IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Patrick P
15.08.23

Outline

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

Executive Summary

Summary of methodologies

- Data collection & wrangling
- EDA (SQL, visualization)
- Interactive map (Folium)
- Dashboard (Plotly Dash)
- Predictive analysis

Summary of all results

- EDA results
- Interactive analytics
- Predictive analysis

Introduction

Project background and context

- SpaceX's Falcon 9 rockets stand out due to their reusability, enabling launches at $62 million compared to competitors' $165 million.

Problems you want to find answers

- Launch site success rates, reliability, prediction

Section 1

Methodology

Methodology

Data collection methodology:

- SpaceX Rest API
- Web Scraping Wikipedia Data

Data wrangling

- One Hot Encoding data fields for machine learning and data cleaning

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

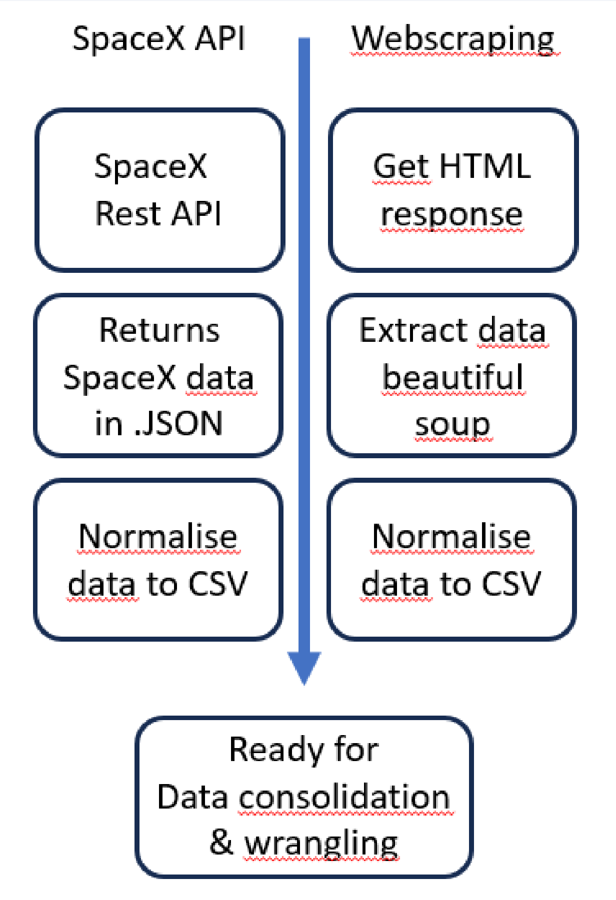
Data Collection

Datasets:

- SpaceX launch data => SpaceX REST API
- Falcon 9 launch data => Wikipedia

Data includes:

- launches, launch specifications
- rocket usage, payload delivered
- landing specifications and landing outcomes



Data Collection – SpaceX API

Data Collection with SpaceX Rest calls examples:

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

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

```
# Use json_normalize meethod to convert th
data = pd.json_normalize(response.json())
```

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
```

```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Getting Response from API

Converting Response to a .json file

Apply custom functions to clean data

Assign list to dictionary and dataframe

Filter dataframe and export to .csv

https://github.com/PaIBM/IBM_Data_Science_Professional_Certification/blob/d2f57f526effbc6ebdbf26f4d228ec2e0d4dae70/IBM-DS0321EN-SkillsNetwork_labs_module_1_L1_labs-jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

- Data Collection with SpaceX Rest calls examples:

```python
# check for successful response
if response.status_code != 200:
    print('Failed to fetch page')
else:
    print('Page fetched successfully')
```

```
Page fetched successfully
```

```python
soup = BeautifulSoup(response.text, 'html.parser')
```

```python
html_tables = soup.find_all('table')
```

```python
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)
```

```python
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 an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
```

Getting Response from HTML

Creating a BeautifulSoup Object

Finding the tables

Getting column names

Creating a dictionary

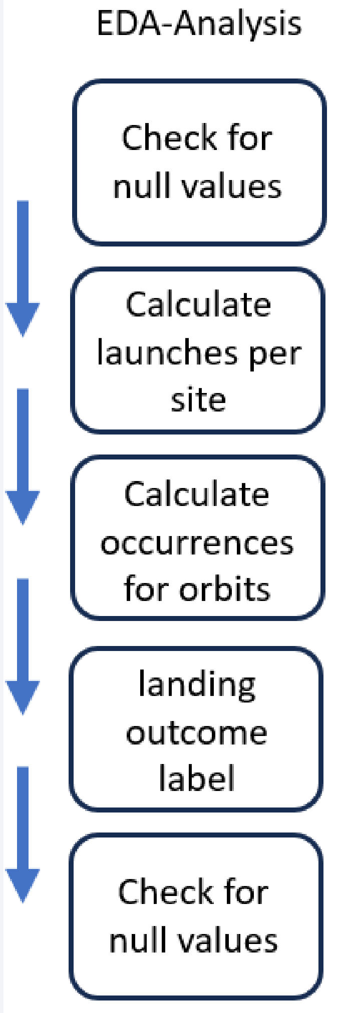
Appending data to keys

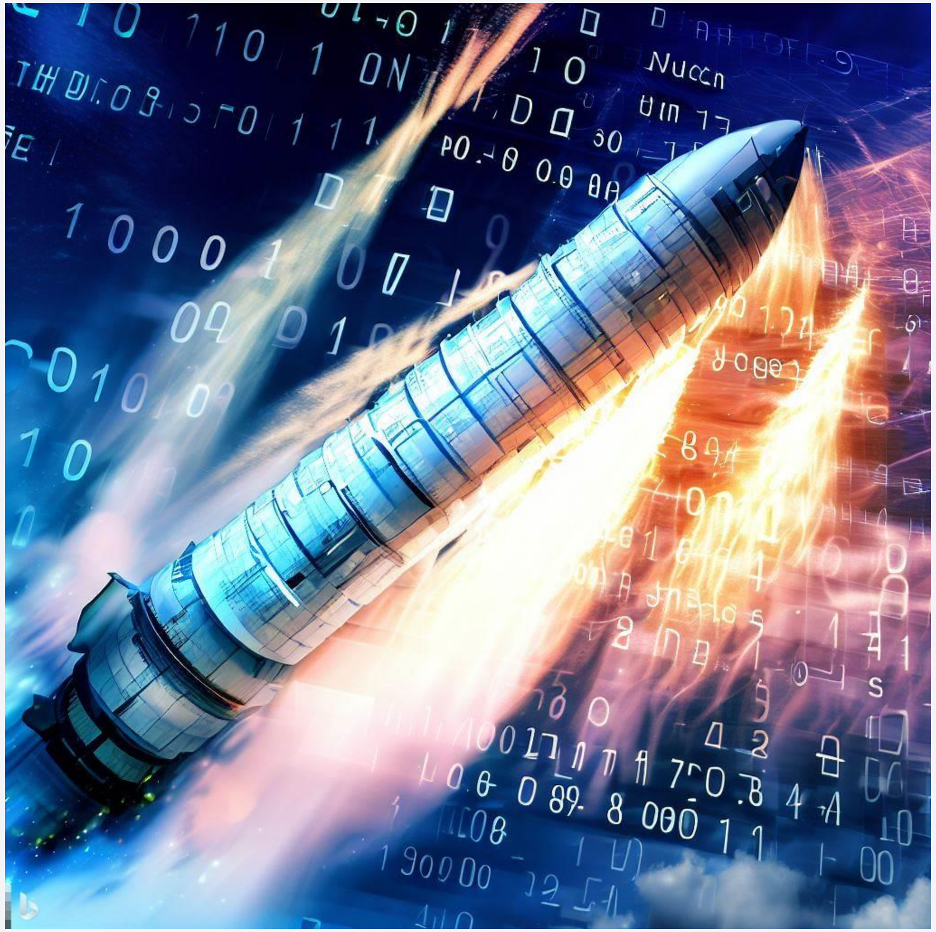
Converting dictionary to dataframe

Dataframe to .csv

https://github.com/PaIBM/IBM_Data_Science_Professional_Certification/blob/d2f57f526effbc6ebdbf26f4d228ec2e0d4dae70/IBM-DS0321EN-SkillsNetwork_labs_module_1_L2_jupyter-labs-webscraping.ipynb

