Winning Space Race with Data Science

Mohamed Ali Selmi

27th July 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection with API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL and Data Visualization
  - Interactive Visual Analysis with Folium and Dash
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result (Machine Learning Lab)

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.

- **Problems you want to find answers**
  - What features determine the landing outcomes?
  - What are the relationships between those features?
  - What state of these features determine the best outcome?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected through SpaceX RESTful API and Web Scraping from Wikipedia
- Perform data wrangling
  - Data processed through one-hot encoding for 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
  - Data split into training testing data.
  - Training Data used with different classification models.
  - Hyperparameter grid search applied for parameter tuning

Data Collection

Data collection is the process of gathering data for use in business decision-making, strategic planning, research and other purposes. It's a crucial part of data analytics applications and research projects: Effective data collection provides the information that's needed to answer questions, analyze business performance or other outcomes, and predict future trends, actions and scenarios.

- We used the SpaceX REST API to collect the Data.
- We also used Web Scraping to collect Data from Wikipedia.
- Next, we present our data collection process.

Data Collection – SpaceX API

- Get request for rocket launch data using API
- Use json_normalize() method to convert JSON result to Dataframe
- For more details:

  Notebook link on Github

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

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

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

```
[10]
response.status_code
```

```
200
```

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

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

Using the dataframe `data` print the first 5 rows

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

Data Collection - Scraping

- Request the Falcon9 Launch Wikipedia page from url
- Create a BeautifulSoup from the HTML response
- Extract all column/variable names from the HTML header
- For more details:
  - Notebook link on Github

```
# use requests.get() method with the provided static_url
# assign the response to a object
resp=requests.get(static_url)
data=resp.text
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup=BeautifulSoup(data,'html.parser')
```

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables=soup.find_all('table')
```

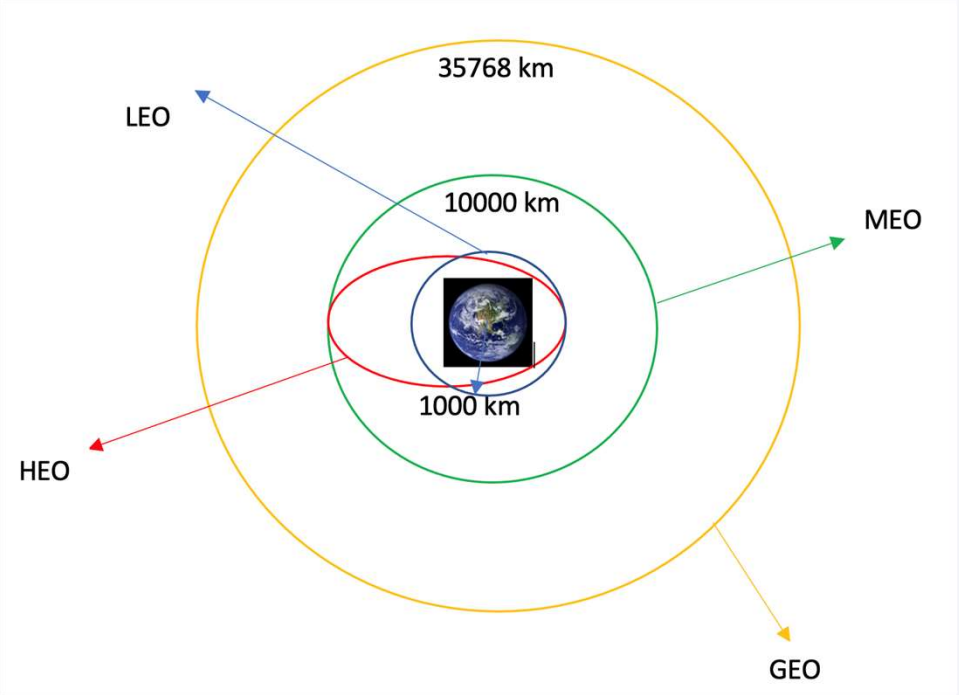
```
column_names = []

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

Data Wrangling

Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis. With the amount of data and data sources rapidly growing and expanding, it is getting increasingly essential for large amounts of available data to be organized for analysis.

- We calculate the number of launches at each site, and the number and occurrence of each orbits
- We create landing outcome label from outcome column and export the results to csv.
- For more details:

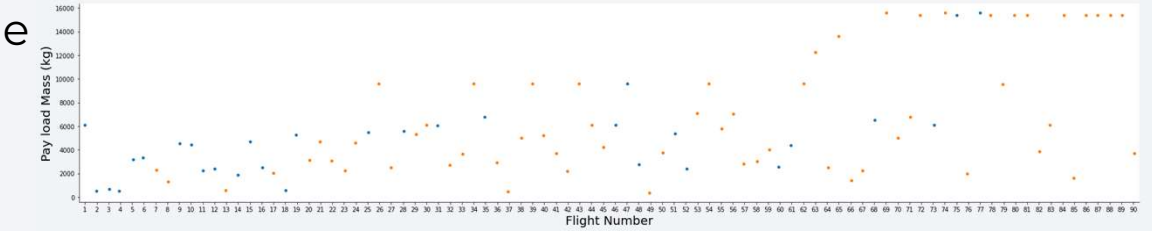
  Notebook link on Github

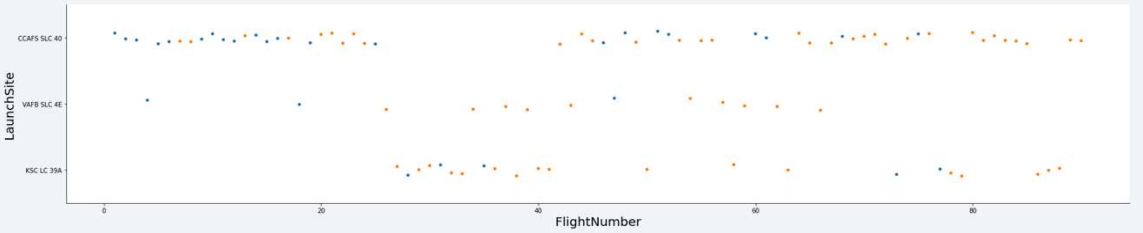


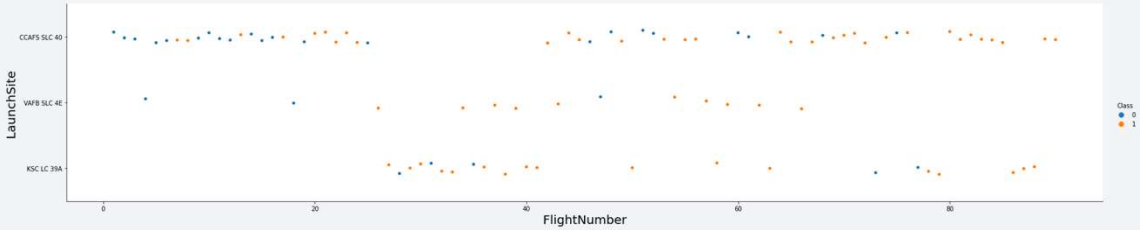
EDA with Data Visualization

- We used scatter graph to find the relationship between the attributes such as between:
  - Payload and Flight Number.
  - Flight Number and Launch Site.
  - Payload and Launch Site.
  - Flight Number and Orbit Type.
  - Payload and Orbit Type.

Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs. It's very easy to see which factors affecting the most to the success of the landing outcomes.



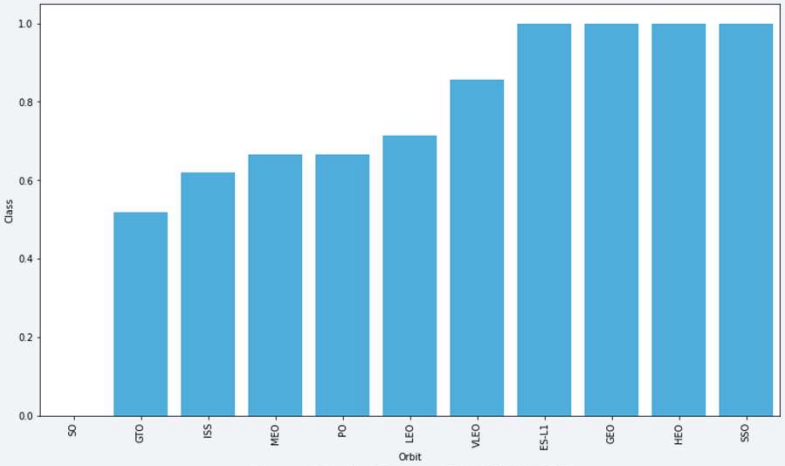


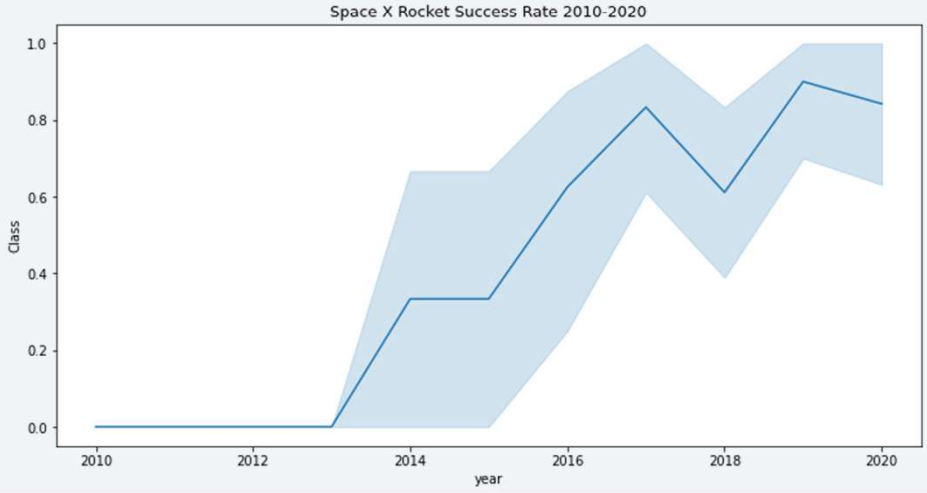


Notebook link on Github

EDA with Data Visualization

- Bar graphs is one of the easiest way to interpret the relationship between the attributes. In this case, we used a bar chart to determine which orbits have the highest probability of success.
- We used a line graph to show a trends or pattern of the attribute over time which in this case, is used for see the launch success yearly trend.





[Notebook link on Github]

EDA with SQL

- **We loaded the SpaceX dataset into an IBM db2 database. We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:**
  - the names of the launch sites.
  - 5 records where launch sites begin with the string 'CCA'.
  - the total payload mass carried by booster launched by NASA (CRS).
  - the average payload mass carried by booster version F9 v1.1.
  - the date when the first successful landing outcome in ground pad was achieved.
  - the names of the boosters which have success in drone ship 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.
  - the failed landing_outcomes in drone ship, their booster versions, and launch sites names for in year 2015.
  - the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order.

Notebook link on Github

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.

