IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Aynur Aytekin
September 2022

Outline

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

Executive Summary

Summary of Methodologies

- Data collection
- Data wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Visualization
- Interactive Visual Analytics Folium
- Machine Learning Prediction

Summary of All Results

- Exploratory Data Analysis Results
- Interactive Visual Analytics Results
- Predictive Analytics Results

Introduction

Founded in 2002 by Elon Musk, Space Exploration Technologies Corporation (SpaceX) has been the most successful spacecraft manufacturer, space launch provider, and a satellite communications corporation of the history. The primary objective of SpaceX is to provide low-cost space transportation which will enable colonization of Mars in the near future.

One of the biggest achievements of SpaceX is Falcon 9, a two-stage rocket. Falcon 9 has a very unique design since the first stage of the rocket is reusable. The average cost of Falcon 9 launch is 62 million dollars, whereas other providers cost about 165 million dollars for each launch.

The goal of this project is accurately predict if the Falcon 9 first stage will land successfully. With this data, we can estimate the cost of the 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

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using the SpaceX REST API and WEB scraping from Wikipedia.
- Data wrangling:
  - The data collected from API is in JSON format. We used json_normalize function to turn it into dataframe. Using the Python BeatifulSoup package, we web scraped some HTML tables that contain Falcon 9 launch records. For further analysis, we converted those tables into Pandas dataframe. The data was filtered so that only the parts related to Falcon 9 remains. The NULL data was replaced with the mean of the PayLoadMass to take care of the missing data.

Methodology

Executive Summary

- Exploratory data analysis (EDA) using visualization and SQL
- Interactive visual analytics using Folium and Plotly Dash
- Predictive analysis using classification models
    - How to build, tune, evaluate classification models

Data Collection

Data collection steps;

- Collecting data using the SpaceX REST API and WEB scraping from Wikipedia.
- The collected data includes the information regarding the launch site names, the longitude and latitude of the sites, the mass of the payload, the orbit that it is going to, the type and the outcome of the landing, the number of the cores with that type, whether gridfins were used, whether the core is reused, whether legs were used, the landing pad used, the block of the core which is a number used to separate version of cores, the number of times this specific core has been reused, and the serial of the core.

Data Collection – SpaceX API

```python
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

Getting the booster name from the rocket column.

```python
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Getting the mass of the payload and the orbit that it is going to from PayLoadMass column.

Getting the name of the launch site being used, the logitude, and the latitude from launchpad column.

```python
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
        Longitude.append(response['longitude'])
        Latitude.append(response['latitude'])
        LaunchSite.append(response['name'])
```

Getting the information from cores column

```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

[GitHub Link: Data Collection]

Data Collection – SpaceX API

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

```
response = requests.get(spacex_url)
```

Requesting rocket launch data from SpaceX API

```
# 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

```
# Get the head of the dataframe
data.head()
```

Normalization of the data using .json_normalize

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9.drop(data_falcon9[data_falcon9["BoosterVersion"] != "Falcon 9"].index, inplace=True)
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.reset_index(drop=True, inplace=True)
```

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

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

Creating the dataframe using the requested data

Dealing with the missing values

```
# Calculate the mean value of PayloadMass column
avg_PayloadMass = data_falcon9['PayloadMass'].astype('float').mean(axis=0)
print("Average of Payload Mass is:", avg_PayloadMass)
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, avg_PayloadMass)
data_falcon9.isnull().sum()
```

```
Average of Payload Mass is: 6123.547647058824
FlightNumber        0
```

[GitHub Link: Data Collection]()

Data Wrangling

Calculating the number of launches on each site

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Calculating the number and occurrence of each orbit

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

Calculating the number and occurence of mission outcome per orbit type

```
# landing_outcomes = values on Outcome column
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
```

Creating a landing outcome label from Outcome column

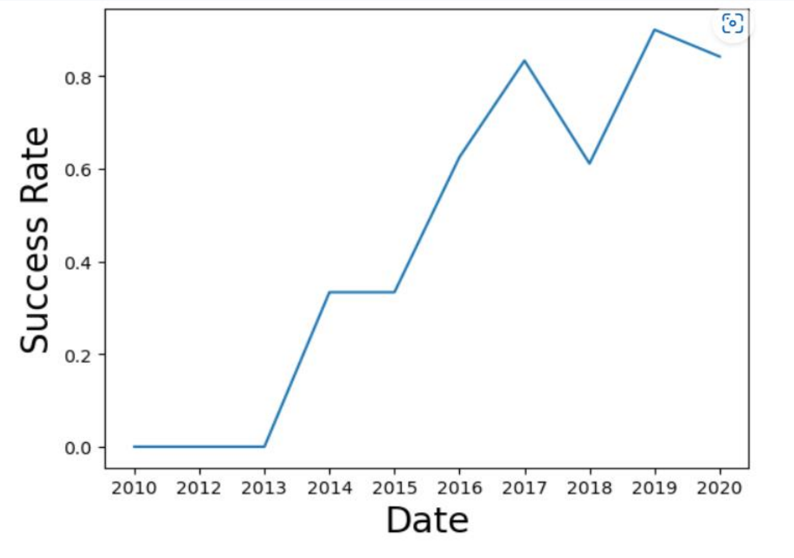
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
df['Class'] = df['Outcome'].apply(lambda x: 1 if x in ["True ASDS", "True RTLS", "True Ocean"] else 0)
df.head(10)
```

[GitHub Link: Data Wrangling]()

EDA with Data Visualization

In the exploratory data analysis part, we used data visualization skills to visualize the data and extract meaningful patterns to guide the modeling process. The following relationships were examined using scatterplot, bar chart, and line charts.

- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Orbit Type vs. Success Rate
- Flight Number vs. Orbit Type
- Payload vs. Orbit Type
- The Yearly Trend of Launch Success



[GitHub Link: EDA with Visualization]

EDA with SQL

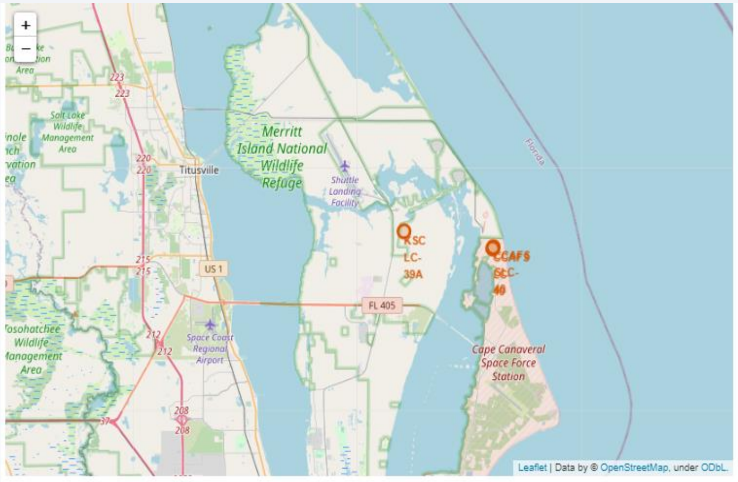
In the exploratory data analysis with SQL part, we completed;

- Displaying the names of the unique launch sites in the space mission,
- Listing the five records where launch sites begin with the string "CCA",
- Finding the total payload mass carried by boosters launched by NASA (CRS),
- Calculating the average payload mass carried by booster version F9 v1.1,
- Finding the date when the first successful landing outcome in ground pad was achieved,
- Finding the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster versions which have carried the maximum payload mass
- Listing the records which will display the month names, failure landing outcomes in drone ship ,booster versions, launch site for the months in year 2015.
- Ranking the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

[GitHub Link: EDA with SQL]

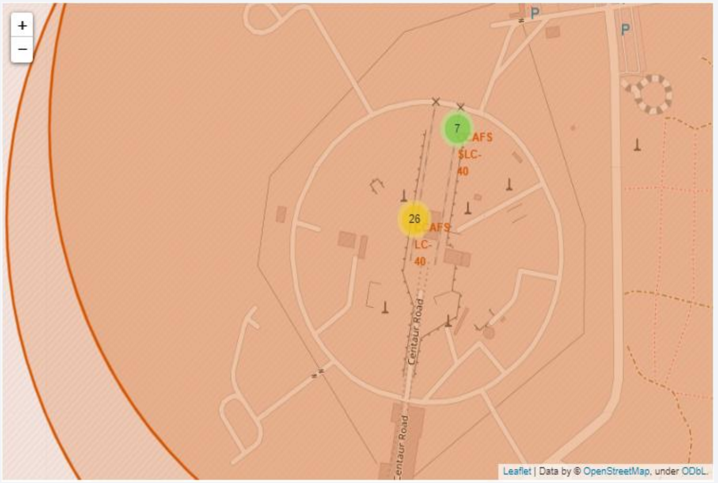
Build an Interactive Map with Folium

Using the Folium lab, we built an interactive map to analyze the launch site proximity.

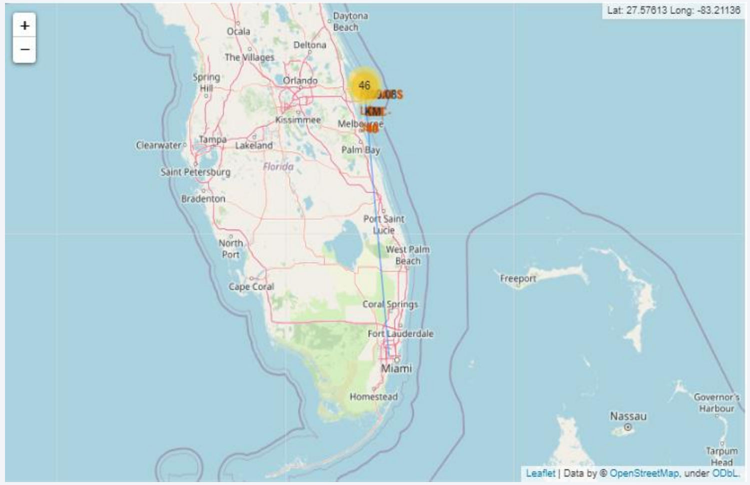


We marked all the launch sites on the map using the coordinates

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745



We marked the success/failed launches for each site on the map



We calculated the distances between a launch site to its proximities

Predictive Analysis (Classification)

Building The Model

- Load and transform the data,
- Create test and train data sets
- Optimize the parameters for each model using Grid Search
- Train the model and perform Grid Search

Evaluating The Model

- Check the accuracy for each model
- Calculate the best Hyperparameter for SVM, Classification Trees, and Logistic Regression
- Plot the confusion matrix for each model

