IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Thi Anh Hong Tran
06th February 2022

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- This project will explore successful landing of rocket launch in aero space industry. SpaceX advertises the Falcon 9 rocket on its website at a much lower cost of 62 millions compared to the cost of 165 millions from other providers. Therefore, understanding how SpaceX can achieve this cost efficiency is crucial to break into this market and bid against SpaceX. Indeed, much of the savings is because SpaceX can reuse the first stage. As a result, if we can determine if the first stage will land successfully, we can determine the cost of a overall launch and have appropriate strategic actions to optimize upon.
- We will use data analysis, visualization and relevant statistical model(s) to answer our main problem – i.e. to predict whether the Falcon 9 first stage will land successfully.

Section 1

Methodology

Methodology

Executive Summary

- **Data collection methodology:** *The data is initially extracted from SpaceX API of launches in the past, then using pre-defined helper functions to transform into a data frame containing relevant data attributes. Finally, the data is filtered to only include Falcon 9 launches which is the type of rocket (i.e. booster version) that the analysis will focus on.*
- **Perform data wrangling:** *We identified that only Payload Mass and Landing Pad attributes have missing information. One of the approaches is to replace missing Payload Mass with the average payload mass value from our dataset. On the other hand, Landing Pad will retain the None value to represent the launches where landing pads were not used.*
- **Perform exploratory data analysis (EDA) using visualization and SQL**
- **Perform interactive visual analytics using Folium and Plotly Dash**
- **Perform predictive analysis using classification models:** *We use relevant model of classification problems (namely logistic regression, SVM, decision tree, etc.) to predict the model performance on training versus testing set. Each model is then find tuned with the Grid Search technique to identify the best parameters to be set within the model. Finally, we evaluate and choose the most accurate model based on comparison of predicted values versus the actual values (i.e. so-called confusion matrix) as well as the performance score on testing dataset.*

Data Collection

- The data of past launches can be extracted from the public SpaceX API URL using GET request. We then decode the response content as a Json and normalize it into a Pandas dataframe. To make the Json result more consistent, the analysis will use static response object from this link.
- Only relevant information such as rocket, payloads, launch pad, cores needs to be kept in the dataframe (i.e. taking a subset of data). Then using defined helper functions, a new dataframe is created with more detailed columns around launch outcome (our target variable) and the independent variables (namely, booster version, payload mass, launch site, flight, etc.)

Data Collection – SpaceX API

- Github URL [here]

1. Make a GET request to the API URL

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

response = requests.get(spacex_url)
```

2. Decode the request response as json and normalize into a Pandas dataframe

```
df = pd.json_normalize(response.json())
```

3. Call this dataframe as 'Data', keep only relevant details and apply helper function to transform into a new dataframe

```
data = df
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4. Combine the columns into a dictionary called 'launch_dict', then create a dataframe from this

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5. Filter out to get only Falcon 9 booster rocket version and reset index. We get the final dataframe called 'data_falcon9'

```
data_falcon9 = data1[data1['BoosterVersion'] != 'Falcon 1']
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Data Collection - Scraping

- Extract a Falcon 9 launch records HTML table from Wikipage (use static url for consistent results through the analysis)
- Github URL here

1. Make a GET request to the HTML page, assign response to an object

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=10276
```

```
falcon9_response = requests.get(static_url)
html_page = falcon9_response.content
```

2. Create a BeautifulSoup object from the HTML response

```
soup = BeautifulSoup(html_page, 'html.parser')
```

3. Extract all column/ variable names from the HTML table header

```
html_tables = soup.find_all('table')
html_tables
```

```
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
headers = first_launch_table.find_all('th')
for header in headers:
    column_name = extract_column_from_header(header)
    if column_name is not None and len(column_name) > 0:
        column_names.append(column_name)
```

```
print(column_names)
```

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

4. Create an empty dictionary with keys from the extracted column names

```
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']=[]
```

5. Fill up the launch_dict with launch records extracted from table rows. Then we can create a dataframe from it

```
df=pd.DataFrame(launch_dict)
```

Data Wrangling

- This task aims to find some patterns in the data and determine the Training labels (1 for successful landing, 0 for unsuccessful landing) for training supervised models
- Github URL here

1. Identify columns with missing values, whether numerical vs. categorical type

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

```
df.dtypes
```

2. Calculate the number of launches on each site

```
df['LaunchSite'].value_counts()
```

3. Calculate the number and occurrence of each orbit (each launch aims to a dedicated orbit, e.g Low Earch Orbit (LEO), Geosynchronous orbit (GTO), Highly elliptical orbit (HEO), etc.)

```
df['Orbit'].value_counts()
```

4. Calculate the number and occurrence of mission outcome per orbit type

```
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
]: True ASDS      41
   None None      19
   True RTLS      14
   False ASDS      6
   True Ocean      5
   False Ocean     2
   None ASDS       2
   False RTLS      1
   Name: Outcome, dtype: int64
```

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

5. Create a landing outcome label from the Outcome column and call it as 'Class'

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for i, outcome in enumerate(df['Outcome']):
    if outcome in bad_outcomes:
        landing_class.append('0')
    else:
        landing_class.append('1')

print(landing_class)
```

```
df['Class']=landing_class
```

EDA with Data Visualization

- Flight Number, Launch Site, Orbit and Payload Mass are key variables that could have impact on the success of first launch. In order to visualize these relationship, we plot the below graphs including the yearly trend of success rate:

    1. *Plot of Flight Number vs Payload across the outcomes of launch*
    2. *Plot of Flight Number vs Launch Site across the outcomes of launch*
    3. *Plot of Payload vs Launch Site across the outcomes of launch*
    4. *Bar plot of success launch rate across Orbit types*
    5. *Plot of Orbit vs Flight Number across the outcomes of launch*
    6. *Plot of Orbit vs Payload across the outcomes of launch*
    7. *Trending plot of yearly launch success rate*

- Github URL here

EDA with SQL

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

```
%sql select distinct LAUNCH_SITE from SPACEXTBL;
```

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

```
%sql select * from SPACEXTBL where LAUNCH_SITE like 'CCA%' LIMIT 5;
```

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

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

- 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%';
```

- List the date when the first successful landing outcome in ground pad was achieved:

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

- 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_ between 4000 and 6000;
```

- List the total number of successful and failure mission outcomes:

```
%sql select count(*), MISSION_OUTCOME from SPACEXTBL group by MISSION_OUTCOME;
```

- List the names of the booster_versions which have carried the maximum payload mass:

```
%sql select distinct BOOSTER_VERSION from SPACEXTBL where (select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

- 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 SPACEXTBL where year(DATE) = 2015;
```

- Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order

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

- Github link here

Build an Interactive Map with Folium

- Create ‘spacex_df’ dataframe to store Launch Site and their Latitude and Longitude coordinates
- Initiate ‘site_map’ map object using folium.Map with nasa cooridinate, then add ‘circle’ and ‘marker’ objects using folium.Circle, folium.map.Marker respectively to add popup label and icon showing the site’s name

