Winning Space Race with Data Science

Alper Sener

30/12/2021

Outline

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

Executive Summary

- The Methodology Path is as follows:
  - Data collection from SPACEX API
  - Data wrangling from Web (Wikipedia)
  - Explanatory Data Analysis (EDA) with data visualization
  - EDA with SQL by using IBM Cloud
  - Building an interactive map with Folium
  - Predictive analysis (Classification)
- Summary of all results
  - First, data is collected from different sources.
  - Second, data is analyzed through different methods.
  - Last, predictive analysis utilized to find the best model for our data.

Introduction

- Project background and context
  - In this capstone, we have predicted if the Falcon 9 first stage will land successfully.
  - 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 are the features of successful rockets that are landed without destruction.
  - Finding out the historic trends from SpaceX's experiences according to location, orbit et cetera.

Section 1

Methodology

Methodology

- Data collection methodology:
  - Via the API of SpaceX (REST)
  - Via Webscrapping (Wikipedia)
- Perform data wrangling
  - Data is transformed by numpy for Machine Learning algorithms.
  - Moreover, One Hot Encoding is utilized to have numeric/binary data.
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Subset of data is obtained through SQL and the initial analysis has done through visuals such as, scatter plots and bar charts.
- Perform interactive visual analytics using Folium
- Perform predictive analysis using classification models
  - Several classification models have build and compared.

Data Collection – SpaceX API

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

Getting Response from API

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

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

```
In [10]: response.status_code
Out[10]: 200
```

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

```
In [12]: # Use json_normalize meethod to convert the json result into a dataframe
response.json()
df=pd.json_normalize(response.json())
```

Convert Response as a Json

```
In [19]: # Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
In [20]: BoosterVersion[0:5]
Out[20]: ['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

```
In [22]: # Call getLaunchSite
getLaunchSite(data)

In [23]: # Call getPayloadData
getPayloadData(data)

In [24]: # Call getCoreData
getCoreData(data)
```

```
In [44]: # Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=df_launch_dict[df_launch_dict['BoosterVersion']!='Falcon 1']

In [48]: print(data_falcon9.BoosterVersion.unique())
print(df_launch_dict.BoosterVersion.unique())

['Falcon 9']
['Falcon 1' 'Falcon 9']
```

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

```
In [49]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Apply Custom Functions

Assign list to dictionary then dataframe

```
In [59]: # Calculate the mean value of PayloadMass column
mean_of_data=data_falcon9["PayloadMass"].mean()
# Replace the np.nan values with its mean value
data_falcon9=data_falcon9["PayloadMass"].replace(np.nan,mean_of_data)

data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

Filter dataframe and export to a file

https://github.com/ilyalper/IBM-Applied-Data-Science-Project/tree/main/AppliedDataScienceCapstone/Week1

Data Collection - Scraping

```
In [6]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
In [7]: # use requests.get() method with the provided static_url
        # assign the response to a object
        response= requests.get(url=static_url)
```

```
In [8]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(response.content, 'html5lib')
```

Getting Response from HTML

Creating Beautifulsoup Object

```
In [10]: # 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")
```

Finding tables in the URL

Starting from the third table is our target table contains the actual launch records.

```
In [11]: # Let's print the third table and check its content
         first_launch_table = html_tables[2]
         print(first_launch_table)
```

```
In [114]: column_names = []

          for columnns in first_launch_table.find_all("th"):
              column_name= extract_column_from_header(columnns)
              if column_name is not None:
                  if len(column_name)>0:
                      column_names.append(column_name)
```

Deducing Column Names

```
In [143]: 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
          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']=[]
```

Creating a new dictionary

```
In [145]: extracted_row = 0
          #Extract each table
          for table_number,table in enumerate(soup.find_all('table',"wikitable pla
             # get table row
              for rows in table.find_all("tr"):
                  #check to see if first table heading is as number corresponding
                  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
                      # TODO: Append the flight_number into launch_dict with key `
                      launch_dict['Flight No.'].append(flight_number)
                      #print(flight_number)
                      datatimelist=date_time(row[0])

                      # Date value
                      # TODO: Append the date into launch_dict with key `Date`
                      date = datatimelist[0].strip(',')
                      launch_dict['Date'].append(date)
```

Append data to the related keys

Convert dictionary to a dataframe and export to a file

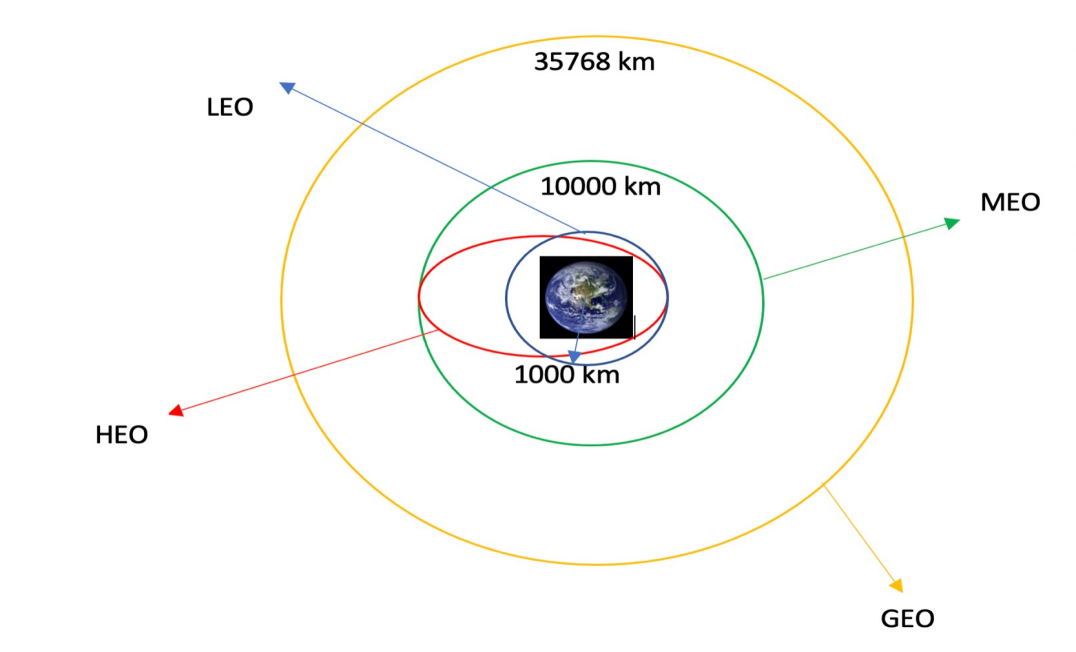
```
In [146]: df=pd.DataFrame(launch_dict)
```