Finding The Best Model

- Determine the model with the highest accuracy score

GitHub Link: Machine Learning Prediction

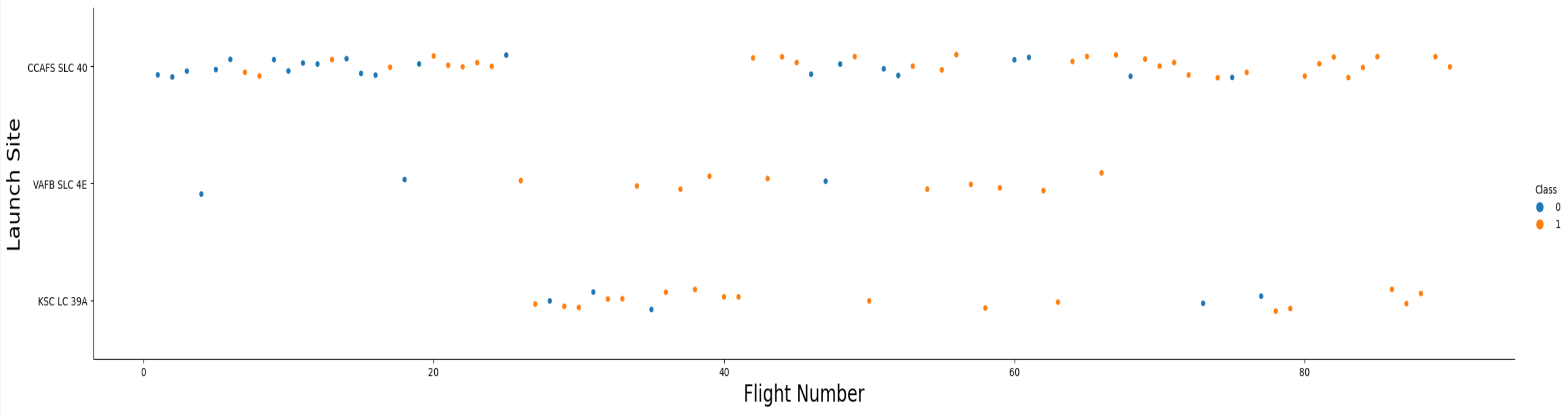
Results

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

Section 2

Insights drawn from EDA

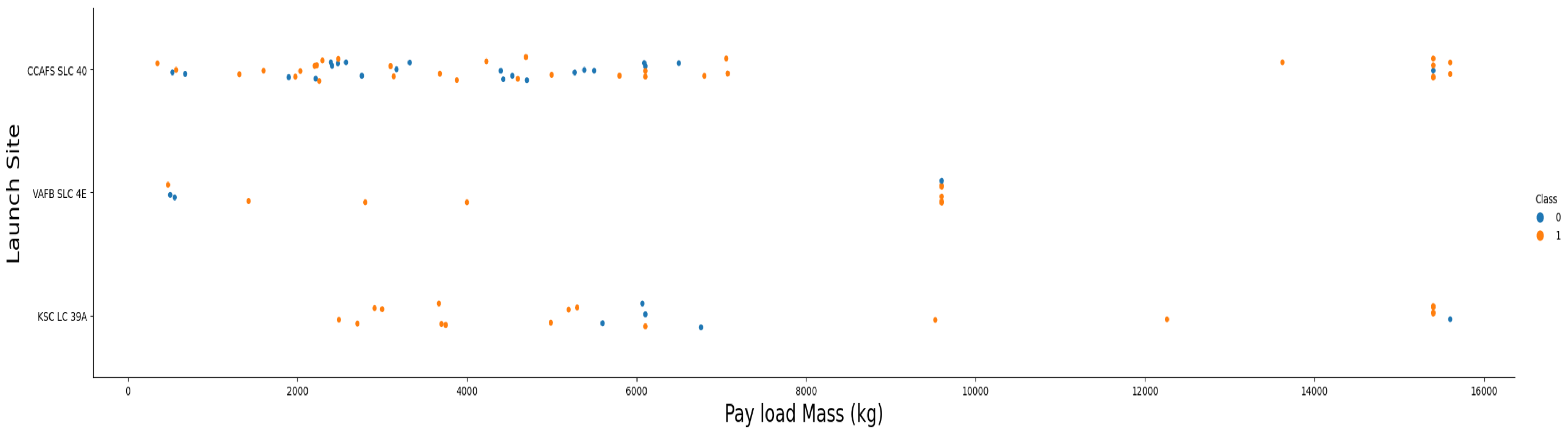
Flight Number vs. Launch Site



Scatter plot of Flight Number vs. Launch Site

- The success rate is higher for the recent flights on all launch sites
- CCAFS SLC 40 has the lowest success rate

