IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Brandon Young
10.10.2023

Outline

Executive Summary

SUMMARY OF METHODOLOGIES

- Data Collection through SpaceX API
- Data Collection with Web Scraping
- Data Wrangling
- EDA with Visualization
- EDA with SQL
- Building an Interactive Map with Folium
- Machine Learning Prediction and Analysis

SUMMARY OF ALL RESULTS

- Preliminary analysis based on EDA
- Interactive Analytics with screenshots
- Machine Learning Predictive analysis results

Introduction

PROJECT BACKGROUND AND CONTEXT

SpaceX has transformed the space industry with its Falcon 9 rocket by prioritizing reusability to reduce launch costs. According to their website, a Falcon 9 launch costs around $62 million, whereas other providers, including NASA, can charge over $165 million for similar launches. One of the key factors in SpaceX's cost-saving approach is their ability to successfully land and reuse the first stage of the Falcon 9. If a new competitor, such as SpaceY, wants to challenge SpaceX in the market, it is essential to understand the success rate of first-stage landings. This project aims to analyze the available data and use machine learning to predict the success of the Falcon 9's first-stage landing. Such insights can be invaluable for competitors looking to effectively bid against SpaceX.

MATTERS TO EXAMINE

- What factors determine if Falcon 9 will land successfully?
- How does the success rate of the Falcon 9 landings compare to other rockets?
- What features in the data are most influential in predicting successful lands?

Section 1

Methodology

Methodology

DATA COLLECTION METHODOLOGY

- Using SpaceX Rest API
- Web Scrapping from Wikipedia

PERFORM DATA WRANGLING

- Data was cleaned and transformed using one-hot encoding for machine learning

PERFORM EDA USING VISUALIZATION AND SQL

- Data was queried with SQL and various plots were used to show patterns in the data

PERFORM INTERACTIVE VISUAL ANALYTICS USING FOLIUM AND PLOTLY DASH

- Folium was used along with Dash to build interactive maps and a dashboard

PERFORM PREDICTIVE ANALYSIS USING CLASSIFICTION MODELS

- Different ML algorithms were used to find the most accurate results on the data.

Data Collection

For data collection, I employed the following methods:

- Request and parse the SpaceX launch data using the GET request via the SpaceX Rest API
- Web scrape SpaceX's Wikipedia page for launch data using Beautiful Soup. The objective was to get the necessary tables needed from it.

Data Collection – SpaceX API

- I used the get request to SpaceX to collect data and did some light data wrangling
- Save data out to CSV file

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

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

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

```python
# Call getBoosterVersion
getBoosterVersion(data)
```

```python
# Call getLaunchSite
getLaunchSite(data)
```

```python
# Call getPayloadData
getPayloadData(data)
```

```python
# Call getCoreData
getCoreData(data)
```

```python
# dictionary creation
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}
```

```python
#data frame creation
launch_df = pd.DataFrame(launch_dict)

#filtering the data
data_falcon9 = launch_df[launch_df['BoosterVersion'] != 'Falcon 1']
data_falcon9.head()
# Calculate the mean value of PayloadMass column
payload_mass_mean = data_falcon9['PayloadMass'].mean()

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

#check  nan values
data_falcon9.isnull().sum()

#writing the dataframe to a csv file
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Link to the completed notebook

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

Data Collection – Scraping

- I used Beautiful Soup to scrape the data from Wikipedia
- Extracted all column/variable names from the HTML table header
- Created a data frame by parsing the launch HTML tables
- Save data to a CSV file

```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
html = response.content
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html, "html.parser")
# Use the find_all function in the BeautifulSoup object, with element type `table`
tables = soup.find_all("table")
# Assign the result to a list called `html_tables`
html_tables = tables
column_names = []

# Apply find_all() function with `th` element on first_launch_table
first_row = first_launch_table.find_all("th")
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for row in first_row:
    column_name = extract_column_from_header(row)
    if column_name:
        column_names.append(column_name)
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'] = []
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']=[]

# Payload
# TODO: Append the payload into launch_dict with key `Payload`
payload = row[3].a.string
launch_dict['Payload'].append(payload)
print(f"Payload: {payload}")

# Payload Mass
# TODO: Append the payload_mass into launch_dict with key `Payload mass`
payload_mass = get_mass(row[4])
launch_dict['Payload mass'].append(payload_mass)
print(f"Payload Mass: {payload_mass}")

# Orbit
# Append the orbit into launch_dict with key `Orbit`
orbit = row[5].a.string
launch_dict['Orbit'].append(orbit)
print(f'Orbit: {orbit}')

# Customer
if row[6].a:
    customer = row[6].a.string
    launch_dict['Customer'].append(customer)
    print(f"Customer: {customer}")
else:
    launch_dict['Customer'].append(None)  # or some default value
    print("No customer found for this row.")

