IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rocket + Data < Science

Rakesh Goel
12-April-2022

Outline

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

Executive Summary

This document explains various methods that are available to perform analysis of datasets to achieve an intended objective through exploratory, interactive and predictive analysis using various tools and methods available under Data Science. We have taken SpaceX as an example to find the reasons behind their success in rocket science and how the same can be replicated. Following are the key steps and milestones I have followed in the study.

Data

- Data collection
- Data wrangling

Methodologies

- EDA and interactive visual analytics methodology
  - EDA with visualization
  - EDA with SQL
  - Interactive map with Folium
- Predictive analysis (classification) methodology

Results

- EDA with visualization results
- EDA with SQL results
- Interactive map with Folium results
- Plotly Dash dashboard results
- Predictive analysis (classification) results

Conclusion

Introduction

Project background and context

Understanding Success Rate of Rocket Science with Data Science –

With this project, we have taken an attempt to understand the success rate of the iconic Falcon 9 stage-1 to land safely back on earth after a mission. To achieve this we have taken into account the SpaceX launch information available in public domain such as Wikipedia and the API based access that POSTMAN API service has provided.

Business Objectives

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.

Section 1

Methodology

Methodology – Executive Summary

There are three levels of analysis performed

Exploratory Analysis
- Data sources identification of SpaceX launches
- Shortlisting Wikipedia and SpaceXData.com APIs.
- Data extracted and wrangled with Web Scraping, CSV and API calls, and analyzed through Pandas and DB2 SQLs

Interactive Analysis
- Interactive and static visual graphs creations
- Understand and find out observable points of interest for story telling

Predictive Analysis
- Loaded the cleaned and wrangled data
- Performed predictive analysis using different classification methods
- Find out best performing model for adoption

Data Collection
- Launch data through REST APIs from SpaceXData.com portal
- Web scraping from Wikipedia

Data Wrangling & Cleanup
- Data understanding and attributes selection
- One hot encoding & De-normalization
- NaN values replacement

Exploratory data analysis (EDA) using Visualization and SQL
- SQL querying on DB2 data
- Explored launch sites, booster versions, payload capacity and outcomes

EDA with interactive visual analytics
- Interactive graphs with Plotly Dash and associated callback functions
- Various static such as scatter plotting, line graph, bar charts

Predictive analysis using classification methods
- Split the data into train & test data sets to build the model
- Used LR, SVC, Decision Tree and KNN classifications

Data Collection

Data collection methodology:

- Launch data available through REST APIs from SpaceXData.com portal
- Automated data collection from Wikipedia

Following are the steps that describe how data sets were collected

- SpaceX launch data was pulled in using **SpaceX (POSTMAN) REST API**, that included data for lunch sites, launch dates, rocket booster versions, payload, landing details and outcome among others.
- Alternate way to collate data was to use **web scrapping** to pull data from **Wikipedia**, which was also explored.

Flow chart of the data collection process

1 Live datasets (JSON)

https://api.spacexdata.com/v4/rockets/
https://api.spacexdata.com/v4/launchpads/
https://api.spacexdata.com/v4/payloads/
https://api.spacexdata.com/v4/cores/

\+

Static Response Object for Consistency

https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json

OR

Live Data of Past Launches (JSON)

https://api.spacexdata.com/v4/launches/past

2 De-normalization Functions

```
getBoosterVersion
getLaunchSite
getPayloadData
getCoreData
```

Data Selection and Cleanup

3 Normalized DataFrame

4 Filter & cleanup

5
- **Falcon 9** data
- NaN replaced with mean values

Dataset Creation

6 Dataset Output For Wrangling & OneHot Coding to perform
- Interactive analysis
- Predictive analysis

Data Collection – SpaceX API

- For the APIs calls, the get method was used to pull in data in the JSON format.
- As the data available on API server is normalized, a set of functions created to join the data together, after filtering (keeping Falcon 9 rows) the needed data in main data set.
- Replaced the blank data in PayLoadMass attribute with mean value to finalize the dataframe for next step.

Past launches data collection API call

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

De-normalizing the data

```
getBoosterVersion
getLaunchSite
getPayloadData
getCoreData
```

Data wrangling and cleanup example

```
# We will remove rows with multiple cores because those
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]
```

Example of de-normalizing function through API calls

```
# Takes the dataset and uses the rocket column to call the API and append the data to t
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json(
        BoosterVersion.append(response['name'])
```

Keeping only Falcon 9 data, deleting rest

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

```
# Calculate the mean value of PayloadMass column
payloadmass_avg = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, payloadmass_avg, inplace=True)
```

Replacing the NaN values with "mean" value of PayLoadMass data

Data Collection Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/Final%20Capstone%20Project%20-%20Spacex%20Data%20Collection.ipynb

Data Collection - Scraping

Following are the steps followed for the web scraping:

- Identified the source data page and studied the tables present on the page. Identified the table location for the Falcon9 data table to be scraped.
- Created data object and loaded the data through get call
- Used BeautifulSoup object to convert the raw data into structured one.
- Extracted the data headers and rows as described in the flowchart

Wikipedia page for scraping

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Page scraping through get call

```
# use requests.get() method with the provided static_url
# assign the response to a object
falcon9_page = requests.get(static_url).text
```

BeautifulSoup object to get structured data object

```
# Use BeautifulSoup() to create a BeautifulSoup object
soup = BeautifulSoup(falcon9_page, 'html5lib')
```

Extracting the third table from html5 object

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

Table header extraction

```
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

Extracting the data from each row

```
for table_number,table in enumerate(soup.find_all('table',"wikitabl
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number correspon
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])
```

Created the DataFrame of extracted data

```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

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

Web Scraping Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/Final%20Capstone%20Project%20-%20Webscraping.ipynb

Data Wrangling

Following are the steps taken while doing data wrangling:

- After analyzing the booster landing data, it is observed that there were few instances of failed landing attempts and the mission outcome was marked as unsuccessful.
- Data had attribute related to ground pad landings, True RTLS meant successful landing, where as False RTLS meant unsuccessful landing. Similar is the case for ASDS values for landing outcome on drone ship.
- It was logical to perform hot encoding on Outcome attributes, where 1 means the booster successfully landed 0 means it was unsuccessful.
- The encoding was added to a new attribute named as "class" in the dataframe

```
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
```

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

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

```
df['Class']=landing_class
df[['Class']].head(8)
```

Data Wrangling Github Notebook link

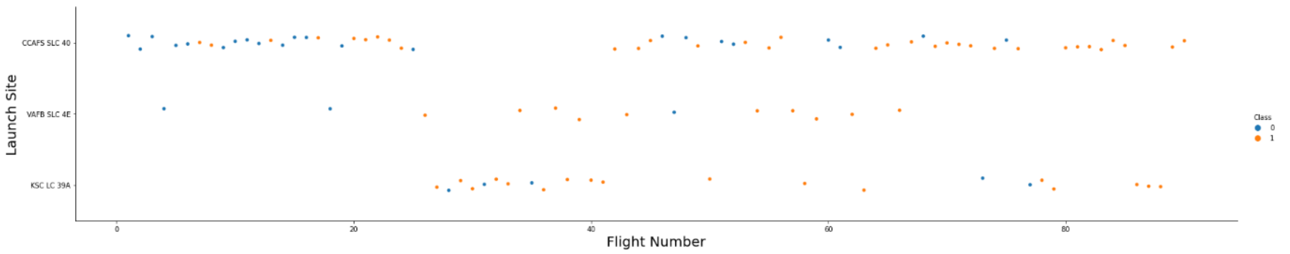
https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/labs-jupyter-spacex-Data-wrangling.ipynb

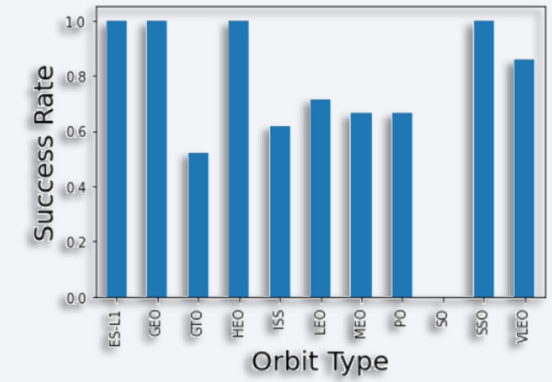
EDA with Data Visualization

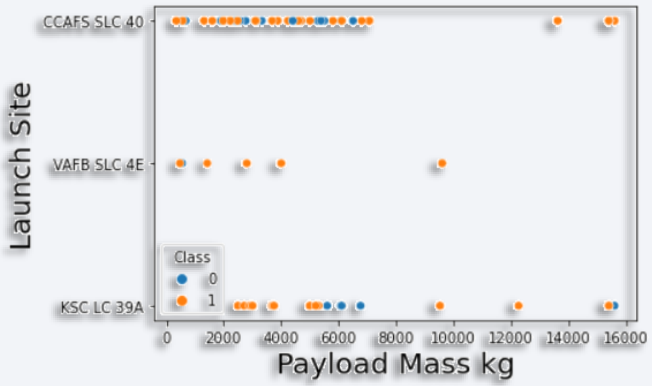
Steps followed for EDA with Data Visualization using MatplotLib and seaborn are:

- Loaded the data after data wrangling where we introduced “class” attribute having hot encoding
- Created scatter plot on overall data
- Scatter plot to visualize the relationship between
    - Flight Number and Launch Site
    - Payload and Launch Site
    - FlightNumber and Orbit type
- Bar chart to visualize relationship between success rate of each orbit type
- Line chart to visualize launch success yearly trend
- Created dummy variable set to apply OneHotEncoder to the column Orbits, LaunchSite, LandingPad, and Serial

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







Data Visualisation with Plotly - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

Steps followed for performing EDA with SQL are:

- Manually loaded the CSV file data table using the database console LOAD tool in DB2. Updated the date/timestamp formats, and datatype for payloadmass attribute.
- Loaded the SQL magic library, established connection with DB2 instance where CSV data is loaded

Performed following tasks

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass. Use a subquery
- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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

EDA with SQL - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

Following objects were added on the map

- Marker for all launch sites on a map
- With help of marker_cluster, marked success/failed launches for each site, green marker representing the success while red as failure.
- Added features, such as mouse curser tracking, showing lat/long values to help identify the location of any valuable point of interest near the launch sites, such as below. Then drawing the polylines to indicate the distance.
    - closest_highway = 28.56335, -80.57085
    - closest_railroad = 28.57206, -80.58525
    - closest_city = 28.10473, -80.64531
- Drawing polylines to show distances w.r.t. to key locations such as coastline.

Reason for using interactive maps and marking the objects

Successful launches also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations.

Interactive Map with Folium - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/lab_jupyter_launch_site_location.ipynb

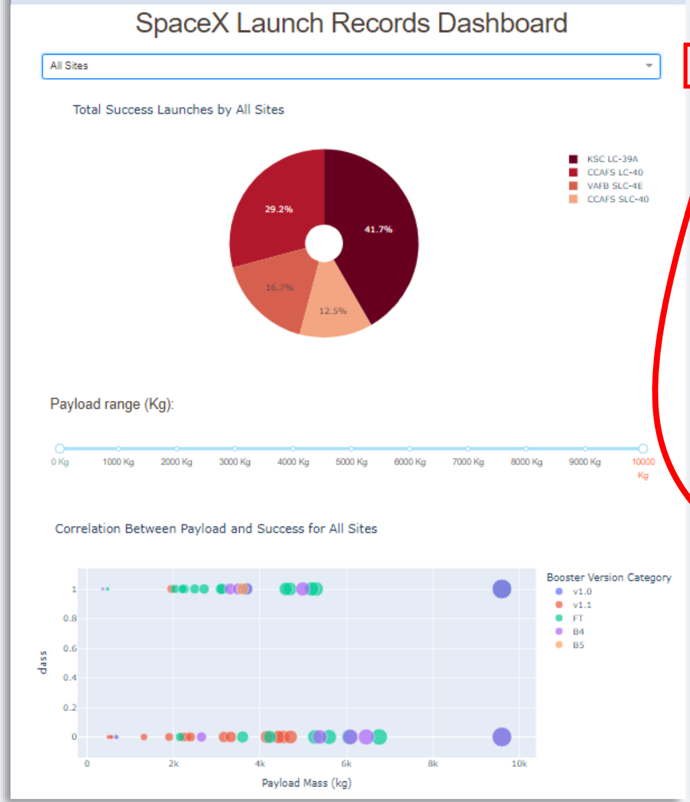
Build a Dashboard with Plotly Dash

Dashboard application created using dash and Plotly libraries, that contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart. This interactive map helped in to know data faster and better while applying filter interactively

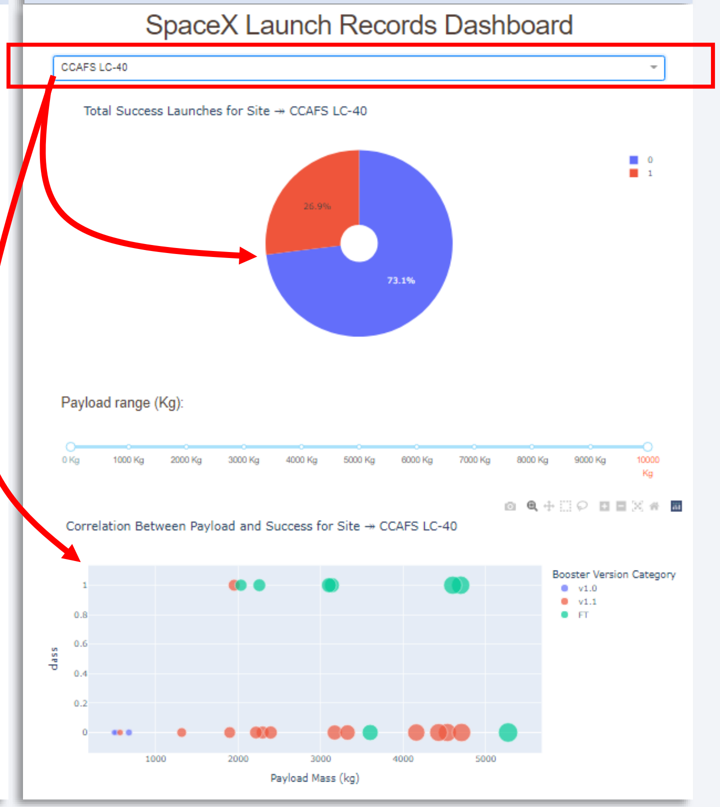
Example code snippet

```
def update_piegraph(site_dropdown):
    if (site_dropdown == 'All Sites' or site_dropdown == 'None'):
        all_sites  = spacex_df[spacex_df['class'] == 1] # All Success only for all sites.
        fig = px.pie(
                all_sites,
                names = 'Launch Site',
                title = 'Total Success Launches by All Sites',
                hole = .2,
                color_discrete_sequence = px.colors.sequential.RdBu
            )
    else:
        site_specific  = spacex_df.loc[spacex_df['Launch Site'] == site_dropdown]
        fig = px.pie(
                site_specific,
                names = 'class',
                title = 'Total Success Launches for Site &#8608; '+site_dropdown,
                hole = .2
            )
    return fig
```

Interactive Dashboard: All Sites



Interactive Dashboard: Filter Applied Site = CCAFS LC-40



Available Filters through call-back function

- Drop Down: Sites
- Multi-Site Selection Checkboxes
- Range Slider: Payload
- Multi-Booster Version Selection Checkboxes

Data Visualisation with Plotly Dash - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/plotly-dash.py

Predictive Analysis (Classification)

Steps Followed

Model building

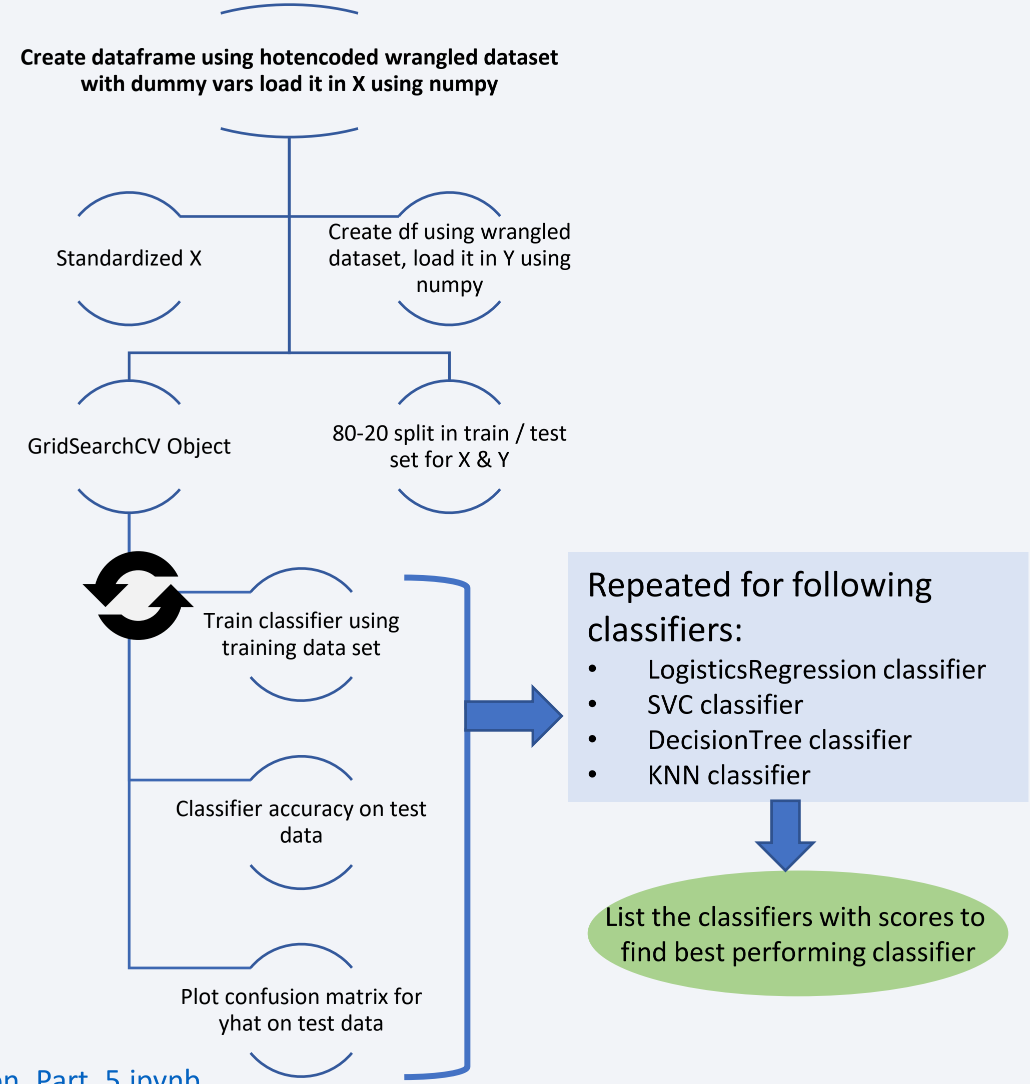
- Load the wrangled and hotencoded datasets in dataframe. Load it in X and Y using numpy.
- Standardize the data in X using sklearn transform method.
- Split X, Y datasets in train and test in 80% 20% ratio respectively
- Create GridSearchCV object with each of the listed classifier (as shown in flowchart) classifier keeping cv=10
- Fit it with training dataset

