IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Michael S. Miranda
April 7, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - EDA results
  - Interactive analytics
  - Predictive analytics

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.

- Problems you want to find answers

The project task is to predicting if the first stage of the SpaceX Falcon 9 rocket will land successfully

Section 1

Methodology

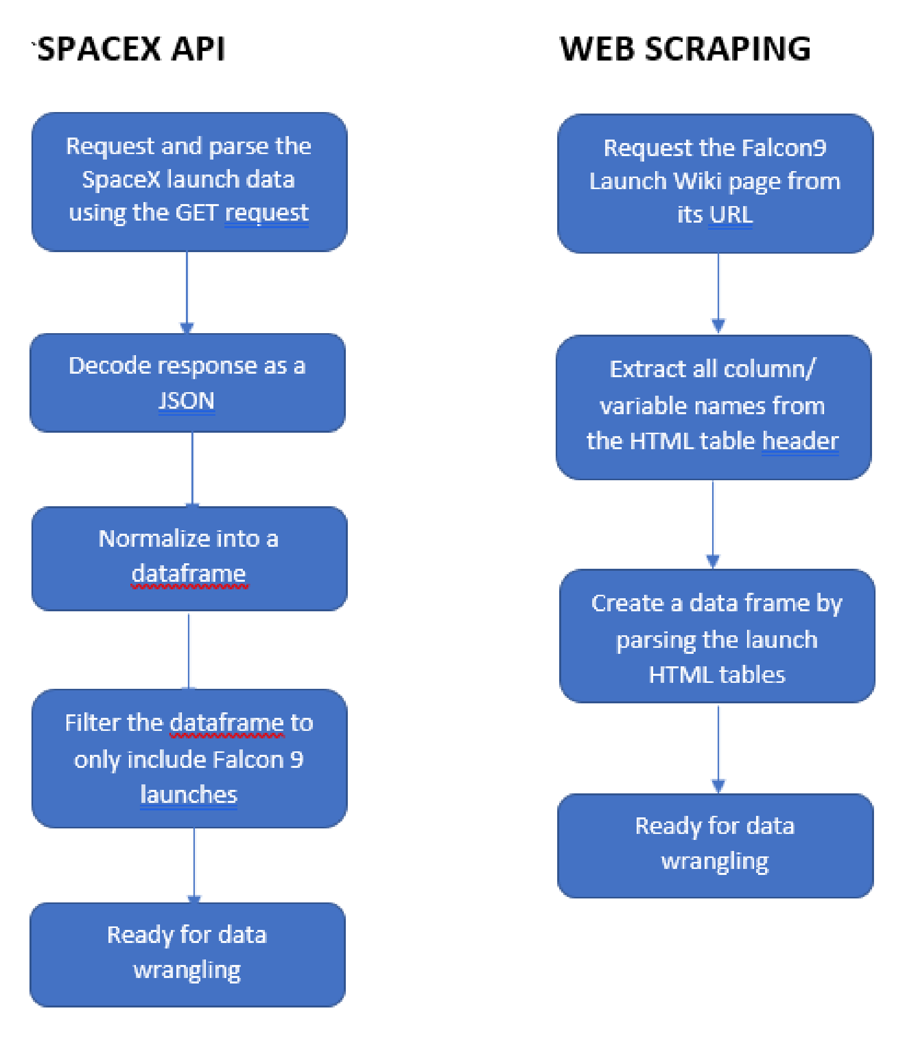
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - One Hot Encoding data fields for Machine Learning and data cleaning of null values and irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - LR, KNN, SVM, DT models have been built and evaluated for the best classifier

Data Collection

- The following datasets was collected:
  - SpaceX launch data that is gathered from the SpaceX REST API.
  - This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
  - The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
  - Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.



Data Collection – SpaceX API

- Data collection with SpaceX REST calls

https://github.com/microbytes2005/course6/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

SPACEX API CALLS

1. **Getting response from API**
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```
2. **Converting Response to a JSON file**
```
data = pd.json_normalize(response.json())
```
3. **Apply custom functions to clean data**
```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```
4. **Assign list to dictionary and convert to dataframe**
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

df = pd.DataFrame(launch_dict)
```
5. **Filter Data**
```
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```
6. **Data Wrangling**
```
data_falcon9.isnull().sum()
mean = data_falcon9['PayloadMass'].mean()
ata_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan,
mean)
```

Data Collection - Scraping

- Web Scraping from Wikipedia

https://github.com/microbytes2005/course6/blob/main/jupyter-labs-webscraping.ipynb

WIKIPEDIA WEB SCRAPING

1. Request the Falcon9 Launch Wiki page from its URL

```
static_url =
"https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_
Falcon_Heavy_launches&oldid=1027686922"
response = requests.get(static_url)Converting Response to a JSON
file
data = pd.json_normalize(response.json())
```

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

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
```

3. Extract column name one by one using extract_column_from_header()

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

4. Create a data frame by parsing the launch HTML tables

```
launch_dict= dict.fromkeys(column_names)
del launch_dict['Date and time ( )']
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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

5. Construct the dictionary