```
In [147]: df.to_csv('spacex_web_scraped.csv', index=False)
```

https://github.com/ilyalper/IBM-Applied-Data-Science-Project/tree/main/AppliedDataScienceCapstone/Week1

Data Wrangling

We have performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.



- Load Data
- Create a dataframe from the Data
- Clean the null and irrelevant values from the data
- Create quantitative columns if there are not (One hot coding)
- Export results to a file

https://github.com/ilyalper/IBM-Applied-Data-Science-Project/tree/main/AppliedDataScienceCapstone/Week1

EDA with Data Visualization

We drawn following scatter plots

- Flight Number & Payload Mass
- Flight Number & Launch Site
- Payload & Launch Site
- Orbit & Flight Number
- Payload & Orbit Type
- Orbit & Payload Mass

We drawn following bar plots

- Success Rate & Orbit Type

We drawn following line charts

- Success Rate & Year

https://github.com/ilyalper/IBM-Applied-Data-Science-Project/tree/main/AppliedDataScienceCapstone/Week2

EDA with SQL

- **Using bullet point format, summarize the SQL queries you performed**
  - The names of the unique launch sites in the space mission
  - Initial 5 records where launch sites begin with the string 'KSC'
  - The total payload mass carried by boosters launched by NASA (CRS)
  - Average payload mass carried by booster version F9 v1.1
  - The date where the successful landing outcome in drone ship was achieved.
  - The names of the boosters which have success in ground pad and have payload mass between 4000 and 6000
  - The total number of successful and failure mission outcomes
  - The names of the booster versions which have carried the maximum payload mass.
  - The records which will display the month names, successful landing outcomes in ground pad, booster versions, launch site for the months in year 2017
  - Ranking the count of successful landing outcomes between two dates in descending order.

https://github.com/ilyalper/IBM-Applied-Data-Science-Project/tree/main/AppliedDataScienceCapstone/Week2

Build an Interactive Map with Folium

- First, we point out the Latitude and Longitude Coordinates of SpaceX sites. Then, we add these as a Circle Marker around each launch site with a label of the name of the launch site.
- It is very good structure to visualize the launch datas.
- Then, we assigned the dataframe classes as 0 and 1 with different colors to have required attention. Moreover, we show them on the map by a MarkerCluster( ).

- Some examples that we analyze:
  - Are launch sites near to railways?
  - Are launch sites near to highways?
  - Are launch sites near to coastline?
  - Do launch sites keep certain distance away from cities?

https://github.com/ilyalper/IBM-Applied-Data-Science-Project/tree/main/AppliedDataScienceCapstone/Week3

Build a Dashboard with Plotly Dash

- I could not write a dashboard…

Predictive Analysis (Classification)

First, we build the model

- Load dataset as Pandas
- Transform it as Numpy Data
- Split data into training and test sets
- Check how many test and train samples are there
- Build different types of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearch objects and train our dataset.

Then, we evaluate the builded models

- Accuracy, f1_score and Jaccard score are calculated for each model
- Hyperparameters are determined for each type of algorithms
- Confusion Matrix

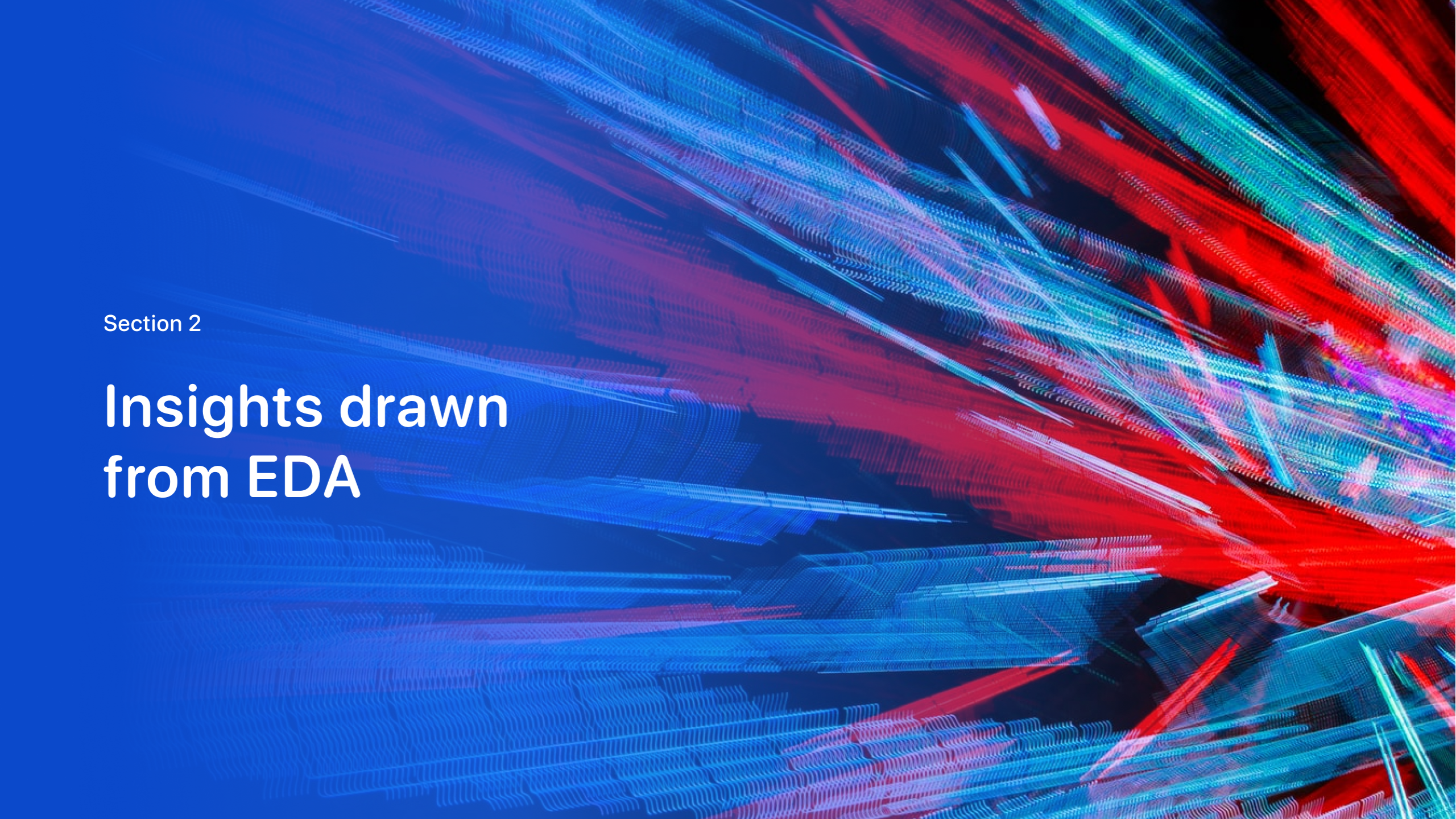
Finally, we find the best performing models

- The model with the best scores (accuracy, f1, jaccard) wins the best performing model

https://github.com/ilyalper/IBM-Applied-Data-Science-Project/tree/main/AppliedDataScienceCapstone/Week4

Results

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



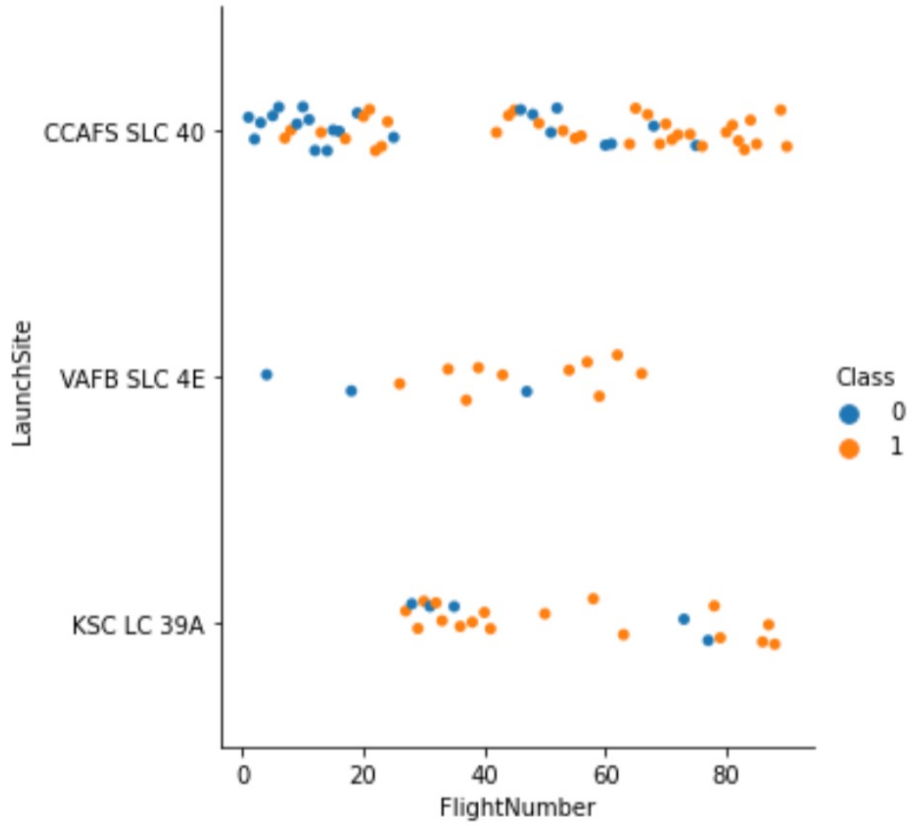
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

Increased number of flights at a launch site is resulted as a greater success rate.

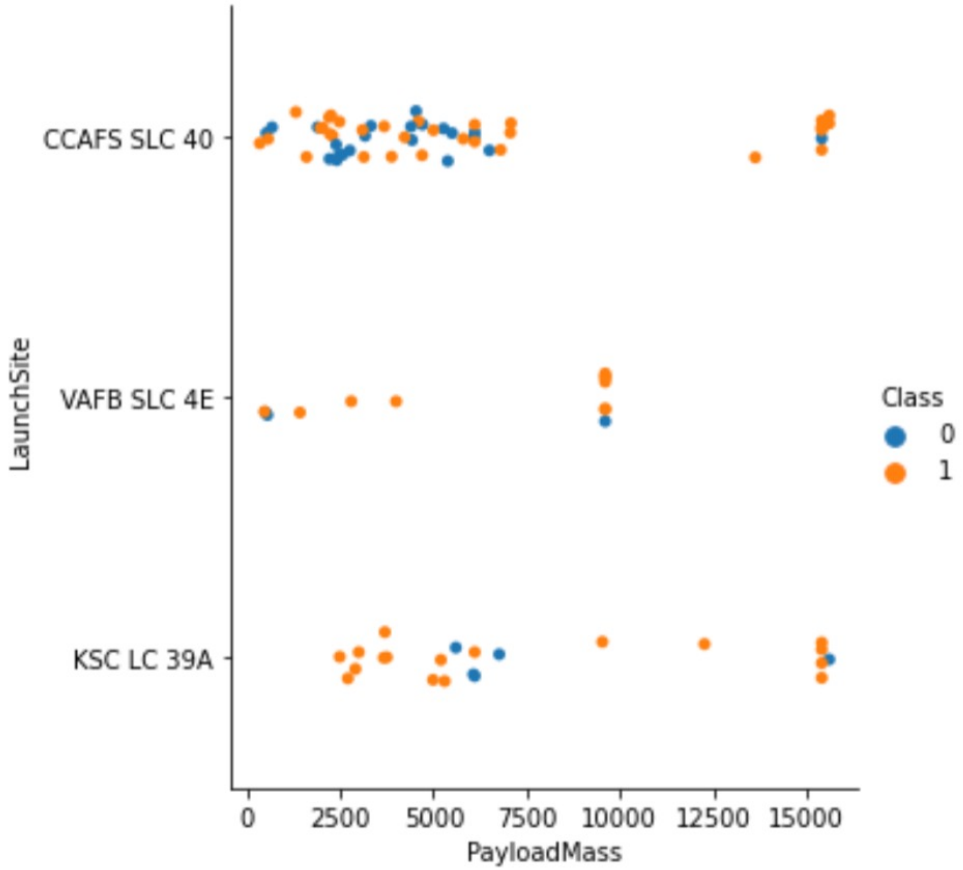
```
In [8]: # Plot a scatter point chart with x axis to be Flight