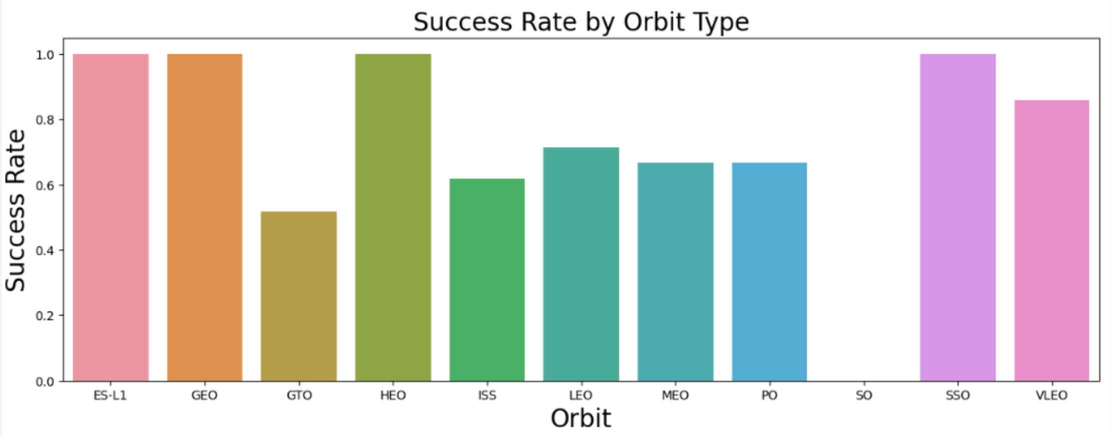
Data Wrangling

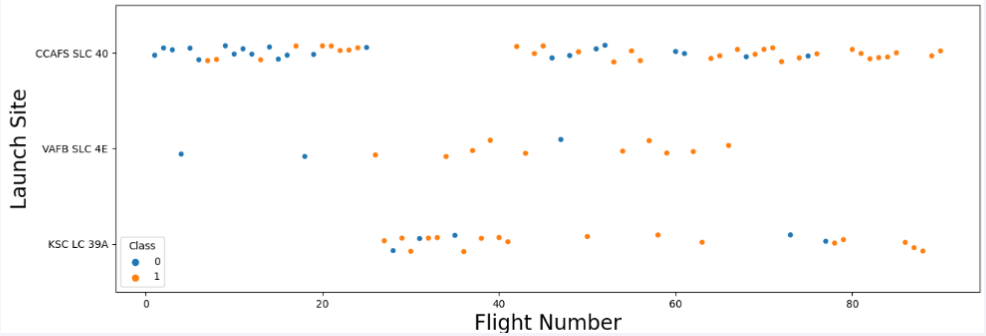


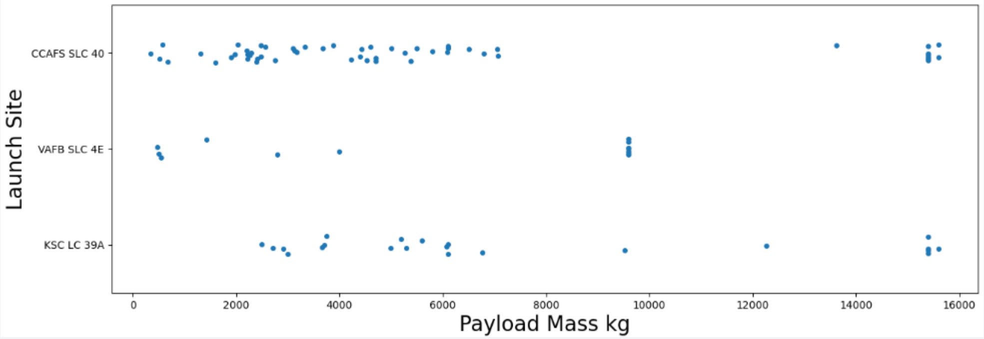


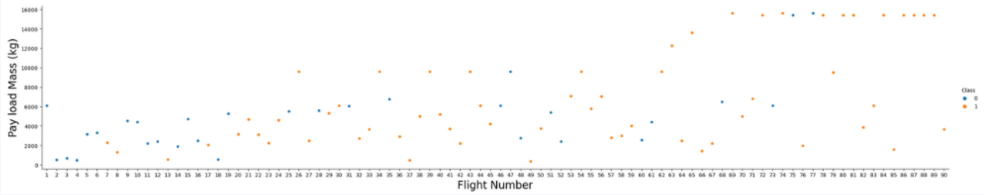
https://github.com/PaIBM/IBM_Data_Science_Professional_Certification/blob/d2f57f526effbc6ebdbf26f4d228ec2e0d4dae70/IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

EDA with Data Visualization









https://github.com/PaIBM/IBM_Data_Science_Professional_Certification/blob/abad505a99dfaed6a0b577de02b5c854441e70e2/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

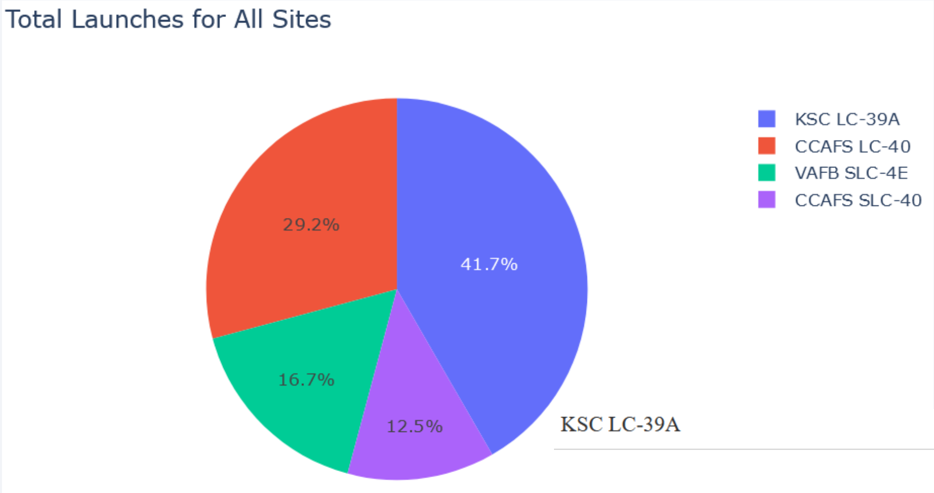
- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string "KSC"
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying the average payload mass carried by booster version F9 v1.1
- Listing dates with successful landing outcomes on drone ships
- Listing the booster names that landed successfully (payload mass > 4000 < 6000)
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster versions that carried maximum payload mass
- Listing the records which will display the month names
- Listing successful landing outcomes on ground pad, booster versions, launch site for the months of 2017
- Ranking the count of successful landing outcomes in descending order (04.06.2010 to 20.03.2017)

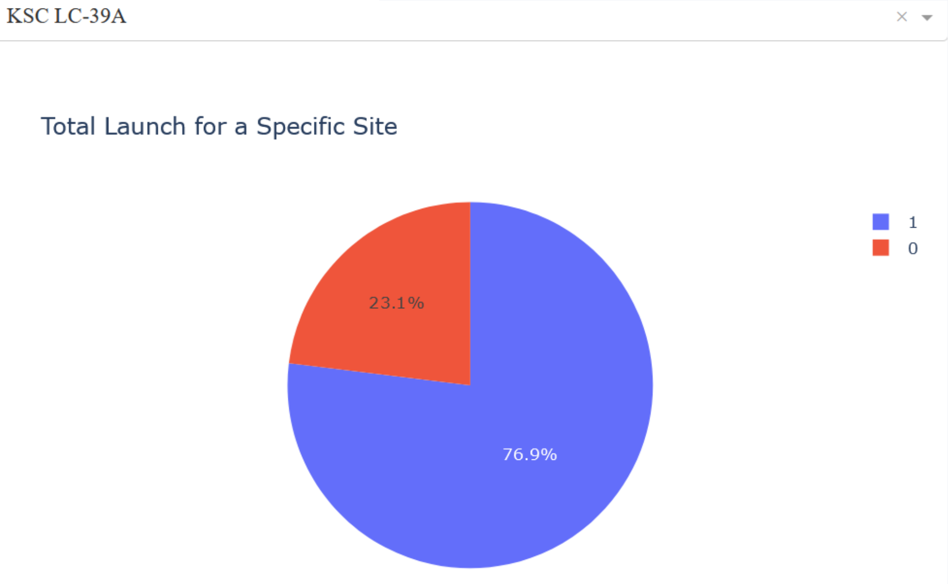
GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