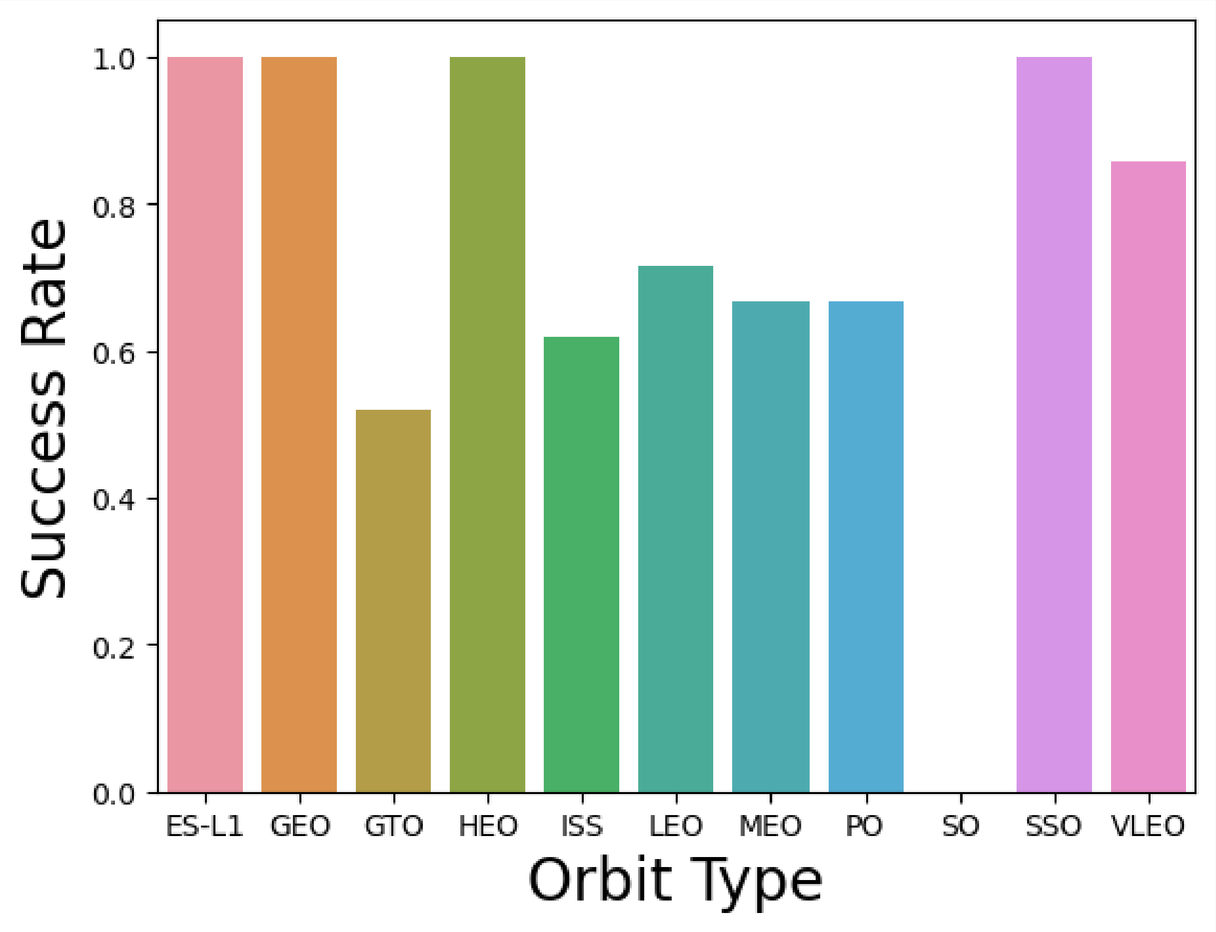
Payload vs. Launch Site



Scatter plot of Payload vs. Launch Site

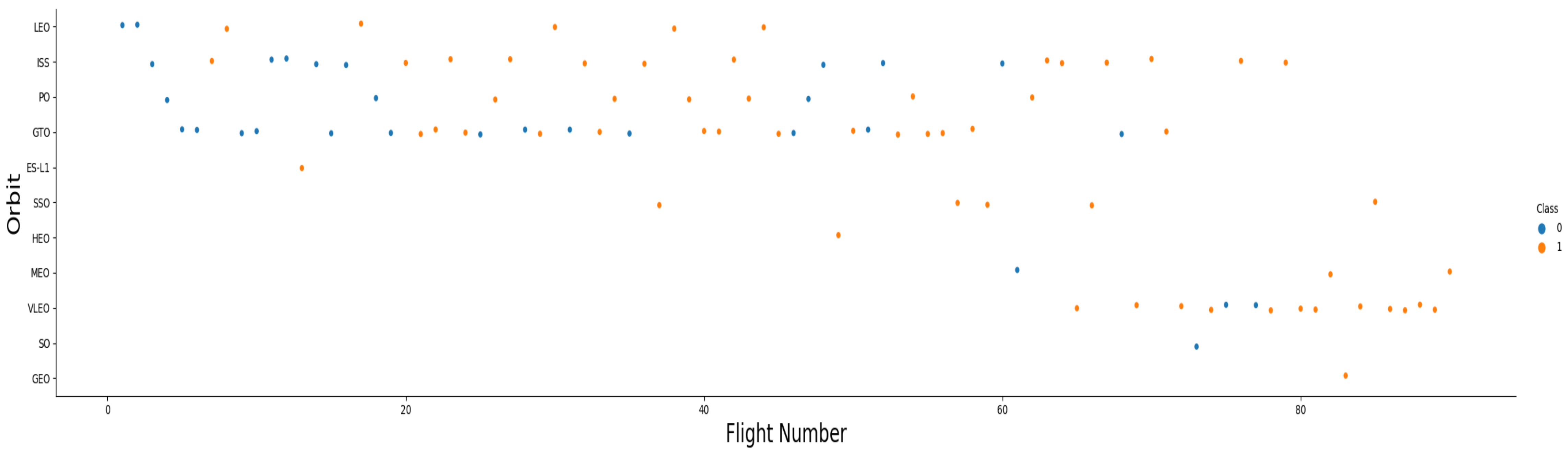
- The success rate increases as the payload mass increases for the launch site VAFB SLC 4E

Success Rate vs. Orbit Type



- The success rate is higher for the orbit types; ES-L1,GEO,HEO,SSO
- The success rate is zero for the orbit type SO

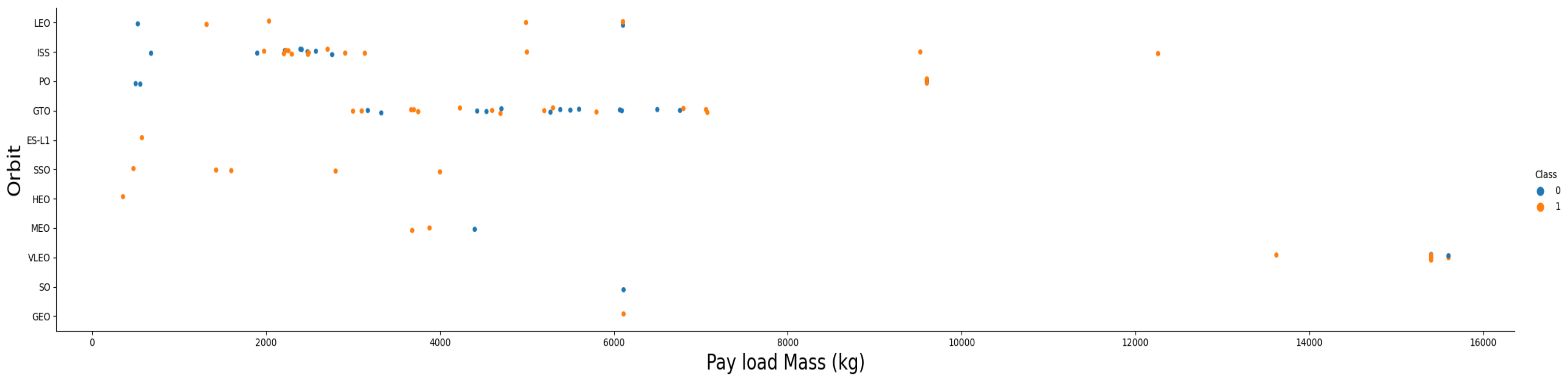
Flight Number vs. Orbit Type



Scatter plot of Flight number vs. Orbit type

- The success rate is higher on LEO orbit, as the flight number increases
- GTO orbit has no correlation with the flight number
- Orbit VLEO has higher rates for higher flight number