```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label
launch_sites = list(launch_sites_df['Launch Site'])
latitudes = list(launch_sites_df['Lat'])
longitudes = list(launch_sites_df['Long'])

for launch_site, lat, long in zip(launch_sites, latitudes, longitudes):
    circle = folium.Circle([lat, long], radius=1000, color='#000000', fill=True).add_child(folium.Popup(launch_site))
    marker = folium.map.Marker([lat, long], icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % launch_site, ))
    site_map.add_child(circle)
    site_map.add_child(marker)

site_map
```

- Define a function to assign market colour as green if a launch was successful (class=1), or red if the launch was not successful (class=0). The use a MarketCluster object and add market to indicate colour of each launch record in the map

```
marker_cluster = MarkerCluster()
def assign_marker_color(launch_outcome):
    if launch_outcome == 1:
        return 'green'
    else:
        return 'red'

spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)
```

```
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)
    marker = folium.Marker(location=[record['Lat'], record['Long']], icon=folium.Icon(color='white', icon_color=record['marker_color']), popup = record['Launch Site'])
    marker_cluster.add_child(marker)

site_map
```

- Define a method to calculate distance between 2 points , then for each launch site, provide coordinates and add marker for its nearest coastline point, railway, highway and city alongside with the distance being returned from the calculation.
- Github link here.

Build a Dashboard with Plotly Dash

- We plotted a dashboard including 2 elements: (1) a pie chart of total success launches across all sites or within a selected site; and (2) a plot showing correlation between payload and launch outcomes across different rocket boosters
- The dashboard includes two interactions: (1) the drop down selection to select a specific launch site to drill down into details of that launch site only versus the default option is 'All sites'; and (2) the slider of payload range to restrict the payload data to shown in the 2nd plot of payload mass and launch outcome.
- These 2 graphs and 2 interactions in the dashboard help to visualize firstly, which site has the best successful rate, which payload mass range and booster version(s) tend to correlate with a successful launch. The interaction helps us visualize easily by reflecting the detailed data in the same dashboard and drill into the specific site/ payload range parameters.
- Github link here.

Predictive Analysis (Classification)

- Since this is classification problem, we will use methods such as Logistic Regression, Support Vector Machine, Decision Tree and K Nearest Neighbors algorithm.

1. Store the data in dataframe called ‘X’, and the predicted variable ‘Class’ in an array ‘Y’. Then standardizing the data using ‘preprocessing.StandardScaler()’ process from sklearn

```
transform = preprocessing.StandardScaler()
```

```
X = transform.fit_transform(X)
```

2. Split the data into training set (80%) and testing set (20%) using train_test_split

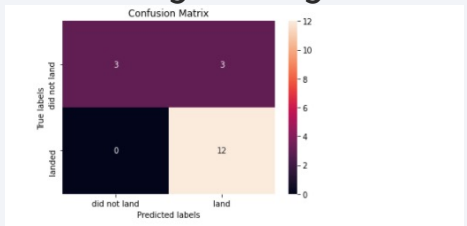
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

3. For each type of algorithm, we create an initial object with certain parameters. Then create a GridSearchCV object to find the best parameters (i.e. model fine-tuning) when fitting with training data. Then calculate the accuracy score of the fine-tuned model on testing data, and display in the confusing matrix. Example below for Logistic Regression:

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

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

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



4. Find the method which performs best (i.e. having highest score on test data)

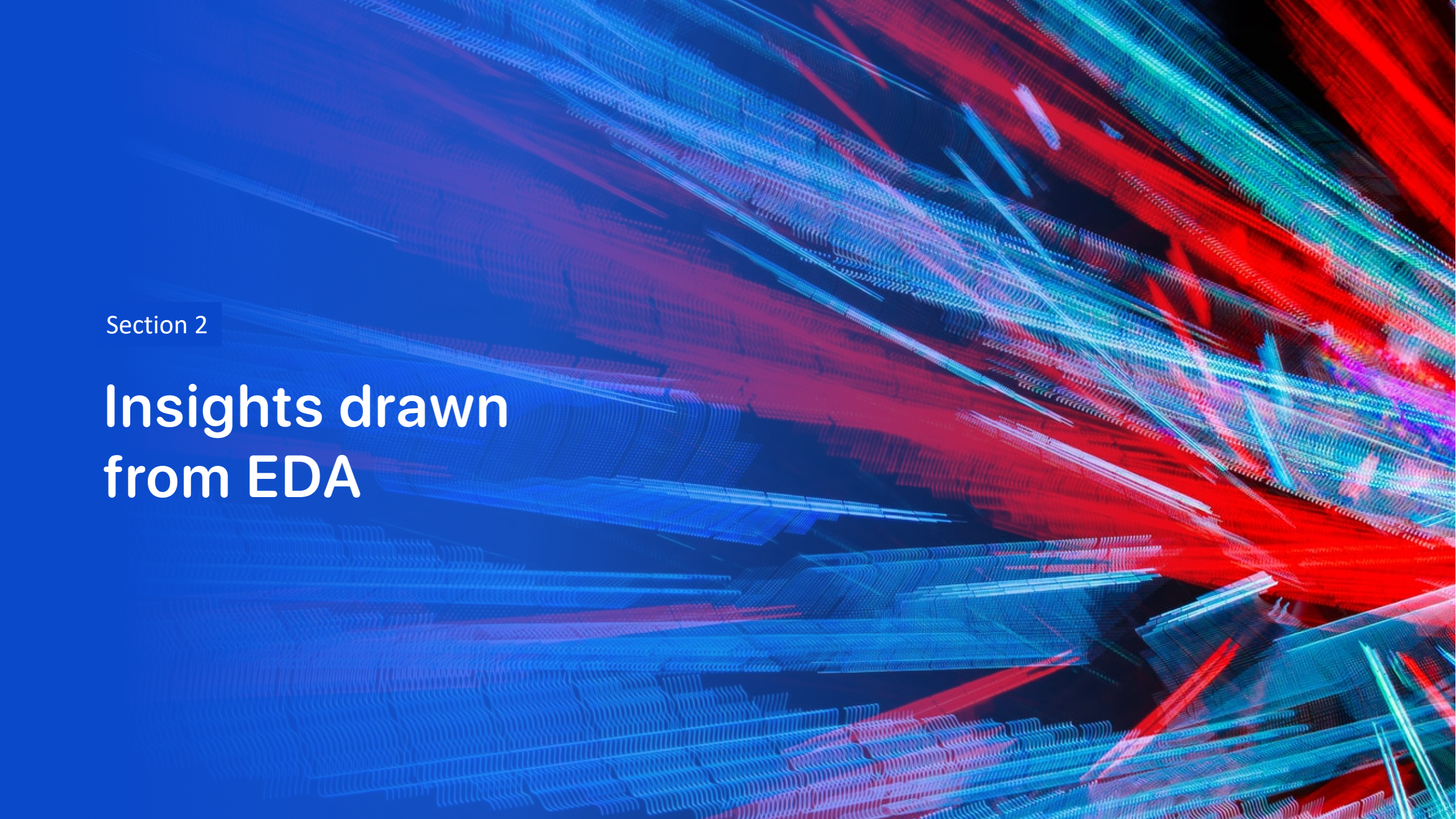
```
predictors = [logreg_cv, svm_cv, knn_cv, tree_cv]
score = []
for predictor in predictors:
    score.append(predictor.score(X_test, Y_test))