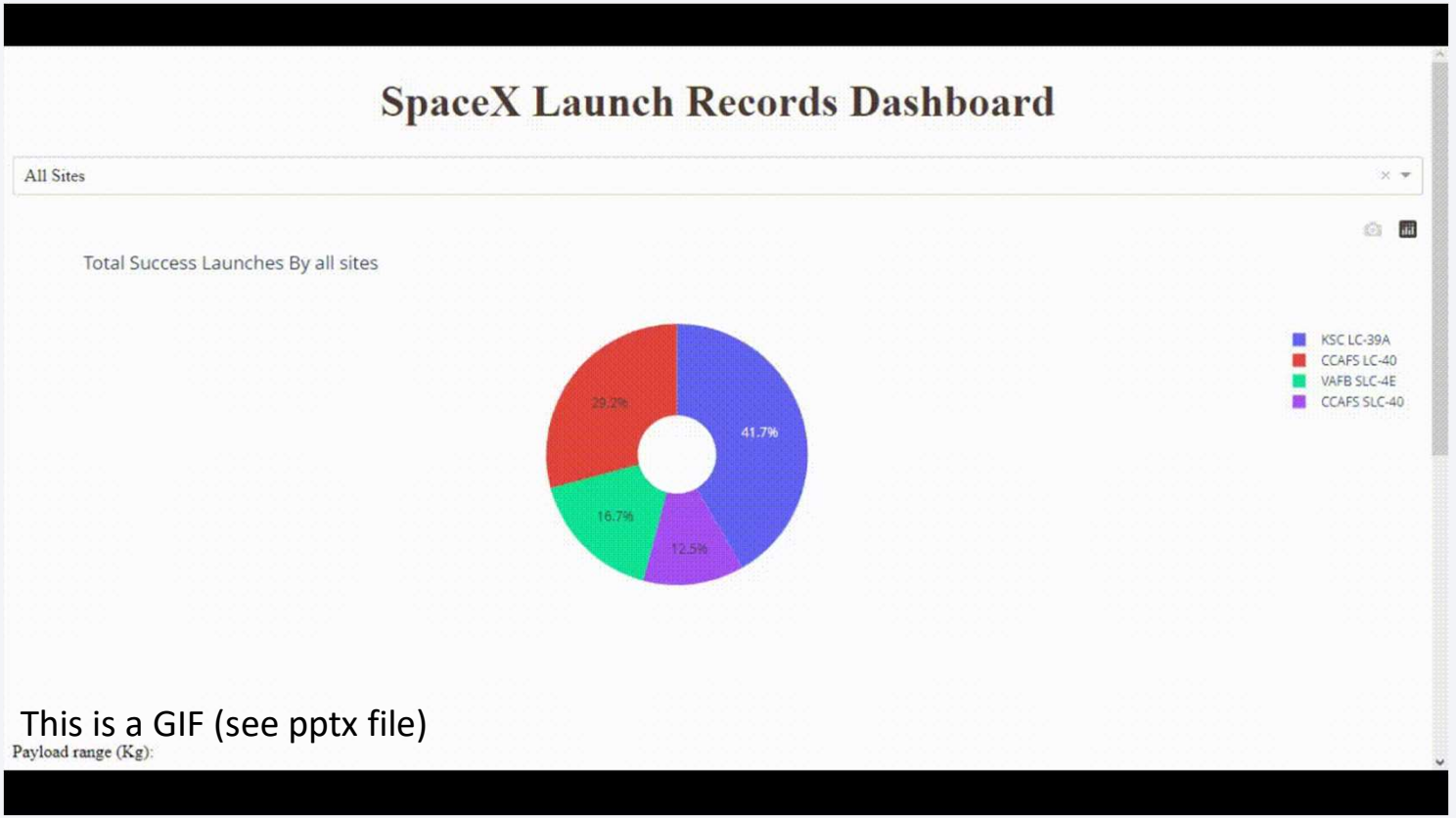
Notebook link on Github

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allowing the user to play around with the data as they need.
- We plotted pie charts showing the total launches by a certain sites.
- We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- Explain why you added those plots and interactions

Notebook link on Github

Build a Dashboard with Plotly Dash



This is a GIF (see pptx file)

[Notebook link on Github]

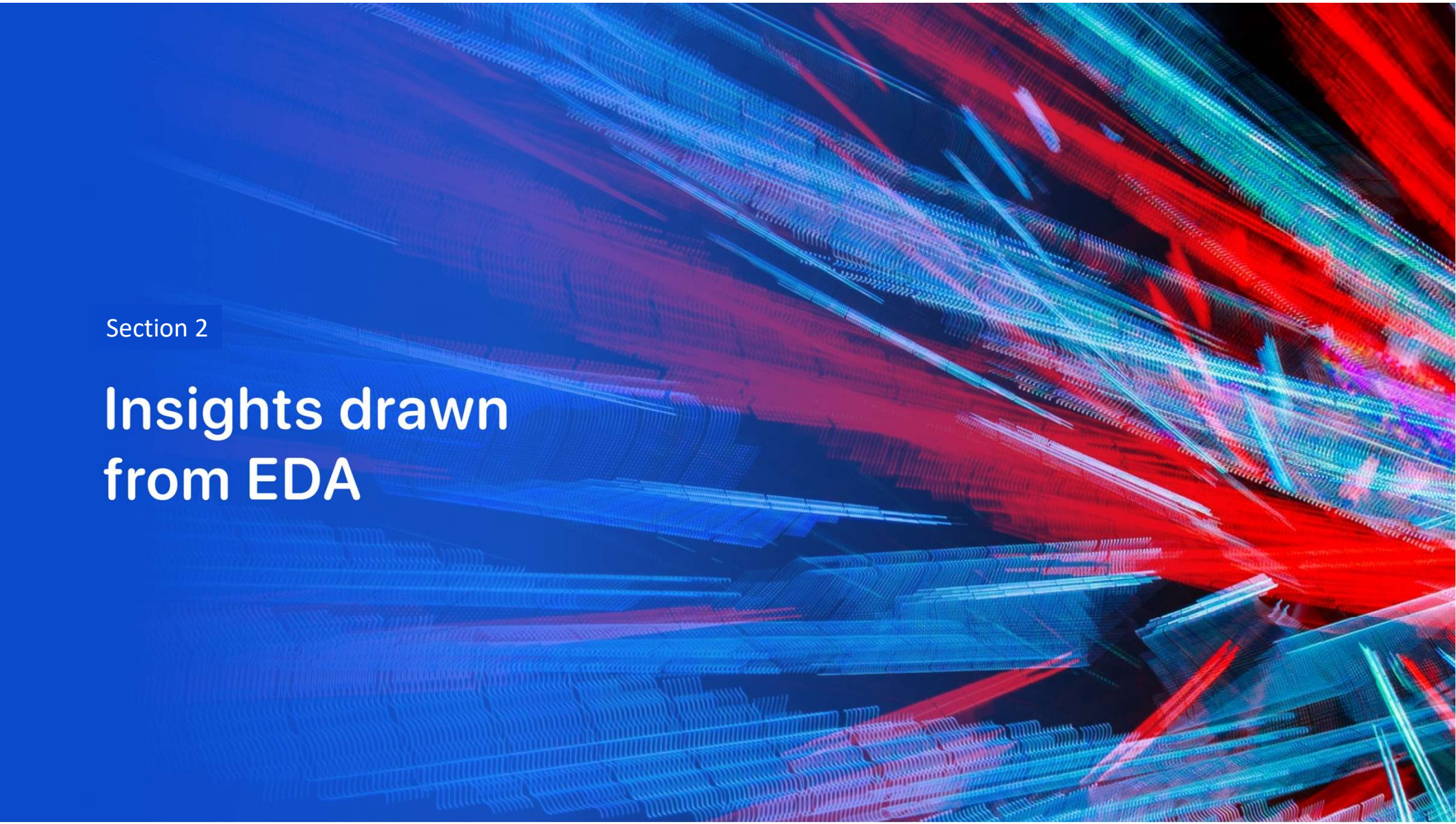
Predictive Analysis (Classification)

- We loaded the data using numpyand pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.

Notebook link on Github

Results

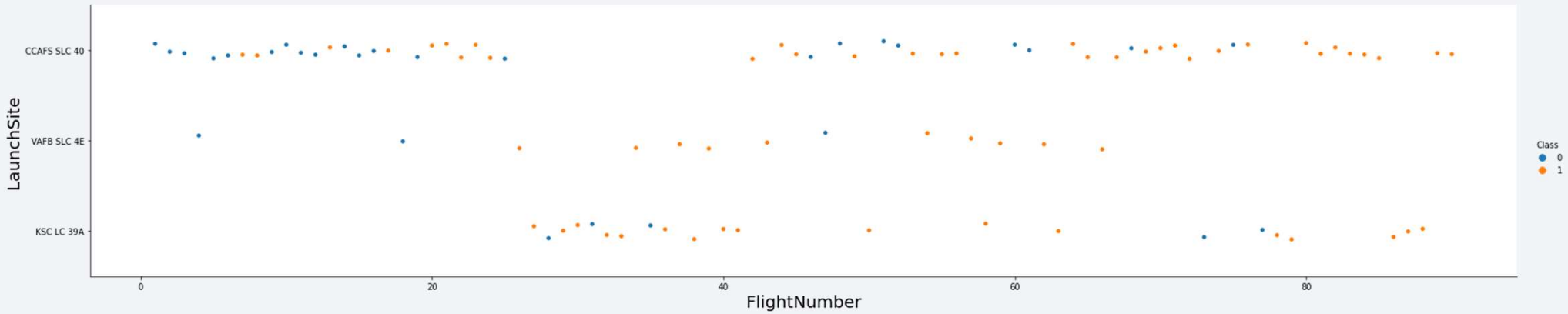
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