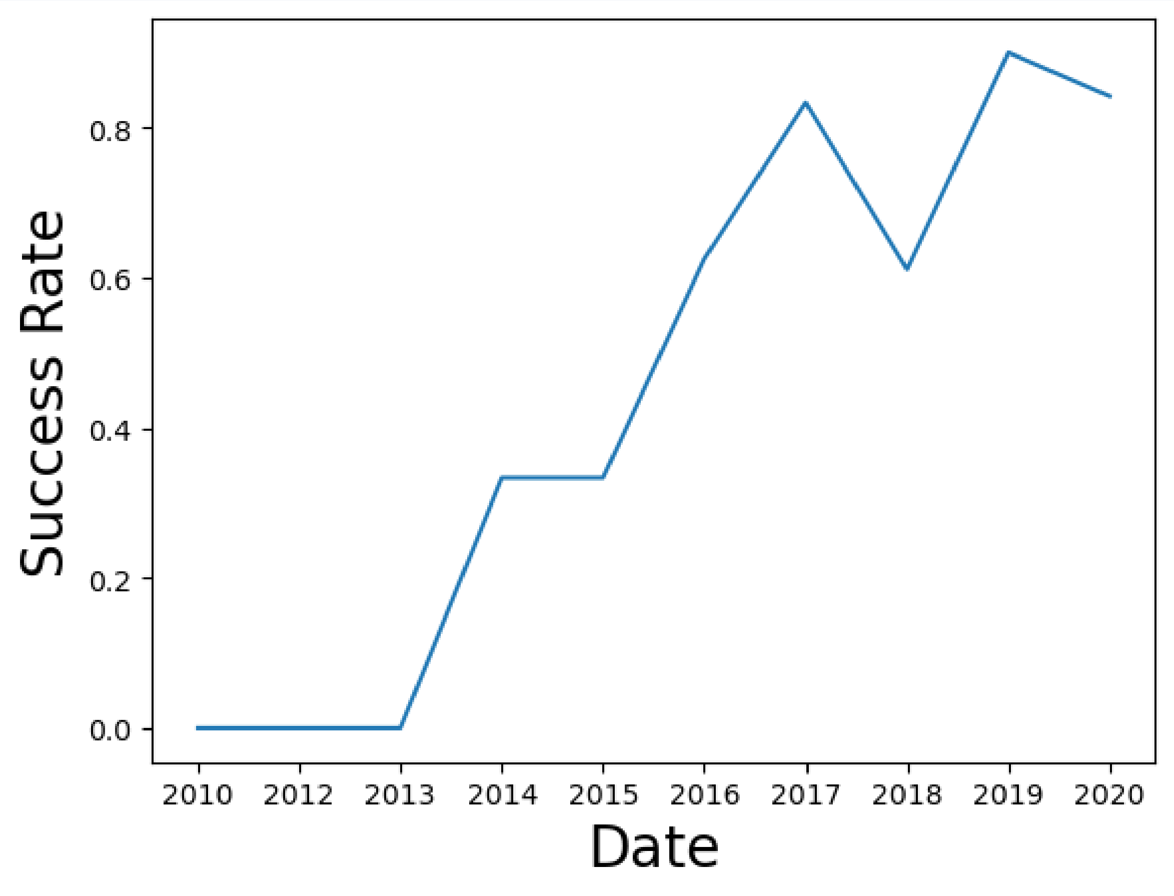
Payload vs. Orbit Type



Scatter plot of payload vs. orbit type

- For the orbits PO, LEO, and ISS the success rate is higher for the heavier payloads
- Orbit GTO has no correlation with the payload mass

Launch Success Yearly Trend



Line chart of yearly average success rate

- The success rate increased between the years 2013 t0 2017
- There was a drop in the success rate during the year 2018
- There is no successful landing before the year 2013

All Launch Site Names

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

```
%%sql

select distinct(launch_site) from spacextbl;
```

```
 * sqlite:///my_data1.db
Done.
```

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

Using SQL to display the unique launch site names

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;
```

```
 * sqlite:///my_data1.db
Done.
```

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

We used "select / where" function form SQL to display the launch sites begin with the string "CCA"

Total Payload Mass

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

```
%%sql select sum(PAYLOAD_MASS__KG_) as total_payload_mass
from spacextbl
where customer = 'NASA (CRS)'
```

```
 * sqlite:///my_data1.db
Done.
```

total_payload_mass
45596

We used "select / sum" function to calculate the total payload mass carried by boosters launched by NASA

Average Payload Mass by F9 v1.1

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

```
%%sql select Booster_Version, avg(PAYLOAD_MASS__KG_) as Average_Mass_F9
from spacextbl
where Booster_Version = 'F9 v1.1'
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version	Average_Mass_F9
F9 v1.1	2928.4

Calculating the average payload mass using "select / avg()"

First Successful Ground Landing Date

```
%%sql

select "Landing _Outcome", min(Date) as First_Successful_Landing_Date from spacextbl
where "Landing _Outcome" = "Success (ground pad)";
```

```
 * sqlite:///my_data1.db
Done.
```

Landing _Outcome	First_Successful_Landing_Date
Success (ground pad)	01-05-2017

Finding the dates of the first successful landing outcome on ground pad using the min() function

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql

select Booster_Version, "Landing _Outcome",PAYLOAD_MASS__KG_ from spacextbl
where "Landing _Outcome" = "Success (drone ship)"
and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_< 6000
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version	Landing _Outcome	PAYLOAD_MASS__KG_
F9 FT B1022	Success (drone ship)	4696
F9 FT B1026	Success (drone ship)	4600
F9 FT B1021.2	Success (drone ship)	5300
F9 FT B1031.2	Success (drone ship)	5200

We used "select / where / and" function to list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

```
%%sql

select Booster_Version, "Landing _Outcome",PAYLOAD_MASS__KG_ from spacextbl
where "Landing _Outcome" = "Success (drone ship)"
and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_< 6000
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version	Landing _Outcome	PAYLOAD_MASS_KG_
F9 FT B1022	Success (drone ship)	4696
F9 FT B1026	Success (drone ship)	4600
F9 FT B1021.2	Success (drone ship)	5300
F9 FT B1031.2	Success (drone ship)	5200

We listed the names of the boosters which have success in drone ship and have the payload mass greater than 4000 but less than 6000

Boosters Carried Maximum Payload

```
%%sql

select Booster_Version, PAYLOAD_MASS__KG_ from spacextbl
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextbl);
```

```
 * sqlite:///my_data1.db
Done.
```

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

We

We calculated the total number of successful and failure mission outcomes using SQL subquery

2015 Launch Records

```
%%sql

select distinct "Landing _Outcome", Booster_Version, Launch_Site, substr(Date, 4, 2) as Month, substr(Date, 7, 4) as Year
from spacextbl
where "Landing _Outcome" = "Failure (drone ship)" and substr(Date, 7, 4) = '2015';
```

```
 * sqlite:///my_data1.db
Done.
```

Landing _Outcome	Booster_Version	Launch_Site	Month	Year
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	01	2015
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	04	2015

To list the names of the booster which have carried the maximum payload mass, we used the substr(Date, 4, 2) since SQLLite does not support month names.

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

```
%%sql
select count("Landing _Outcome") as Count, Date, "Landing _Outcome" from spacextbl
where Date between '04-06-2010 00:00:00' AND '20-03-2017 12:00:00'
and "landing _Outcome" like "%Success"
group by Date
order by count("Landing _Outcome") desc
```