        g = sns.catplot(x="FlightNumber", y="LaunchSite", hue=
```



Payload vs. Launch Site

- The greater the payload mass for all launch sites gives better launch results.
- Small mass amount in KSC LC is also provides better launches.
- However, a significant pattern cannot be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch.

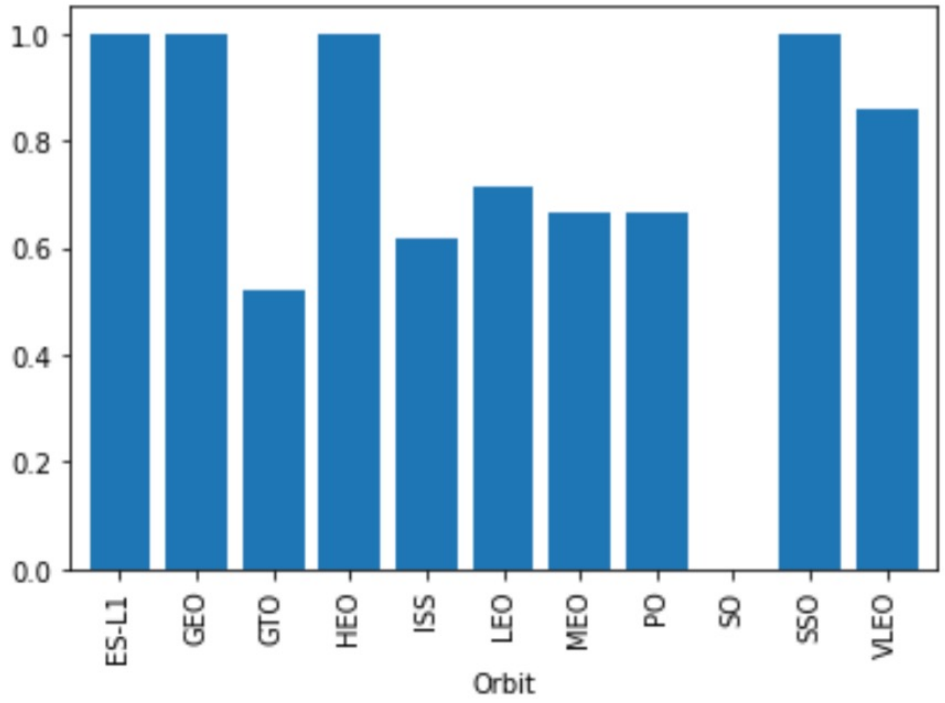
```
In [14]: # Plot a scatter point chart with x axis to be Pay Lo
         g = sns.catplot(x="PayloadMass", y="LaunchSite", hue=
```



Success Rate vs. Orbit Type

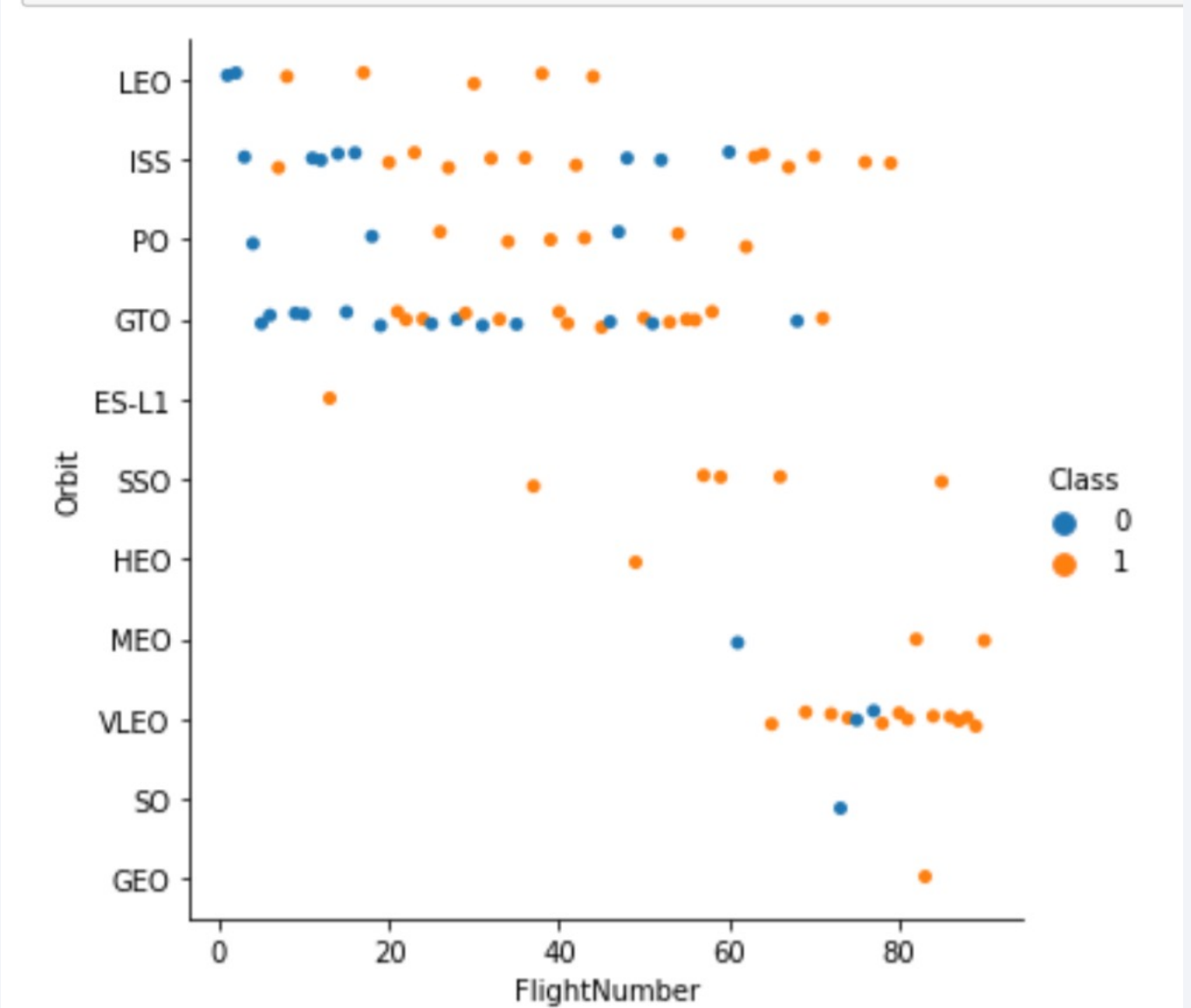
- The orbits with the highest success rate
  - GEO
  - HEO
  - SSO
  - ES-L1
- The orbit with the lowest success rate
  - SO