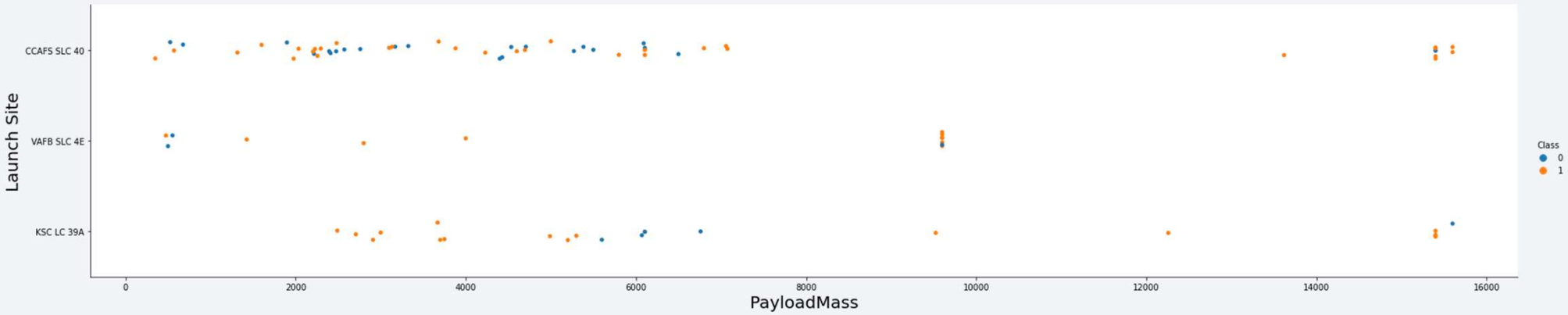
Insights drawn from EDA

Flight Number vs. Launch Site



This plot shows that the larger the flights number of each launch site, the greater the success rate will be.

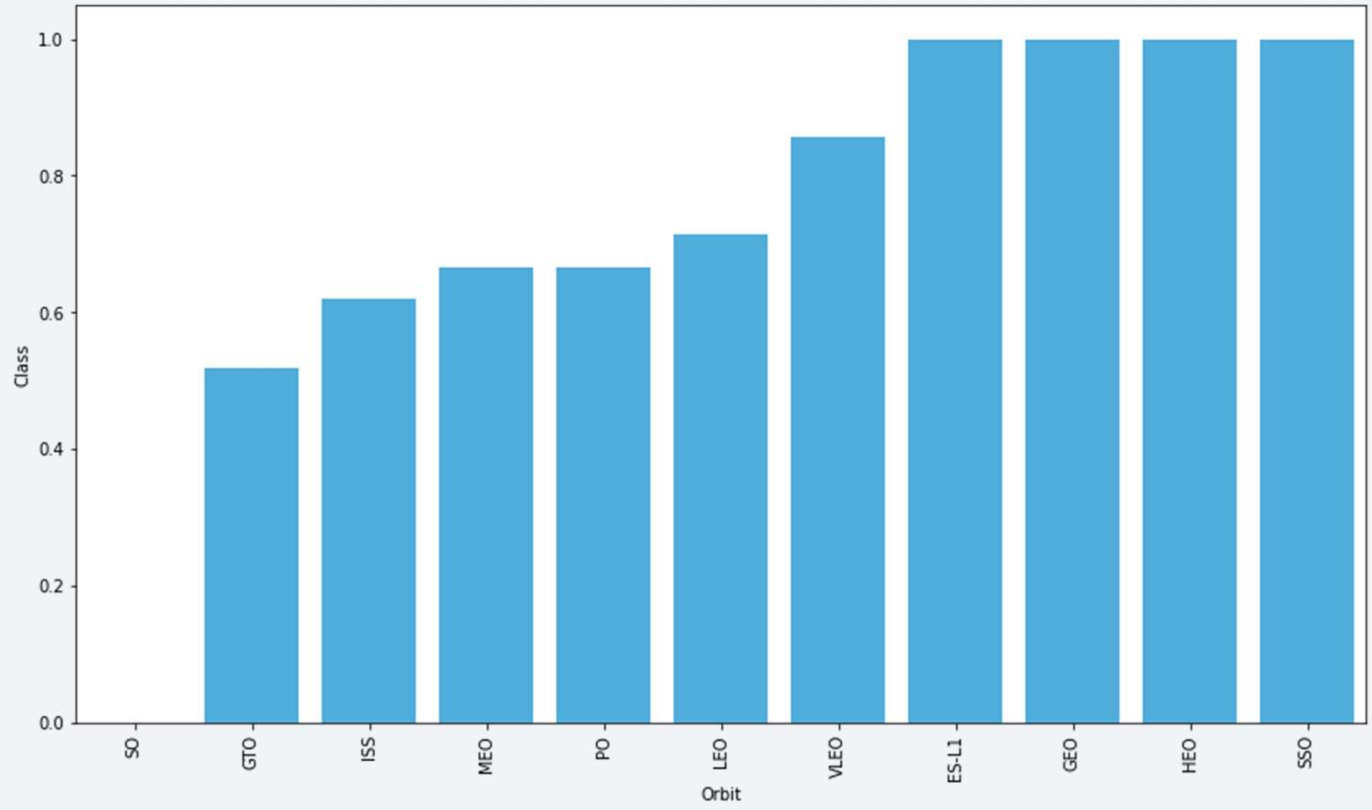
Payload vs. Launch Site



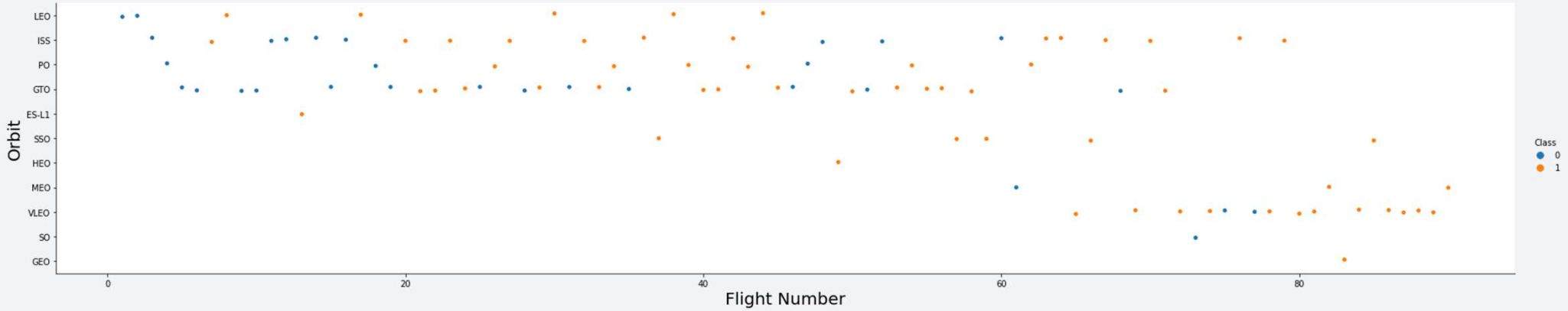
This scatter plot shows once the pay load mass is greater than 7000kg, the probability of the success rate will be highly increased.
However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.