```
extracted_row = 0
for table_number,table in
enumerate(soup.find_all('table',"wikitable plainrowheaders
collapsible")):
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        row=rows.find_all('td')
        if flag:
            extracted_row += 1
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict['Version Booster'].append(bv)
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)
            customer = ''
            if row[6].a != None:
                customer = row[6].a.string
            launch_dict['Customer'].append(customer)
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome'].append(launch_outcome)
            booster_landing = landing_status(row[8])
            launch_dict['Booster
landing'].append(booster_landing)
```

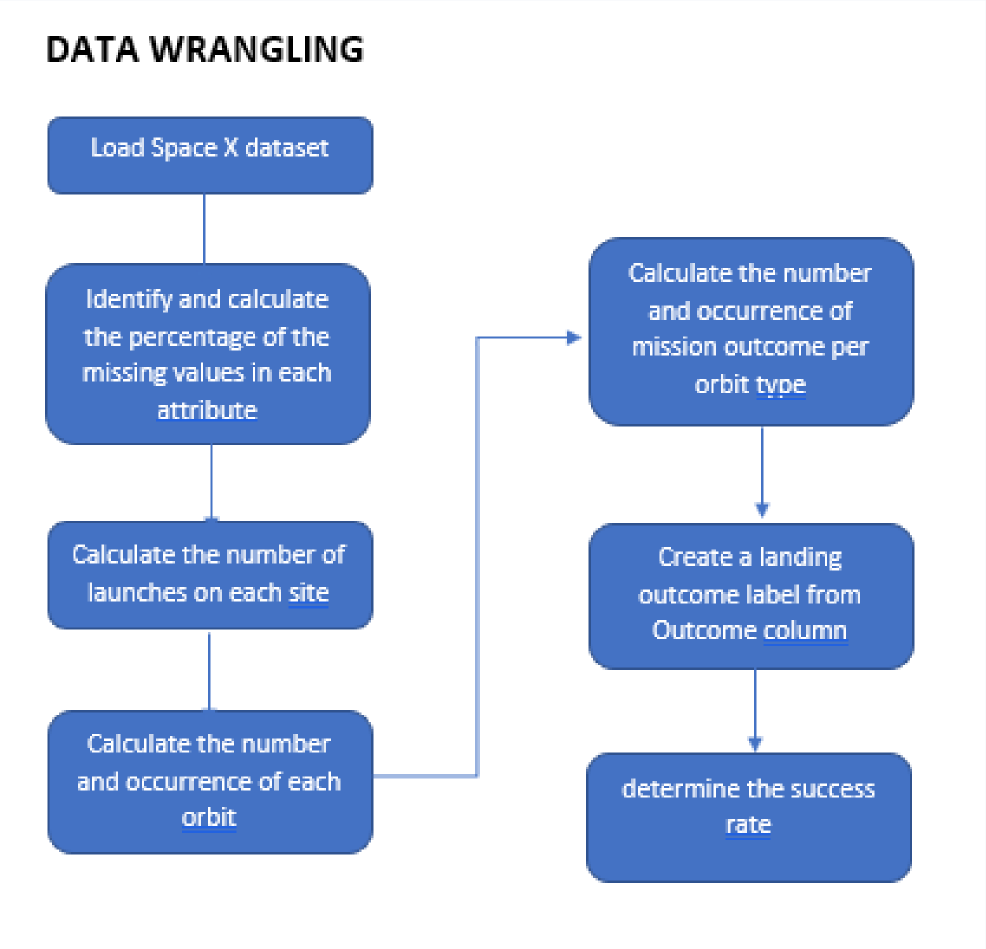
6. Create dataframe from dictionary

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

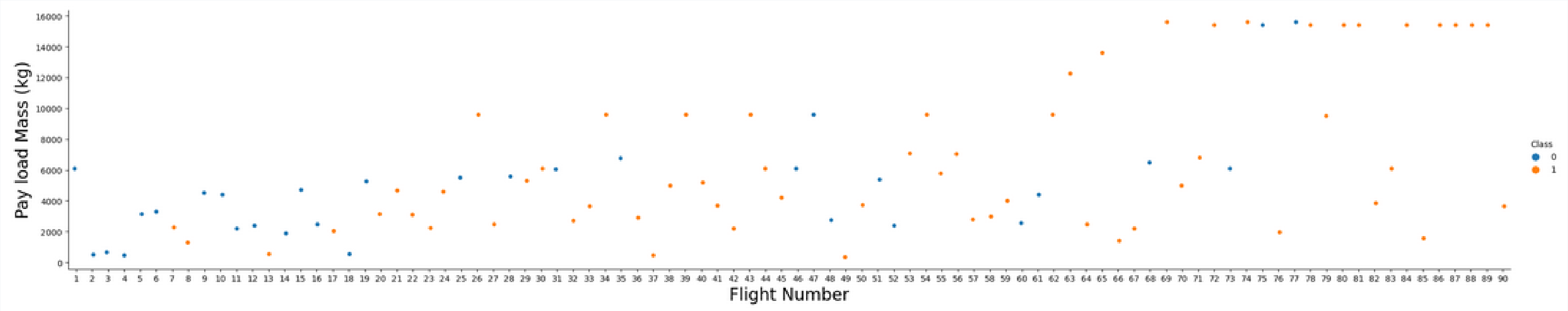
Data Wrangling

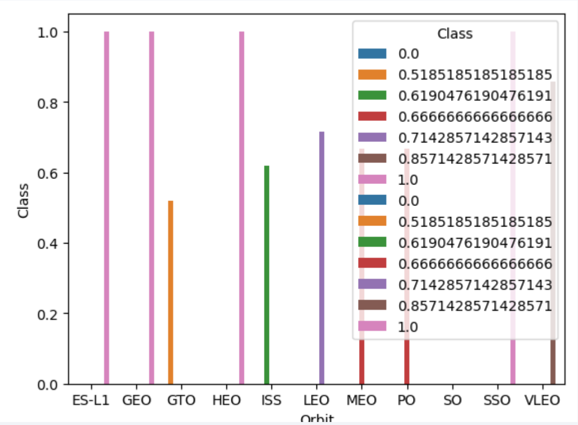
- Data Wrangling

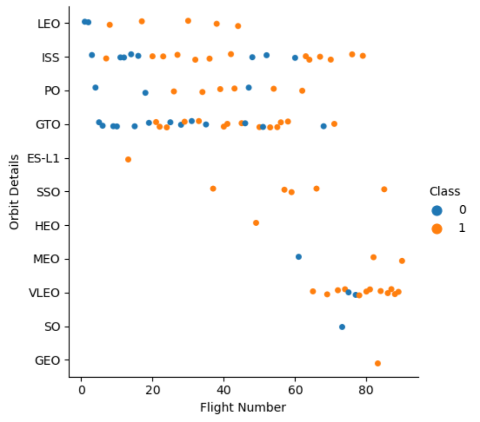
https://github.com/microbytes2005/course6/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

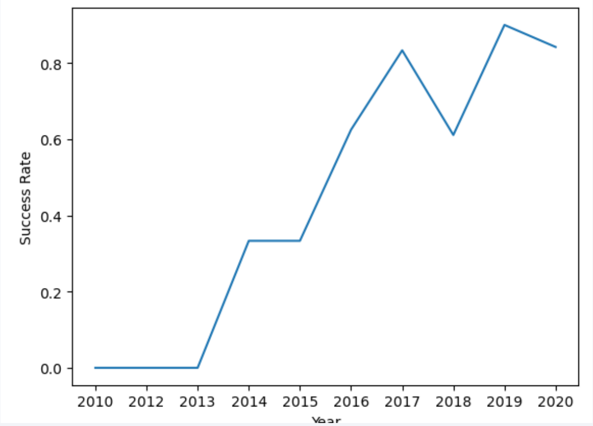


EDA with Data Visualization









https://github.com/microbytes2005/course6/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- **Display the names of the unique launch sites in the space mission**