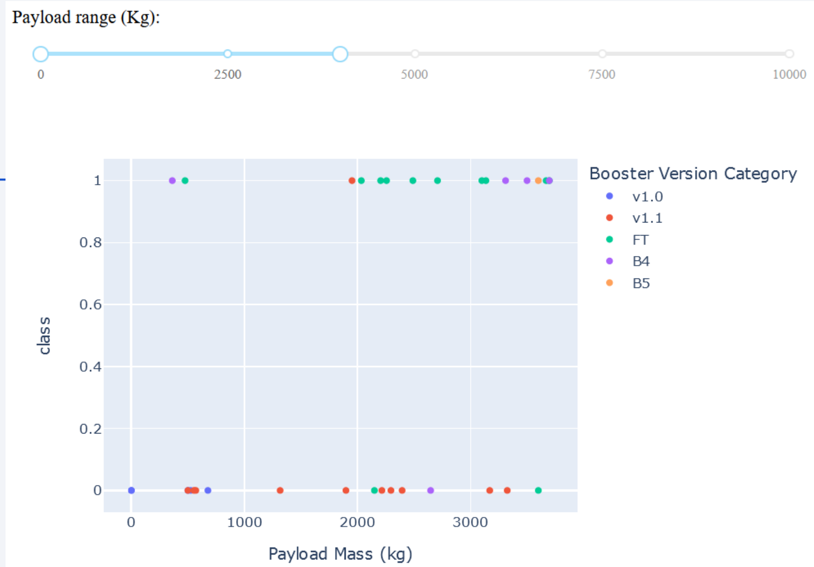
Build an Interactive Map with Folium

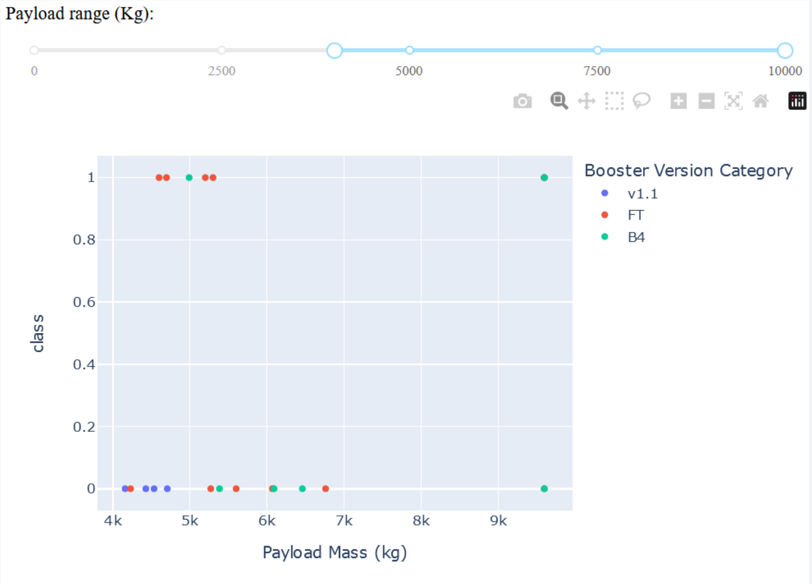
https://github.com/PaIBM/IBM_Data_Science_Professional_Certification/blob/3db94d220448e10b5c80c1cfd01e181c8aa5244f/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash





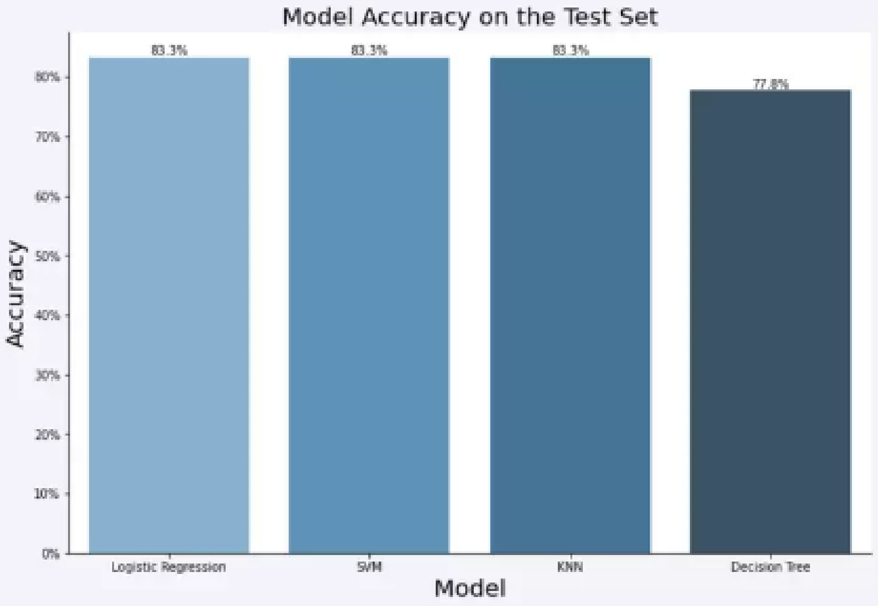


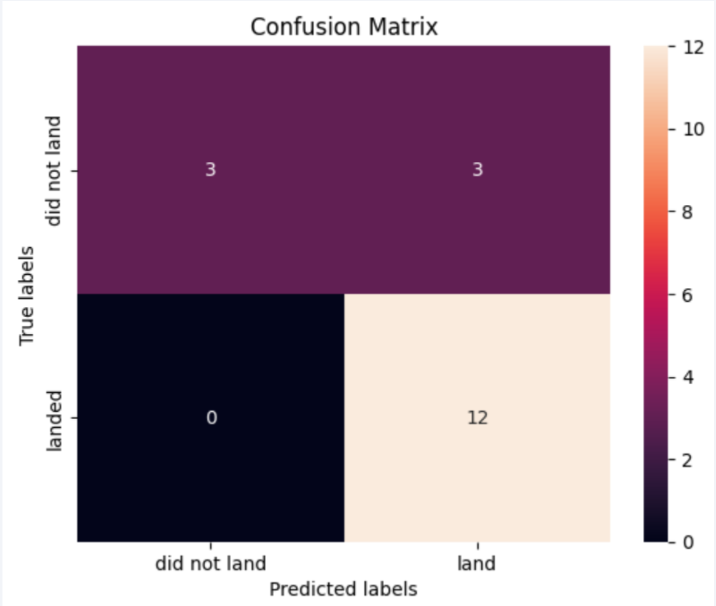


https://github.com/PaIBM/IBM_Data_Science_Professional_Certification/blob/abad505a99dfaed6a0b577de02b5c854441e70e2/spacex_dash_app.py

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/PaIBM/IBM_Data_Science_Professional_Certification/blob/d2f57f526effbc6ebdbf26f4d228ec2e0d4dae70/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

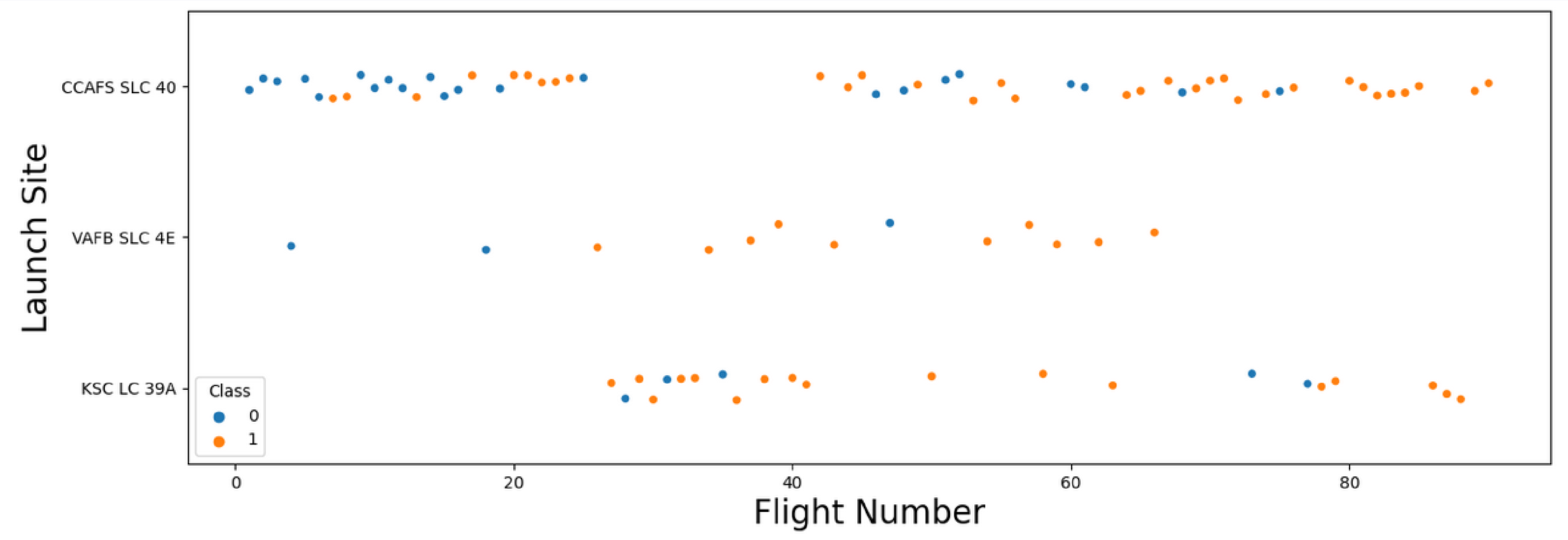
- The SVM, KNN and Logistic Regression models the best in terms of prediction accuracy for this dataset.
- Low weighted payloads perform better than the heavier payloads
- KSC LC 39A had the most successful launches from all the sites
- Orbit GEO, HEO, SSO, ES L1 had the best success rates