Model evaluation

- Check the trained model accuracy using test dataset
- Validate yhat with confusion matrix

Model improvement

- After building the models for LR, SVC, DT and KNN classifiers. Validate the accuracy with full dataset
- Run through above steps after tweaking grid parameters associated with each classifier
- List out the best performing model based on best scoring



Predictive Analysis - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/SpaceX_Machine-Learning-Prediction_Part_5.ipynb

Results

- Exploratory data analysis results

- Interactive analytics demo in screenshots

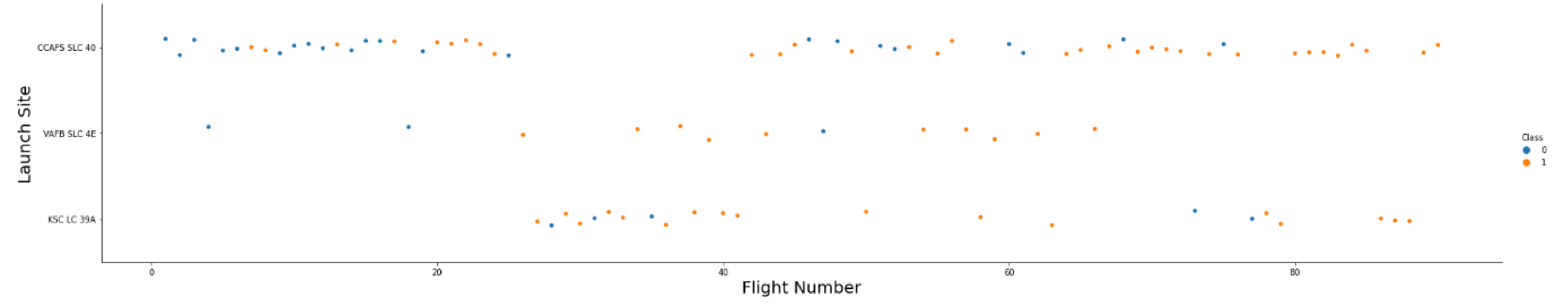
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

Scatter plot of Flight Number vs. Launch Site



Observations:

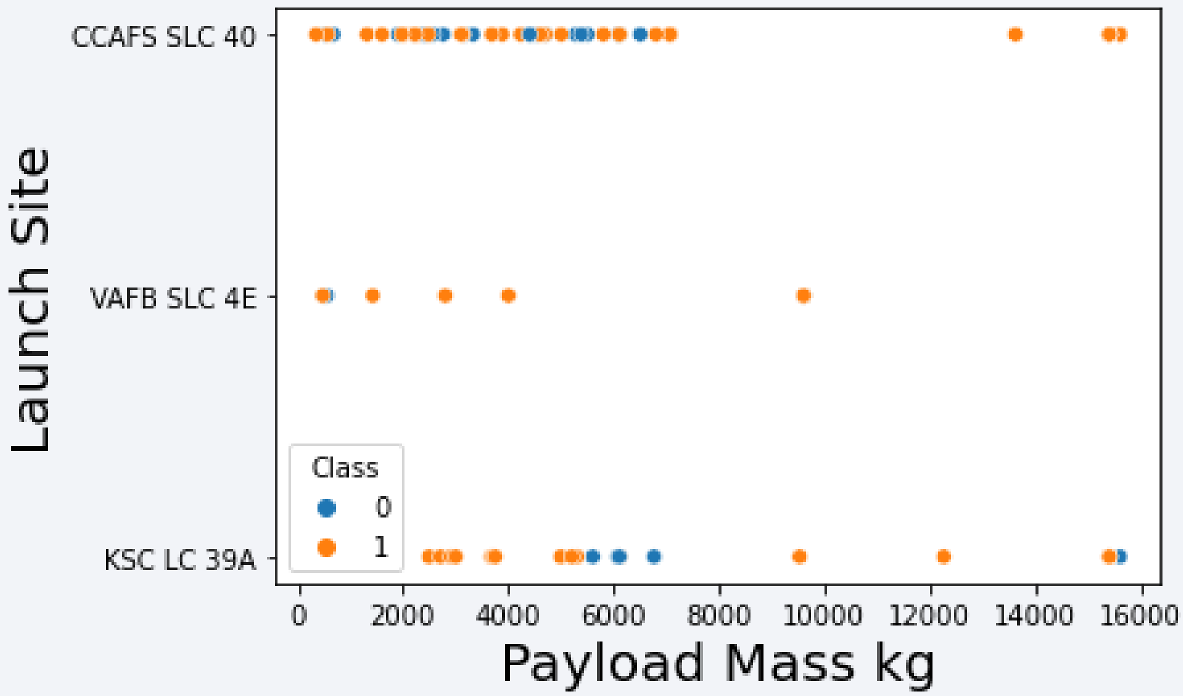
- CCAFS LC-40 launch site was busiest in first 20 flights and after 40 flights
- CCAFS LC-40 launch site had a failure rate of >75% for the first 20 flights
- VAFB SLC 4E site is least utilized, and not used at all after overall ~70th flight
- Currently two sites are operational KSC LC-39A and CCAFS LC-40, that did not see any failure after ~78th flight

Payload vs. Launch Site

Observations:

- VAFB-SLC launch site is not used for launches with heavy payload mass(greater than 10000)
- CCAFS LC-40 launch site has success rate of 100% for launches with heavy payloads >10000
- CCAFS LC-40 launch site is not used for payloads between 8000 to 13000

Scatter plot of Payload vs. Launch Site



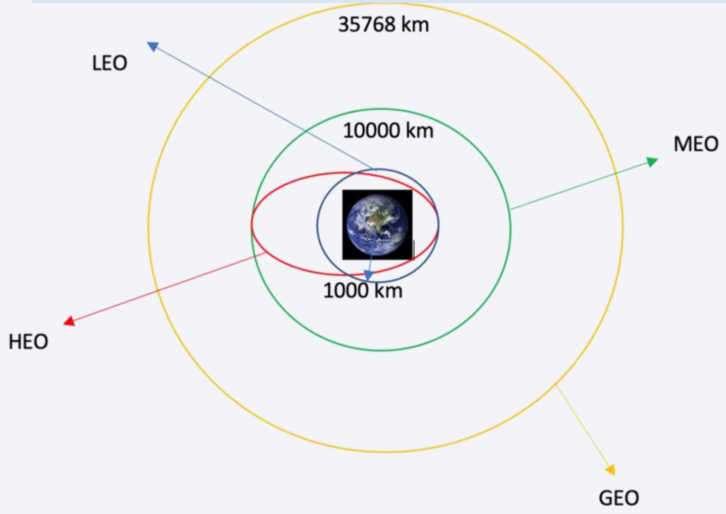
Success Rate vs. Orbit Type

Observations:

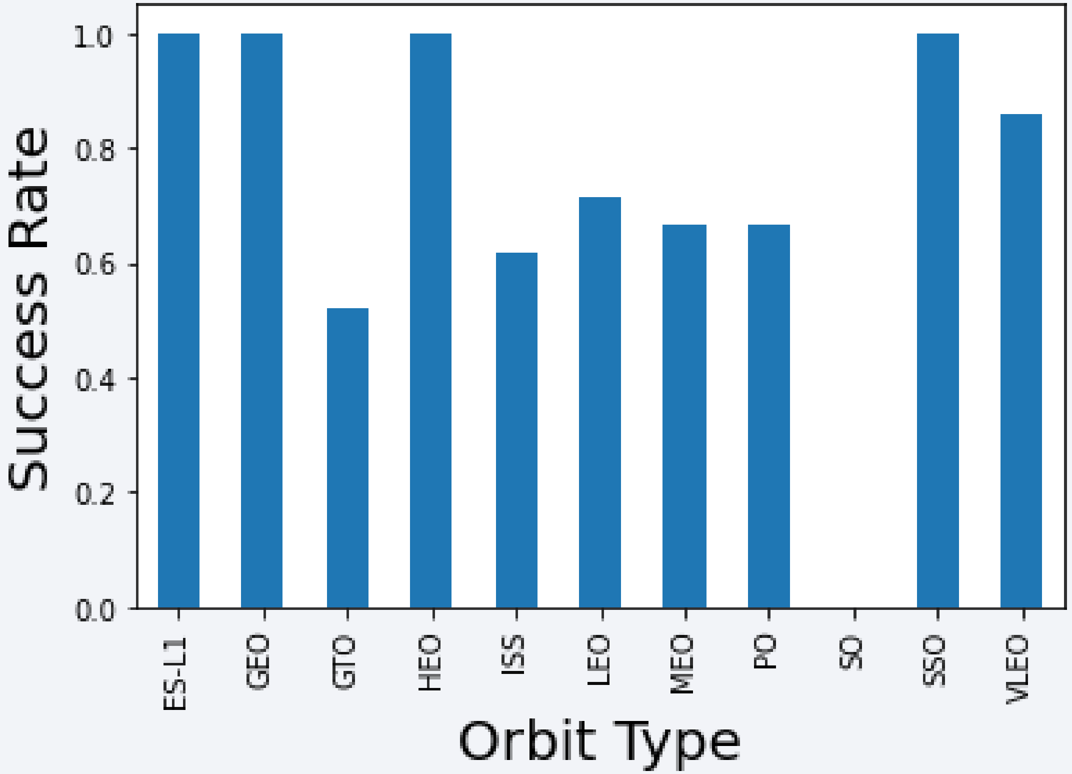
- ESL1, GEO, HEO and SSO orbits have better success rate compared with other orbits.

- There is no launch for SO orbit

Orbital map for reference

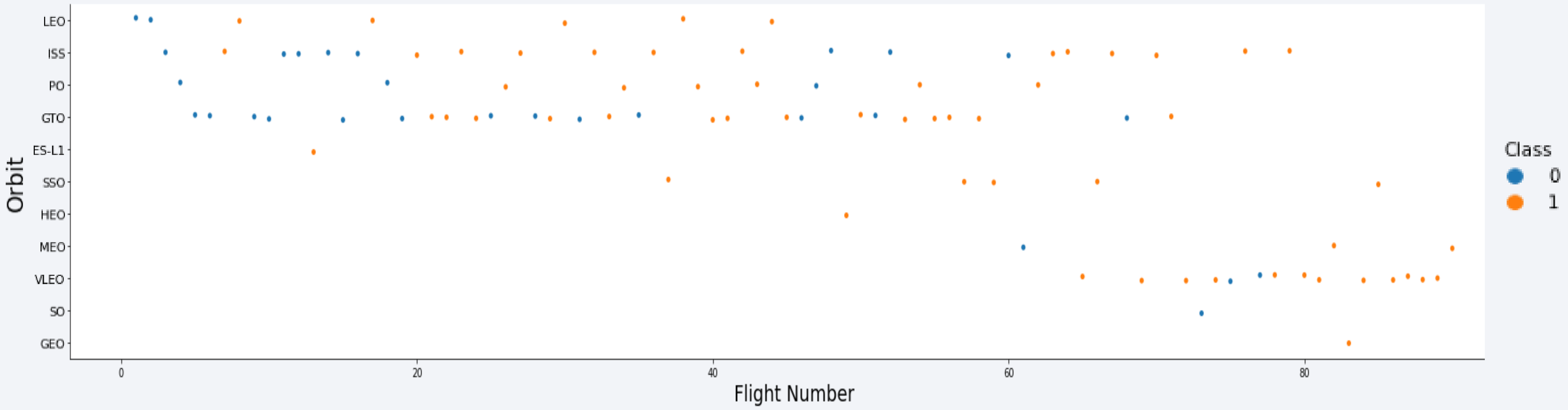


Bar chart for the success rate of each orbit type



Flight Number vs. Orbit Type

Scatter Plot of Flight Number vs. Orbit type

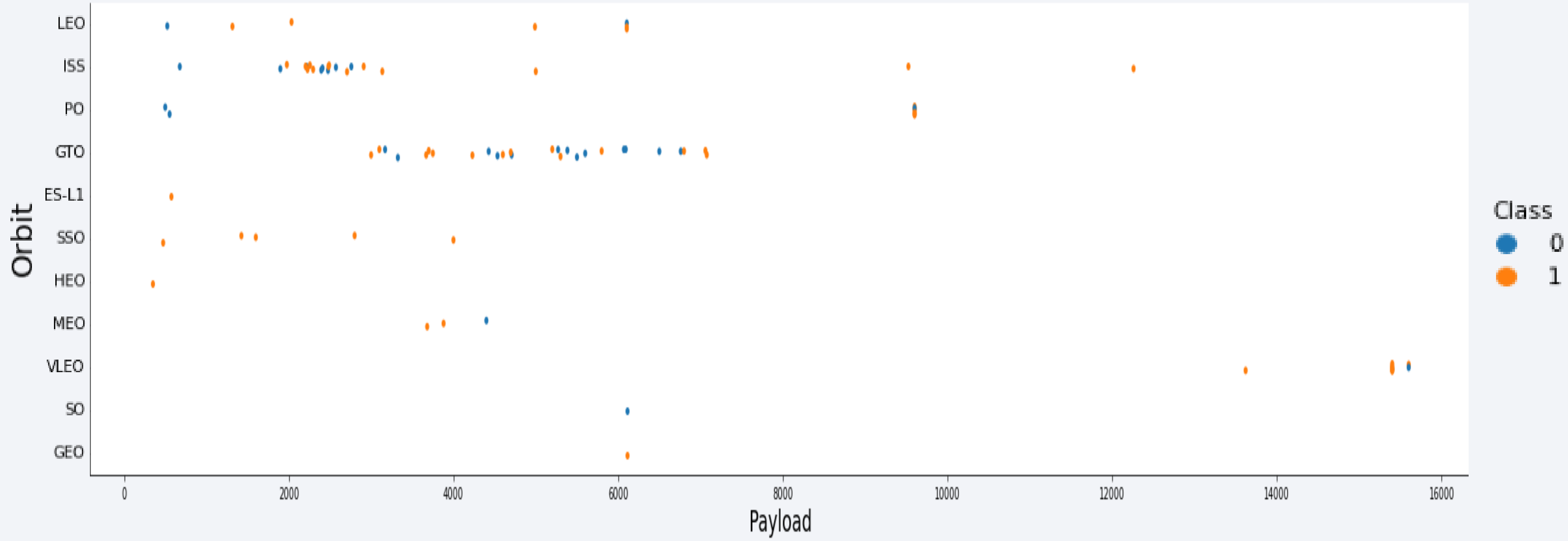


Observations:

- In the LEO orbit the Success appears related to the number of flights
- There seems to be no relationship between flight number when in GTO orbit

Payload vs. Orbit Type

Scatter Plot of Payload Vs Orbit Type

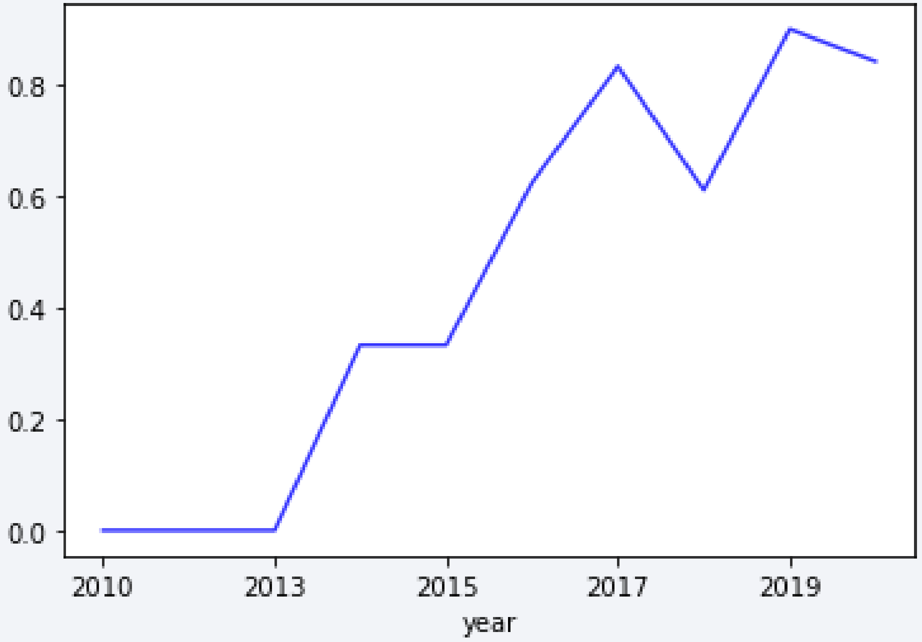


Observations:

- 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) and scattered scross.