  `%sql select Unique(LAUNCH_SITE) from SPACEX;`
- **Display 5 records where launch sites begin with the string 'KSC'**
  `%sql SELECT LAUNCH_SITE from SPACEX where (LAUNCH_SITE) LIKE 'KSC%' LIMIT 5;`
- **Display the total payload mass carried by boosters launched by NASA (CRS)**
  `%sql select sum(PAYLOAD_MASS_KG_) as payloadmass from SPACEX;`
- **Display average payload mass carried by booster version F9 v1.1**
  `%sql select avg(PAYLOAD_MASS_KG_) as payloadmass from SPACEX;`
- **List the date where the first successful landing outcome in drone ship was acheived.**
  `%sql select min(DATE) from SPACEX;`
- **List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000**
  `%sql select BOOSTER_VERSION from SPACEX where LANDING_OUTCOME='Success' and PAYLOAD_MASS_KG_ BETWEEN 4000 and 6000;`
- **List the total number of successful and failure mission outcomes**
  `%sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEX GROUP BY MISSION_OUTCOME;`
- **List the names of the booster_versions which have carried the maximum payload mass. Use a subquery**
  `%sql select BOOSTER_VERSION as boosterversion from SPACEX where PAYLOAD_MASS_KG_=(select max(PAYLOAD_MASS_KG_) from SPACEX);`
- **List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017**
  `%sql SELECT MONTH(DATE),LANDING_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEX where EXTRACT(YEAR FROM DATE)='2017';`