Section 2

Insights drawn from EDA

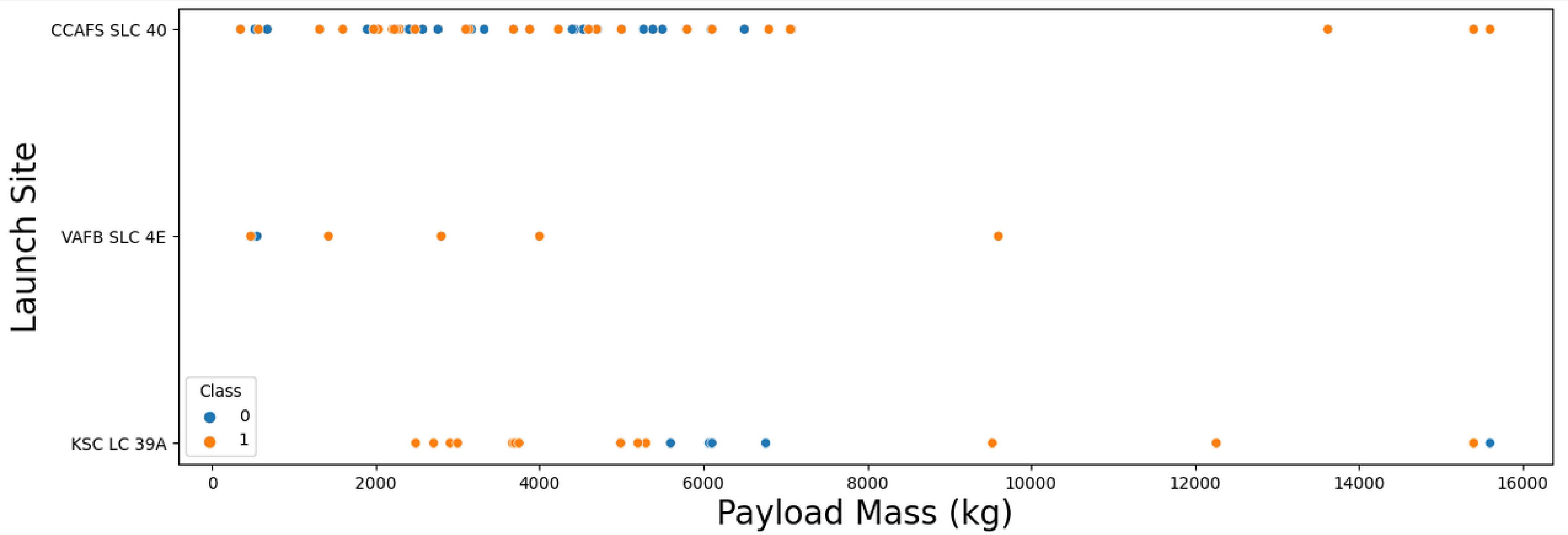
Flight Number vs. Launch Site

- CCAFS SLC 40 has more launches than the other two combined



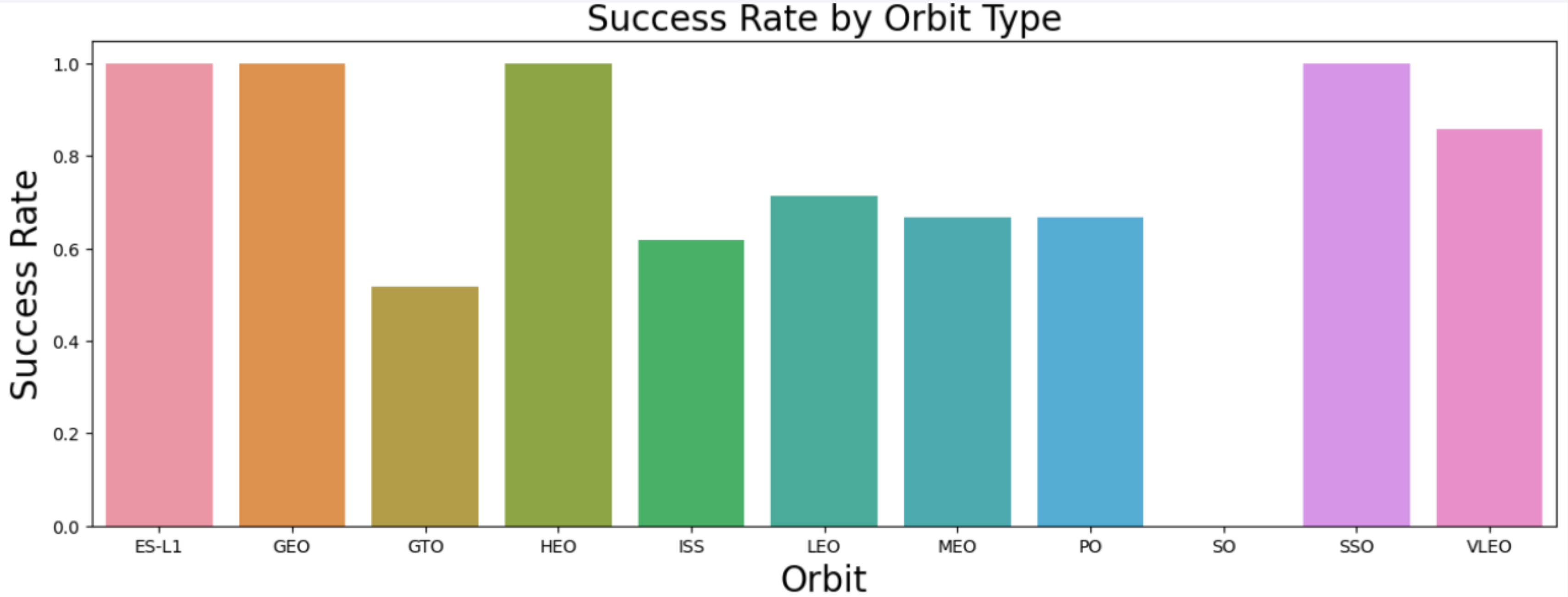
Payload vs. Launch Site

- CCAFS SLC 40 was used for the majority of lower payload masses



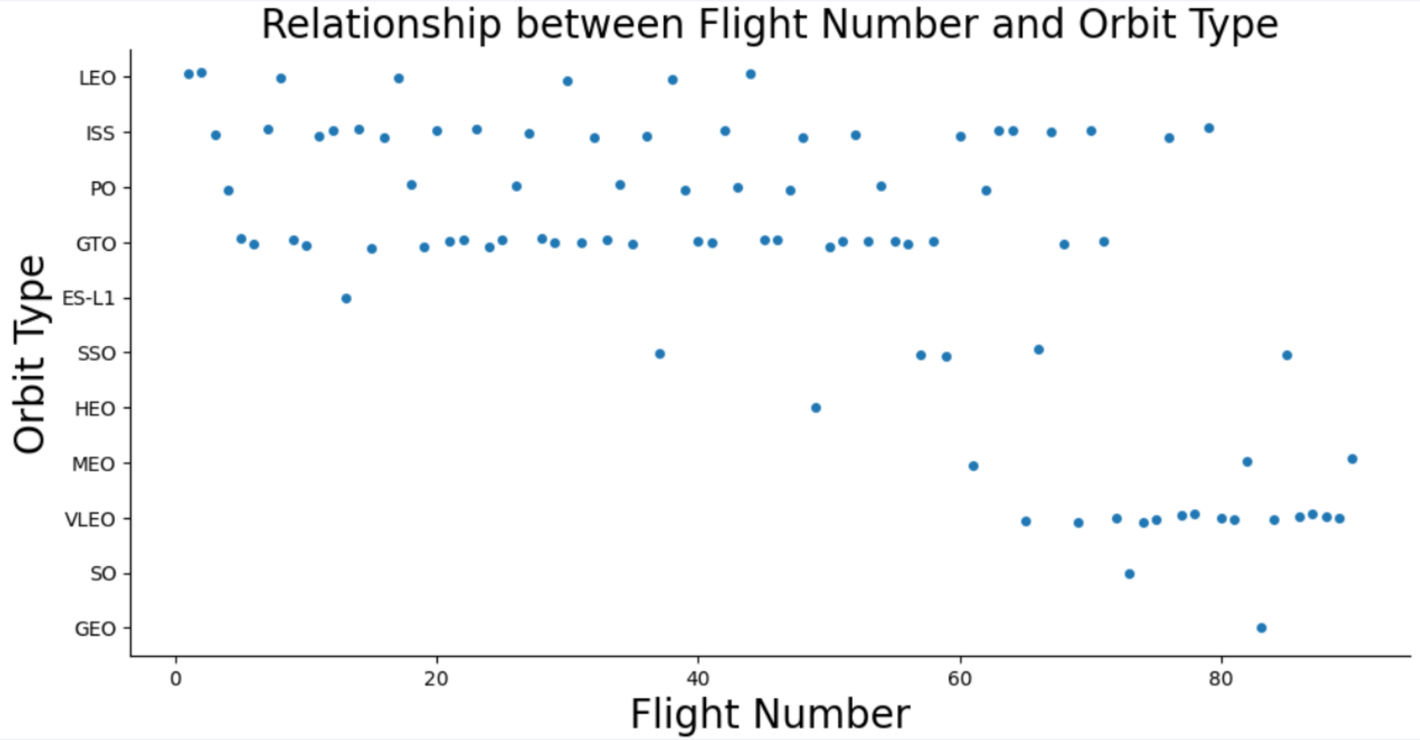
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO have the highest success rates



