queries, charts, Notebook outputs, data sets are at the link https://github.com/Sergey-Misyura/IBM-Data-Science/tree/main/10.%20Data%20Science%20and%20Machine%20Learning%20Capstone%20Project

Thank you!