- **Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.**
  `%sql SELECT LANDING_OUTCOME FROM SPACEX WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;`

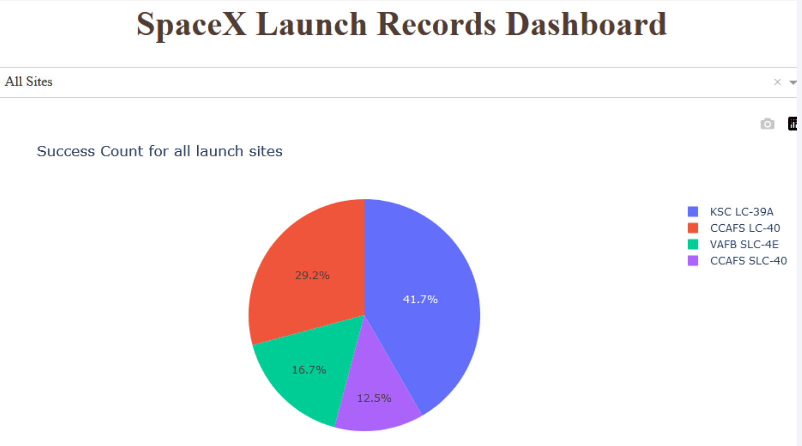
https://github.com/microbytes2005/course6/blob/main/jupyter-labs-eda-sql-edx.ipynb

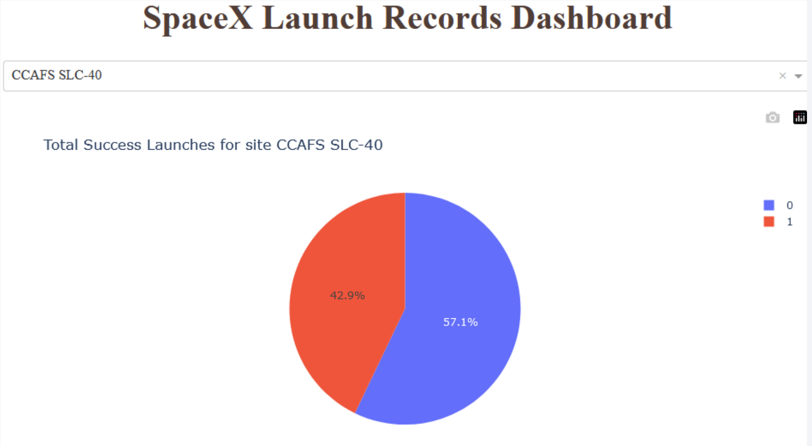
Build an Interactive Map with Folium

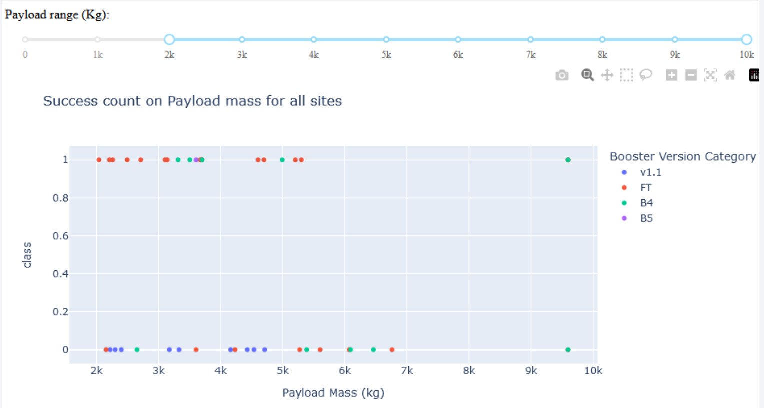
- Create and add folium.Circle and folium.Marker for each launch site on the site map
- folium.Circle is added to highlight a circle area with a text label on a specific coordinate.
- Created markers for all launch records to shhowf a launch was successful (class=1), by using green marker and if a launch failed by using a red marker (class=0).
- Draw a PolyLine between a launch site to the selected point to show distances

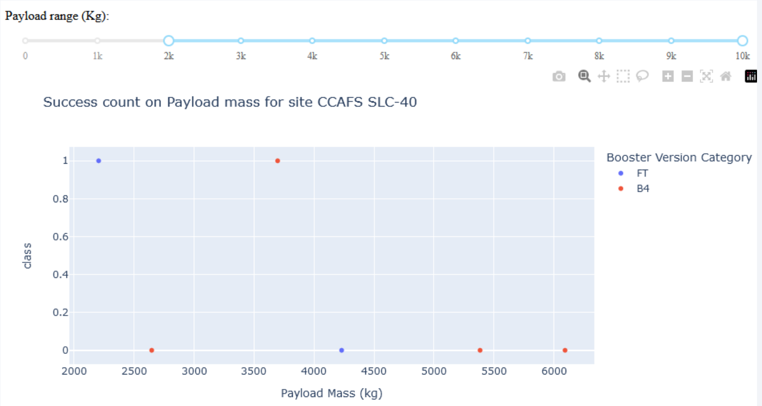
http://localhost:8889/notebooks/jupyter_launch_site_location.jupyterlite.ipynb#

Build a Dashboard with Plotly Dash







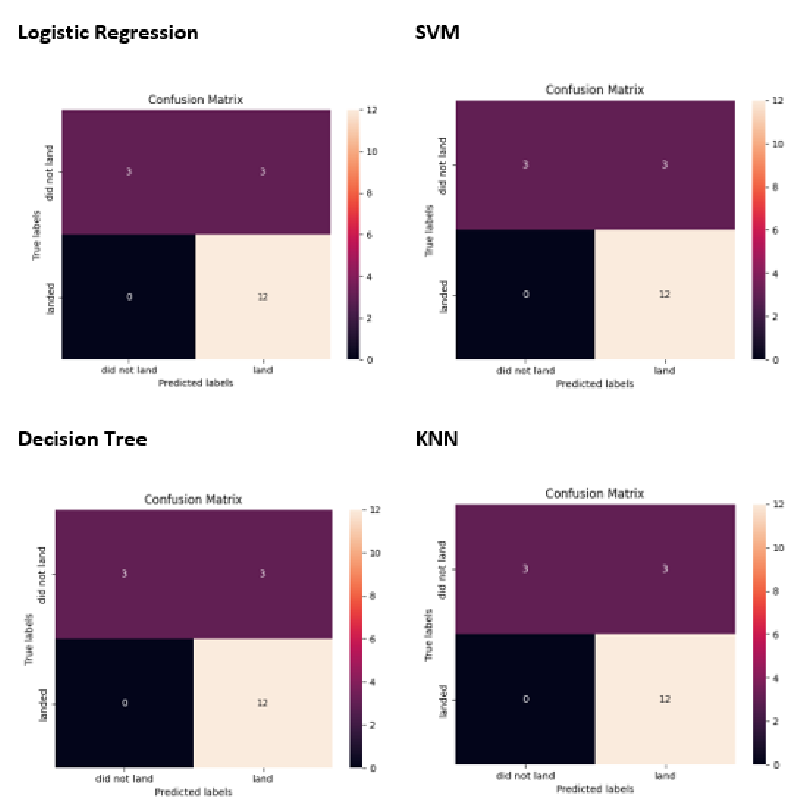


https://github.com/microbytes2005/course6/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- The SVM, KNN, and Logistic Regression model achieved the highest accuracy at 83.3%, while the SVM performs the best in terms of Area under the Curve at 0.958.