```
In [40]: # HINT use groupby method on Orbit column and get t
dd=df.groupby("Orbit")["Class"]
dd.mean()
pllot = dd.mean().plot(kind='bar', zorder=2, width=
```



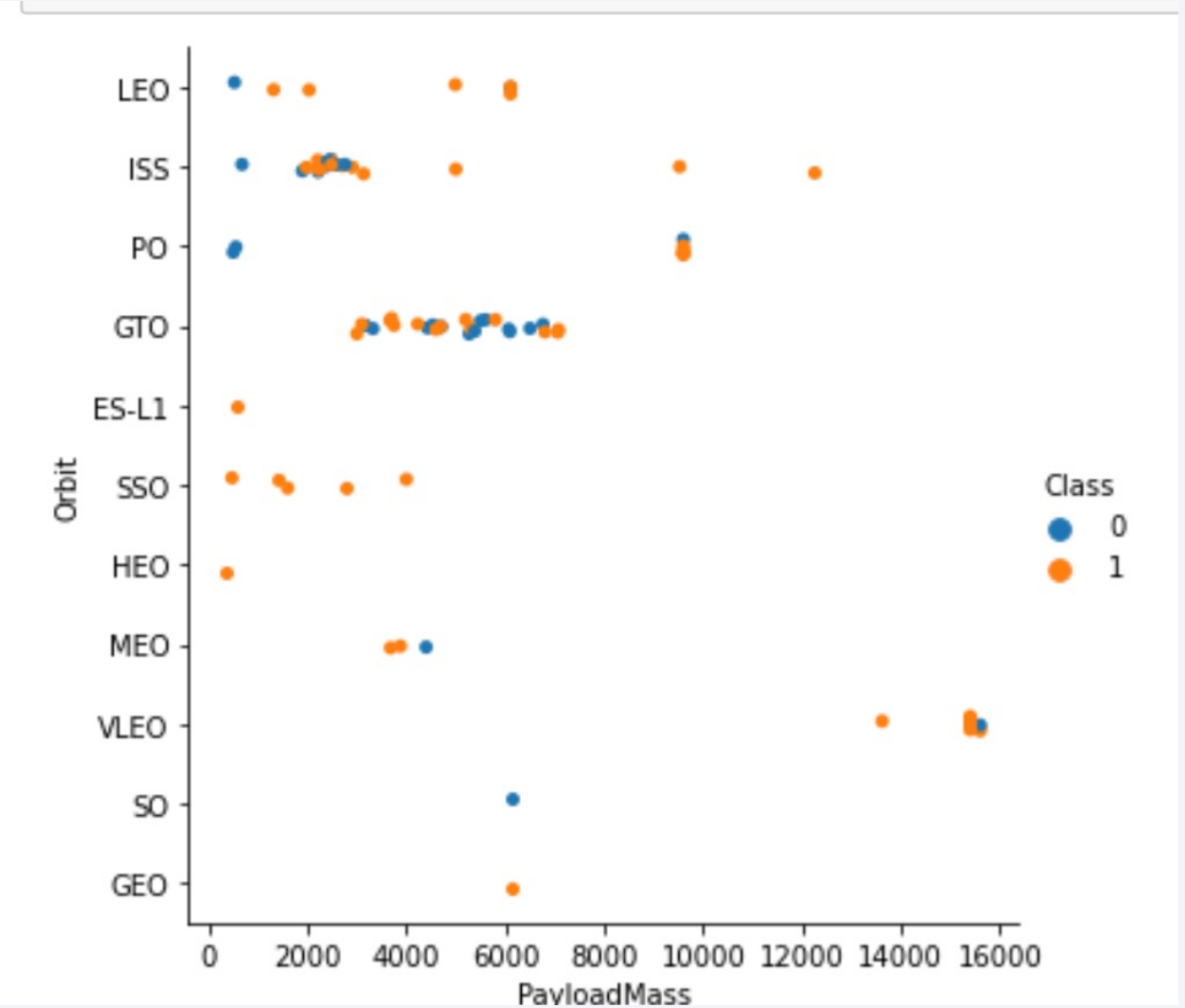
Flight Number vs. Orbit Type

- 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

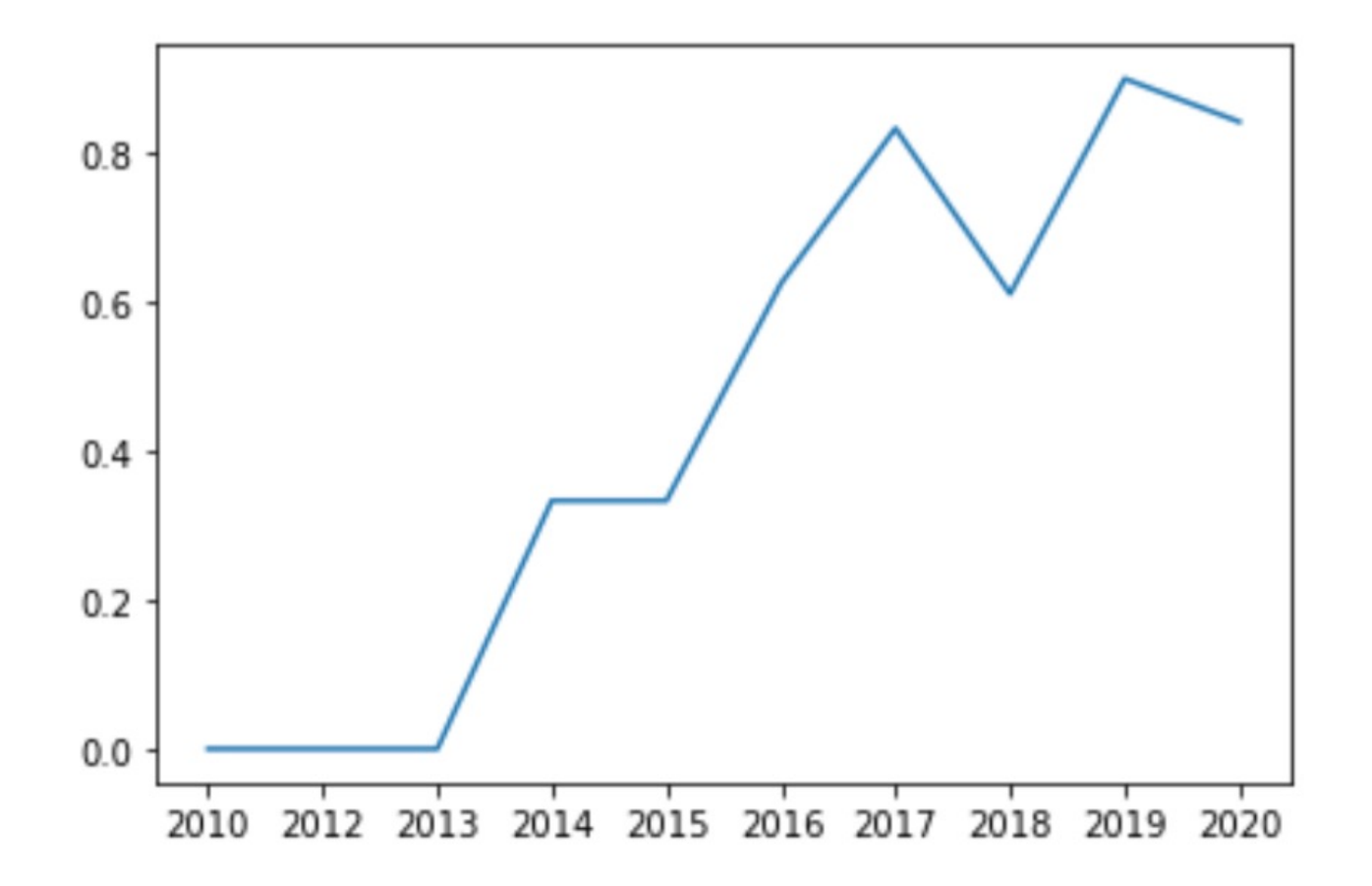
- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



Launch Success Yearly Trend

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

There is no launch data for the year 2011



All Launch Site Names

UNIQUE
in the query means that it will only show Unique values in a column

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