# Launch outcome
# TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
launch_outcome = list(row[7].strings)[0]
launch_dict['Launch outcome'].append(launch_outcome)
print(f"Launch outcome: {launch_outcome}")

(variable) booster_landing: Any into launch_dict with key `Booster landing`
booster_landing = landing_status(row[8]).strip('\n')
launch_dict['Booster landing'].append(booster_landing)
print(f"Booster landing: {booster_landing}")
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        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)
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            print(f"Flight Number: {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)
            print(f"Date: {date}")

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            print(f"Time: {time}")

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict['Version Booster'].append(bv)
            print(f"Version Booster: {bv} ")

            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)
            print(f"Launch Site: {launch_site}")

df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df.replace("Success\n","Success",inplace=True)
df.replace("No attempt\n","No attempt",inplace=True)
df.to csv('spacex web scraped.csv', index=False)
```

Link to the completed notebook

https://github.com/byvfx/ibm_ds_capstone/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

With the data from the SpaceX API I needed to do the following:

- Calculate the number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate the number and occurrence of mission outcome of the orbits
- Create a landing outcome label from the 'Outcome' column, classifying it as successful or failure.
- Save data to a CSV file

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()

 for i,outcome in enumerate(landing_outcomes.keys()):
     print(i,outcome)
```

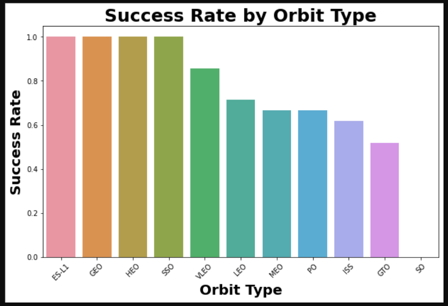
```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

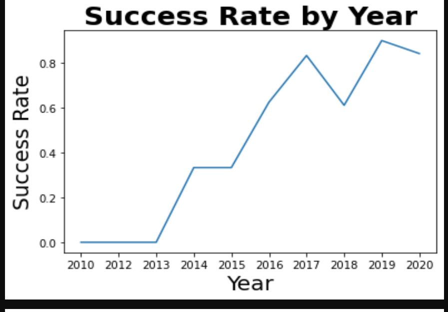
```
# landing_class = 0 if bad_outcome
landing_class = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
# landing_class = 1 otherwise
landing_class = df['Outcome'].apply(lambda x: 1 if x not in bad_outcomes else 0)