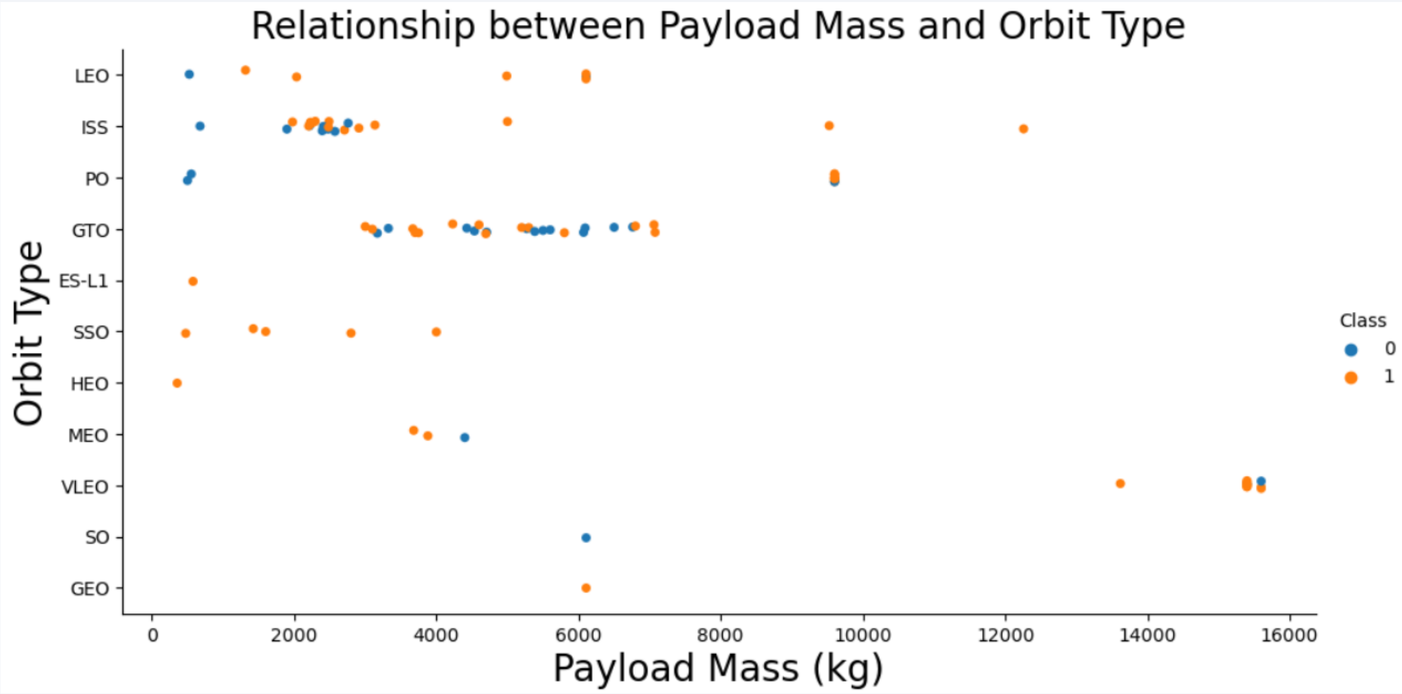
Flight Number vs. Orbit Type

- ISS and GTO were evenly distributed for the first 60-80 flights
- VLEO Orbits have been more dominant in the recent years



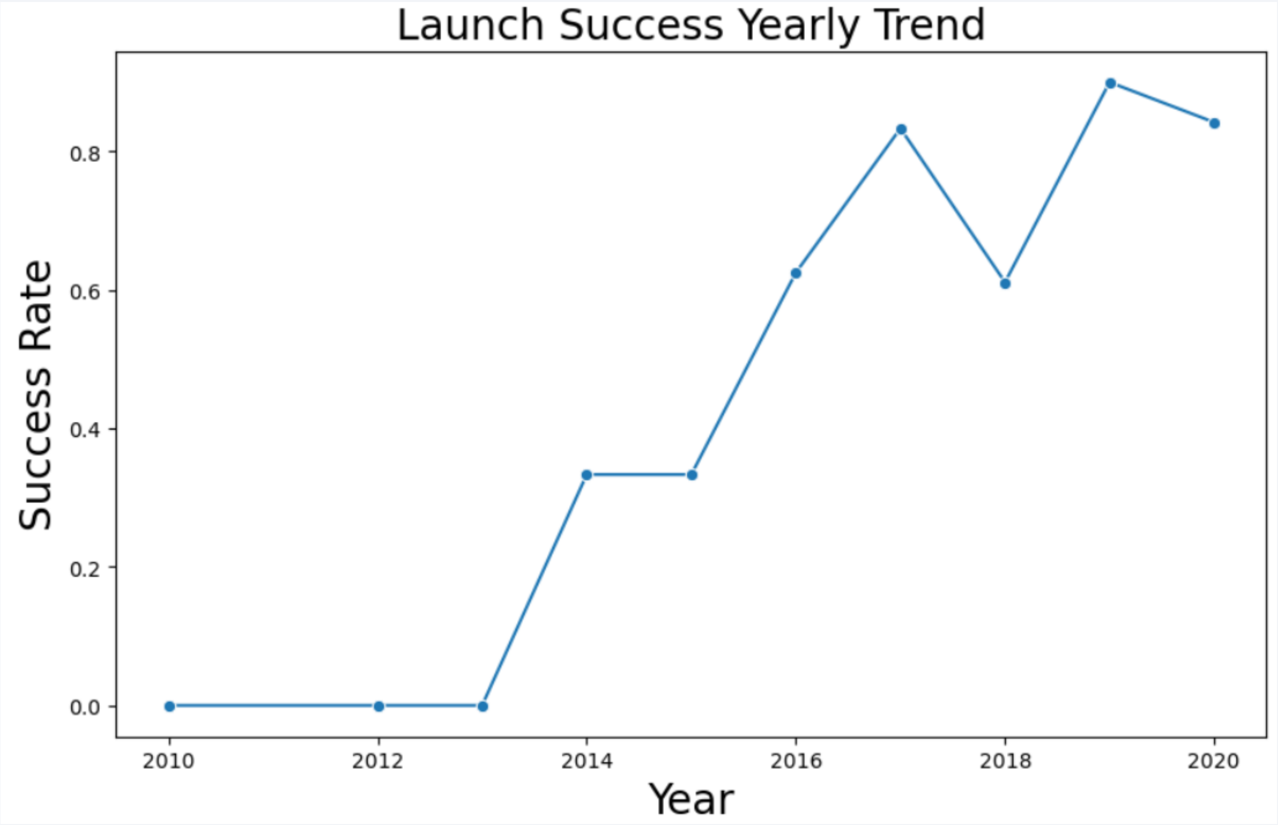
Payload vs. Orbit Type

- ISS received mostly payloads of around 2000kg
- GTO orbits got between 3000 and 8000kg
- With heavy payloads the successful landings are more for Polar, LEO and ISS.