Launch Success Yearly Trend

Line chart of yearly average success rate



Observations:

- Success rate since 2013 kept increasing till 2020 with a slight dip seen during 2018

All Launch Site Names

```
In [6]:
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXDATASET;

 * ibm_db_sa://hjx74932:***@b0ae
Done.
```

Out[6]:

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

The SQL statement used Distinct function to list all the unique values in the LAUNCH_SITE column.

Site Details:

1. **LC-40 and SLC-40, both are same site**. As information available in Wikipedia says, Space Launch Complex 40 (SLC-40), previously Launch Complex 40 (LC-40) is a launch pad for rockets located at the north end of Cape Canaveral Space Force Station, Florida.
2. **KSC LC-39A** - Kennedy Space Center Launch Complex 39A, located at NASA's Kennedy Space Center in Merritt Island, Florida
3. **VAFB SLC-4E** - Vandenberg Space Launch Complex 4, located at Vandenberg Space Force Base, California, U.S.

Launch Site Names Begin with 'CCA'

```
%%sql
SELECT *
FROM SPACEXDATASET
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

- The SQL statement used “Like” operator in the where clause, in that we defined “CCA” text as fixed, followed by “%” symbol representing any text.

Out[7]:

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

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

- The SQL statement used SUM function on "PAYLOAD_MASS__KG_" attribute, which is an **integer** type. A filter was applied using where clause to show total payload mass launched by NASA (CRS).
- The result of the query was **45596**, i.e. 45596 KG was the total payload mass launched by NASA (CRS)

Task 3

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

In [8]:

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

```
 * ibm_db_sa://hjx74932:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c3r
Done.
```

Out[8]:

1
45596

Average Payload Mass by F9 v1.1

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

- The SQL statement used AVG function on “PAYLOAD_MASS__KG_” attribute, which is an **integer** type, to calculate the average payload mass. A filter was applied in where clause along with “LIKE” operator to show average payload mass carried by Booster names starting with F9 v1.1.
- The result of the query was **2534**, i.e. 2534 KG was the average payload mass carried by F9 v1.1

Task 4

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

In [9]:
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXDATASET
WHERE Booster_Version LIKE 'F9 v1.1%';
```

```
 * ibm_db_sa://hjx74932:***@b0aebb68-94fa-46ec-a1fc-1c999e
Done.
```

Out[9]:

1
2534

First Successful Ground Landing Date

```
%%sql
SELECT MIN(Date)
FROM SPACEXDATASET
WHERE Landing__Outcome = 'Success (ground pad)';
```

Task 5

List the date when the first successful landing outcome in g

Hint:Use min function

```
In [10]:
%%sql
SELECT MIN(Date)
FROM SPACEXDATASET
WHERE Landing__Outcome = 'Success (ground pad)';

 * ibm_db_sa://hjx74932:***@b0aebb68-94fa-46ec-a1fc-
Done.

Out[10]:
1
2015-12-22
```

- The SQL statement used MIN function on "Date" attribute, which is Date data type, to find the first launch date. A filter was applied on "Landing__Outcome" attribute in where clause to limit the records in scope to "Success (ground pad)".
- The result of the query was **2015-12-22**, i.e. the first launch that landed safely on the ground pad happened 22nd December 2015.

One of the key landmark event for the mankind in space exploration in sustainable way!! ☺

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXDATASET
WHERE LANDING__OUTCOME = 'Success (drone ship)'
    AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000);
```

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

- The SQL statement selected the "booster_version" attributed, while adding filter on Landing_Outcome and PayLoad_Mass__KG_ attributes, to limit the search results to Successful landings on drone ship for which mass was between 4000 AND 6000

Total Number of Successful and Failure Mission Outcomes

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXDATASET
GROUP BY MISSION_OUTCOME;
```

Task 7

List the total number of successful and failure mission outcomes

In [12]:

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXDATASET
GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://hjx74932:***@b0aebb68-94fa-46ec-a1fc-1c999edb6187.c
Done.
```

Out[12]:

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

- The SQL statement selected the "mission_outcome" attributed and COUNT function to calculate the grouped occurances for each unique mission outcome
- GROUP BY clause was used to define which attribute to be grouped while counting the occurrences.

Boosters Carried Maximum Payload

```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXDATASET
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXDATASET) ORDER BY BOOSTER_VERSION;
```

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

- To get the Booster_Version for the maimum payload delivered, we used sub-query option.
- The main query string had selection criteria on DISTINCT Booster_Version, where Payload mass attributed is equated to the result of the sub-query.
- In the Sub-query, a maximum value identified in the PayLoad mass attributed is 15600 KG, that is carried by primarily **F9 B5** boosters with different sub-variants.

2015 Launch Records