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

df["Class"].mean()
df.to_csv('dataset_part_2.csv', index=False)
```

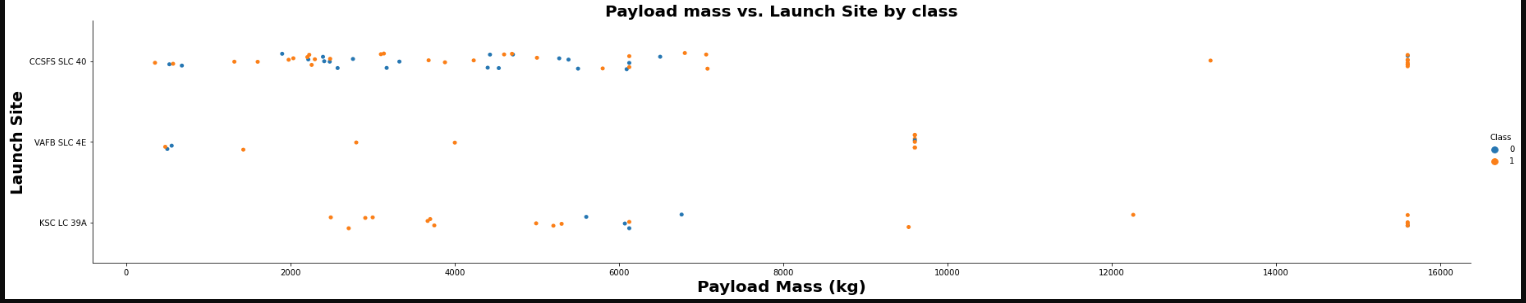
Link to the completed notebook

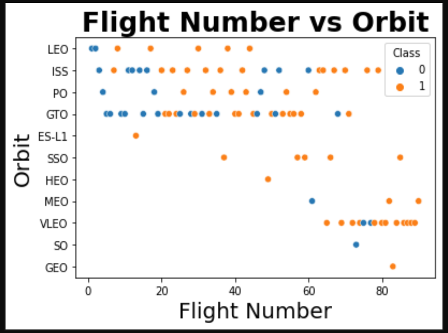
https://github.com/byvfx/ibm_ds_capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

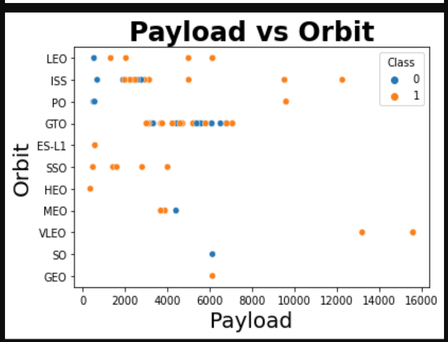
EDA with Data Visualization

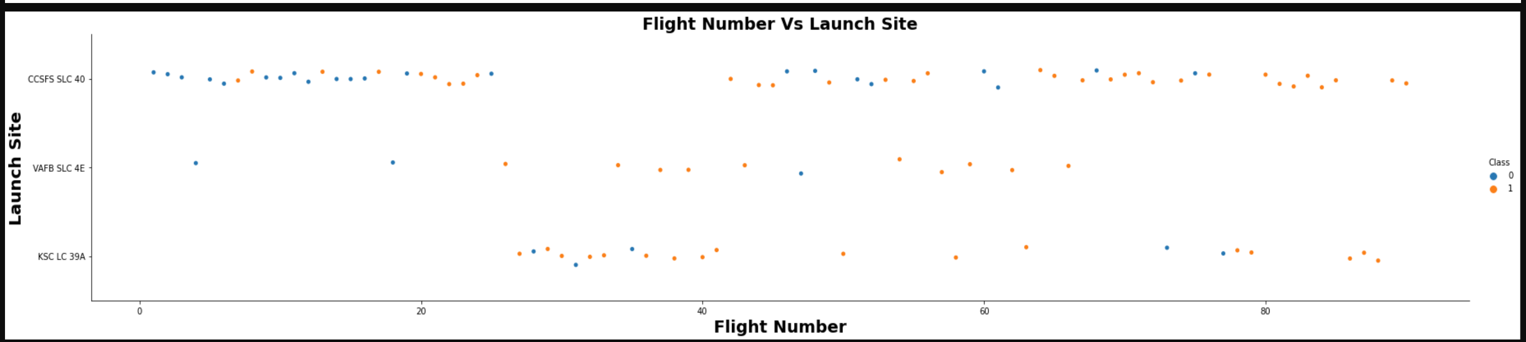


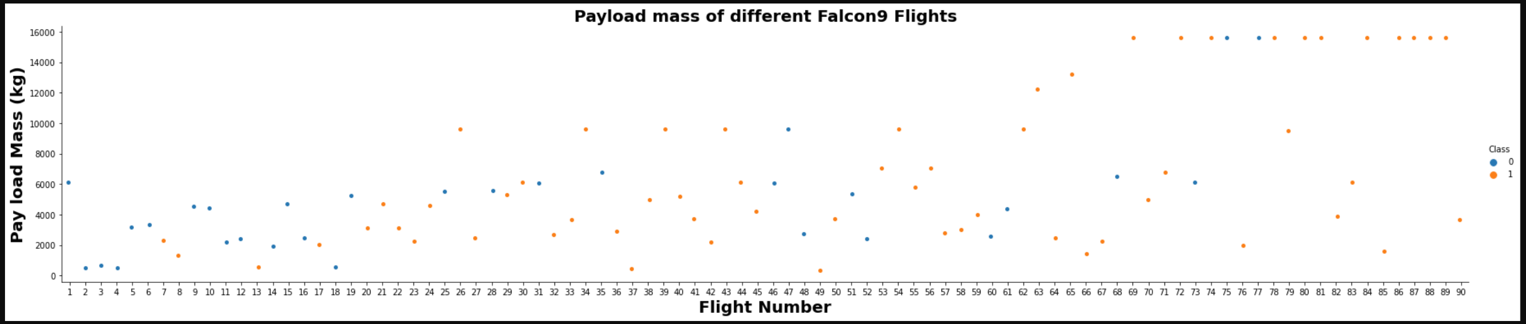












I explored the data relationship between flight number, launch site, payload and success rate of each orbit type, and the launch success yearly trend.

Link to the completed notebook https://github.com/byvfx/ibm_ds_capstone/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

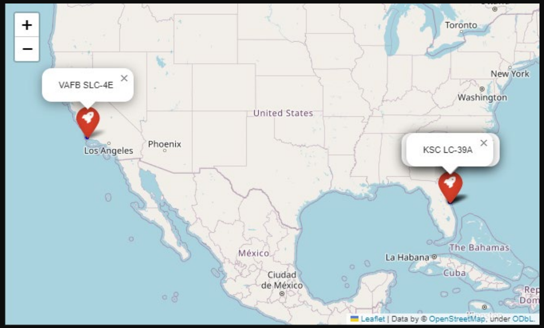
SQL QUERIES PERFORMED

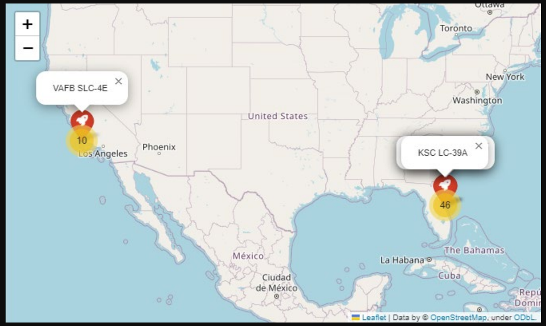
- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in the ground pad was achieved.
- List the total number of successful and failed mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months 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

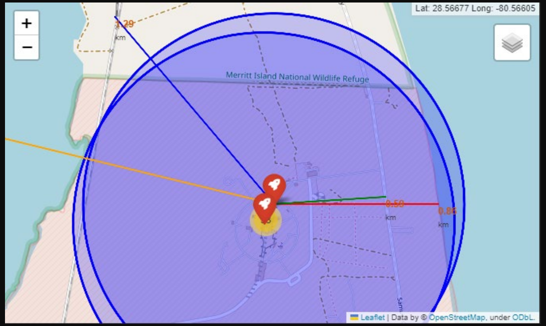
Link to the completed notebook https://github.com/byvfx/ibm_ds_capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

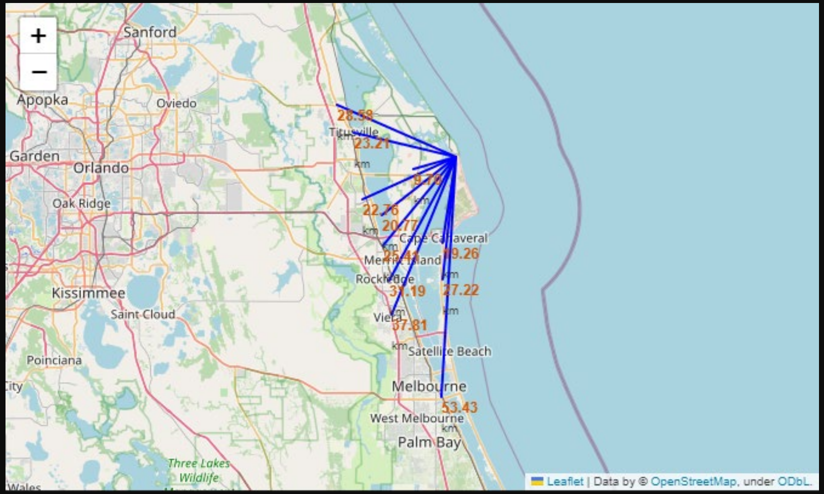
Build an Interactive Map with Folium

- We marked where rockets launch on a map.
- We used different symbols on the map to show whether a rocket launch was successful.
- We gave each launch a label: 0 means it failed, and 1 means it was successful.
- We grouped launch sites by color to see which ones succeed more often.
- We looked at launch sites' proximity to things like train tracks, roads, and the ocean.
- We checked if the launch sites are far from cities.









Link to the completed notebook

https://github.com/byvfx/ibm_ds_capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a dashboard with plotly dash

PIE CHART

- It shows the success rate of all launch sites.
- It displays the proportion of success and failures of given launch site.

SCATTER PLOT

- It shows the correlation between mission outcome and payload mass for different booster versions for all sites or selected site.
- The payload mass can be filter by weight range using the slider control.

Link to the completed notebook https://github.com/byvfx/ibm_ds_capstone/blob/main/dash_spacex_lab.py

Predictive Analysis (Classification)

- I loaded in the data using numpy and pandas, transformed the data, split out data into training and testing.
- I built different machine-learning models and tuned different hyperparameters using GridSearch CV.
- I used accuracy as the metric for our model improved the model using feature engineering and algorithm tuning.
- I found the best-performing classification model. Which was the decision tree