```
In [8]: %sql Select UNIQUE(launch_site) From SPACEXDATA
```

```
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86
98/BLUDB
Done.
```

Out[8]:

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

Launch Site Names Begin with 'CCA'

LIMIT
in the query means that it will only show selected amount of rows in order

%
in a string is utilized to search a certain part of a string

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

In [14]:
```
%sql Select * From SPACEXDATA WHERE launch_site LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB
Done.
```

Out[14]:

DATE	Time (UTC)	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

SUM
in the query means that it will summates the total in the column.

WHERE
clause filters the dataset to only perform calculations on certain rows

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

```
In [26]: %sql Select  SUM(payload_mass__kg_) From SPACEXDATA WHERE customer = 'NASA (CRS)';
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databa
98/BLUDB
Done.
```

Out[26]:

1
45596

Average Payload Mass by F9 v1.1

AVG
in the query means that it will averages the values in a column.

WHERE
clause filters the dataset to only perform calculations on certain rows

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

In [28]:
```
%sql Select  AVG(payload_mass__kg_) as Averagge From SPACEXDATA WHERE booster_version = 'F9 v1.1';
```

```
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appc
98/BLUDB
Done.
```

Out[28]:

averagge
2928

Successful Drone Ship Landing with Payload between 4000 and 6000

AND
clause specifies additional
filter conditions

“<“ “>”
we can do logical operations
in numbered columns

```
In [55]: %sql Select  booster_version From SPACEXDATA WHERE "Landing _Outcome"='Success (drone ship)' \
AND payload_mass__kg_>4000 AND payload_mass__kg_<6000 ;
```