Success Rate vs. Orbit Type

This chart depicted the possibility of the orbits to influences the landing outcomes as some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.



Flight Number vs. Orbit Type

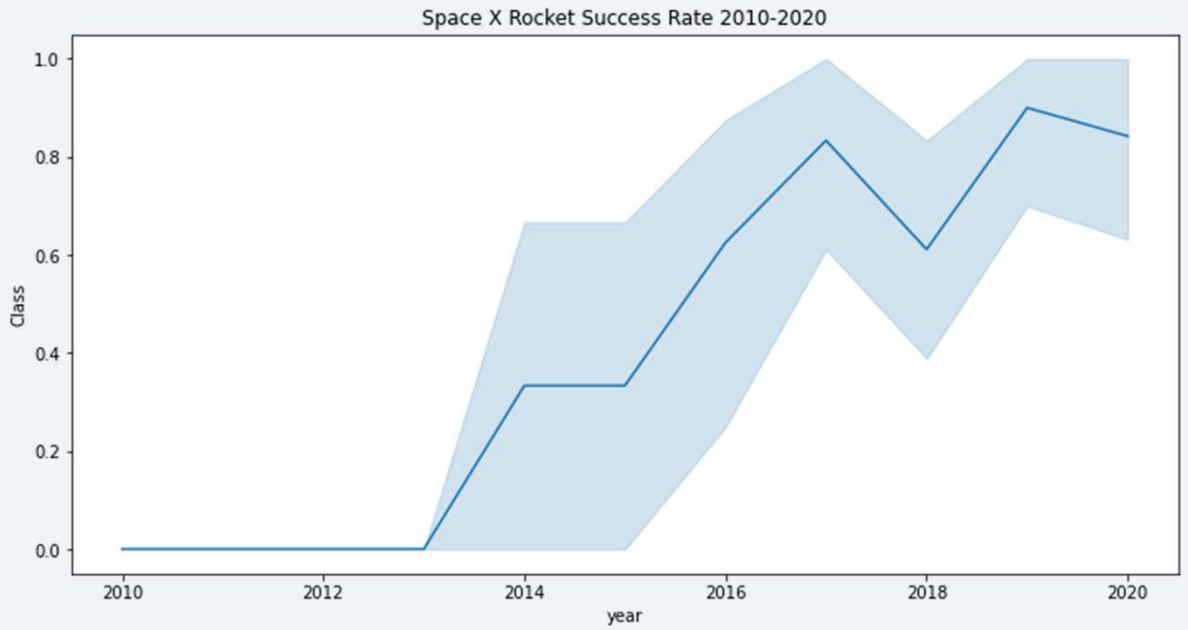


We see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type

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

Launch Success Yearly Trend



you can observe that the success rate since 2013 kept increasing till 2020

All Launch Site Names

```
selectQuery = "SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;"
selectStmt = ibm_db.exec_immediate(conn, selectQuery)
while ibm_db.fetch_row(selectStmt) != False:
   print (" Launch_Site:",  ibm_db.result(selectStmt, 0))
```

[7]

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

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

Launch Site Names Begin with 'CCA'

```
selectQuery = "SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;"
selectStmt = ibm_db.exec_immediate(conn, selectQuery)
while ibm_db.fetch_row(selectStmt) != False:
    for i in range(10):
      print(ibm_db.result(selectStmt, i),end='')
    print("")
```

[16]

```
2010-06-0418:45:00F9 v1.0  B0003CCAFS LC-40Dragon Spacecraft Qualification Unit0LEOSpaceXSuccessFailure (parachute)
2010-12-0815:43:00F9 v1.0  B0004CCAFS LC-40Dragon demo flight C1, two CubeSats, barrel of Brouere cheese0LEO (ISS)NASA (COTS) NROSuccessFailure (parachute)
2012-05-2207:44:00F9 v1.0  B0005CCAFS LC-40Dragon demo flight C2525LEO (ISS)NASA (COTS)SuccessNo attempt
2012-10-0800:35:00F9 v1.0  B0006CCAFS LC-40SpaceX CRS-1500LEO (ISS)NASA (CRS)SuccessNo attempt
2013-03-0115:10:00F9 v1.0  B0007CCAFS LC-40SpaceX CRS-2677LEO (ISS)NASA (CRS)SuccessNo attempt
```

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

Total Payload Mass

```
[25]
    selectQuery = "SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';"
    selectStmt = ibm_db.exec_immediate(conn, selectQuery)
    while ibm_db.fetch_row(selectStmt) != False:
        print("total payload mass carried by boosters launched by NASA (CRS) :",ibm_db.result(selectStmt, 0))

... total payload mass carried by boosters launched by NASA (CRS) : 45596
```

We calculated the total payload carried by boosters from NASA as 45596 using the query below

Average Payload Mass by F9 v1.1

```
    selectQuery = "SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';"
    selectStmt = ibm_db.exec_immediate(conn, selectQuery)
    while ibm_db.fetch_row(selectStmt) != False:
        print("average payload mass carried by booster version F9 v1.1 :",ibm_db.result(selectStmt, 0))
[29]
...  average payload mass carried by booster version F9 v1.1 : 2928
```

We calculated the average payload mass carried by booster version F9 v1.1 as 2928

First Successful Ground Landing Date

```
[36]
    selectQuery = "SELECT min(DATE) FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)';"
    selectStmt = ibm_db.exec_immediate(conn, selectQuery)
    while ibm_db.fetch_row(selectStmt) != False:
        print("the date when the first successful landing outcome in ground pad was acheived :",ibm_db.result(selectStmt, 0))

... the date when the first successful landing outcome in ground pad was acheived : 2015-12-22
```

We use the min() function to find the result.
The dates of the first successful landing outcome in ground pad was 22nd December 2015

Successful Drone Ship Landing with Payload between 4000 and 6000

```
selectQuery = """SELECT BOOSTER_VERSION
                 FROM SPACEXTBL
                 WHERE LANDING__OUTCOME = 'Success (drone ship)'
                 AND PAYLOAD_MASS__KG_ > 4000
                 AND PAYLOAD_MASS__KG_ < 6000;"""
selectStmt = ibm_db.exec_immediate(conn, selectQuery)
print('the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000')
while ibm_db.fetch_row(selectStmt) != False:
    print(ibm_db.result(selectStmt, 0))
```

[38]

```
the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
F9 FT B1022
F9 FT B1026
F9 FT  B1021.2
F9 FT  B1031.2
```

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

Total Number of Successful and Failure Mission Outcomes

```
[46]
    selectQuery = "SELECT count(*) FROM SPACEXTBL WHERE MISSION_OUTCOME like 'Success%';"
    selectQuery1 = "SELECT count(*) FROM SPACEXTBL WHERE MISSION_OUTCOME like 'Failure%';"
    selectStmt = ibm_db.exec_immediate(conn, selectQuery)
    while ibm_db.fetch_row(selectStmt) != False:
        print("the total number of successful mission outcomes",ibm_db.result(selectStmt, 0))
    selectStmt1 = ibm_db.exec_immediate(conn, selectQuery1)
    while ibm_db.fetch_row(selectStmt1) != False:
        print("the total number of failure mission outcomes",ibm_db.result(selectStmt1, 0))

... the total number of successful mission outcomes 100
    the total number of failure mission outcomes 1
```

We used wildcard like '%' to filter for WHERE Mission Outcome was a success or a failure.