```
%%sql
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXDATASET
WHERE Landing__Outcome = 'Failure (drone ship)'
    AND YEAR(DATE) = 2015;
```

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

- We selected the LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE attributes, by applying the data filter using where clause
- The data selection was limited to Failure happened on drone ships using LANDING__OUTCOME. Additional filter included with AND operator to limit the results only for the year 2015 using YEAR function applied on DATE attribute.

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

In [15]:
```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_COUNT
FROM SPACEXDATASET
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY TOTAL_COUNT DESC
```

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

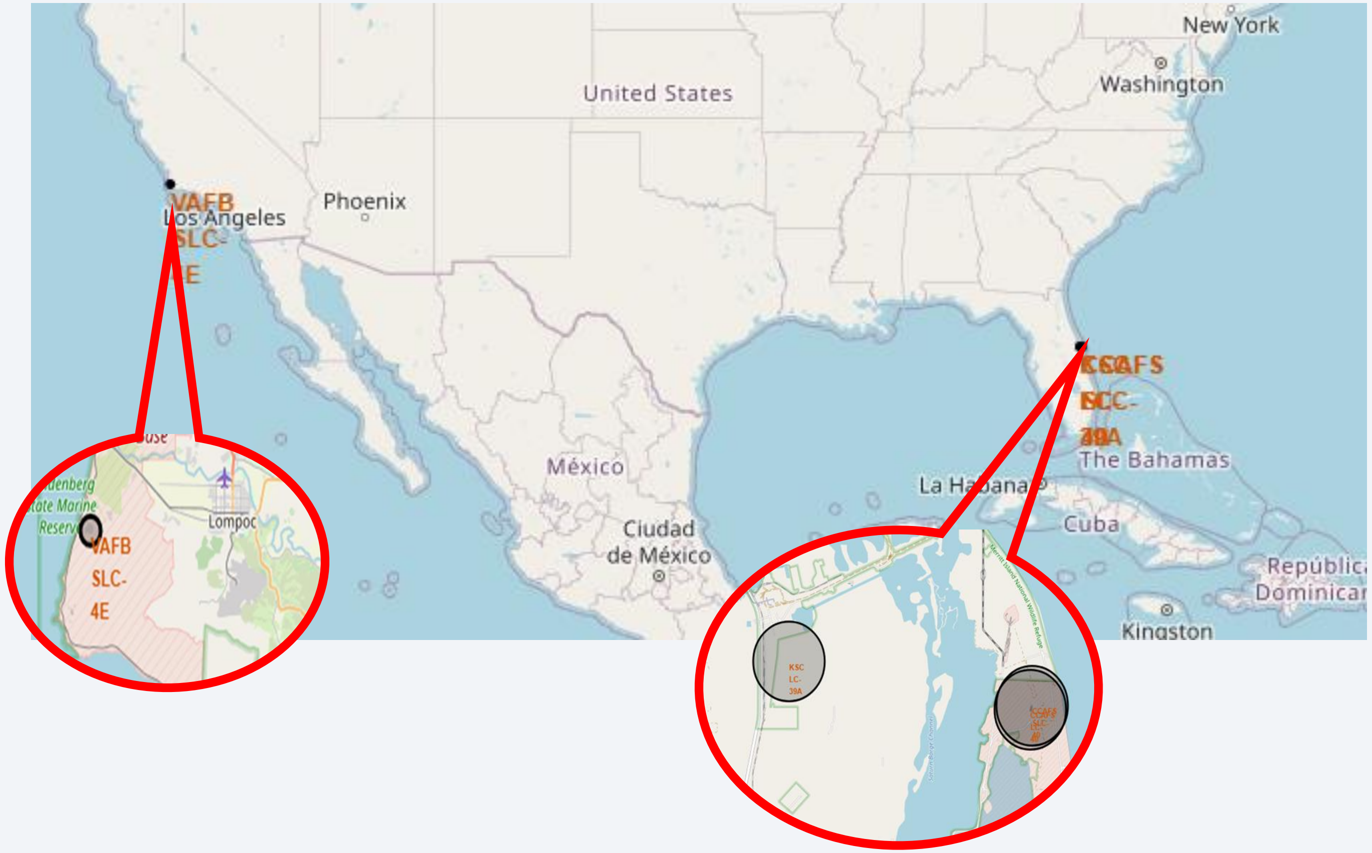
- We selected the LANDING__OUTCOME and calculated the count for LANDING__OUTCOME using the GROUP BY clause on the same attribute.
- The COUNT function result was given a name "TOTAL_COUNT"
- The data selection was limited to the event dates on DATE attribute using BETWEEN operator for the given dates.
- ORDER BY clause with DESC operator was used to list the results in order on TOTAL_COUNT alias given to the count of LANDING__OUTCOME attribute.

Section 3

Launch Sites Proximities Analysis

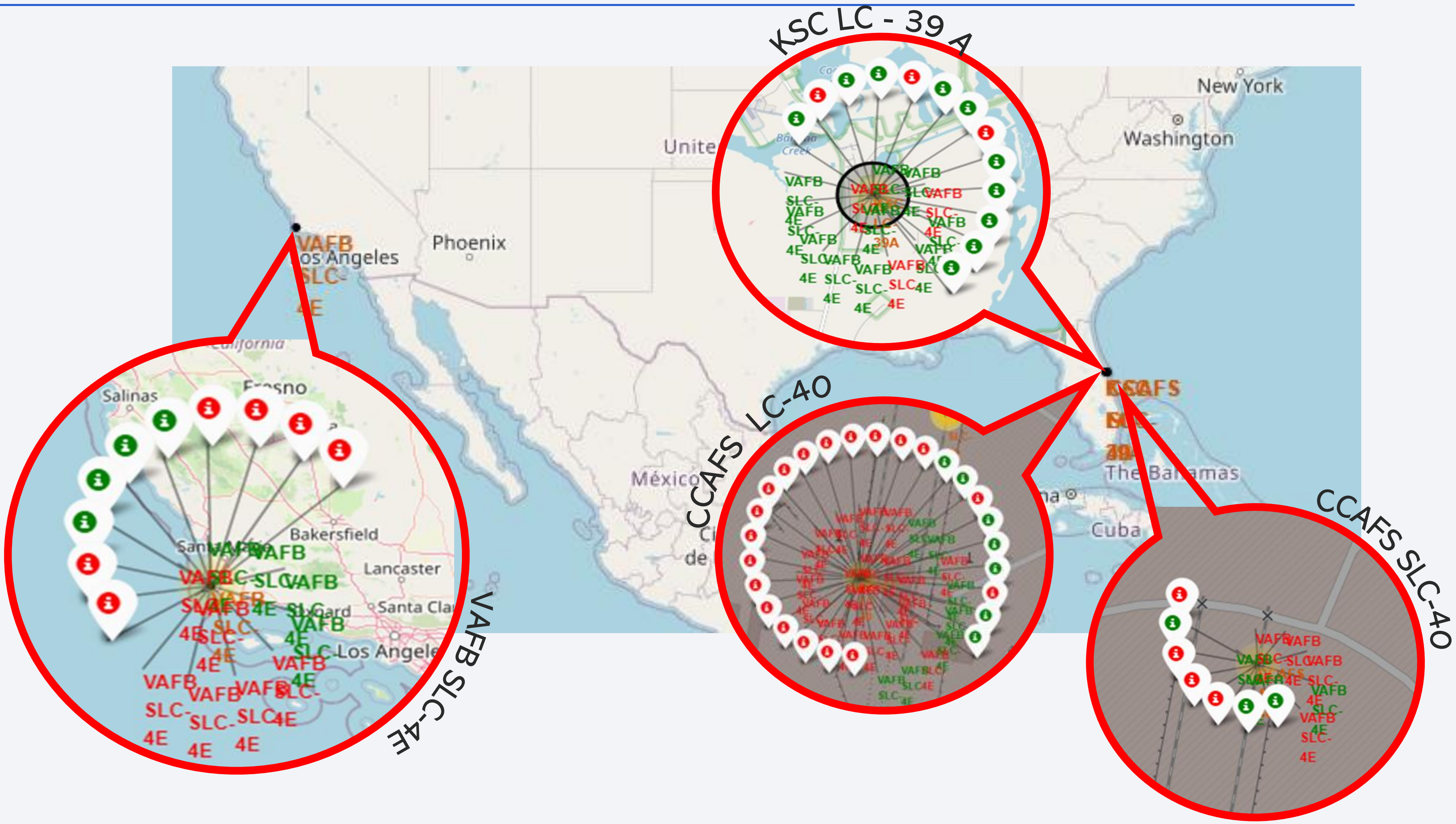
Folium Map For All Launch Sites

- We can see that one of the launch site is facing the PACIFIC Ocean, while three are located in Florida facing the Atlantic Ocean. Based on the Orbital coverage & objectives, launch site can be decided accordingly



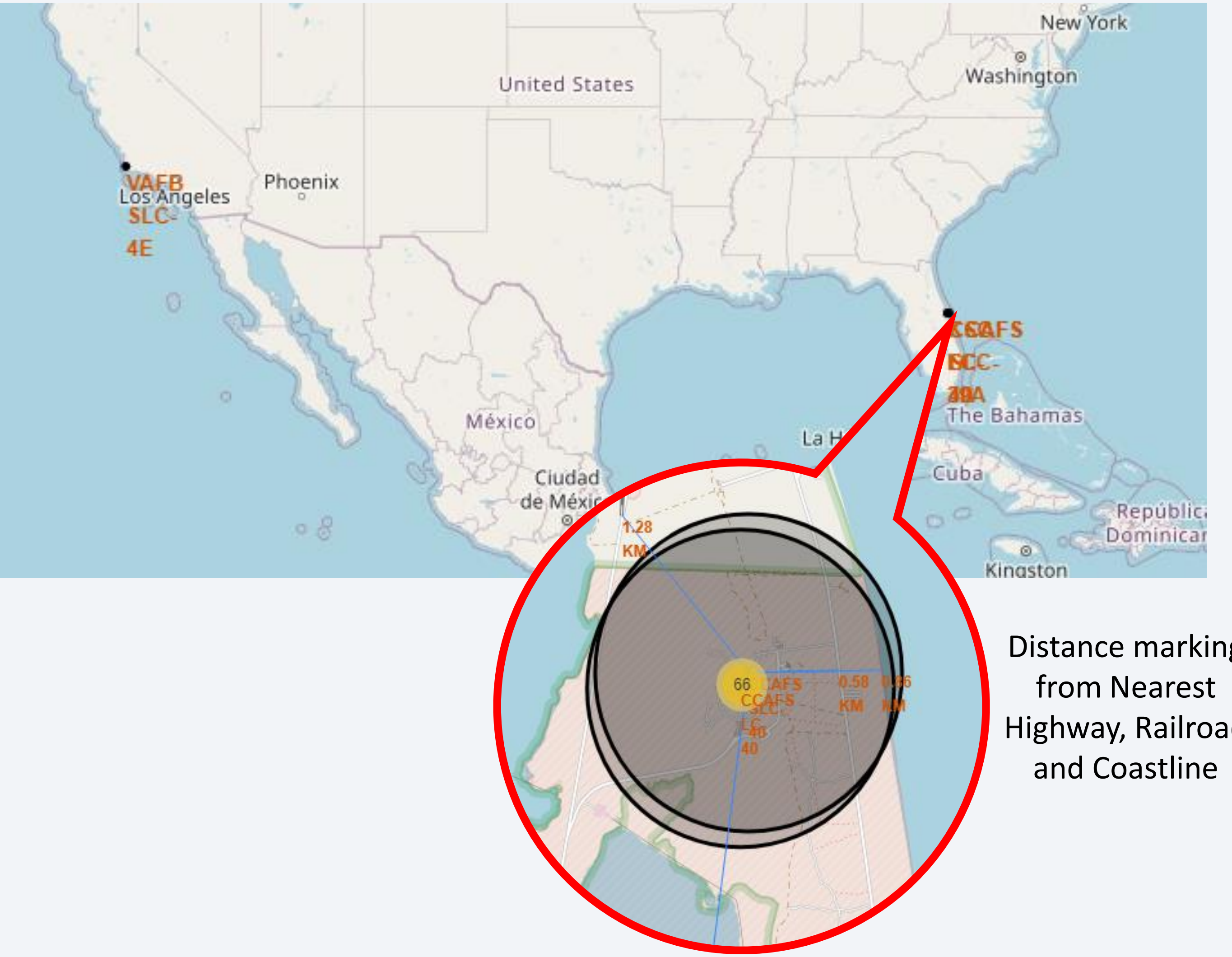
Launch Sites With Outcome in Color Coding

- We can see that Majority of the launched happened at CCAFS LC-40 did not have successful outcome.
- Whereas KSC LC-39A has better outcome as it is evident in the Interactive Dashboard as well.



Calculating and Showing Distance Markings for POI