```
 * sqlite:///my_data1.db
Done.
```

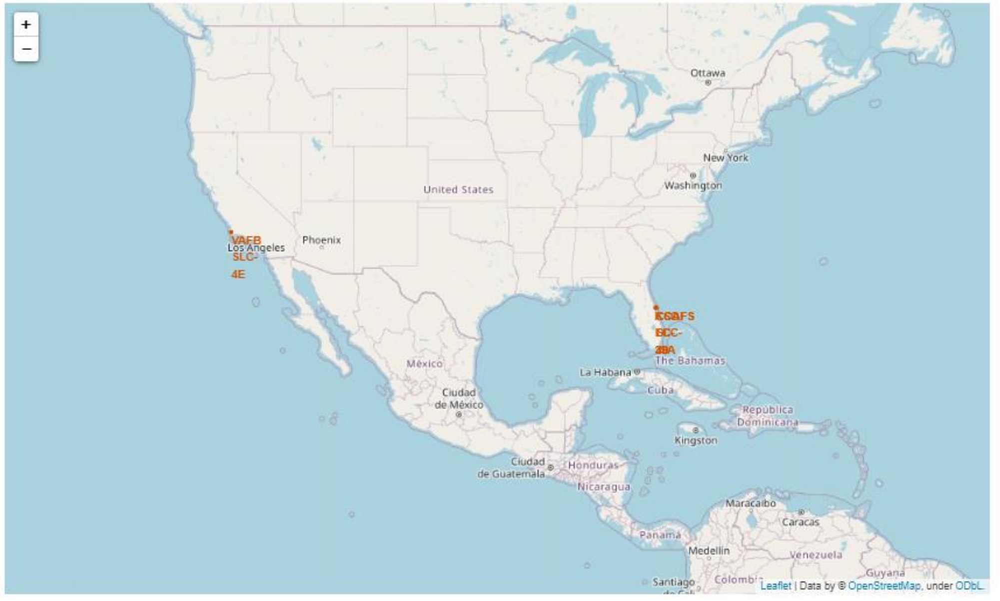
Count	Date	Landing _Outcome

- We used the code listed above to 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

Section 3

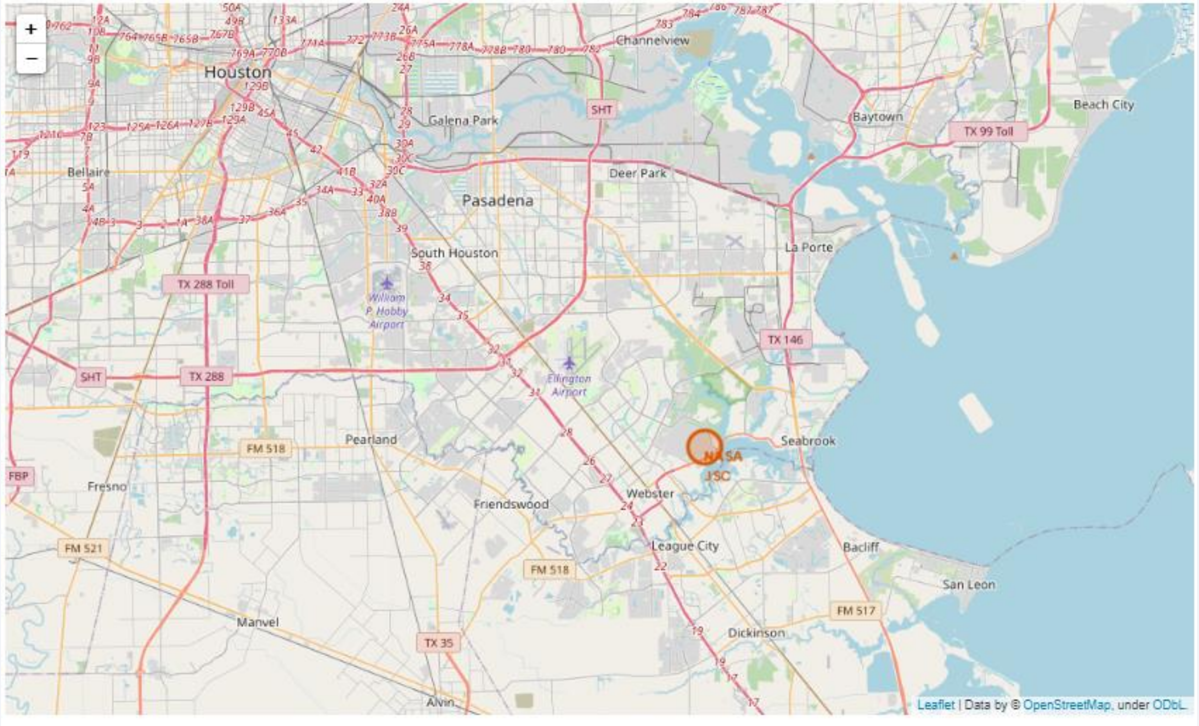
Launch Sites Proximities Analysis

Marking The Launch Sites Using Folium



We used "folium.circle" to add a highlighted circle area with a text label on a specific coordinate.

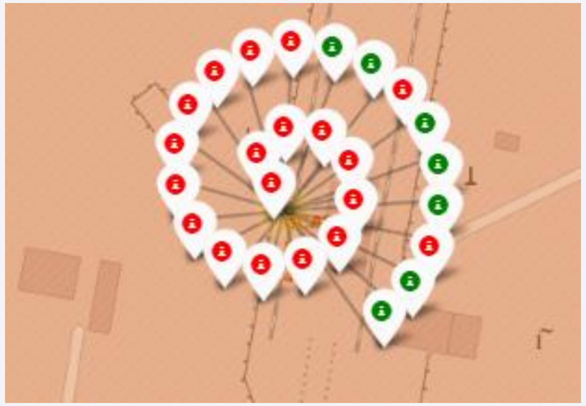
We added all launch sites on a map.



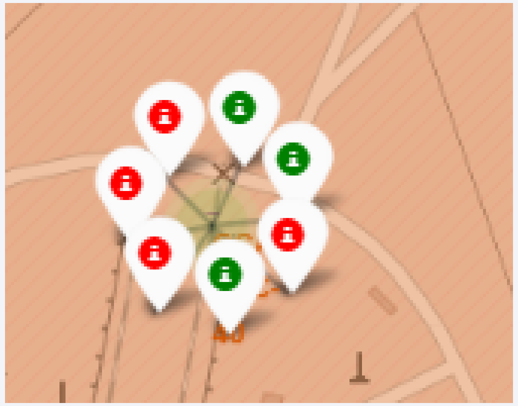
Marking The Success/Failed launches On The Map

```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for index, record in spacex_df.iterrows():
    folium.Marker((record['Lat'] , record['Long']) , icon=folium.Icon(color='white', icon_color=record['marker_color'])).add_to(marker_cluster)