Boosters Carried Maximum Payload

We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

```
selectQuery="""SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_=(
                SELECT MAX(PAYLOAD_MASS__KG_)
                FROM SPACEXTBL);"""
selectStmt = ibm_db.exec_immediate(conn, selectQuery)
print("names of the booster_versions which have carried the maximum payload mass.")
while ibm_db.fetch_row(selectStmt) != False:
    print("Booster_version :",ibm_db.result(selectStmt, 0))
```

[47]

```
names of the booster_versions which have carried the maximum payload mass.
Booster_version : F9 B5 B1048.4
Booster_version : F9 B5 B1049.4
Booster_version : F9 B5 B1051.3
Booster_version : F9 B5 B1056.4
Booster_version : F9 B5 B1048.5
Booster_version : F9 B5 B1051.4
Booster_version : F9 B5 B1049.5
Booster_version : F9 B5 B1060.2
Booster_version : F9 B5 B1058.3
Booster_version : F9 B5 B1051.6
Booster_version : F9 B5 B1060.3
Booster_version : F9 B5 B1049.7
```

2015 Launch Records

```
[49]
    selectQuery="""SELECT BOOSTER_VERSION, LAUNCH_SITE
                   FROM SPACEXTBL
                   WHERE LANDING__OUTCOME = 'Failure (drone ship)'
                   AND Date BETWEEN '2015-01-01' AND '2015-12-31';"""
    selectStmt=ibm_db.exec_immediate(conn,selectQuery)
    while ibm_db.fetch_row(selectStmt):
        print("Booster_version : ",ibm_db.result(selectStmt, 0),"LAUNCH_SITE : ",ibm_db.result(selectStmt, 1))
```

```
... Booster_version :  F9 v1.1 B1012 LAUNCH_SITE :  CCAFS LC-40
    Booster_version :  F9 v1.1 B1015 LAUNCH_SITE :  CCAFS LC-40
```

We used a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

```
[57]
    selectQuery="""SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) as cnt
            FROM SPACEXTBL
            WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
            GROUP BY LANDING__OUTCOME
            ORDER BY cnt DESC;"""
    selectStmt=ibm_db.exec_immediate(conn,selectQuery)
    while ibm_db.fetch_row(selectStmt):
        print("LANDING__OUTCOME : ",ibm_db.result(selectStmt, 0),"count :",ibm_db.result(selectStmt, 1))
...
    LANDING__OUTCOME :  No attempt count : 10
    LANDING__OUTCOME :  Failure (drone ship) count : 5
    LANDING__OUTCOME :  Success (drone ship) count : 5
    LANDING__OUTCOME :  Controlled (ocean) count : 3
    LANDING__OUTCOME :  Success (ground pad) count : 3
    LANDING__OUTCOME :  Failure (parachute) count : 2
    LANDING__OUTCOME :  Uncontrolled (ocean) count : 2
    LANDING__OUTCOME :  Precluded (drone ship) count : 1
```

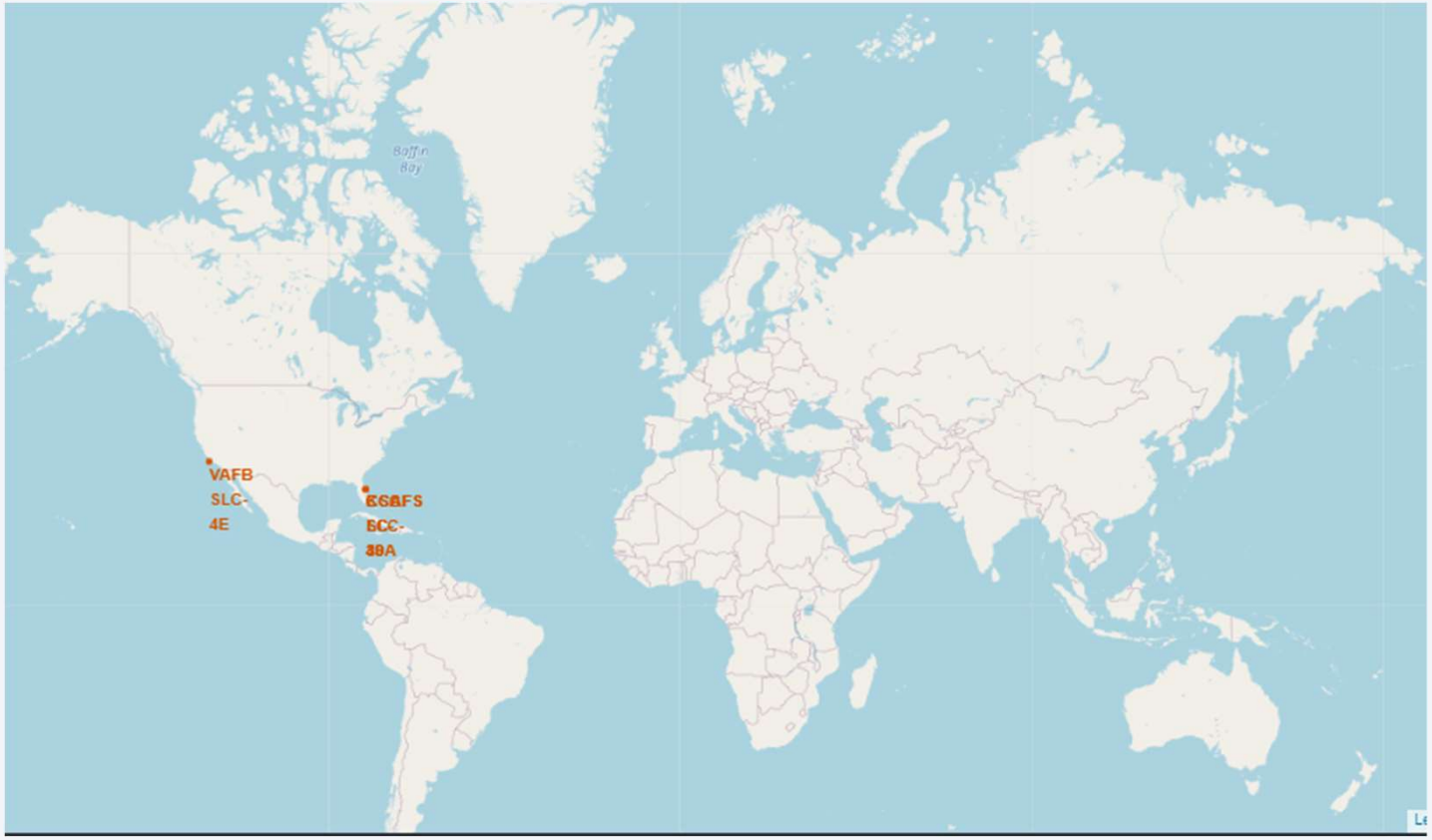
We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.
We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending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