- In the distance marking, we can see that CCAFS LC-40 is 1.28KM away from the nearest railroad, while 0.86 KM away from the coastline.



Distance marking from Nearest Highway, Railroad and Coastline

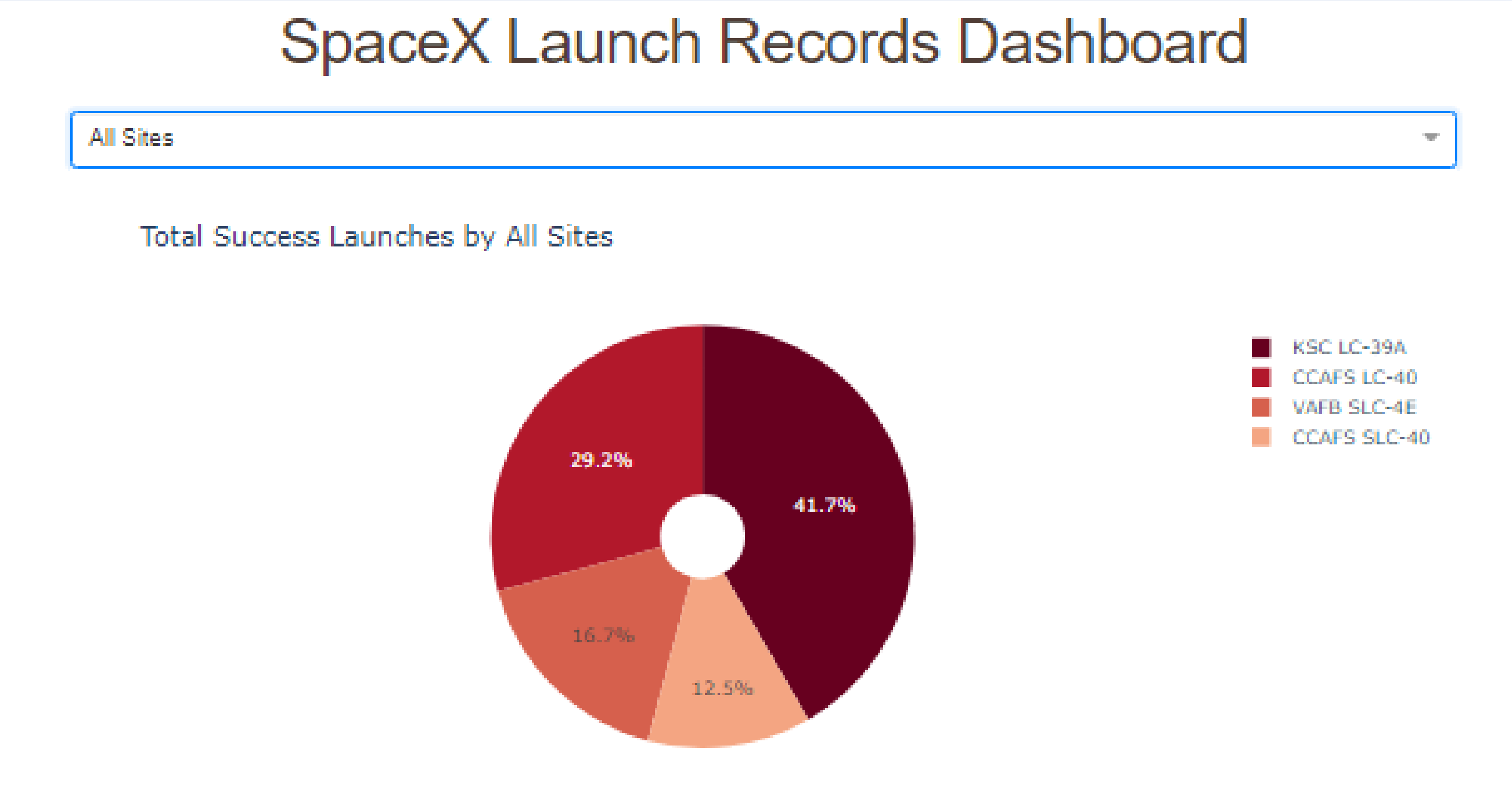
Section 4

Build a Dashboard with Plotly Dash

Successful Launch Ratio Comparison for Each Site

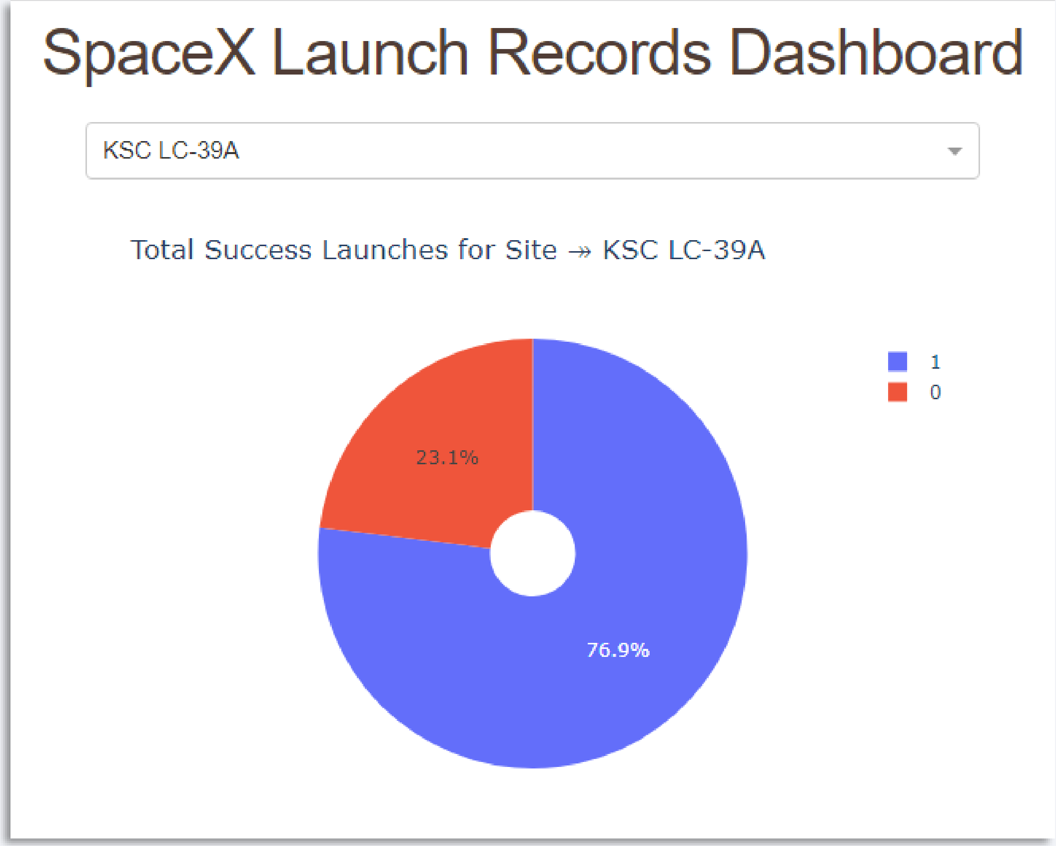
- Maximum successful launches done are at Kennedy Space Center Launch Complex 39A (KSC LC-39A)
- CCAFS LC-40 & CCAFS SLC-40 launch sites (which is same) have contribution of combined success rate of 41.7% which is surprisingly equal to that of KSC LC-39A
- Lowest success rate is of VAFB SLC-4E site

Interactive Dashboard: All Sites



Success vs. Failure Ratio Comparison for KSC LC-39A Site

- Pie graph has covered KSC LC-39A launch site, that has maximum success ratio
- 76% of overall launches taken place at KSC LC-39A site had been successful

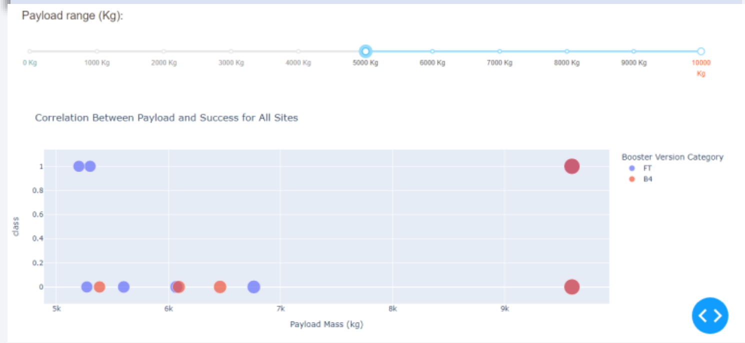


Payload vs. Launch Outcome Plot with Selection Slider for EDA

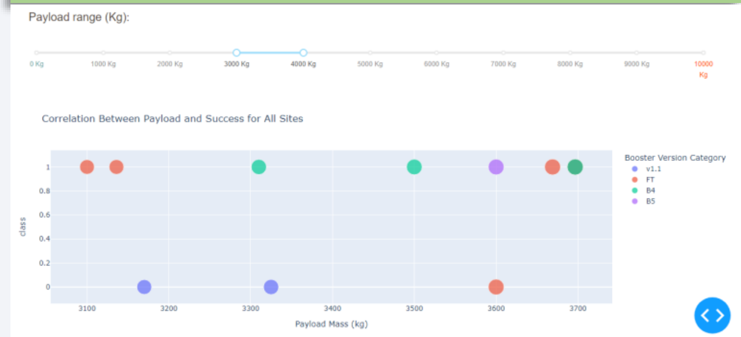
- The scatterplot graphs shown here are for the success rate for each booster category for a payload mass range, selected through slider.
- We can see that the payload range of 3,000 to 4,000 has better success rate compared with other payload range selected

Scatterplot for 0 to 10,000 KG Payload

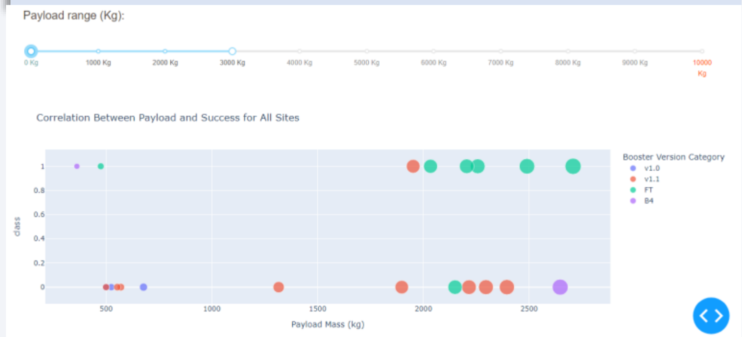
Scatterplot for 5,000 to 10,000 KG Payload



Scatterplot for 3,000 to 4,000 KG Payload



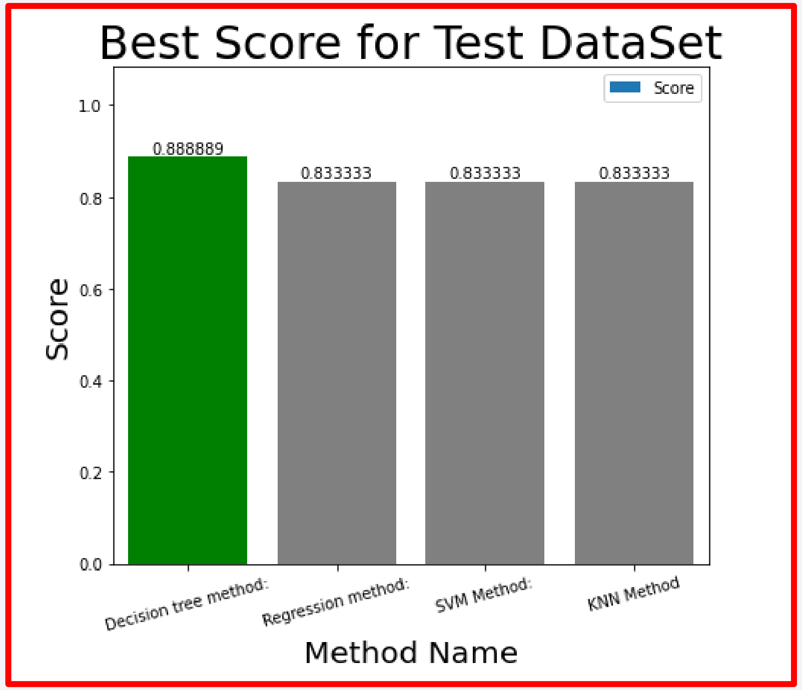
Scatterplot for 0 to 3,000 KG Payload

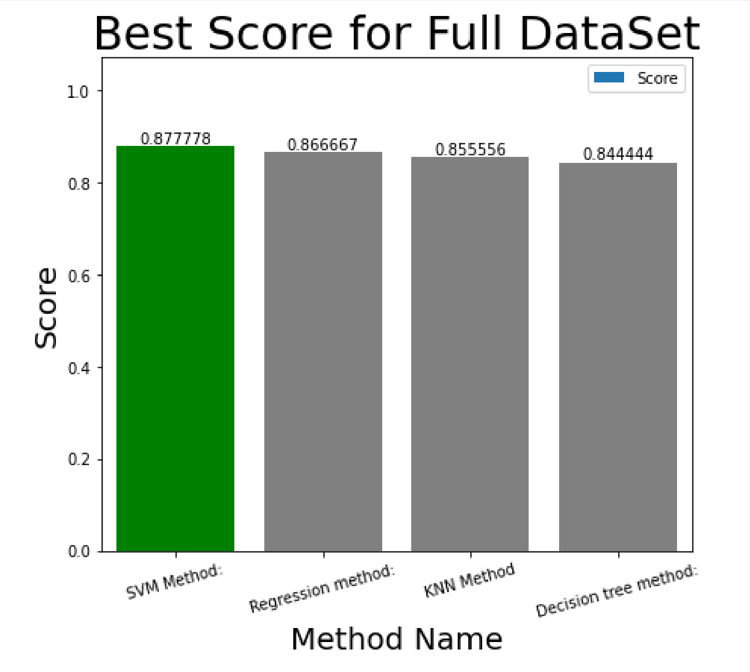


Section 5

Predictive Analysis (Classification)

Classification Accuracy





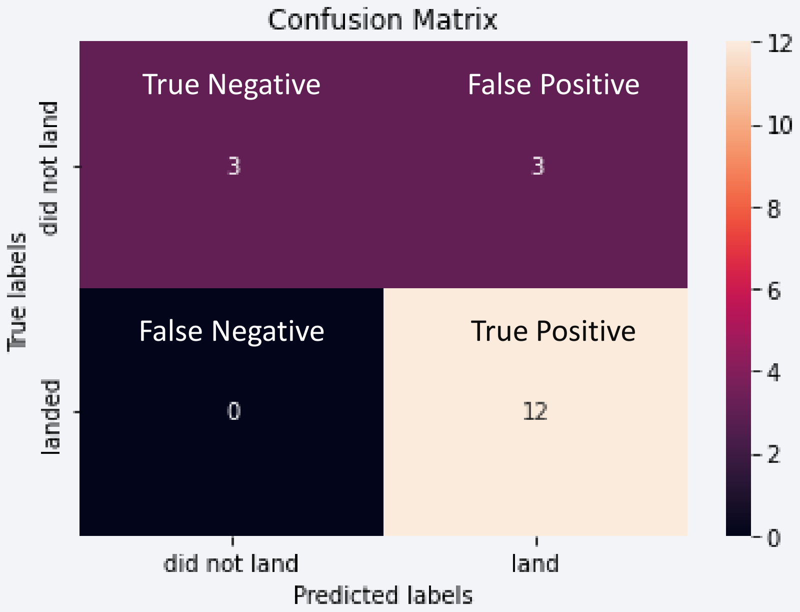
- Classification accuracy for Test data set was consistent and equal for the LR, SVM and KNN model, however for Decision Tree model it was variable due to max_depth parameter set to random. In the last run, surprisingly, Decision Tree gave the best result.
- When validated the best score using the full data set on trained model, SVM model found to be the best performing.