```
# define hyperparameters to tune
parameters_lr ={"C":[0.01,0.1,1],
               'penalty':['l2'],
               'solver':['lbfgs']}# l1 lasso l2 ridge

# define the model
lr = LogisticRegression(random_state = 12345)

# define the grid search object
grid_search_lr = GridSearchCV(
    estimator = lr,
    param_grid = parameters_lr,
    scoring = 'accuracy',
    cv = 10
)
# execute search
logreg_cv = grid_search_lr.fit(X_train,Y_train)

 accuracy = logreg_cv.score(X_test,Y_test)
 print("accuracy :",accuracy)
✓ 0.0s

accuracy : 0.8333333333333334
```

```
tree = DecisionTreeClassifier()
parameters_tree = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train, Y_train)
best_params = tree_cv.best_params_
   accuracy = knn_cv.score(X_test,Y_test)
   print("accuracy :",accuracy)
✓ 0.0s

accuracy : 0.7777777777777778
```

```
# define hyperparameters to tune
parameters_svm = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
                 'C': np.logspace(-3, 3, 5),
                 'gamma':np.logspace(-3, 3, 5)}

# define the model
svm = SVC(random_state = 12345)

# define the grid search object
grid_search_svm = GridSearchCV(
    estimator = svm,
    param_grid = parameters_svm,
    scoring = 'accuracy',
    cv = 10
)
# execute search
svm_cv = grid_search_svm.fit(X_train,Y_train)

 accuracy = svm_cv.score(X_test,Y_test)
 print("accuracy :",accuracy)
✓ 0.0s

accuracy : 0.8333333333333334
```

```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if  (variable) bestalgorithm: Any
                                   cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
✓ 0.0s