https://github.com/microbytes2005/course6/blob/main/labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb



Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

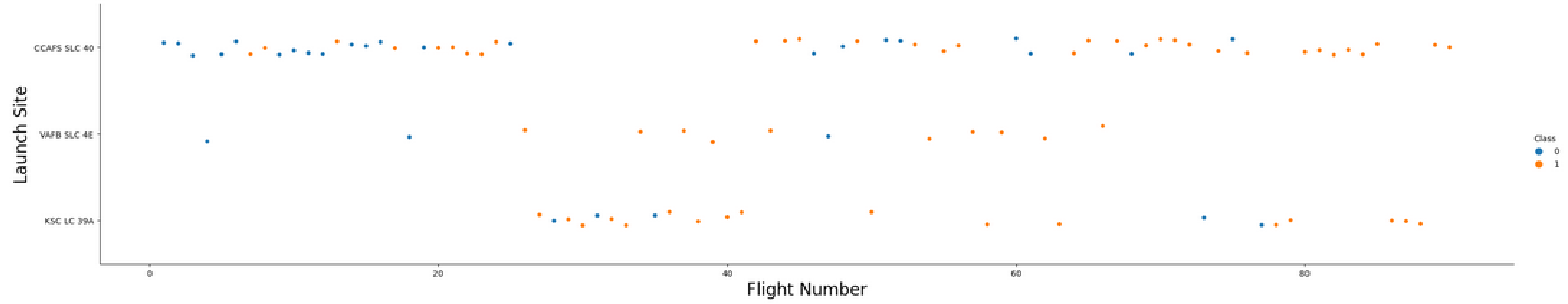
Results

- The SVM, KNN, and Logistic Regression models are the best in terms of prediction accuracy for this dataset.
- Low weighted payloads perform better than the heavier payloads.
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches.
- KSC LC 39A had the most successful launches from all the sites.
- Orbit GEO,HEO,SSO,ES LI has the best Success Rate.

Section 2

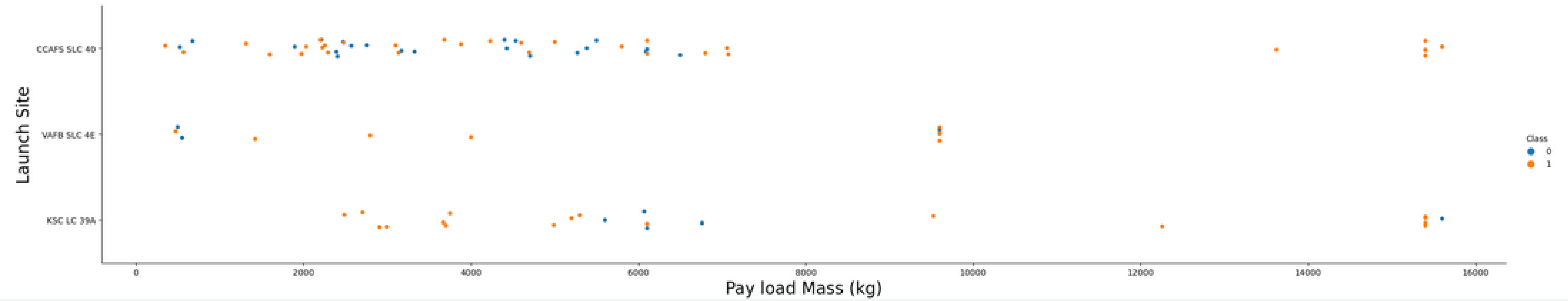
Insights drawn from EDA

Flight Number vs. Launch Site



- Launches from the site of CCAFS SLC 40 are significantly higher than launches form other sites.

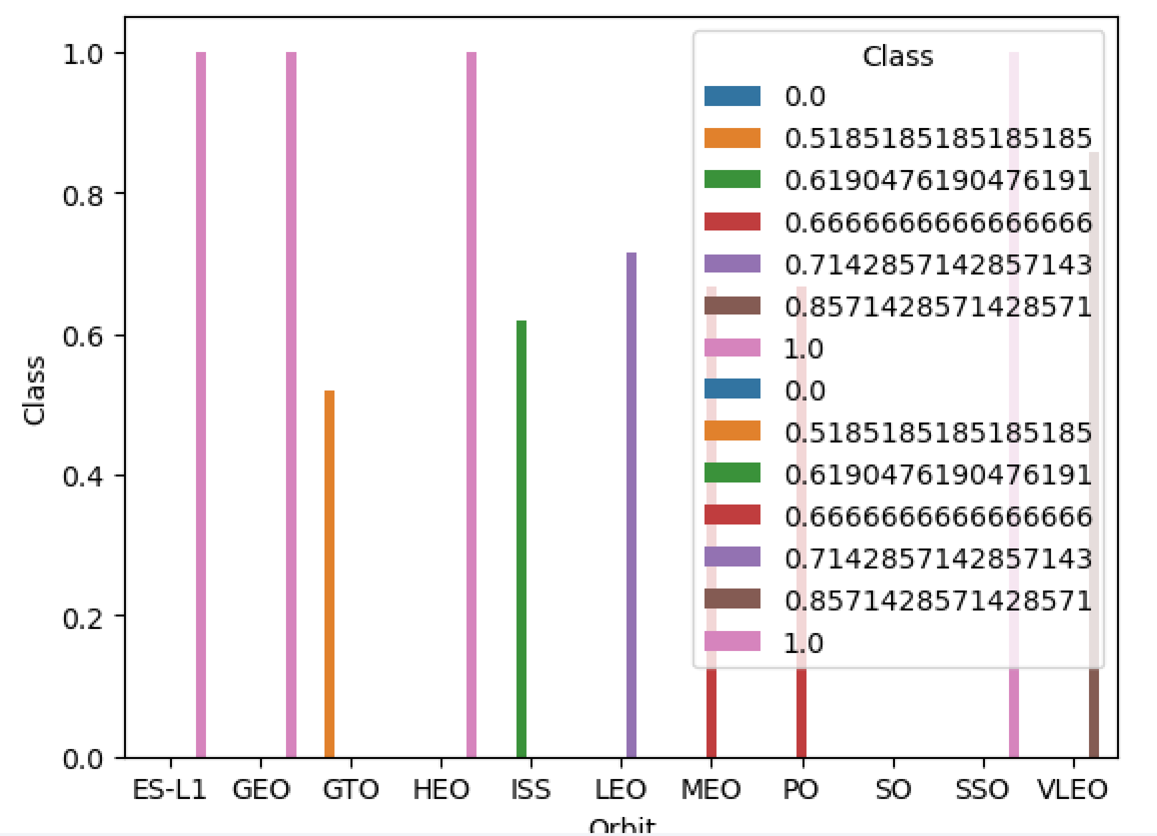
Payload vs. Launch Site



- The majority of IPay Loads with lower mass have been launched from CCAFS SLC 40.

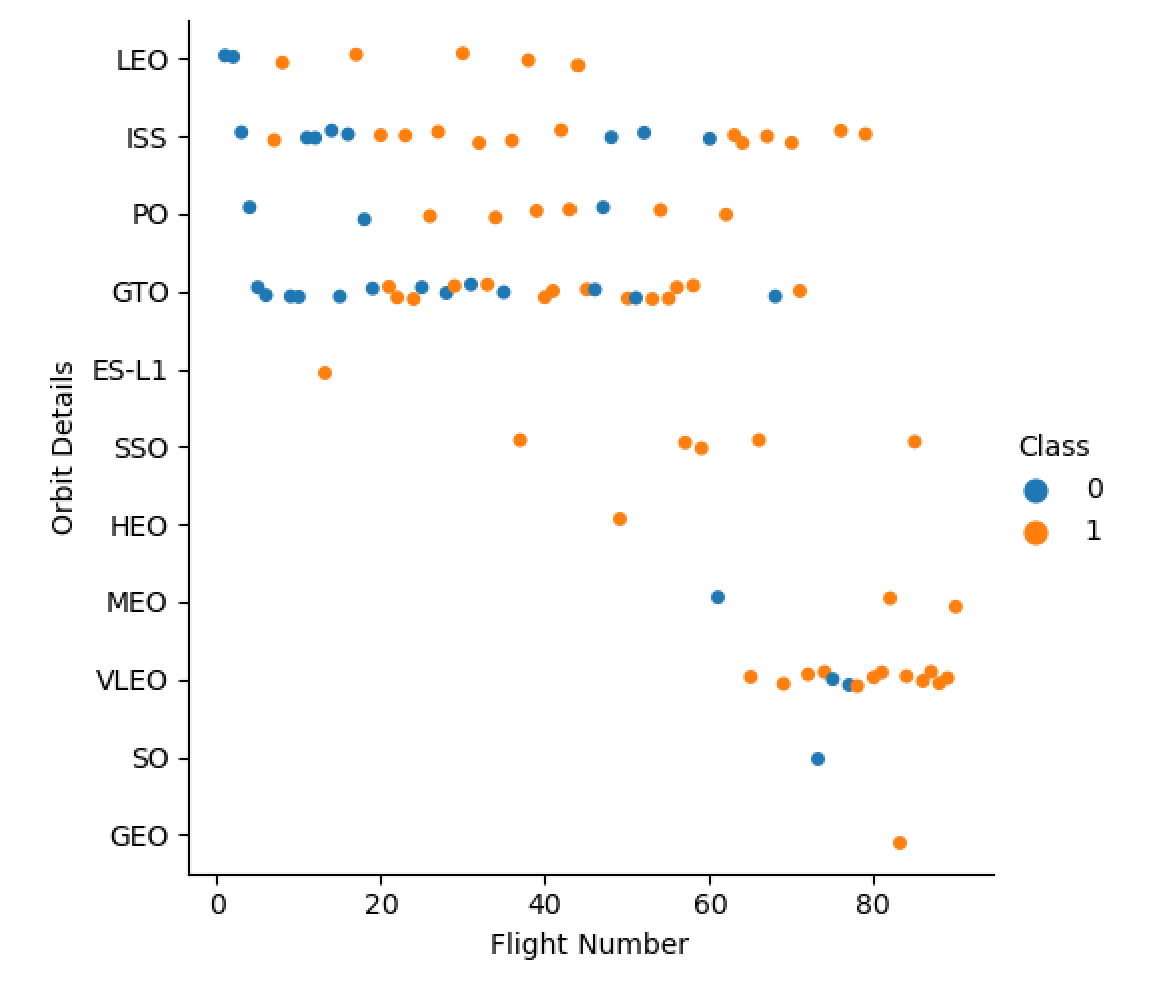
Success Rate vs. Orbit Type

- The orbit types of ES-LI , GEO, HEO, SSO are among the highest success rate.



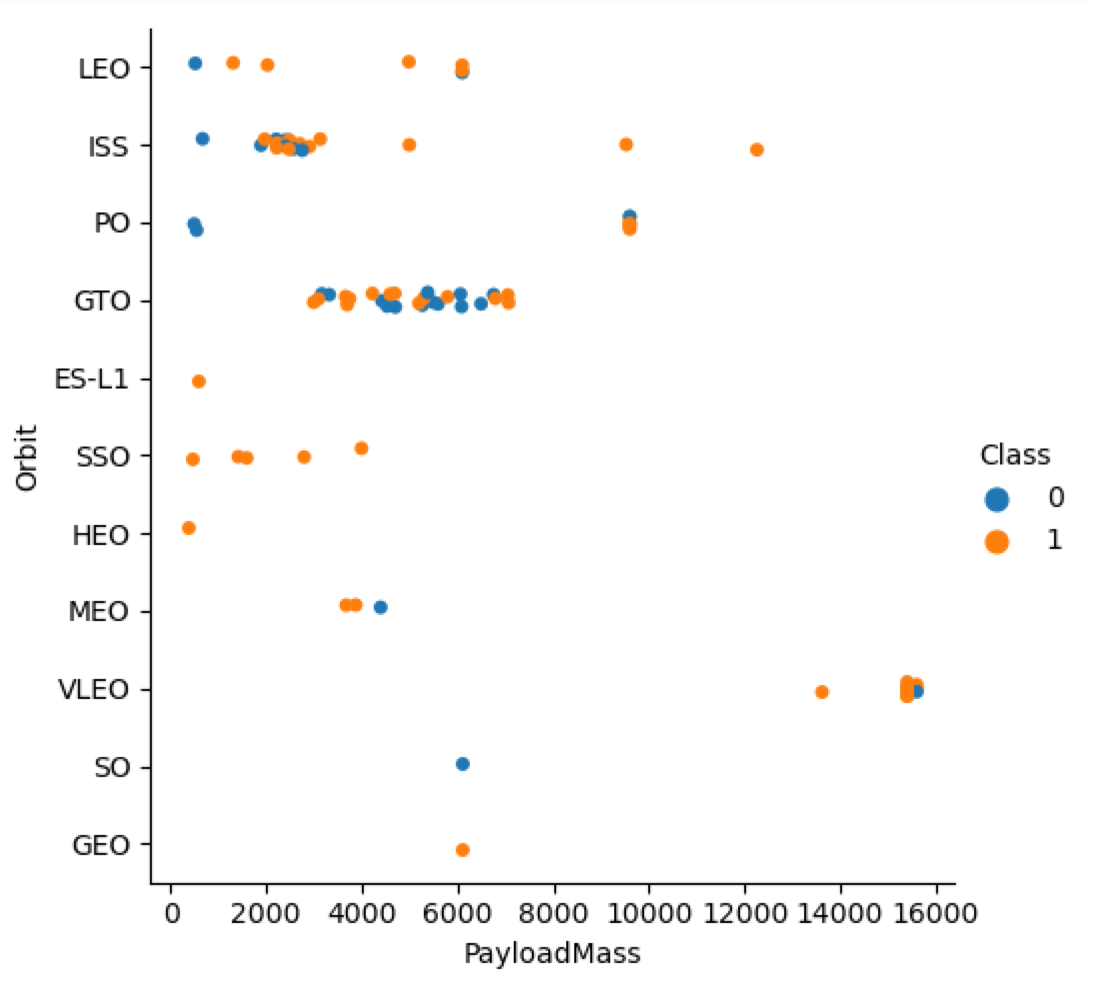
Flight Number vs. Orbit Type

- In LEO orbit the success appears related to the number of flights



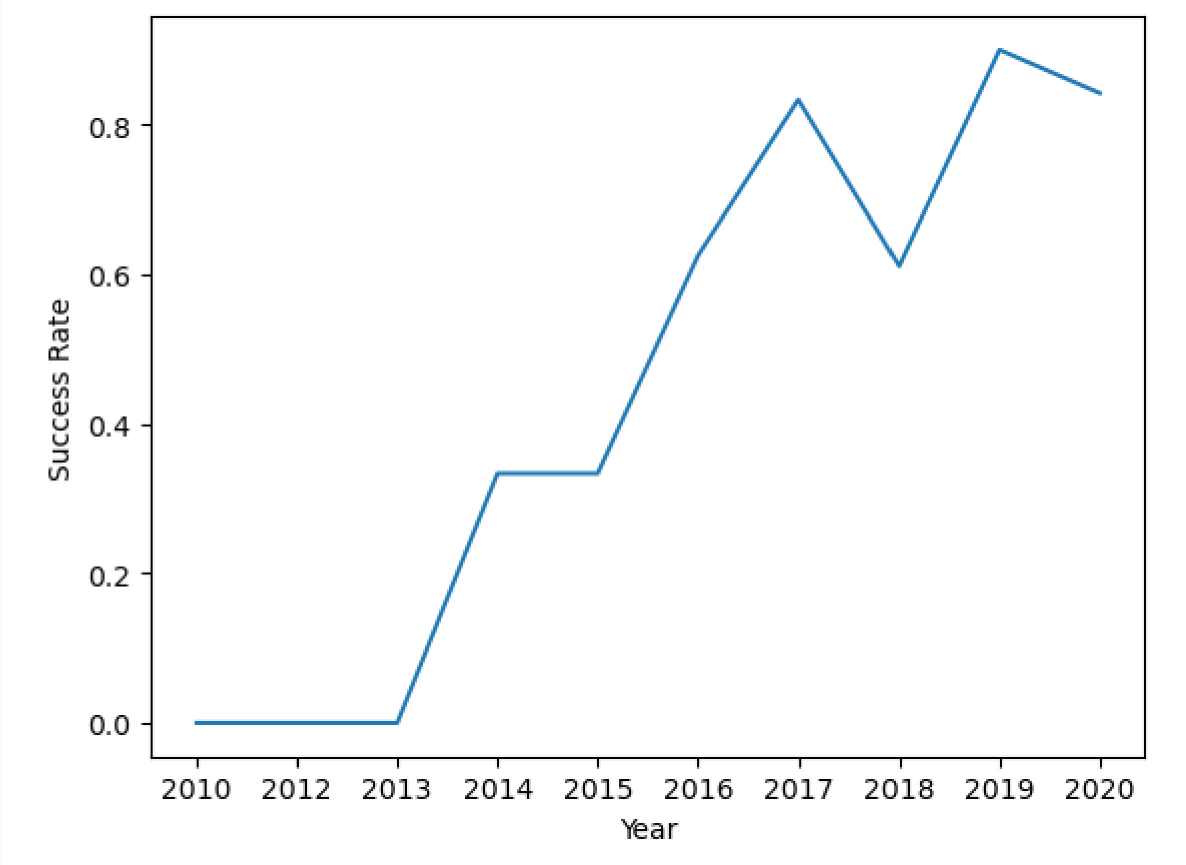
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.



Launch Success Yearly Trend

- Success rate since 2013 kept increasing till 2020



All Launch Site Names