Predictive Analysis - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/SpaceX_Machine-Learning-Prediction_Part_5.ipynb

Confusion Matrix

- Confusion Matrix of all the four models gave same outlook.
- There are few false positive cases also observed in each of the model.



Conclusions

- Launch Pads are located on east coast as well as west coast, providing option for Eastern trajectory or the Western trajectory, to help in achieving intended objectives.
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS orbits.
- Overall ESL1, GEO, HEO and SSO orbits have better success rate compared with other orbits.
- Location of launch sites is an important factor for it’s proximity with the coastline, railroads, highways and cities.
- KSC LC-39A launch site has better outcome as it is evident in the Interactive Dashboard
- Decision Tree classifier performed well, whereas other classifiers also can’t be ruled out as they were also equally good when compared on the Confusion Matrix

Appendix

Data Collection Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/Final%20Capstone%20Project%20-%20Spacex%20Data%20Collection.ipynb

Web Scraping Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/Final%20Capstone%20Project%20-%20Webscraping.ipynb

Data Wrangling Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/labs-jupyter-spacex-Data-wrangling.ipynb

Data Visualisation with Plotly - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/jupyter-labs-eda-dataviz.ipynb

EDA with SQL - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/jupyter-labs-eda-sql-coursera.ipynb

Interactive Map with Folium - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/lab_jupyter_launch_site_location.ipynb

Data Visualisation with Plotly Dash - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/plotly-dash.py

Predictive Analysis - Github Notebook link

https://github.com/rakesh-goel/IBM-Data-Science-Project/blob/master/SpaceX_Machine-Learning-Prediction_Part_5.ipynb

Thank you!