print('Highest score on test data: ', max(score))
```

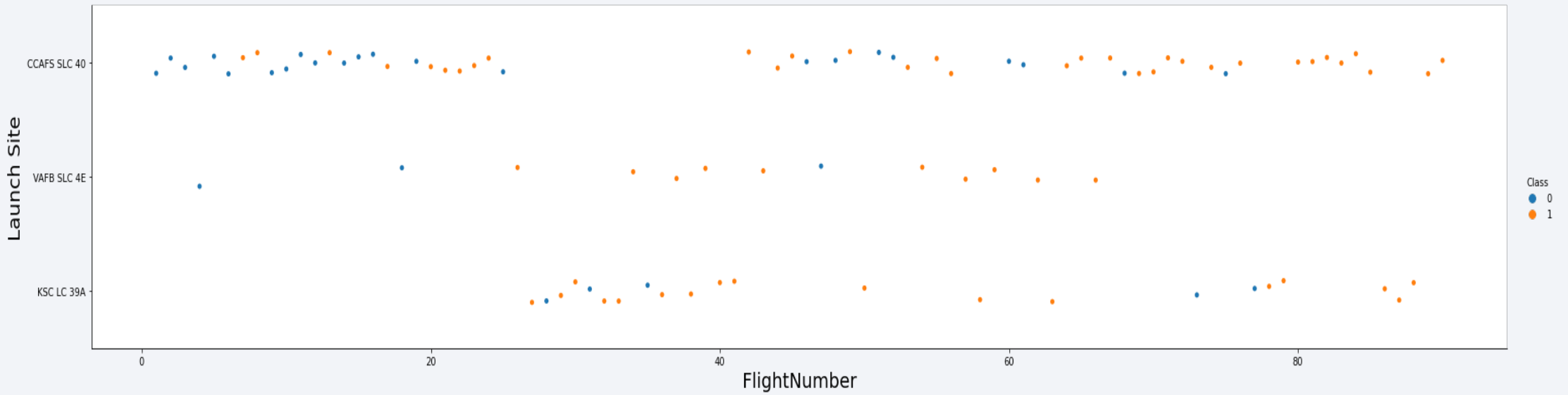

- Github link [here]

Results

- Flight Number, Launch Site, Payload, Orbit have correlation with the launch outcomes
- The interactive dashboard can identify the launch site with most successful rate which is KSC LC-39A; and that smaller payload mass that tends to yield more successful launches
- Predictive analysis identified that Decision Tree model is the one with highest accuracy score on testing set.


Section 2
Insights drawn from EDA

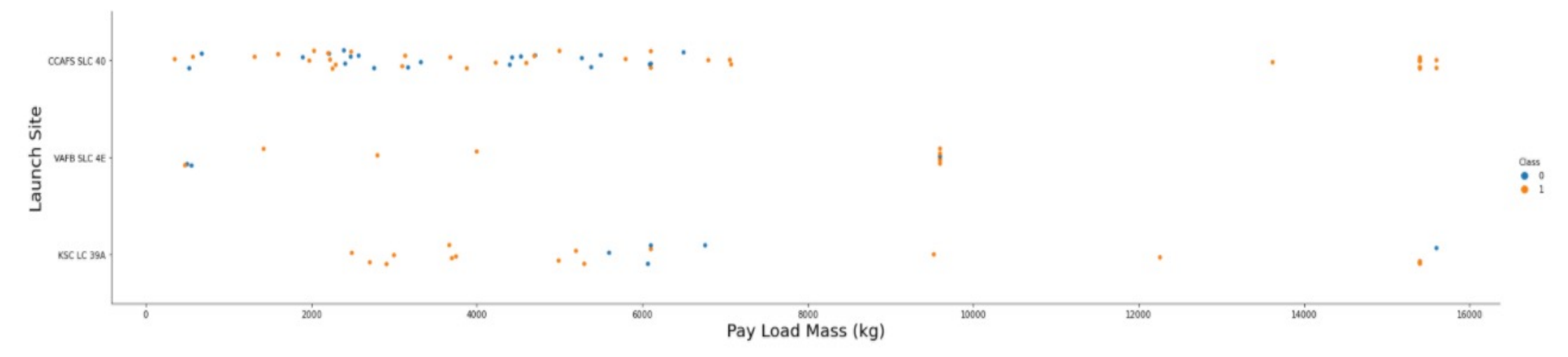
Flight Number vs. Launch Site



Plot of Flight Number vs Launch Site across the outcomes of launch.

There seems to be relationship of the launch site on outcome of launch. CCAFS –C-40 has most flight numbers with almost even successful and unsuccessful first launches. KSC LC 39A seems to have higher successful rate of its launches

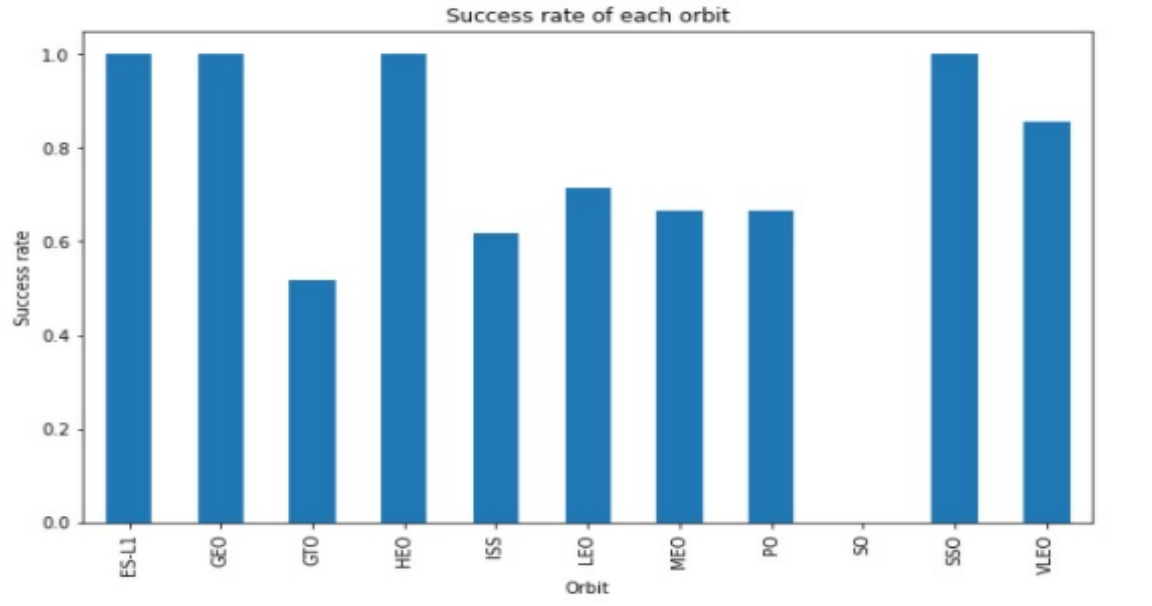
Payload vs. Launch Site



Plot of Payload vs Launch Site across the outcomes of launch

VAFB-SLC launch site does not have heavy rocket payload (beyond 10,000 kgs)

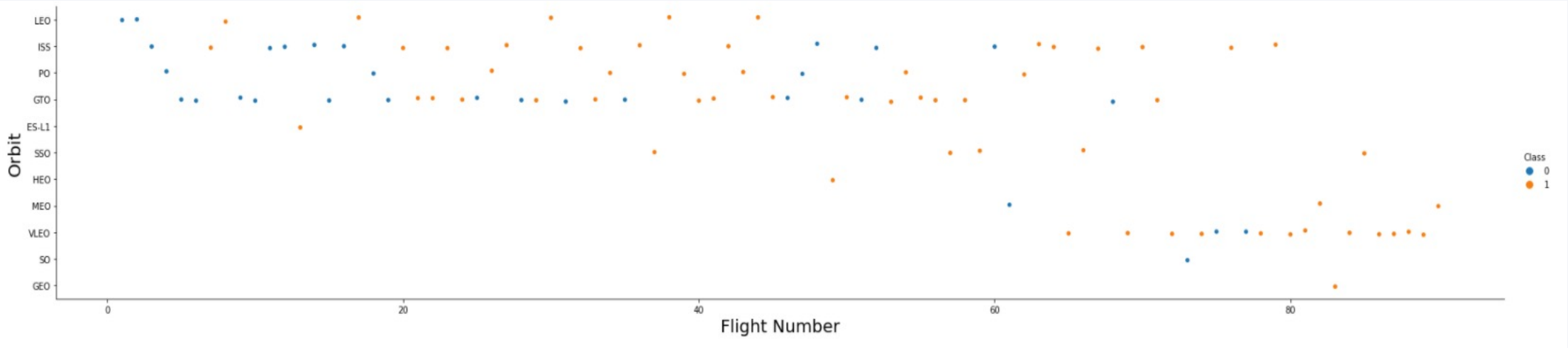