Best model is DecisionTree with a score of 0.8732142857142857
```

Link to the completed notebook
https://github.com/byvfx/ibm_ds_capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

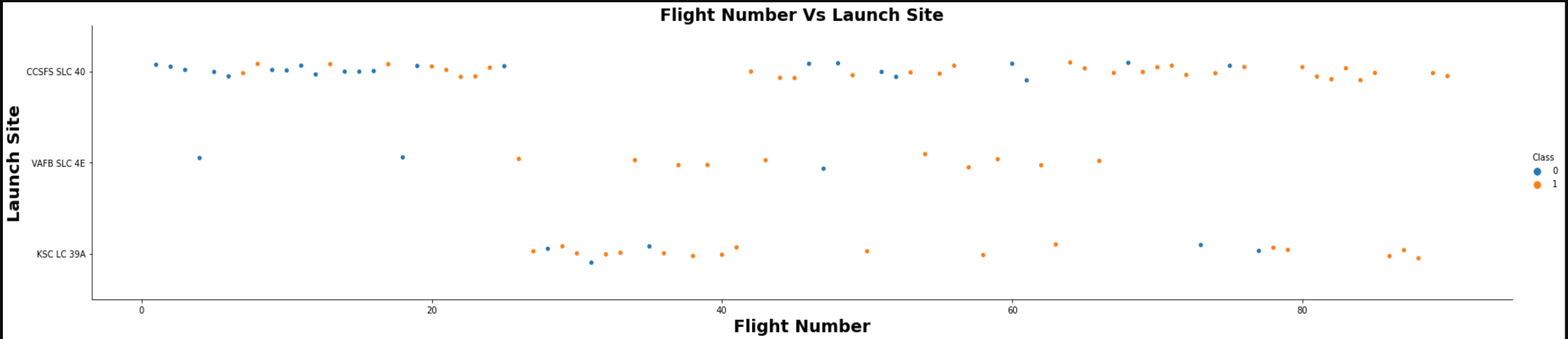
Results

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

Section 2

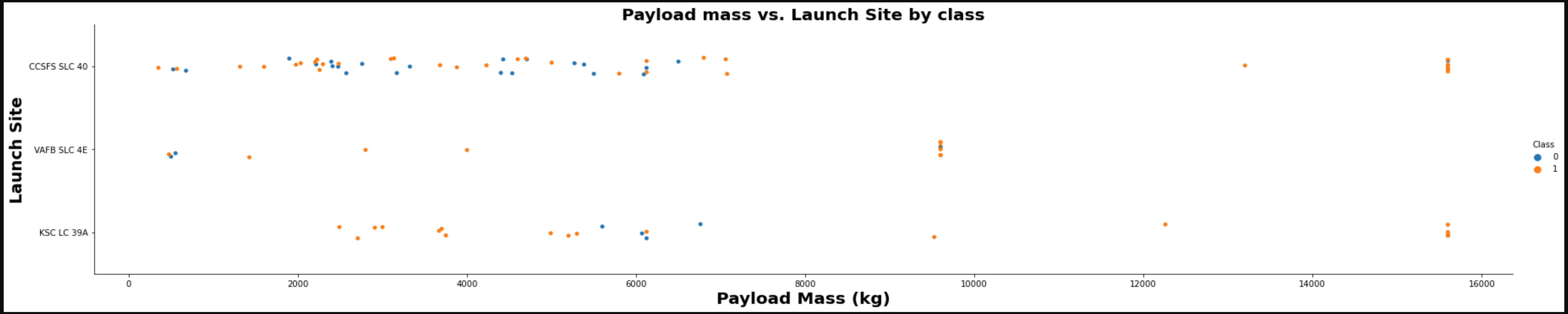
Insights drawn from EDA

Flight Number vs. Launch Site



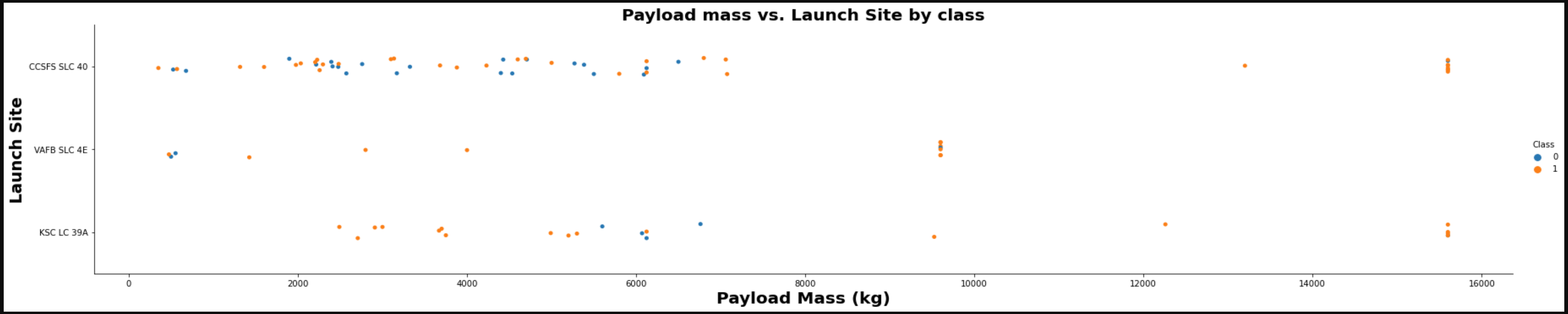