```
%sql select Unique(LAUNCH_SITE) from SPACEX;
```

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

Launch Site Names Begin with 'KSC'

- %sql SELECT LAUNCH_SITE from SPACEX where (LAUNCH_SITE) LIKE 'KSC%' LIMIT 5;

launch_site
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A

Total Payload Mass

- %sql select sum(PAYLOAD_MASS_KG_) as payloadmass from SPACEX;

payloadmass
619967

Average Payload Mass by F9 v1.1

- %sql select avg(PAYLOAD_MASS_KG_) as payloadmass from SPACEX;

payloadmass
6138

First Successful Ground Landing Date

- %sql select min(DATE) from SPACEX;

1
2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

- %sql select BOOSTER_VERSION from SPACEX where LANDING_OUTCOME='Success' and PAYLOAD_MASS_KG_ BETWEEN 4000 and 6000;

booster_version
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1046.3
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

Total Number of Successful and Failure Mission Outcomes

- %sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEX GROUP BY MISSION_OUTCOME;

missionoutcomes
1
99
1

Boosters Carried Maximum Payload

- %sql select BOOSTER_VERSION as boosterversion from SPACEX where PAYLOAD_MASS_KG_=(select max(PAYLOAD_MASS_KG_) from SPACEX);

boosterversion
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

2017 Launch Records

- %sql SELECT MONTH(DATE), LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX where EXTRACT(YEAR FROM DATE)='2017';