Success Rate vs. Orbit Type



Bar plot of success launch rate across orbit types

Orbits such as ES-L1, GEO, HEO and SSO have the highest rate of successful (almost 100%). Next is VLEO which is over 80% success rate. Other orbits have lower success rates varying between 50– 70%

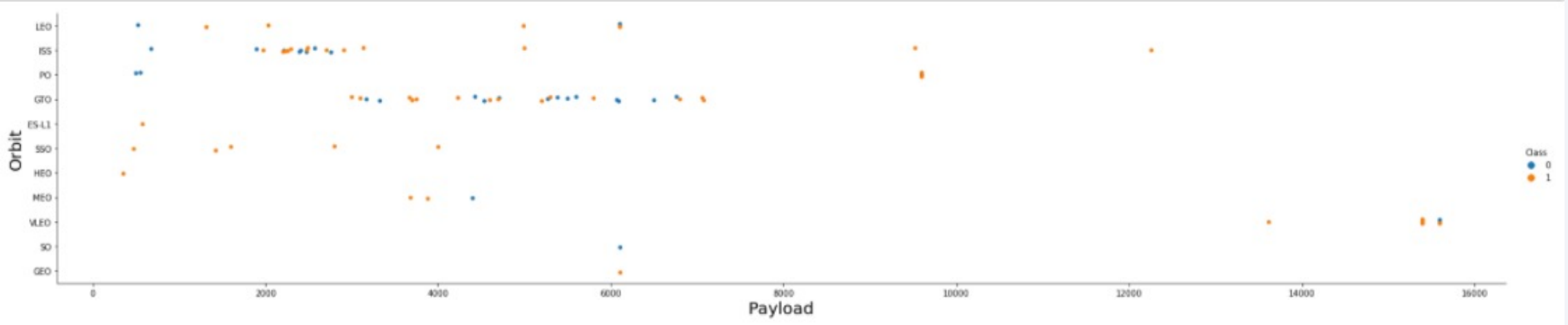
Flight Number vs. Orbit Type



Plot of Orbit vs Flight Number across the outcomes of launch

The relationship seems to vary based on specific orbit, e.g. in the LEO orbit, the Success rate appears to be related to the number of flights. Whereas there seems to be no relationship between flight number when in the GTO orbit

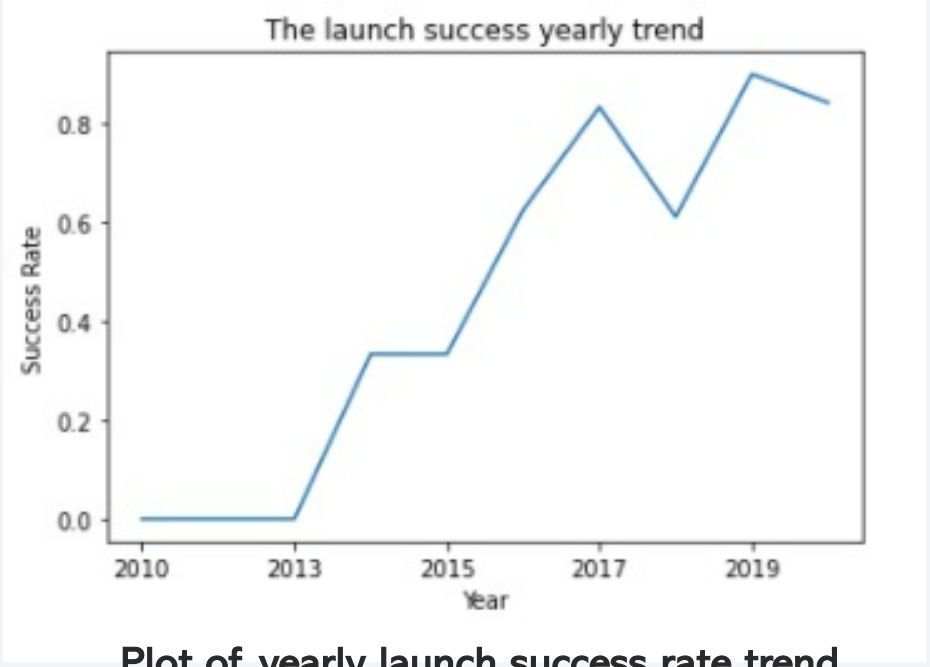
Payload vs. Orbit Type



Plot of Orbit vs Payload across the outcomes of launch

The relationship seems to vary based on specific orbit again here. With heavy payloads the successful landing are more for Polar, LEO and ISS. On the other hand, there is no clear distinction when looking at GTO

Launch Success Yearly Trend



Plot of yearly launch success rate trend

The overall successful rate increases steadily during 2013 – 2019.

All Launch Site Names