- More recent missions have higher success rates at all launch sites
- There's an apparent trend where certain launch sites are more commonly used during specific flight number ranges.
- There seems to be a transition from more Class 0 flights to more Class 1 flights as the flight numbers increase.

Payload vs. Launch Site



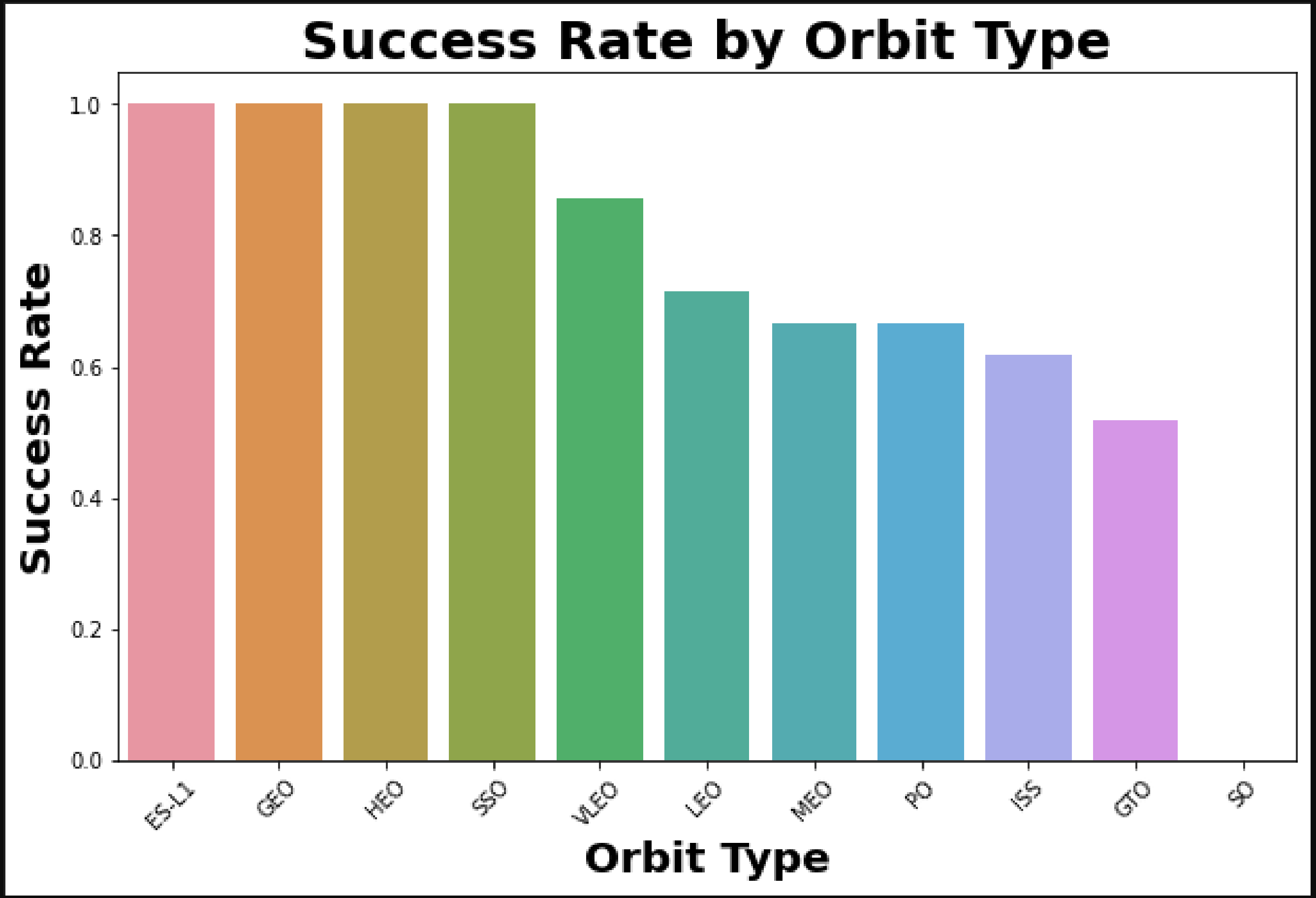
- There isn't a strong linear correlation between payload mass and launch success or failure for individual launch sites based on the visual inspection.