1	landing_outcome	booster_version	launch_site
1	Success (drone ship)	F9 FT B1029.1	VAFB SLC-4E
2	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
3	No attempt	F9 FT B1030	KSC LC-39A
3	Success (drone ship)	F9 FT B1021.2	KSC LC-39A
5	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
5	No attempt	F9 FT B1034	KSC LC-39A
6	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
6	Success (drone ship)	F9 FT B1029.2	KSC LC-39A
6	Success (drone ship)	F9 FT B1036.1	VAFB SLC-4E
7	No attempt	F9 FT B1037	KSC LC-39A
8	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
8	Success (drone ship)	F9 FT B1038.1	VAFB SLC-4E
9	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
10	Success (drone ship)	F9 B4 B1041.1	VAFB SLC-4E
10	Success (drone ship)	F9 FT B1031.2	KSC LC-39A
10	Success (drone ship)	F9 B4 B1042.1	KSC LC-39A
12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40
12	Controlled (ocean)	F9 FT B1036.2	VAFB SLC-4E

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

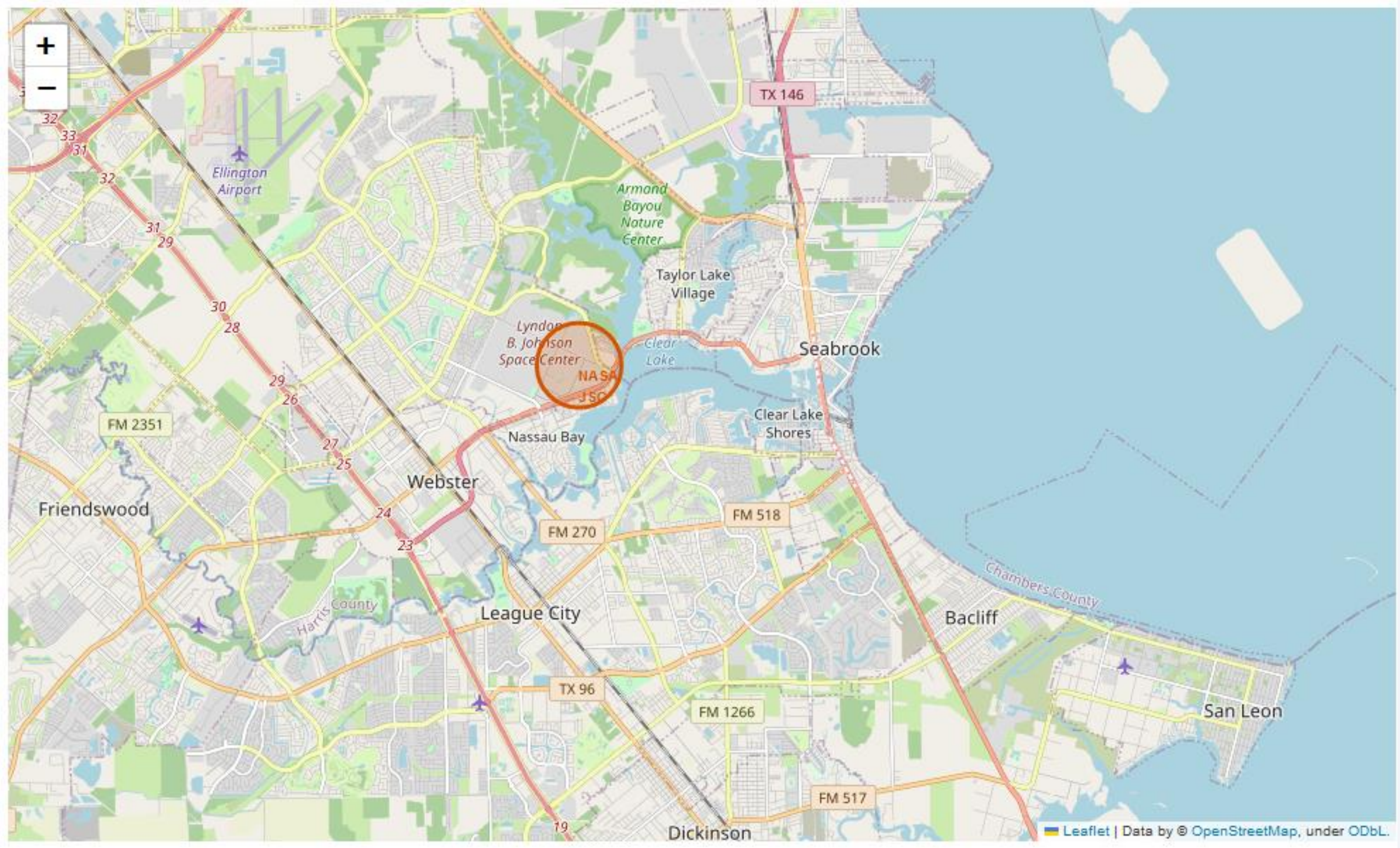
- %sql SELECT LANDING_OUTCOME FROM SPACEX WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;

landing_outcome
No attempt
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)
Precluded (drone ship)
No attempt
Failure (drone ship)
Failure (drone ship)
No attempt
Controlled (ocean)
Failure (drone ship)
Uncontrolled (ocean)
No attempt
No attempt
Controlled (ocean)
Controlled (ocean)
No attempt
No attempt
Uncontrolled (ocean)