```
%sql select distinct LAUNCH_SITE from SPACEXTBL;
```

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

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

- There are 4 unique launch sites from the dataset:
  - CCAFS LC-40
  - CCAFS SLC-40
  - KSC LC-39A
  - VAFB SLC-4E

Launch Site Names Begin with 'CCA'

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

```
%sql select * from SPACEXTBL where LAUNCH_SITE like 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31929/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

- Above are example 5 records where launch sites begin with ‘CCA’, we can see that they took place in 2010, 2012, 2013 in LEO, LEO(ISS) orbits.

Total Payload Mass

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

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

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

)]:

1
45596

- The total payload mass carried by boosters launched by NASA (CRS) is 45,596 kgs

Average Payload Mass by F9 v1.1

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%';
```

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

]:

1
2534

- The average payload mass carried by booster version F9 v1.1 is 2,534 kgs.

First Successful Ground Landing Date

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

Hint:Use min function

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

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

1
2015-12-22

- The first successful landing outcome in ground pad was achieved on 22nd December 2015.

Successful Drone Ship Landing with Payload between 4000 and 6000

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_ between 4000 and 6000;
```

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

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

- The booster versions which have success in drone ship and have payload mass greater than 4000 but less than 6000 are:
  - F9 FT B1021.2
  - F9 FT B1031.2
  - F9 FT B1022
  - F9 FT B1026

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql select count(*), MISSION_OUTCOME from SPACEXTBL group by MISSION_OUTCOME;
```

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

]:

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

- The total number of successful and failure mission outcomes are:
  - 1 failure (in flight)
  - 99 success ones
  - 1 success (payload status unclear)

Boosters Carried Maximum Payload

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 (select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90108kqblod8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

booster_version
F9 B4 B1039.2
F9 B4 B1040.2
F9 B4 B1041.2
F9 B4 B1043.2
F9 B4 B1039.1
F9 B4 B1040.1
F9 B4 B1041.1
F9 B4 B1042.1
F9 B4 B1043.1
F9 B4 B1044

- There are multiple booster versions with maximum payload mass (e.g. F9 B4 B1039.2, etc.)

2015 Launch Records

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 SPACEXTBL where year(DATE) = 2015;
```

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Controlled (ocean)	F9 v1.1 B1013	CCAFS LC-40
No attempt	F9 v1.1 B1014	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
No attempt	F9 v1.1 B1016	CCAFS LC-40
Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40
Success (ground pad)	F9 FT B1019	CCAFS LC-40

- The failed landing outcomes in drone ship in year 2015 were at launch site CCAFS LC-40, and with booster_versions F9 v1.1 B1012, F9 v1.1 B1015

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

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 count(*), LANDING__OUTCOME from SPACEXTBL where DATE between '2010-06-04' and '2017-03-20' group by LANDING__OUTCOME  order by count(*) desc;
```