site_map.add_child(marker_cluster)
site_map
```

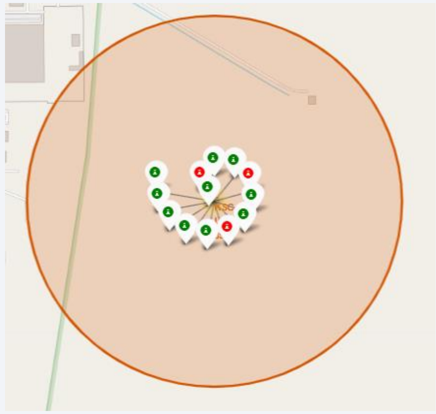
We marked all successful landings in green and failed landings in red.



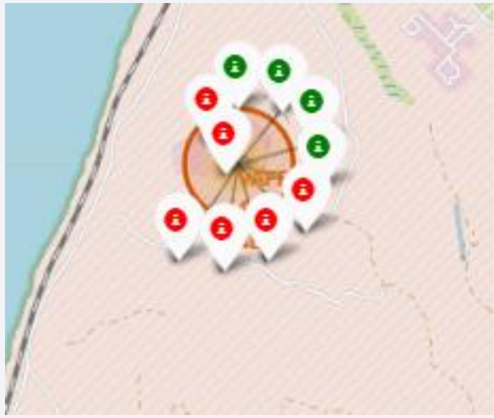
CCAFS LC-40



CCAFS LC-40

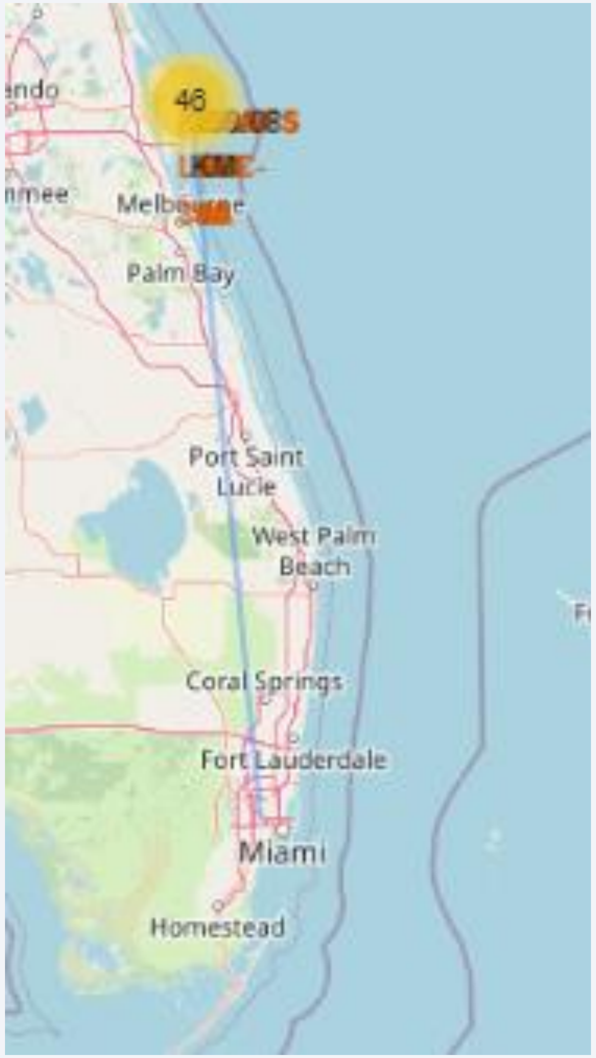


KSC LC-39A



VAFB SLC-4E

Calculating The Distances Between A Launch Site to Its Proximities



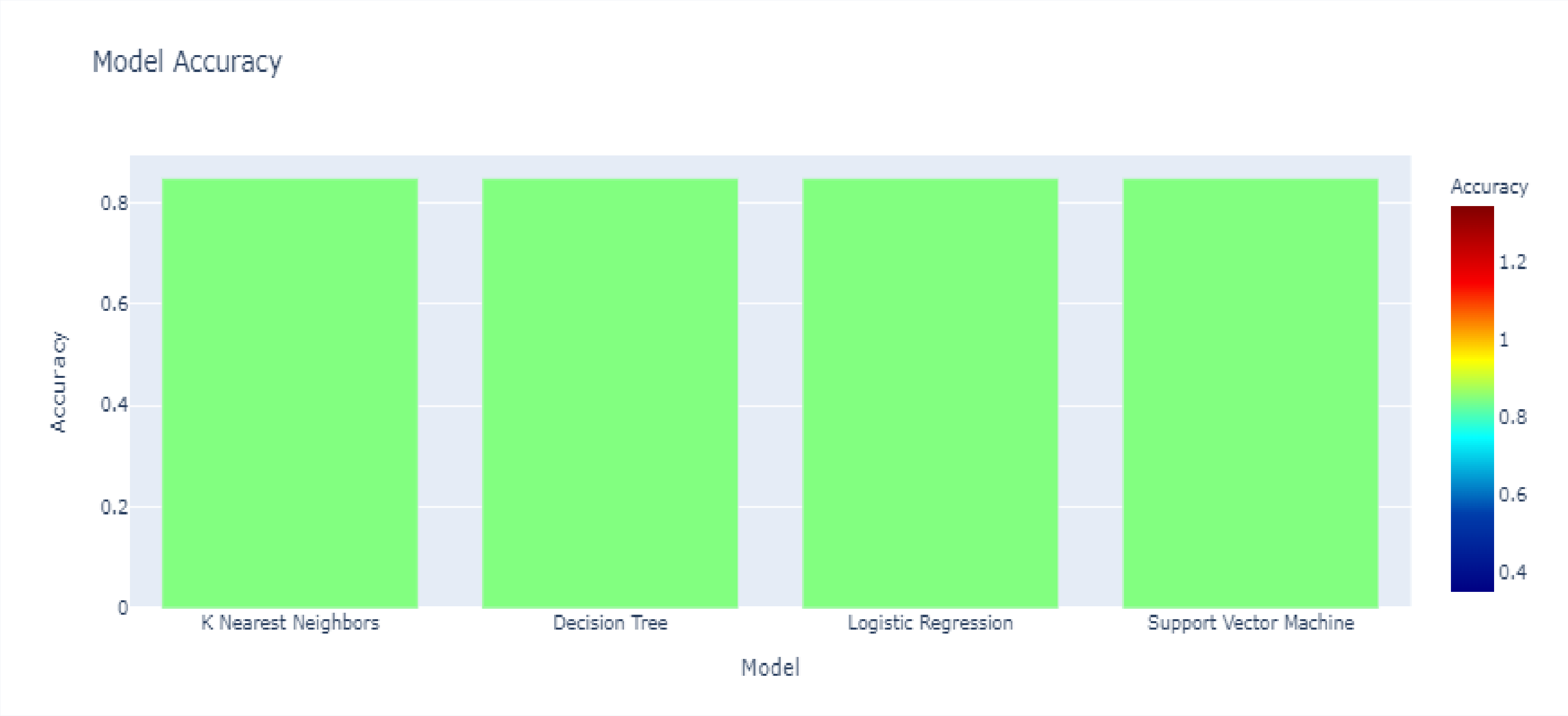