Launch Success Yearly Trend

- Success rates increased significantly since 2013 stabilising on a high level since 2019



All Launch Site Names

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
```

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

Launch Site Names Begin with 'CCA'

```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	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_) AS Total_Payload_Mass FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

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

Avg_Payload_Mass
2534.6666666666665

First Successful Ground Landing Date

```
%sql SELECT MIN(Date) AS First_Successful_Landing FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';
```

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

%sql SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' AND Payload_Mass__kg_ > 4000 AND Payload_Mass__kg_ < 6000;

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

Total Number of Successful and Failure Mission Outcomes

%sql SELECT Landing_Outcome, COUNT(*) AS Total_Outcomes FROM SPACEXTBL WHERE Landing_Outcome IN ('Success', 'Failure') GROUP BY Landing_Outcome;

Landing_Outcome	Total_Outcomes
Failure	3
Success	38

Boosters Carried Maximum Payload

```
%sql SELECT Booster_Version FROM SPACEXTBL
WHERE Payload_Mass__kg_ = (SELECT
MAX(Payload_Mass__kg_) FROM SPACEXTBL);
```

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

```
%sql SELECT substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE Landing_Outcome LIKE 'Failure (drone ship)' AND substr(Date, 1, 4) = '2015';
```

Month	Landing_Outcome	Booster_Version	Launch_Site
10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

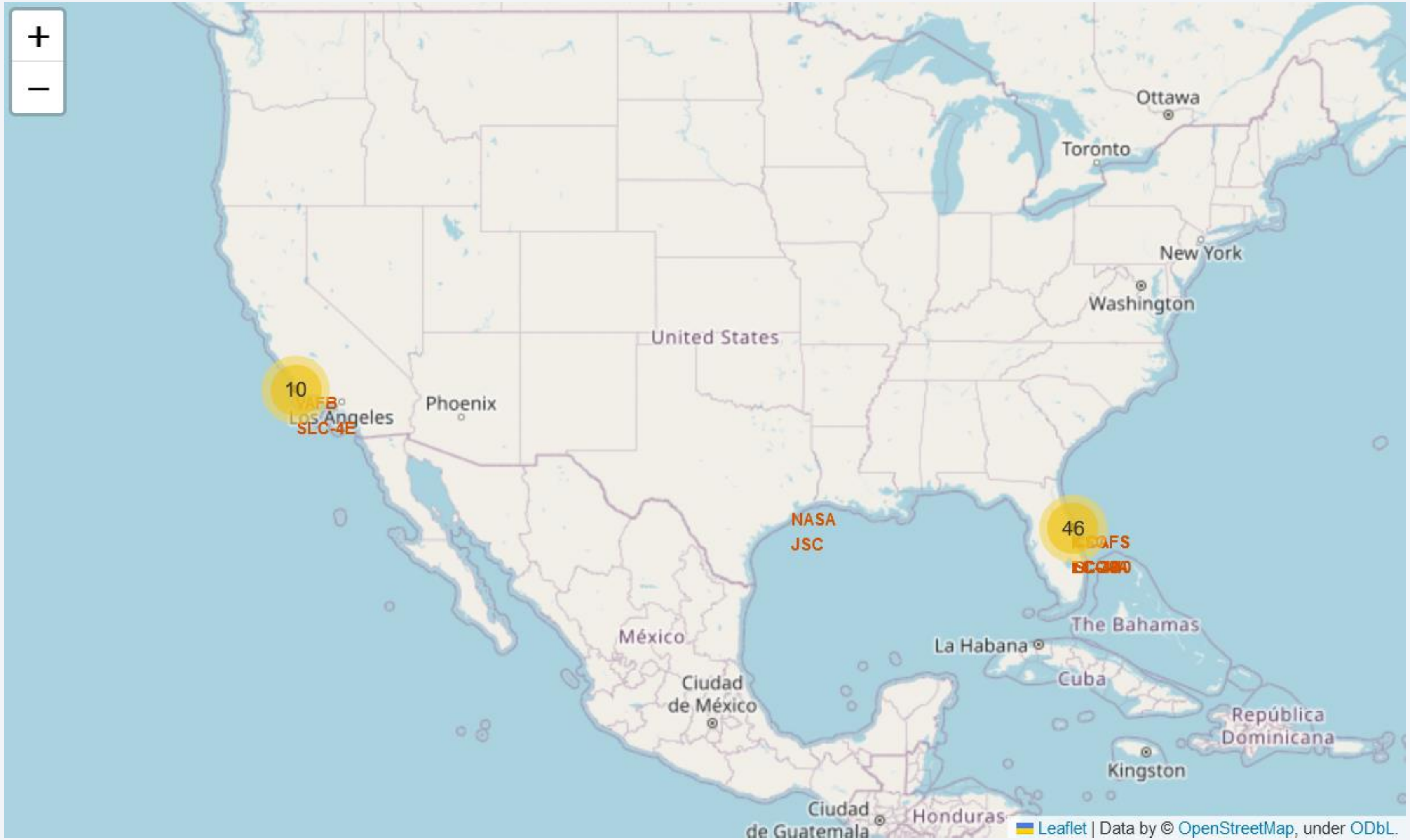
%sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Outcome_Count DESC;

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

Section 3

Launch Sites Proximities Analysis

All launch sites marked on a map



Successful/failed launches marked on the map

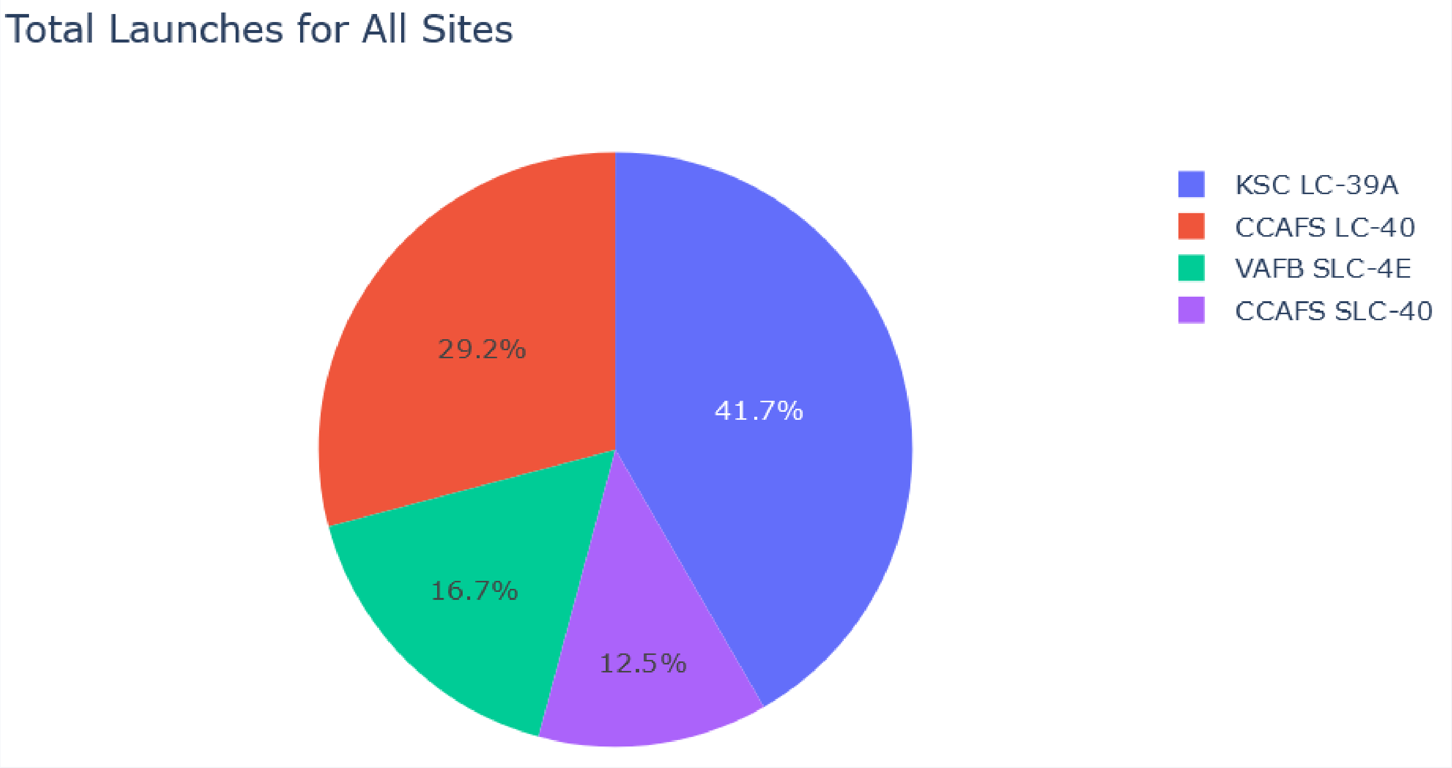
Distances between a launch site to its proximities