Section 3

Launch Sites Proximities Analysis

Location of all the Launch Sites

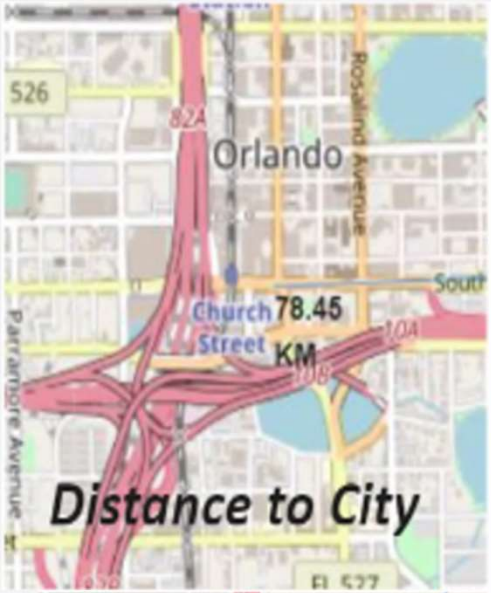
We can see that all the SpaceX launch sites are located inside the United States

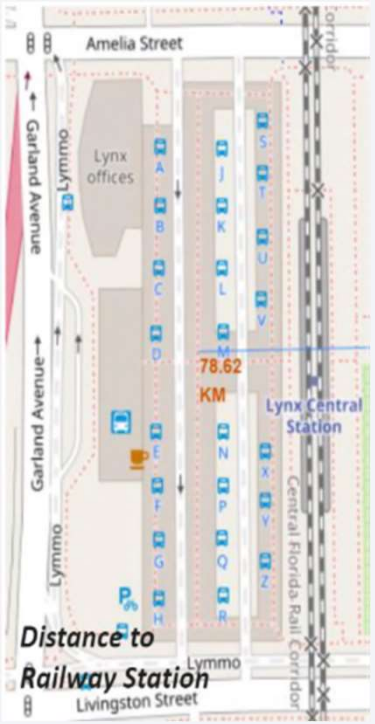


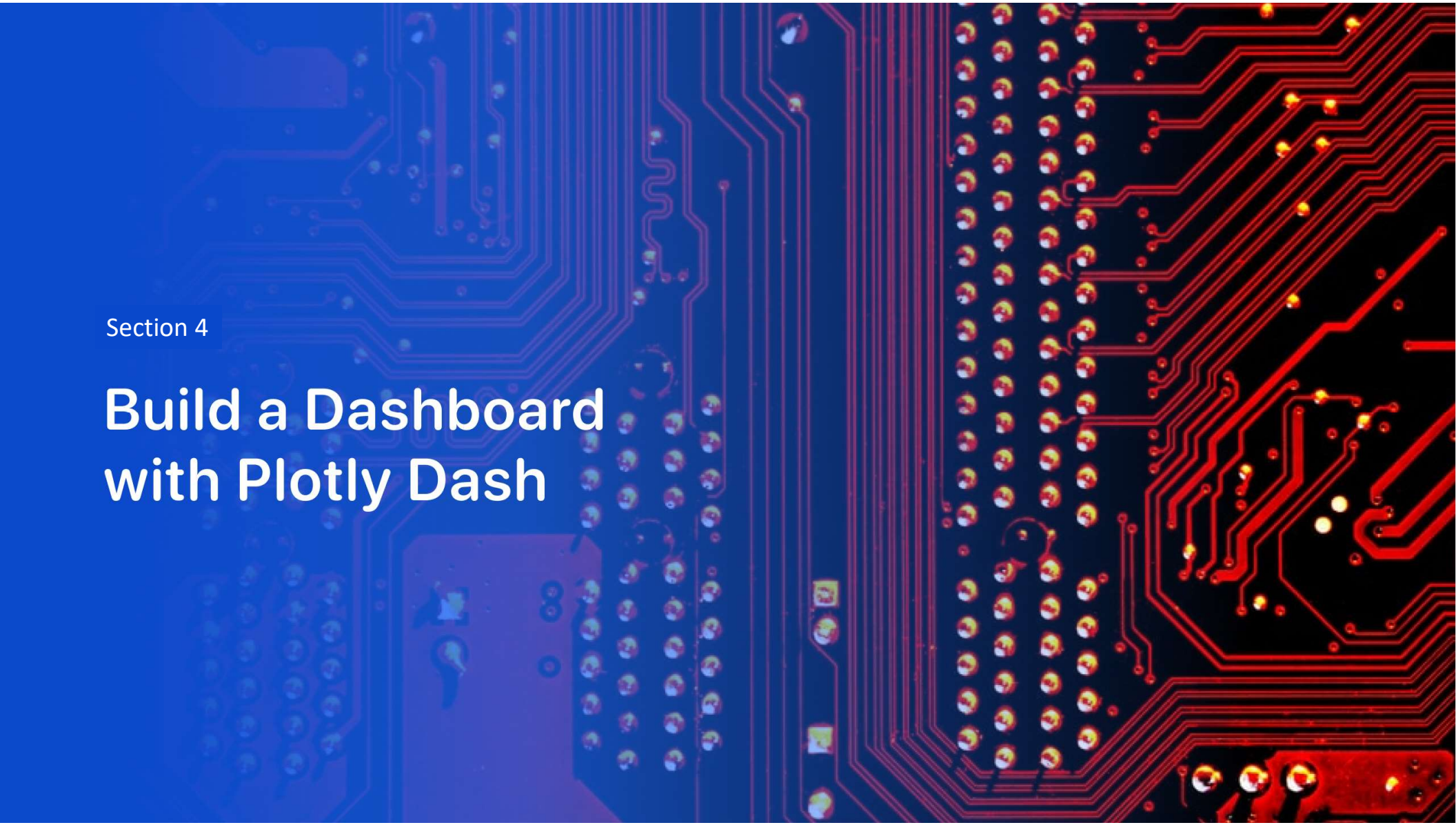
Markers showing launch sites with color labels