```
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.
98/BLUDB
Done.
```

Out[55]:

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

Total Number of Successful and Failure Mission Outcomes

GROUP BY
clause groups the same values in a certain columns

COUNT
Calculates the total amount of instances in a column

List the total number of successful and failure mission outcomes

```
In [64]: %sql Select  mission_outcome, Count(*)  as Results From SPACEXDATA Group by mission_outcome;
```

```
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.a
98/BLUDB
Done.
```

Out[64]:

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

Boosters Carried Maximum Payload

Sub-Query
In SQL we can use subqueries to get the wanted results

MAX
Gives the maximum value in a column

```
In [70]: %sql Select max(payload_mass__kg_) From SPACEXDATA
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l0
98/BLUDB
Done.
```

Out[70]:

1
15600

```
In [72]: %sql Select  booster_version  From SPACEXDATA where  payload_mass__kg_= \
(Select max(payload_mass__kg_) From SPACEXDATA ) ;
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l0
98/BLUDB
Done.
```

Out[72]:

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

2015 Launch Records

ORDER BY
Sorts the values for a certain column

WHERE
clause filters the dataset to only perform calculations on certain rows

Task 9

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

In [82]:
```
%sql Select DATE, "Landing _Outcome",booster_version, launch_site \
From SPACEXDATA WHERE "Landing _Outcome"='Failure (drone ship)' AND DATE LIKE '%2015%';
```

```
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.data
98/BLUDB
Done.
```

Out[82]:

DATE	Landing _Outcome	booster_version	launch_site
10-01-2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
14-04-2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

ORDER BY
Sorts the values for a certain column

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 [84]:
```
%sql Select  "Landing _Outcome", Count(*)  as Results From SPACEXDATA Group by "Landing _Outcome" ORDER BY Results ;
```

```
 * ibm_db_sa://cbj03800:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB
Done.
```

Out[84]:

Landing _Outcome	results
Precluded (drone ship)	1
Failure (parachute)	2
Uncontrolled (ocean)	2
Failure	3
Controlled (ocean)	5
Failure (drone ship)	5
Success (ground pad)	9
Success (drone ship)	14
No attempt	22
Success	38

Section 4

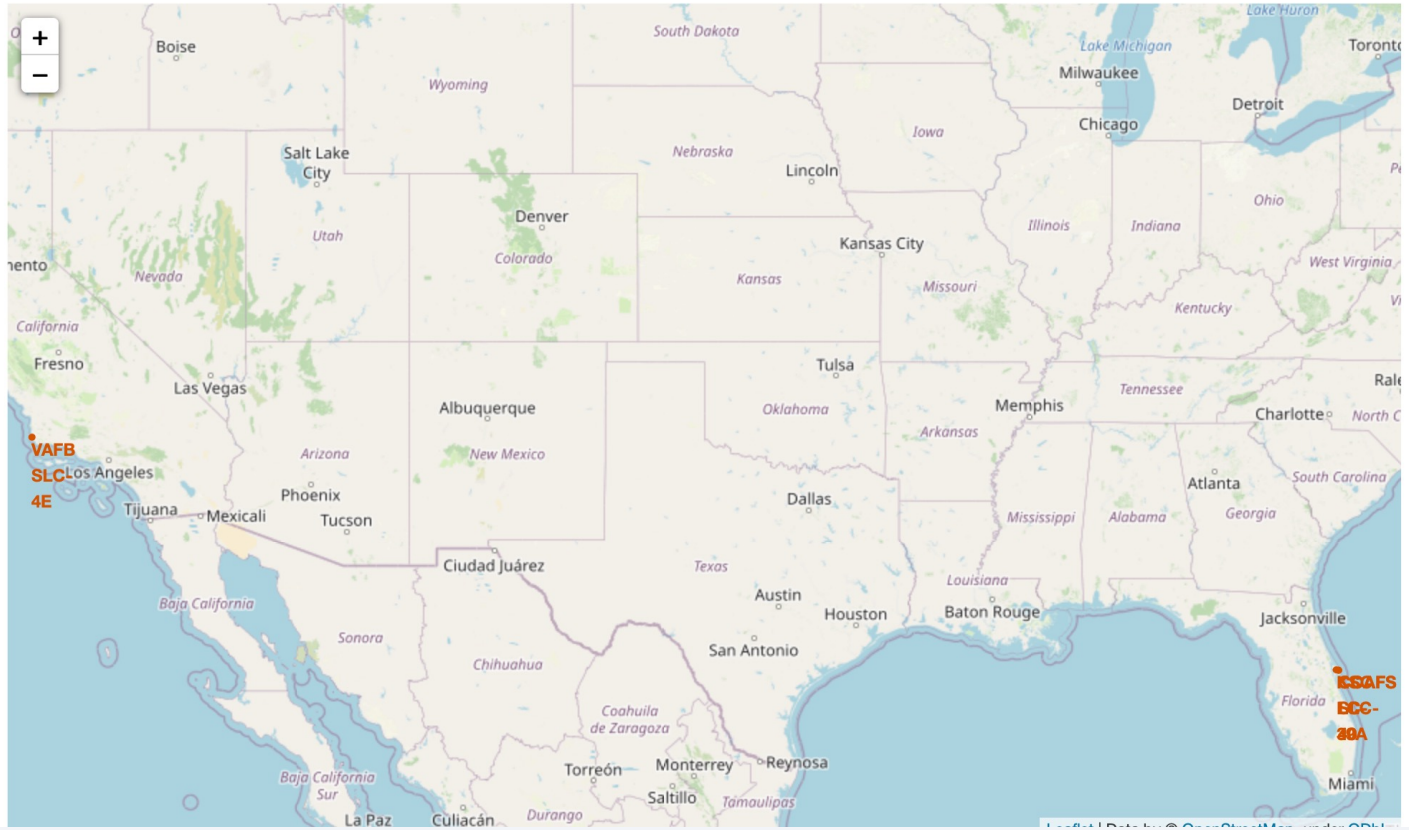
Launch Sites Proximities Analysis

Folium Map Screenshot 1

SpaceX launch sites are located in the western and eastern United States of America.

In [28]:
```
VAFB_SLC_4E = [35, -100]
site_map = folium.Map(location=VAFB_SLC_4E, zoom_start=4.5)
for name,coordinate in launch_sites_dict.items():
    add_marker_on_map(name, coordinate)
site_map
```

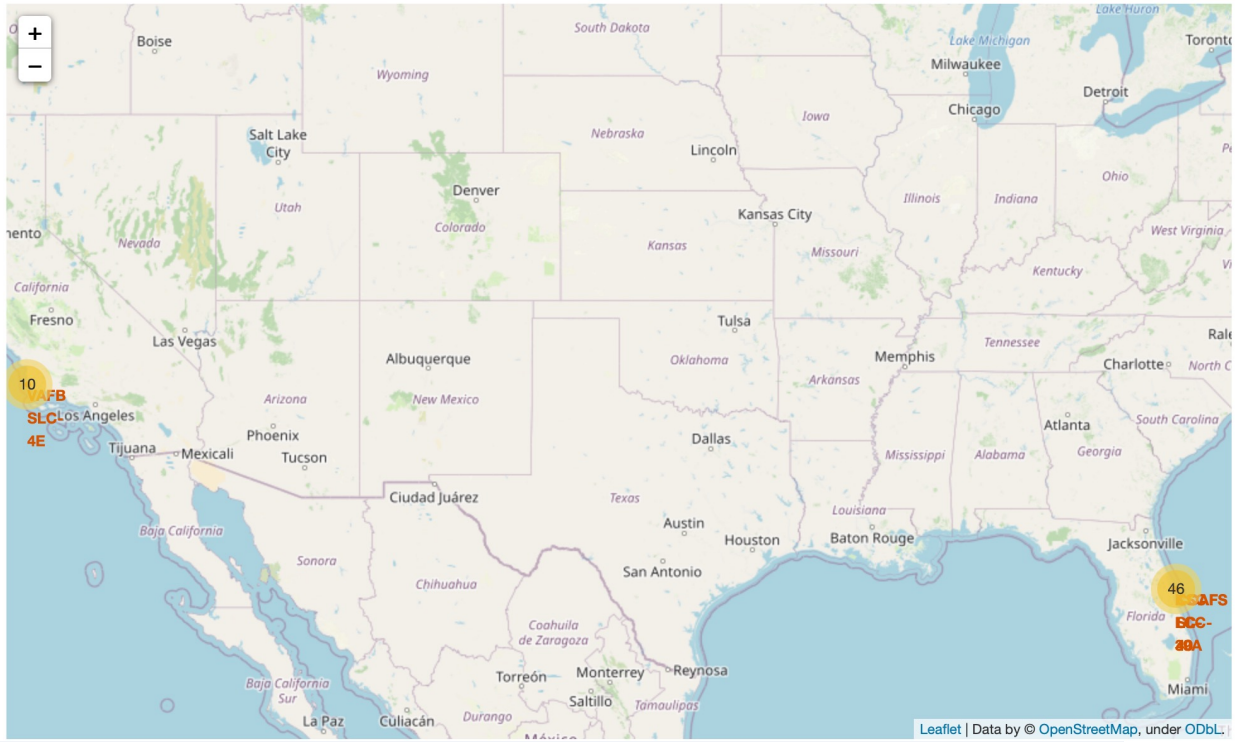
Out[28]:



Folium Map Screenshot 2

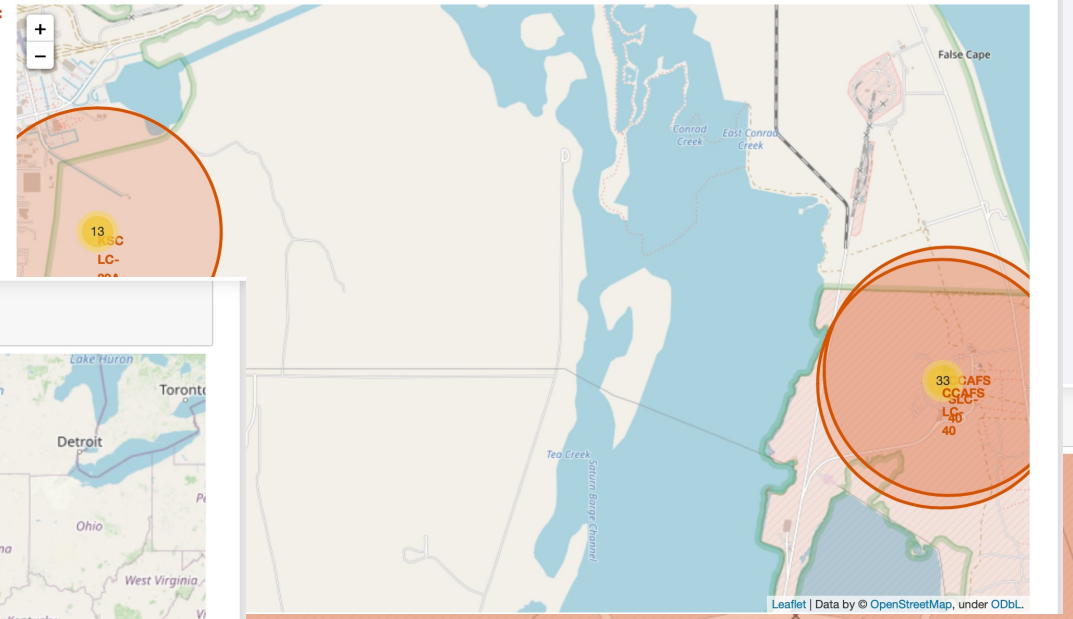
```
    add_marker_cluster_on_map(name_of_1, coordinate_of_1, color_of_1)