Section 4

Build a Dashboard with Plotly Dash

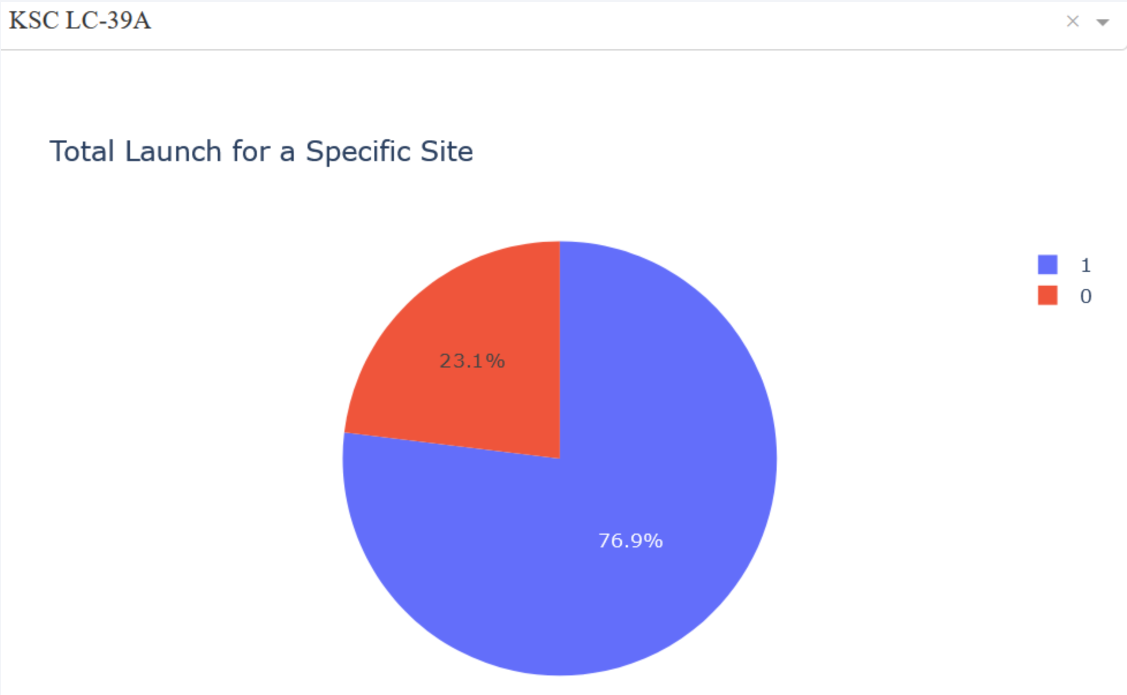
Total success launches by all sites

- KSC LC-39A has the highest percentage of all successful launches.
- CCAFS SLC-40 has the lowest percentage of all successful launches.

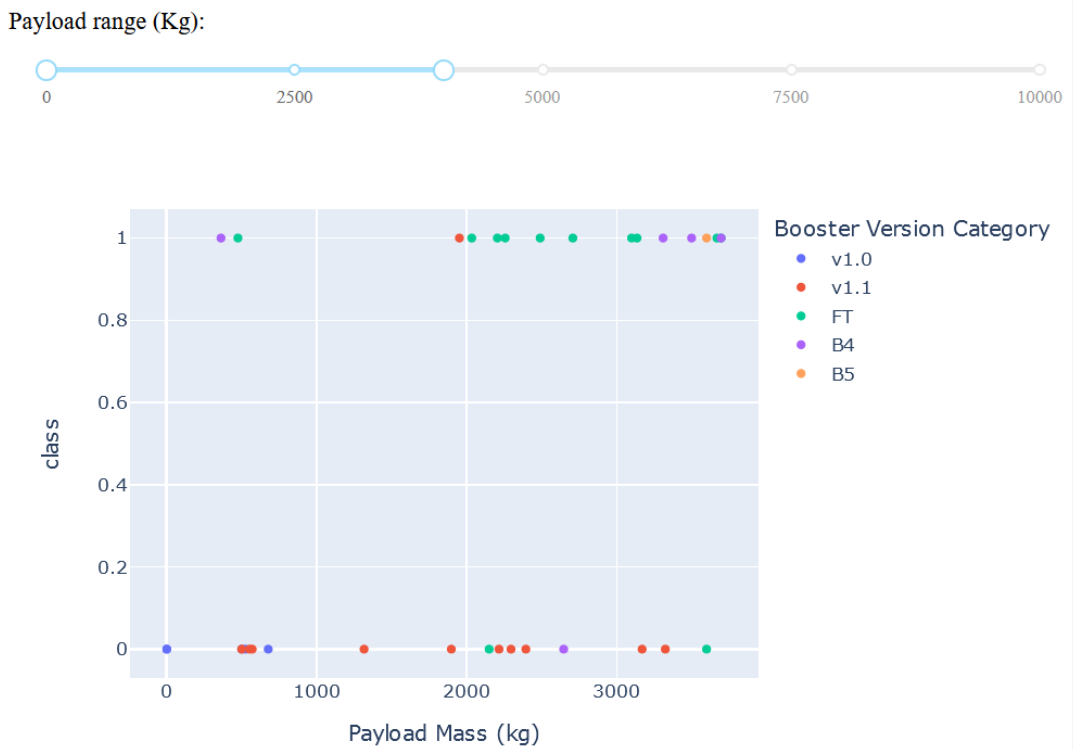


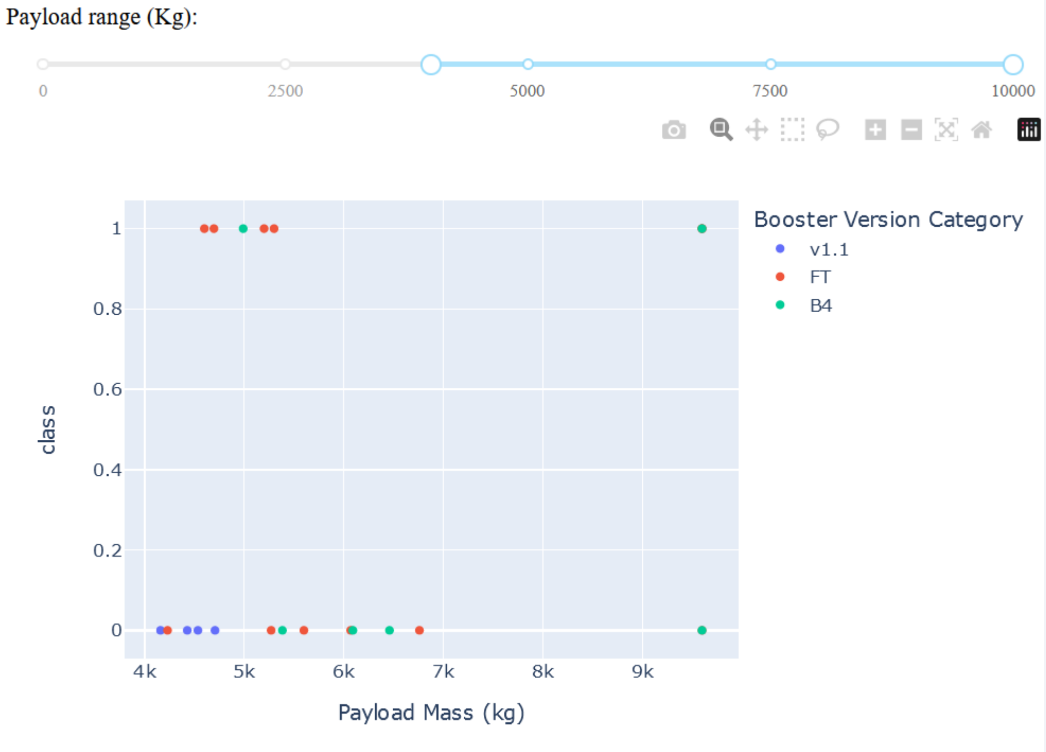
Success rate by site (highest launch success ratio)