```
 * ibm_db_sa://ppr91239:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90108kqblod8lcg.databases.appdomain.cloud:31929/bludb
Done.
```

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

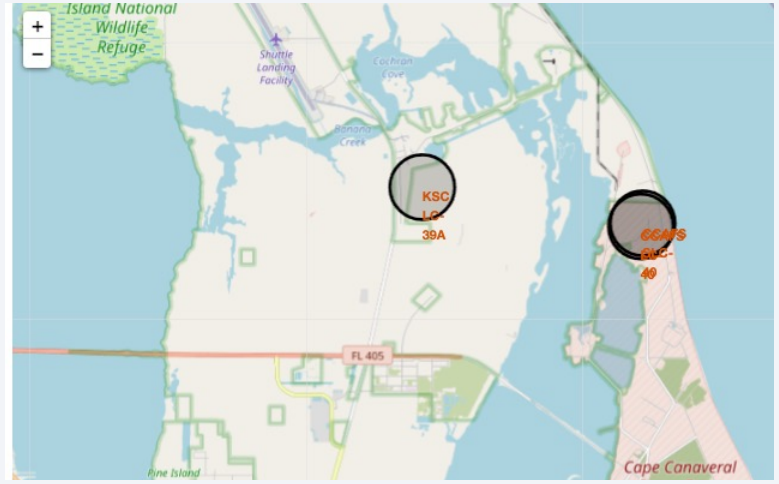
- Between 04/06/2010 and 20/03/2017, there are below counts of landing outcomes in descending order:
  - 10x no attempt
  - 5x failure (drone ship)
  - 5x success (drone ship)
  - 3x controlled (ocean)
  - 3x success (ground pad)
  - 2x failure (parachute)
  - 2x uncontrolled (ocean)
  - 1x precluded (drone ship)


Section 3
Launch Sites
Proximities Analysis

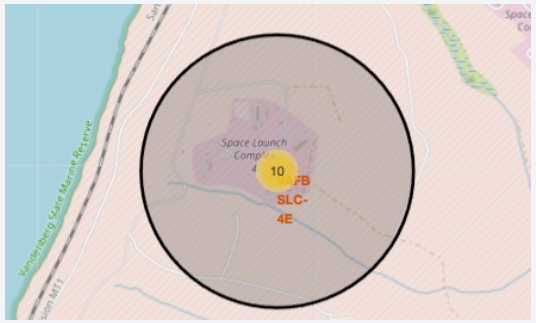
Map of all launch sites

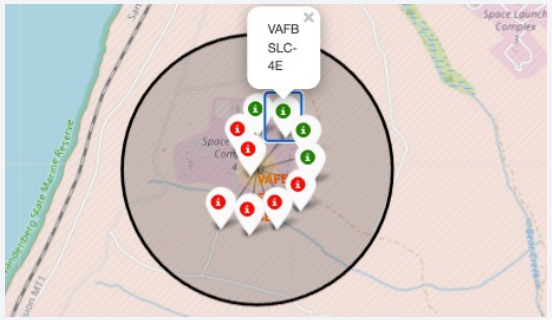
1 out of 4 unique launch sites is on the west coast (VAFB SLC-4E)

Below are zoomed-in screenshots of remaining 3 launch sites in the east coast (CCAFS LC-40, CCAFS SLC-40, KSC LC-39A)



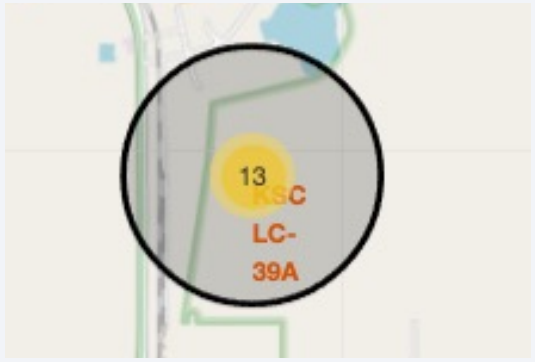
Map of launch outcomes across launch sites

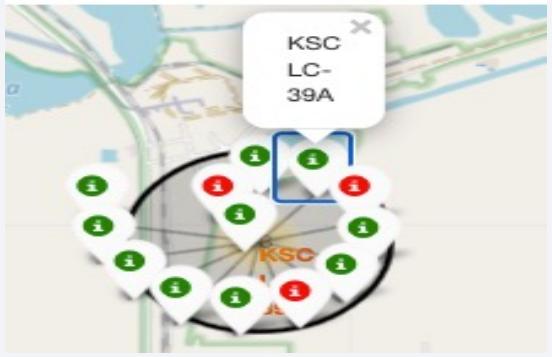




4 success vs 6 failure launches in VAFB SLC-4E (total launch = 10)

7 success vs 19 failure launches in CCAFS LC-40 (total launch = 26)

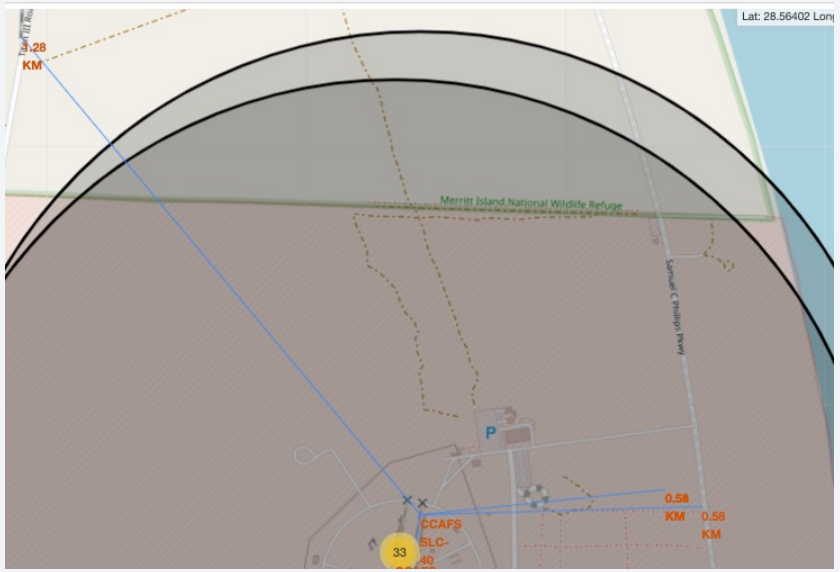