site_map.add_child(marker_cluster)
```

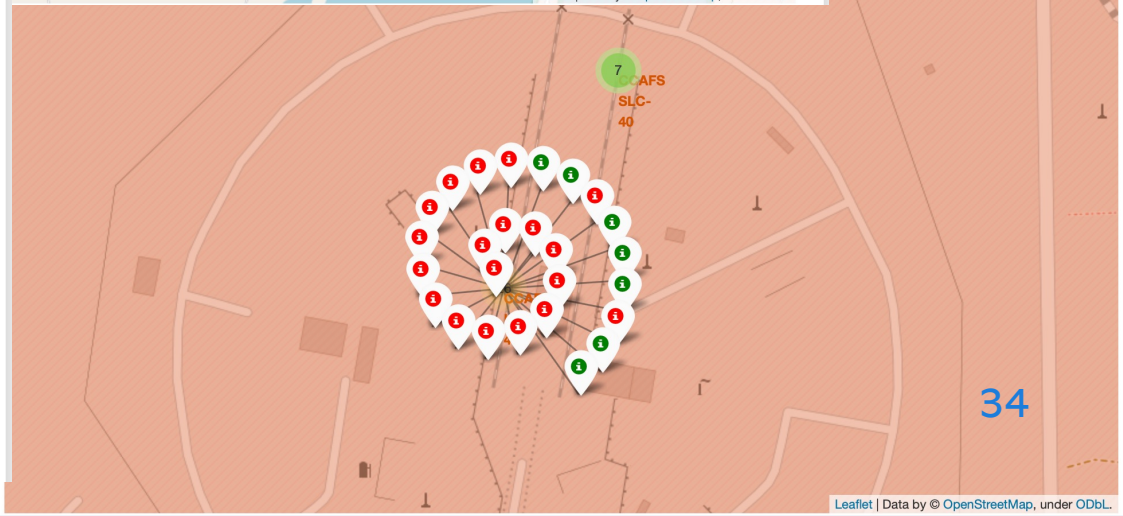
Out[46]:




```
    add_marker_cluster_on_map(name_of_1, coordinate_of_1, color_of_1)

site_map.add_child(marker_cluster)
```

Out[46]:





Section 6

Predictive Analysis (Classification)

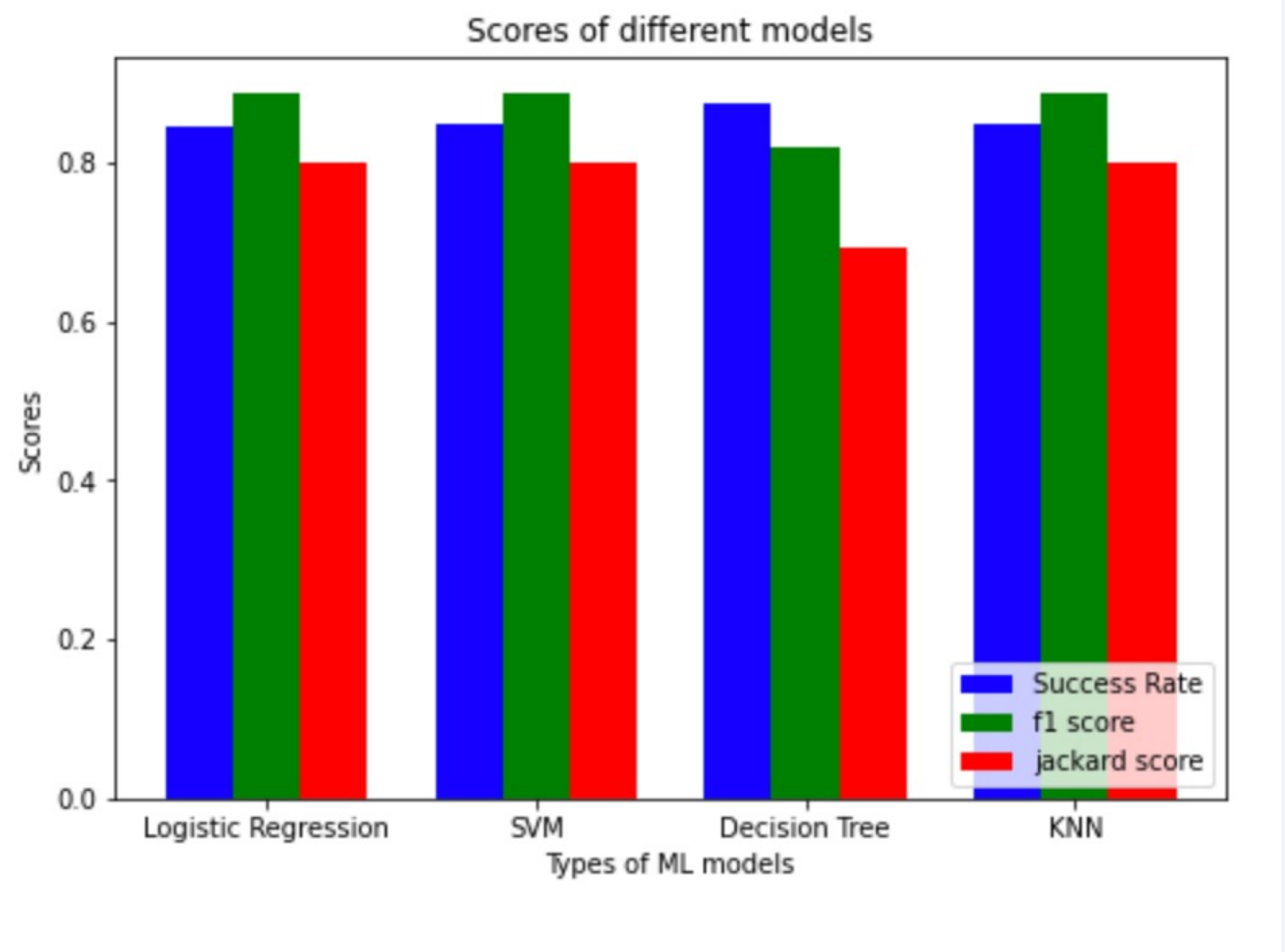
Classification Accuracy

As we look at different tests, each model shows various characteristics on these tests.

Since we are asked to find the best model in terms of accuracy decision tree algorithm is the winner.

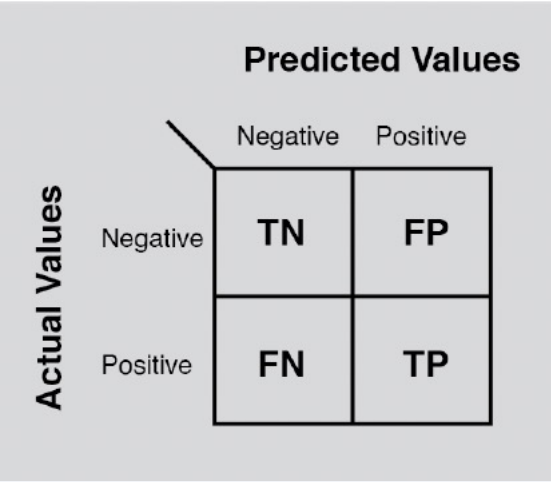
In terms of f1_score Logistic regression is the winner.

In terms of jaccard score SVM is the winner



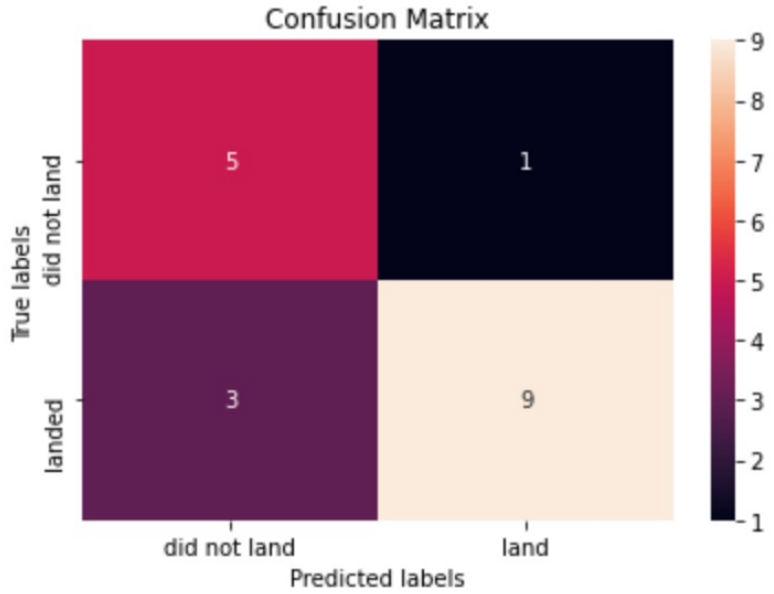
Confusion Matrix

- Confusion matrix results of the models are pretty same.
- Here, we show the confusion matrix of decision tree algorithm since it has the highest success ratio.
- Note that, accuracy = (TP+TN)/Total



We can plot the confusion matrix

```
In [26]: yhat = tree_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- The Decision Tree Algorithm is the best for Machine Learning for this dataset in terms of accuracy.
- In terms of f1_score and jaccard score; logistic regression and SVM are the winners, respectively.
- •The success rates for SpaceX launches is directly proportional time in years and it is increased to the levels of 100% in the recent years.
- Orbits named GEO,HEO,SSO,ES-L1 has the best success rates and SO has the lowest success rate.
- We can see that KSC LC-39A had the most successful launches among all the sites.

Thank you!