- KSC LC-39A achieved a 76,9% success rate while getting a 23,1% failure rate



Payload vs. launch outcome



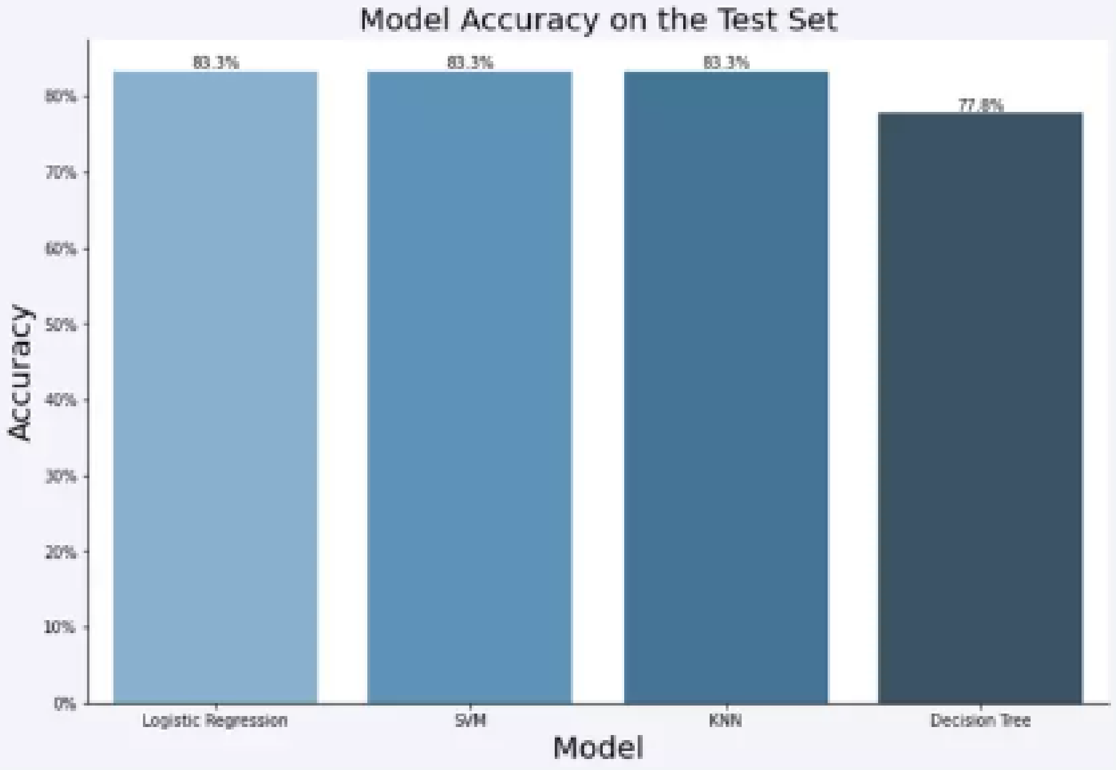


Section 5

Predictive Analysis (Classification)

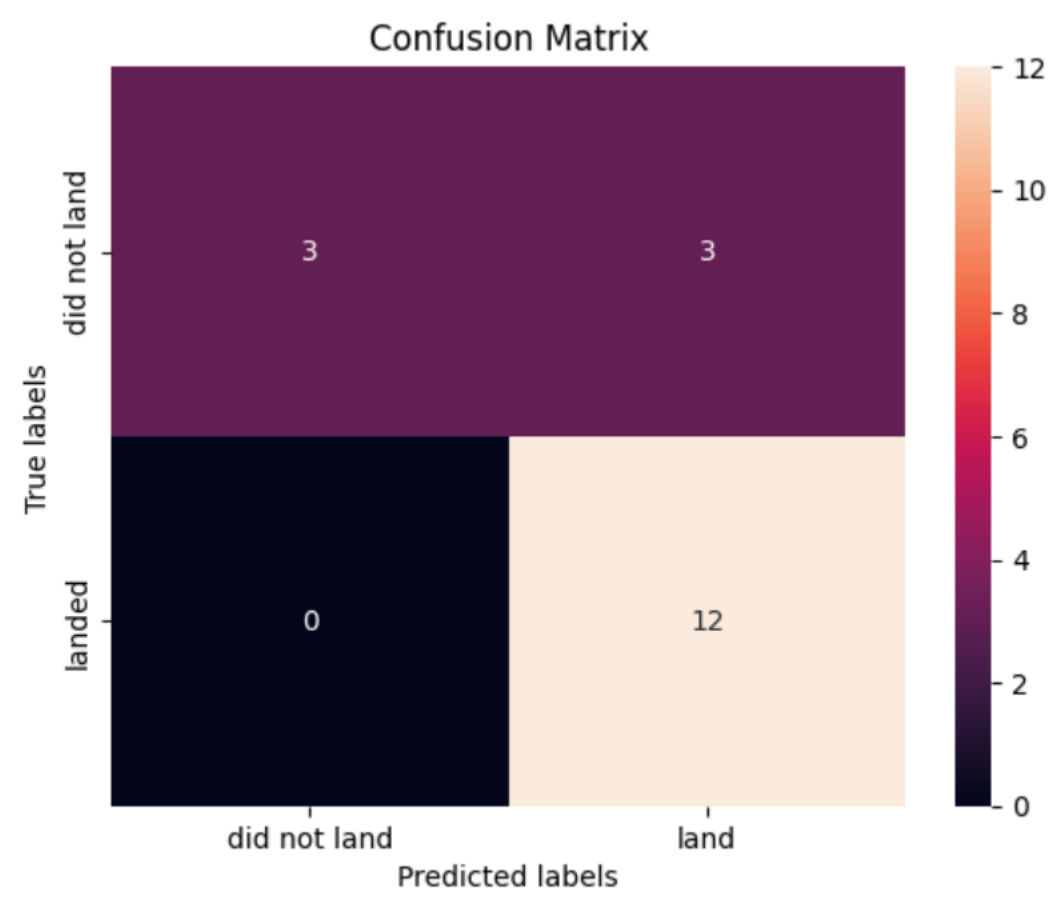
Classification Accuracy

- SVM, KNN and Logistic Regression are the best models in terms of prediction accuracy for this dataset



Confusion Matrix

- SVM, KNN and Logistic Regression are the best models in terms of prediction accuracy for this dataset



Conclusions

- SVM, KNN and Logistic Regression are the best models in terms of prediction accuracy for this dataset
- Low weighted payloads perform better than the heavier payloads
- KSC LC 39A had the most successful launches from all the sites
- Orbits GEO, HEO, SSO ES L1 have the best success rates

Appendix (examples)

TASK 10

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object t
from the dictionary `parameters` .

```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

```
knn_cv = GridSearchCV(KNN,parameters,cv=10)
knn_cv.fit(X_train, Y_train)
```

```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

```
# 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
spacex_df = pd.read_csv("spacex_launch_dash.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__)

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                # dcc.Dropdown(id='site-dropdown',...)
                                dcc.Dropdown(id='site-dropdown',
                                             options=[
                                                 {'label': 'ALL SITES', 'value': 'ALL'},
                                                 {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
                                                 {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
                                                 {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
                                                 {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'}
                                             ],
```

https://github.com/PaIBM/IBM_Data_Science_Professional_Certification/blob/abad505a99dfaed6a0b577de02b5c854441e70e2/spacex_dash_app.py

Thank you!