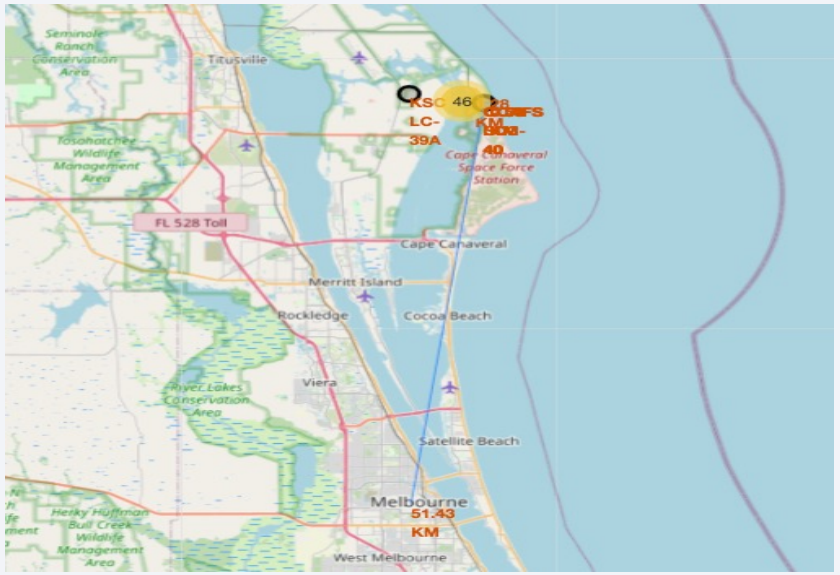


10 success vs 3 failure launches in KSC LC-39A (total launch = 13) – this is the launch site with highest success rate – 78%

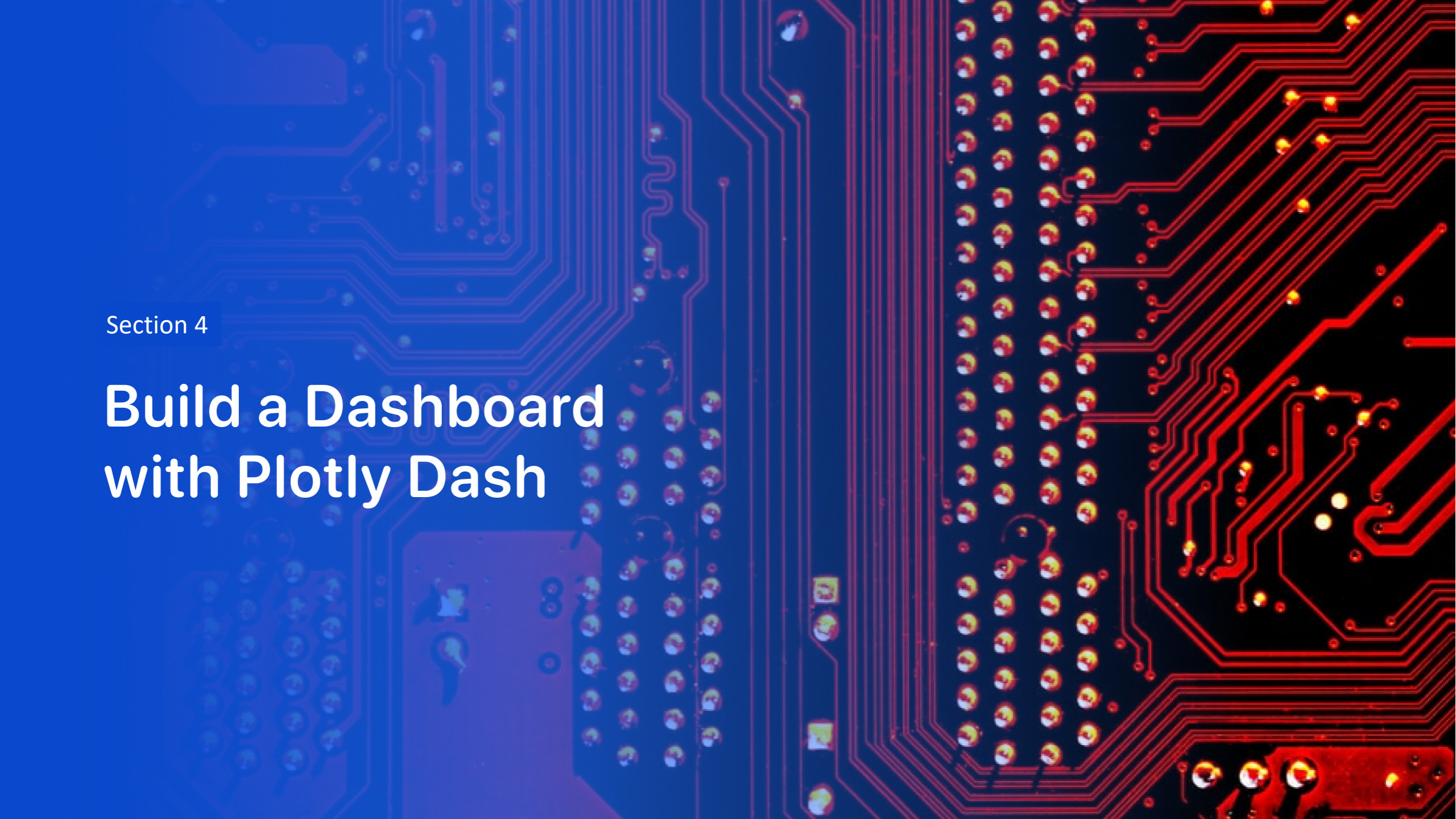
3 success vs 4 failure launches in CCAFS SLC-40 (total launch = 7)

Map of distance proximities of the CCAFS SLC-40

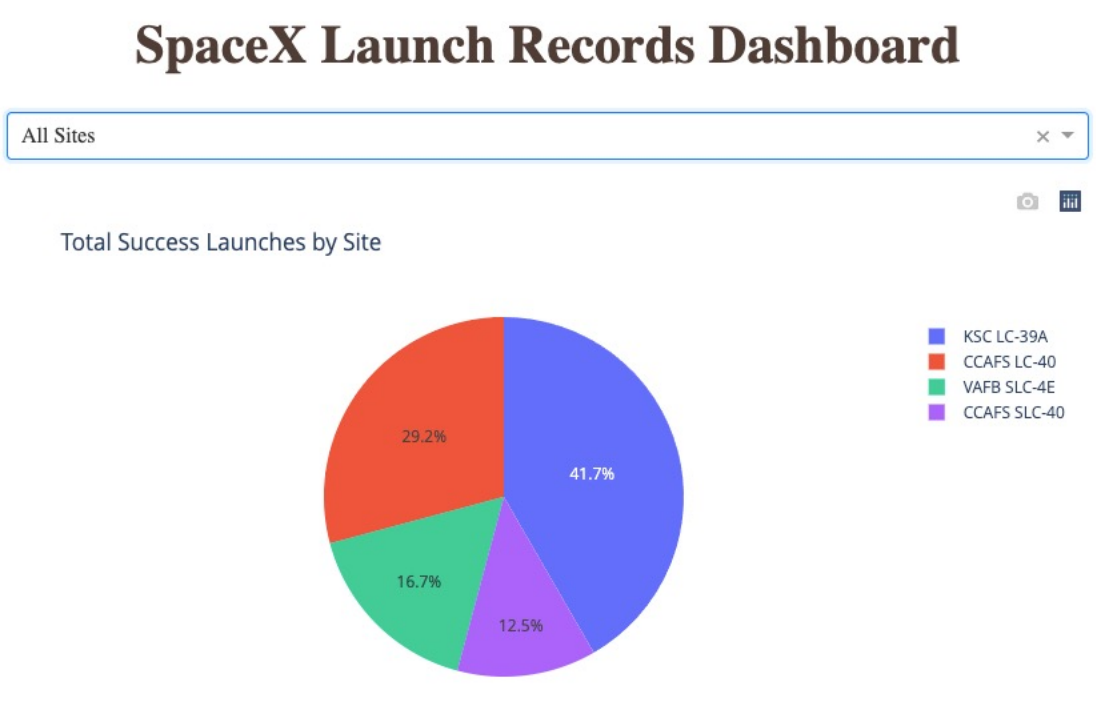




This launch site is closed to highway, railway coastline potentially for easy accessibility to transport, launch pad. On the other hand, it is far away from city which is expected for safety reason.

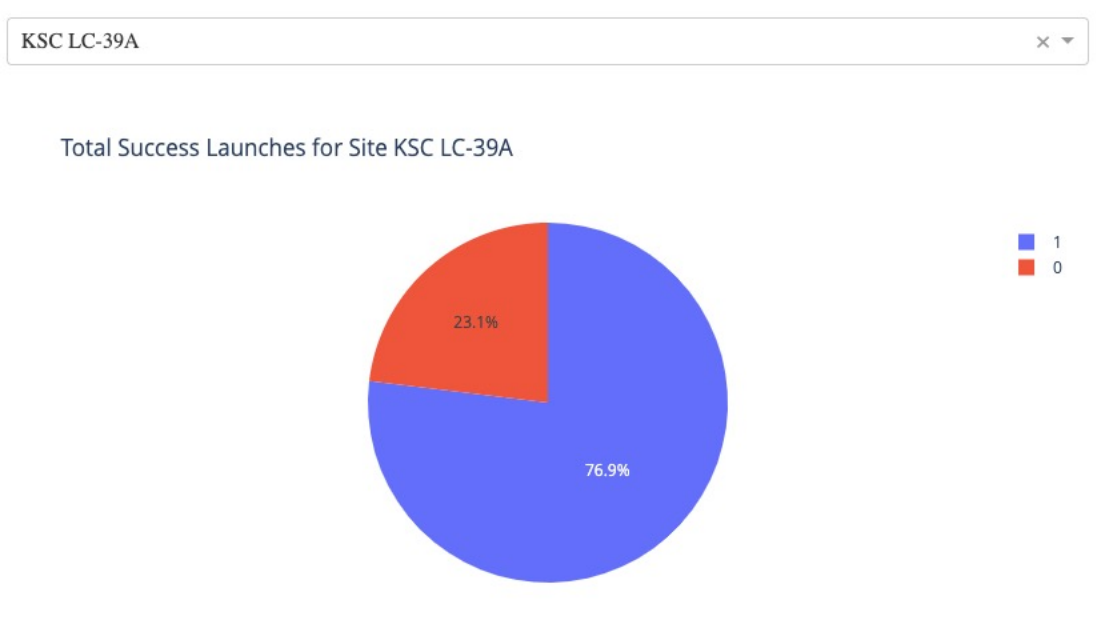

Section 4
Build a Dashboard with Plotly Dash

Total Success Launches by Site



The KSC LC-39A launch site contributes the highest percentage to total success launches across all sites (41.7%).
On the other hand, CCAFS SLC-40 controls the smallest amount to total success launches across all sites (12.5%).

Total Success Launches for Site KSC LC-39A



The KSC LC-39A launch site has almost 80% success launch rate for launches taking place at its site, whereas only 20% failed.

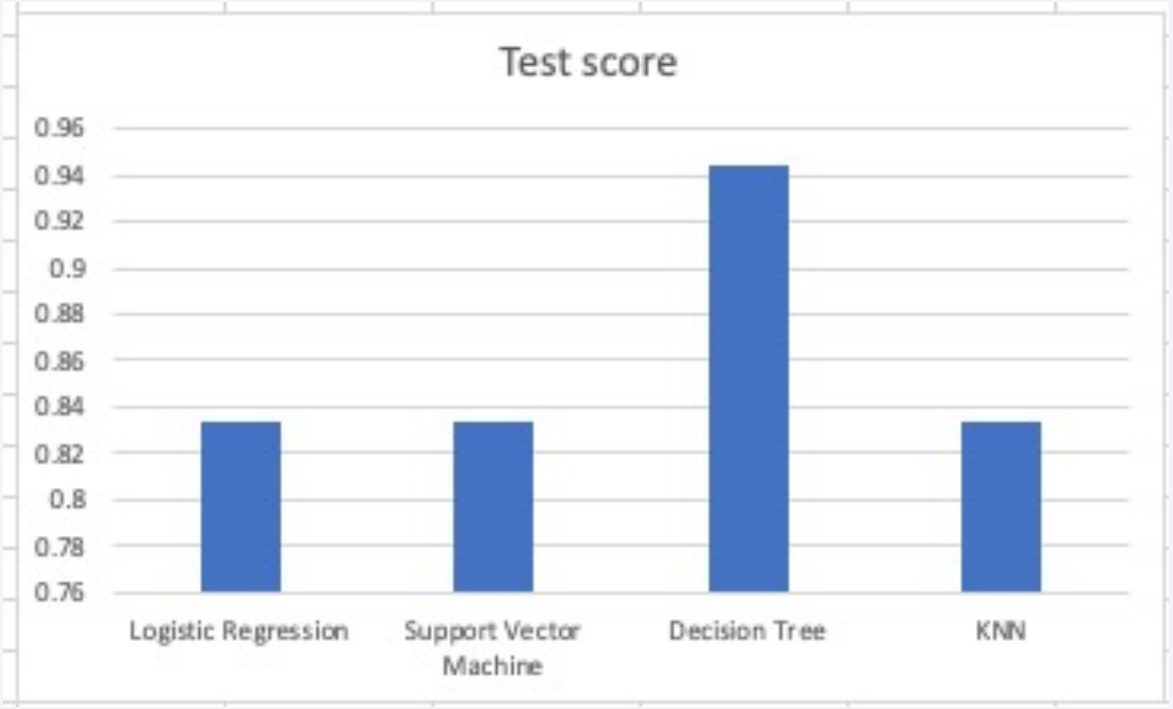
Launch outcomes between 2000-4000 kgs across all sites