No attempt
No attempt
No attempt
Failure (parachute)
Failure (parachute)

Section 3

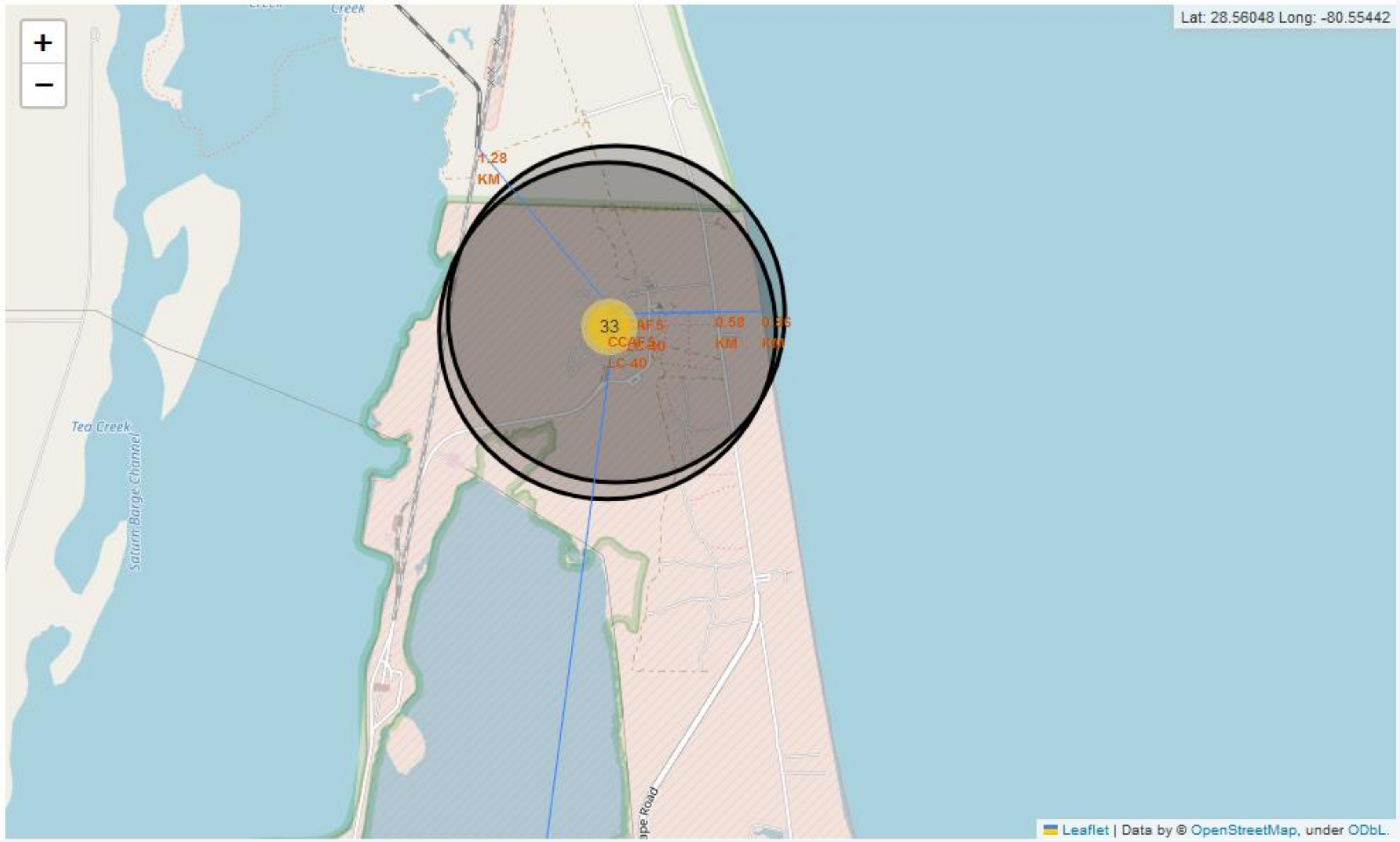
Launch Sites Proximities Analysis

All launch sites marked on a map



Success/failed launches marked on the map

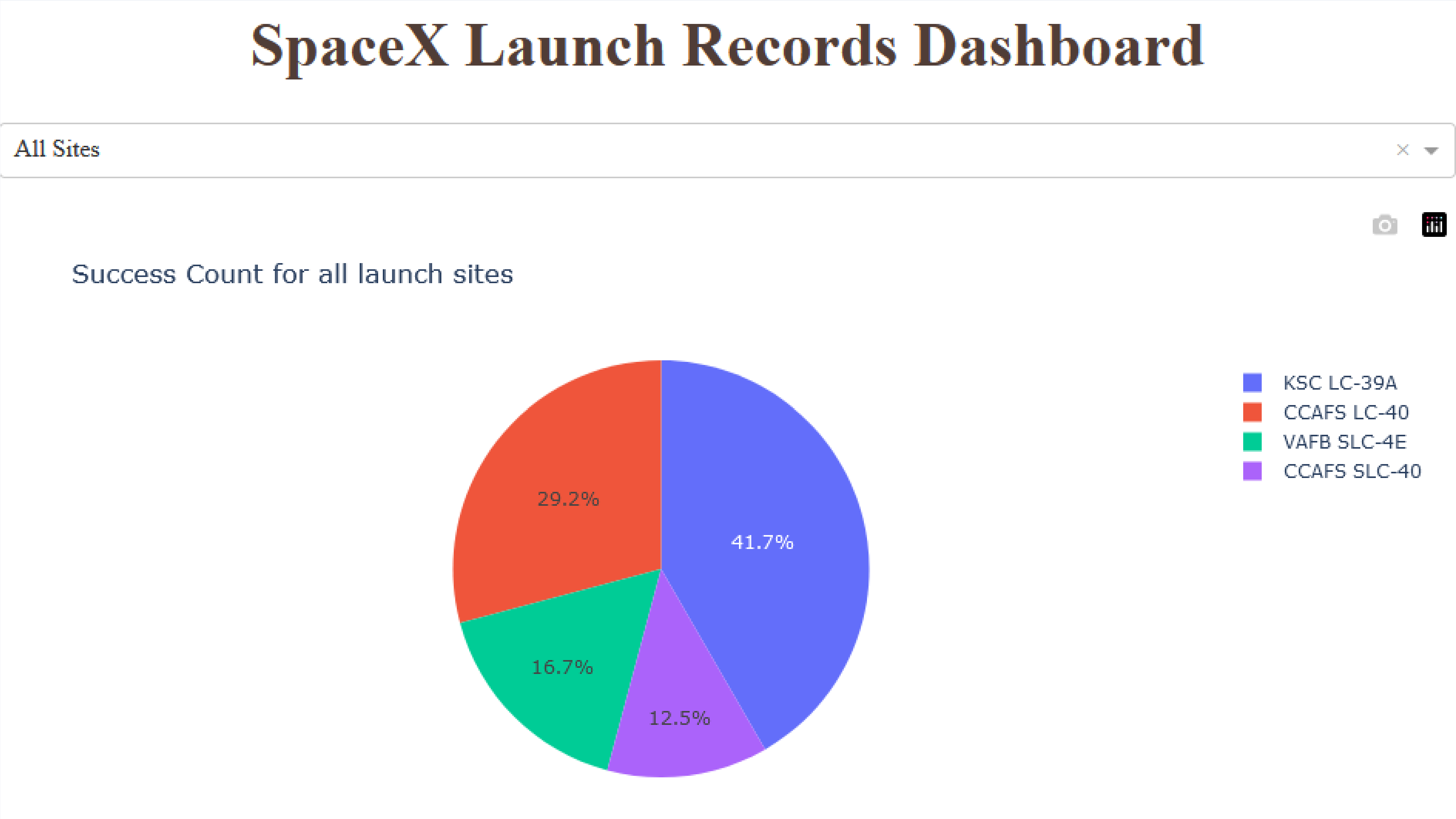
Distances between a launch site to its proximities



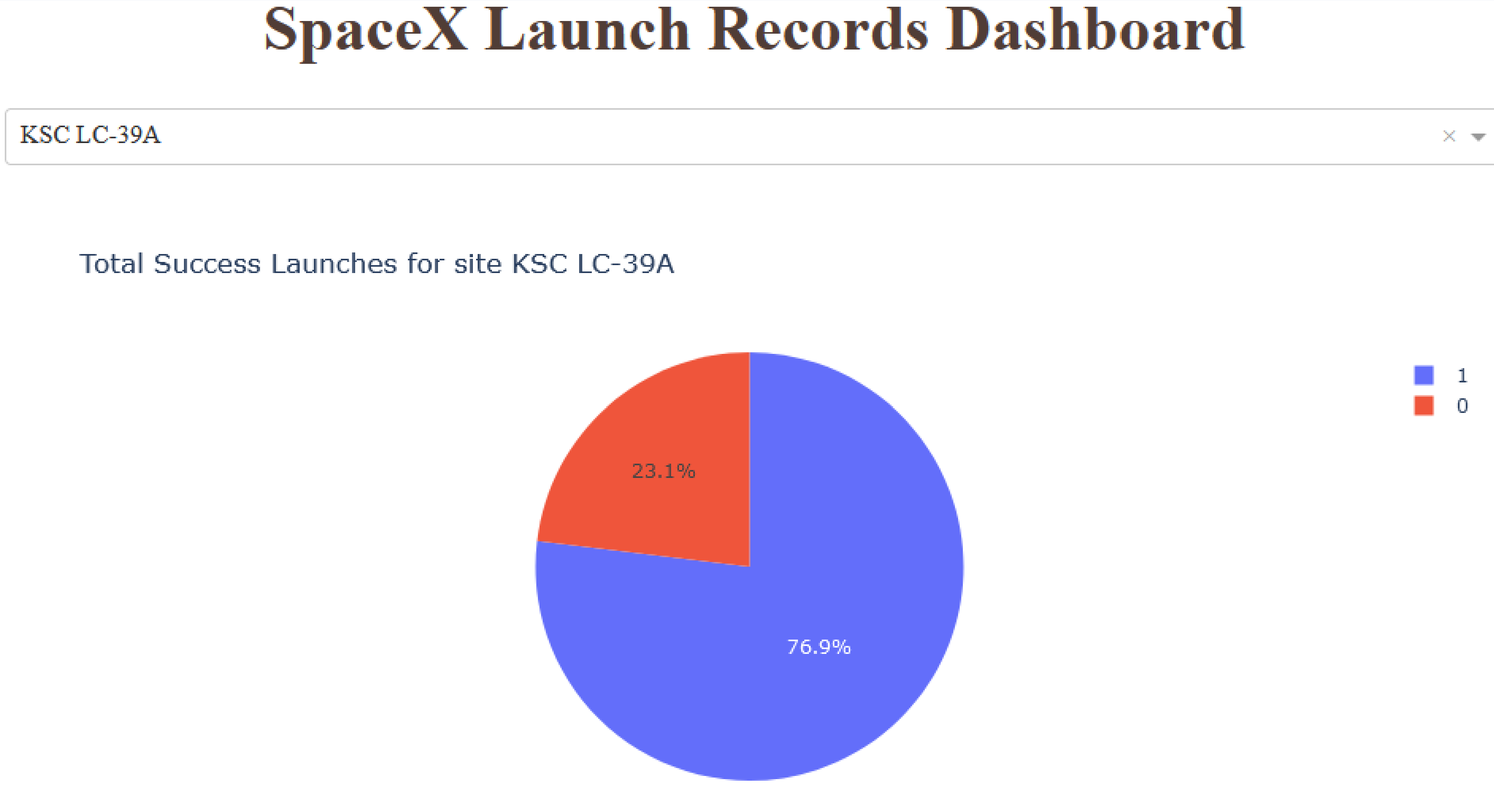
Section 4

Build a Dashboard with Plotly Dash

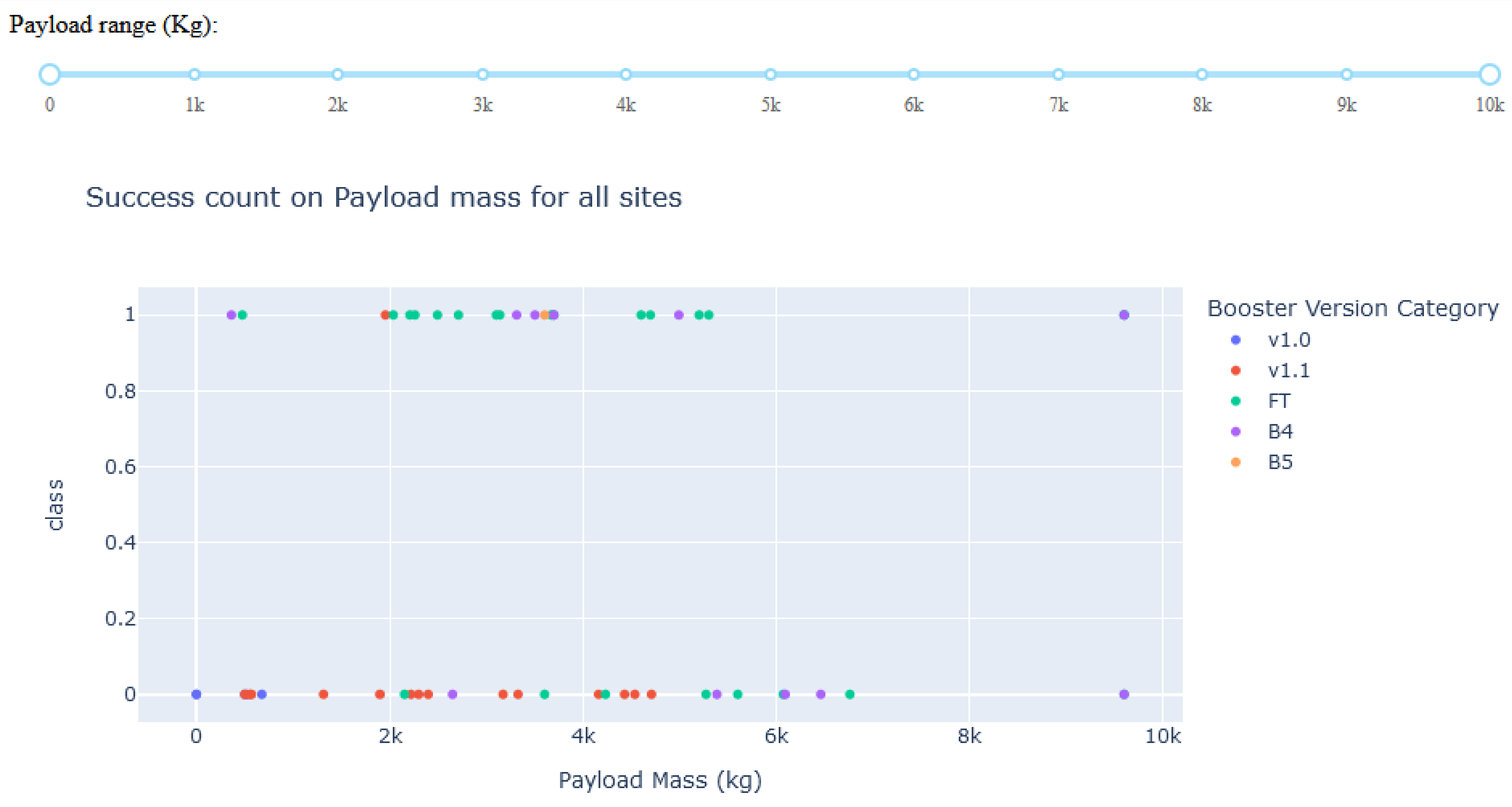
Total success launches by all sites



Success rate by site



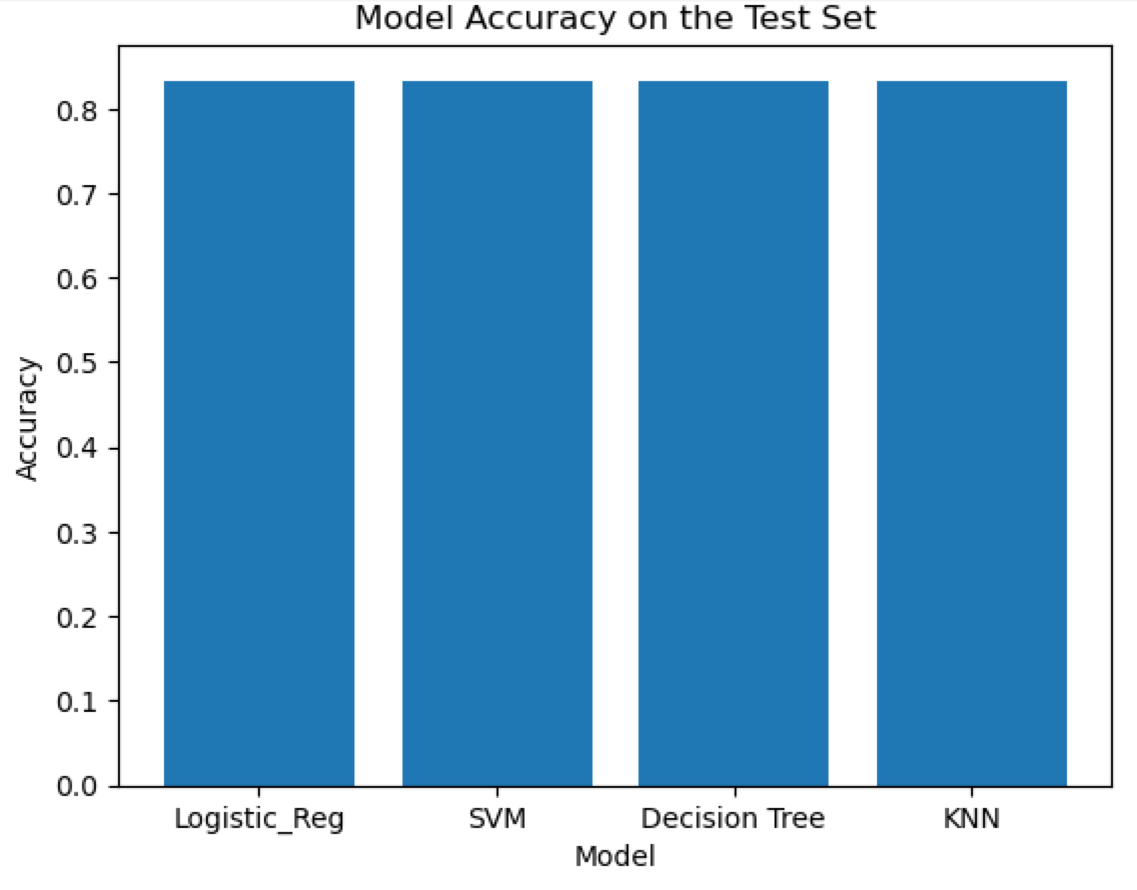
Payload vs Launch Outcome



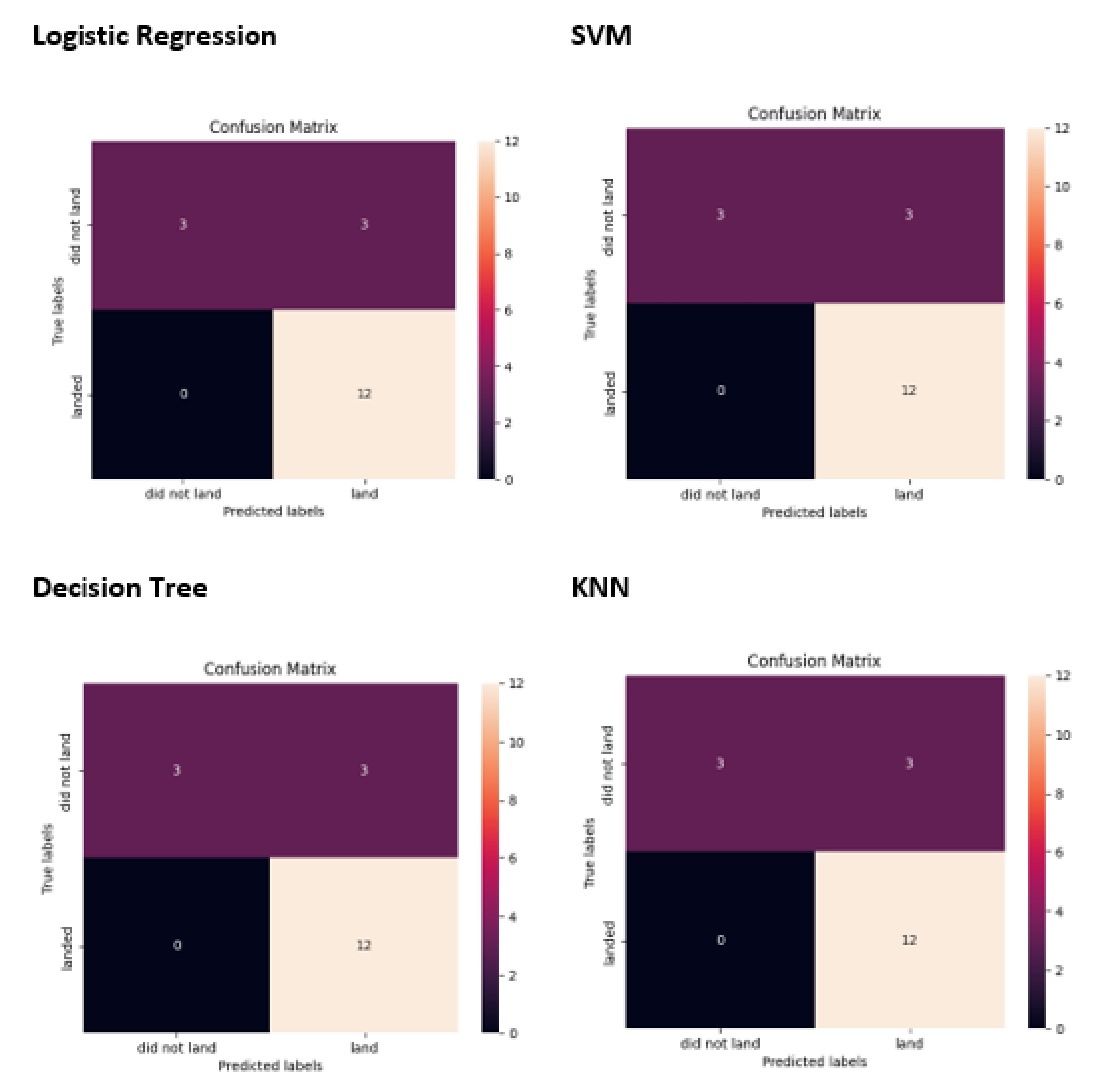
Section 5

Predictive Analysis (Classification)

Classification Accuracy



Confusion Matrix



Conclusions

- The SVM, KNN, and Logistic Regression models are the best in terms of prediction accuracy for this dataset.
- Low weighted payloads perform better than the heavier payloads.
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches.
- KSC LC 39A had the most successful launches from all the sites.
- Orbit GEO,HEO,SSO,ES LI has the best Success Rate.

Appendix

```python
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px

# Read the airline data into pandas dataframe
csv = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_dash.csv"
spacex_df = pd.read_csv(csv)
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

# Create a dash application
app = dash.Dash(__name__)
```

Thank you!