Launch Site distance to landmarks

Are launch sites near railways?
NO
Are launch sites near highways?
NO
Are launch sites near coastline?
YES
Do launch sites keep certain distance away from cities?
YES

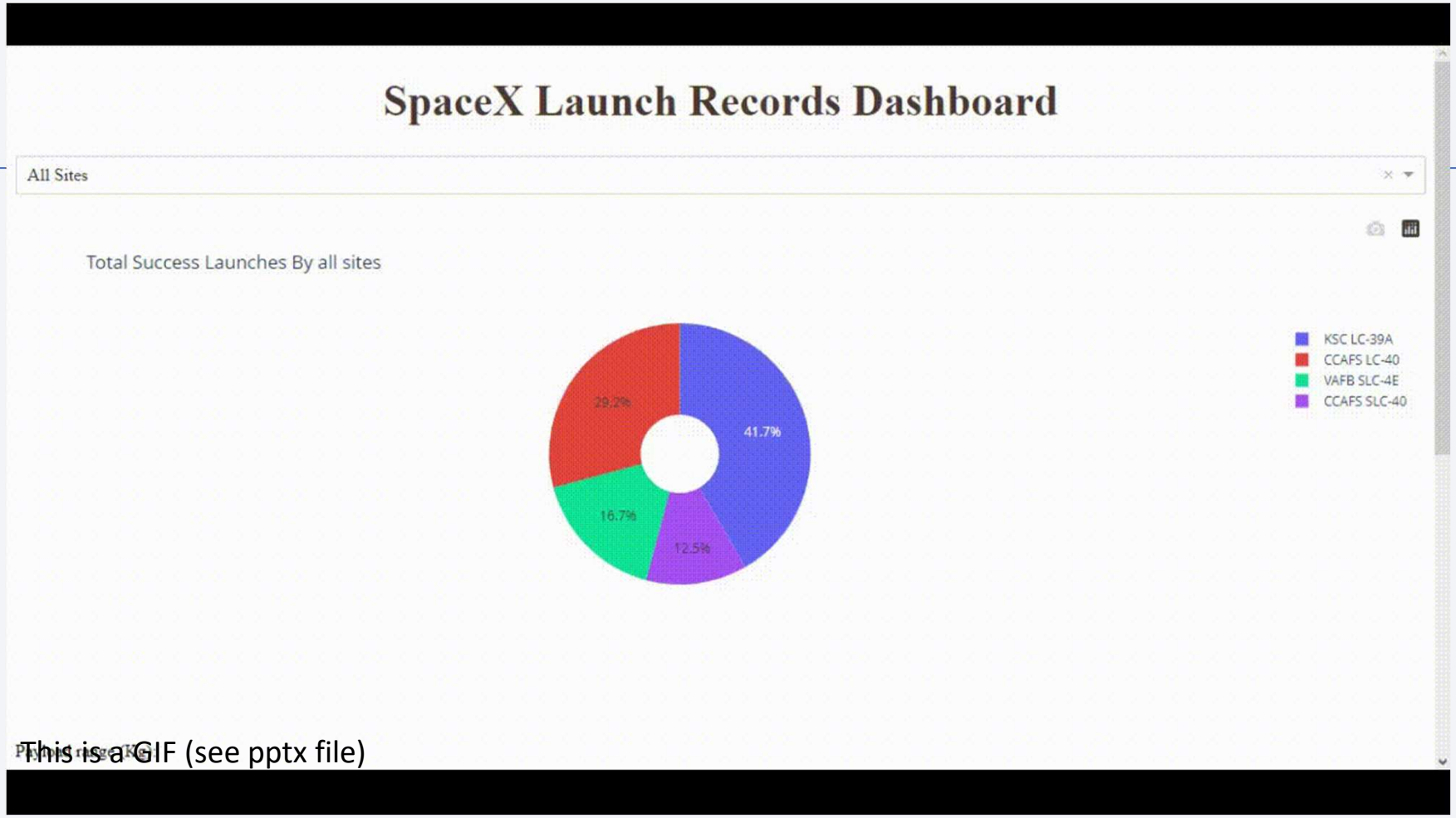






Section 4

Build a Dashboard with Plotly Dash



This is a GIF (see pptx file)

- We can see the success percentage by each sites.
- We can see Payload vs Launch Outcome Scatter Plot

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
algorithms = {'KNN':knn_cv.best_score_,
              'Tree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_}

bestalgorithm = max(algorithms, key=algorithms.get)

print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

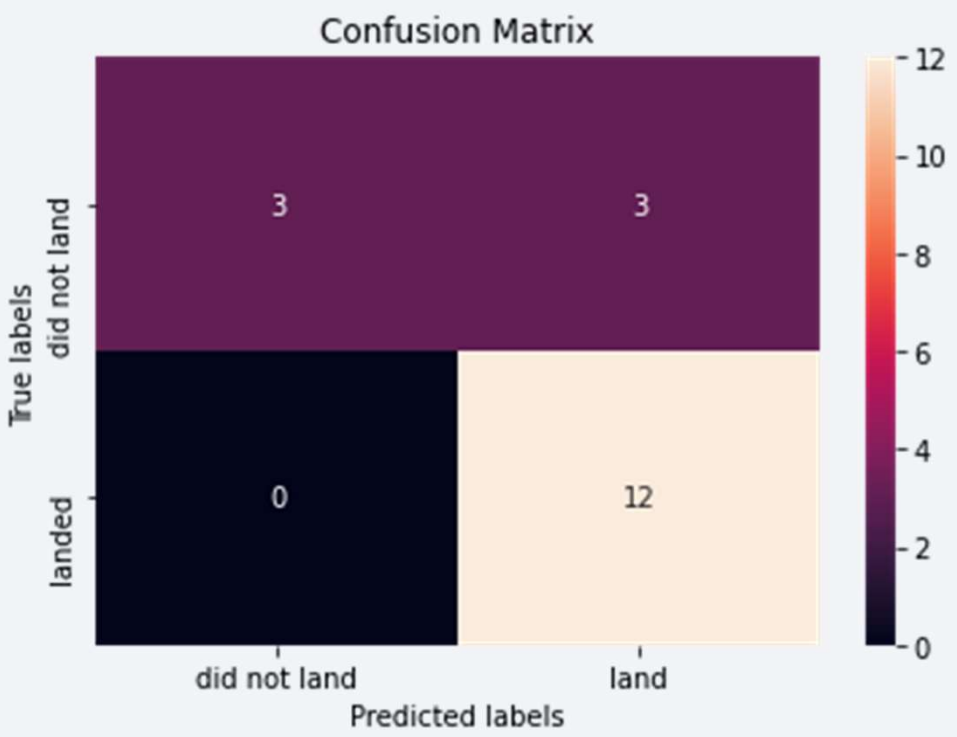
[38]

```
Best Algorithm is Tree with a score of 0.875
Best Params is : {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'best'}
```

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

Confusion Matrix

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



Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC 39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!