```
#Distance to Miami Airport

coordinates = [
    [28.56259, -80.59572],
    [25.79732, -80.28838]]

lines=folium.PolyLine(locations=coordinates, weight=1)
site_map.add_child(lines)
distance = calculate_distance(coordinates[0][0], coordinates[0][1], coordinates[1][0], coordinates[1][1])
distance_circle = folium.Marker(
    [28.56259, -80.59572],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#252526;"><b>%s</b></div>' % "{:10.2f} KM".format(distance),
        )
    )
site_map.add_child(distance_circle)
site_map
```

We calculated the distance from the launch site to a chosen point using the folium map.

Section 4

Predictive Analysis (Classification)

Classification Accuracy

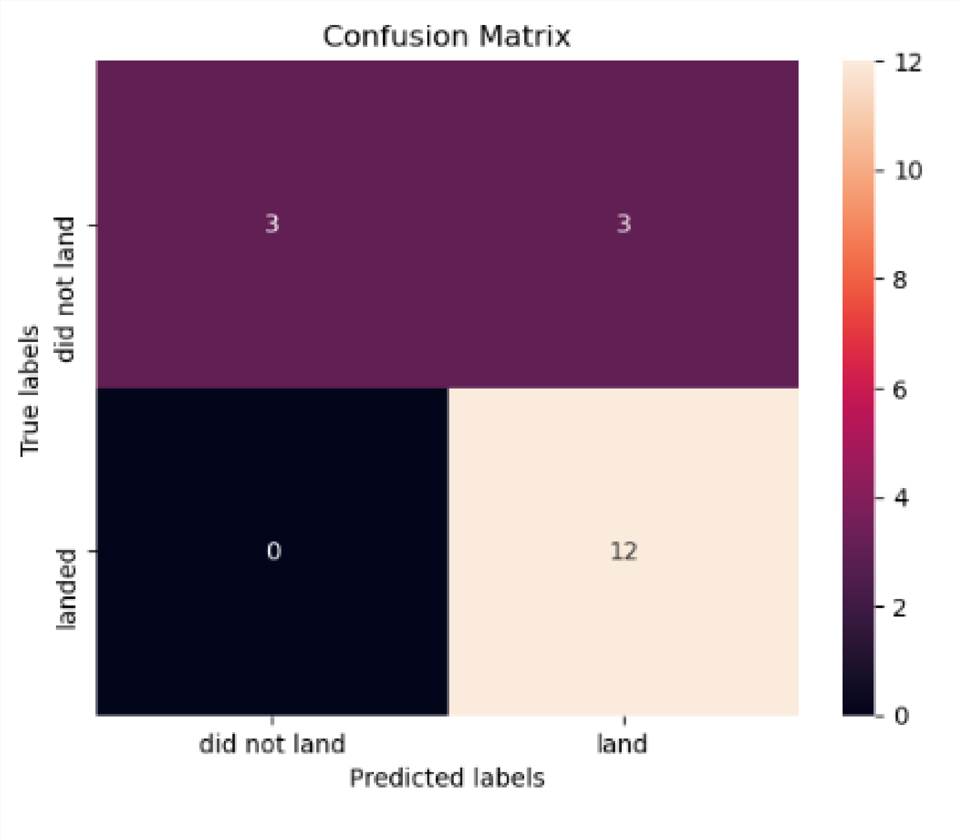


	Model	Accuracy
0	K Nearest Neighbors	0.847222
1	Decision Tree	0.847222
2	Logistic Regression	0.847222
3	Support Vector Machine	0.847222

With the selected sample (n=18) the accuracy of the models were the same. If we run the test with a different sample group, the results will be different.

The accuracy rate for all models was 84.7%

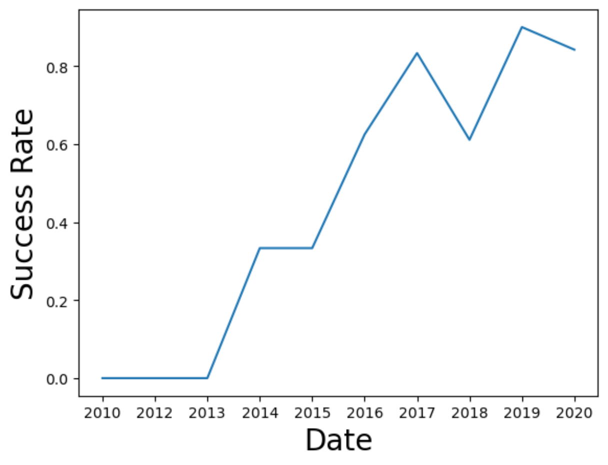
Confusion Matrix



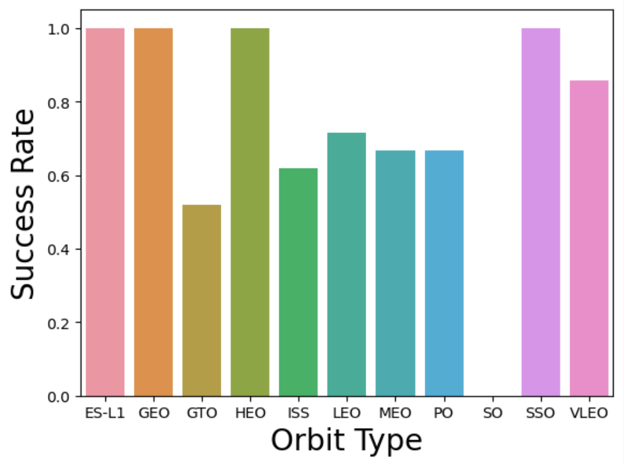
The confusion matrix for all the model types were the same since the accuracy rates were the same.

The models successfully predicted 84.7% of the landings.

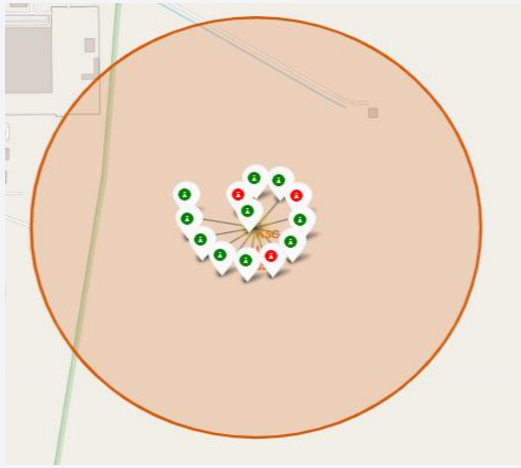
Conclusions



The success rate of SpaceX has been increasing over the time.



The success rates are higher for the orbits; ES-L1, GEO, HEO, and SSO.



KSC LC-39A has a higher success rate.

All four of the models successfully predicted 84.7% of the landing cases.

Appendix

The GitHub Link for the Project

Thank you!