When the payload mass slider is adjusted to between 2000 – 4000 kgs, the success outcomes increases compared to the failed outcomes. The performance seems to also be best for FT booster version based on colour indicated in the graph.

Section 5

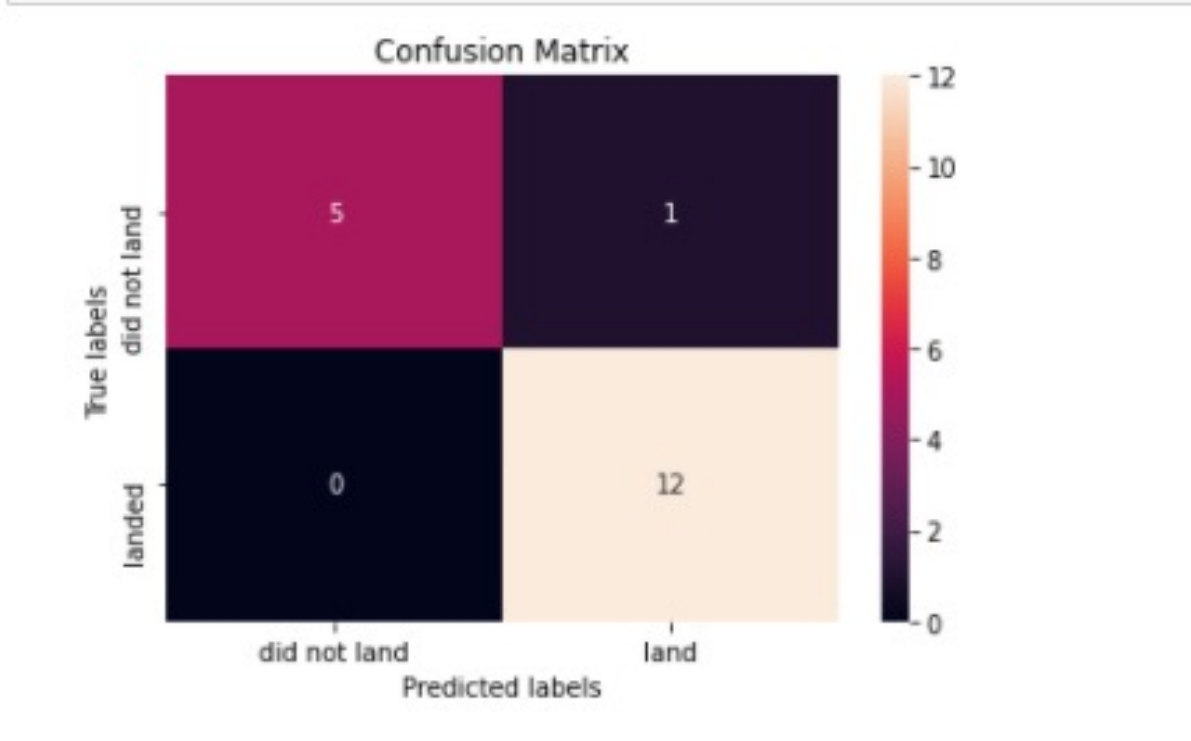
Predictive Analysis (Classification)

Classification Accuracy



Decision tree yields the highest accuracy score on testing data.

Confusion Matrix



From the confusion matrix of tree model, it correctly predicts 12 successful landing, 5 non-successful landing. Only 1 landing is predicted as false positive (i.e. predicted as success but actually failed to land).

Conclusions

- As flight number increases, the launch tends to be more successful.
- From 4 unique launch sites examined in the dataset, KSC LC-39A has the highest successful rate of almost 80%
- Orbits such as ES-L1, GEO, HEO and SSO have the highest rate of successful (almost 100%)
- Smaller payload masses (e.g. between 2,000– 4,000 kilograms) tends to correlate with more successful launch than heavier payload
- Launch sites are closer to highway, railway and coastline but far away from city
- Decision tree model is the predictive algorithm with the highest accuracy score on the dataset being used

Thank you!