- All launch sites have both successful and unsuccessful launches (Class 0 and Class 1) across various payload masses.

Payload vs. Launch Site



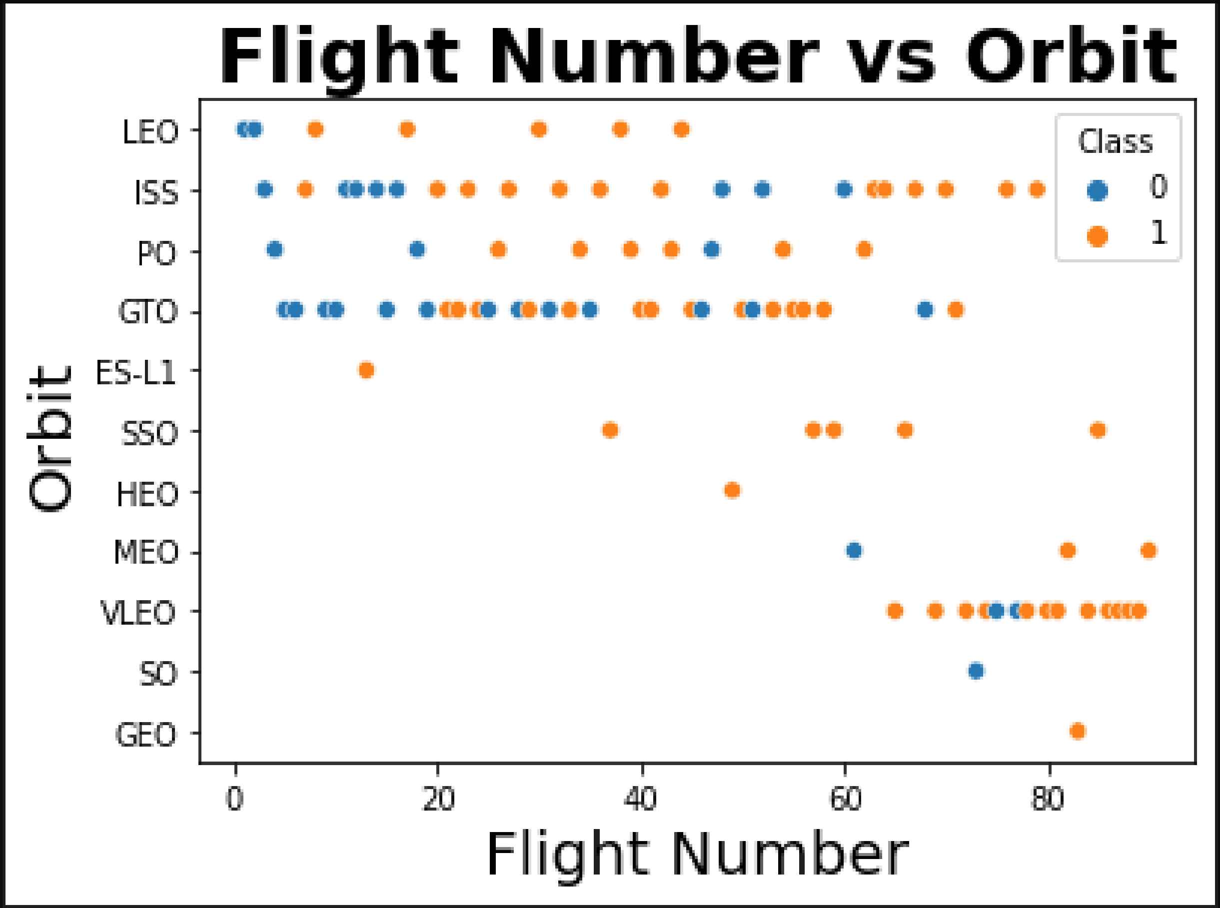
- There isn't a strong linear correlation between payload mass and launch success or failure for individual launch sites based on the visual inspection.
- All launch sites have both successful and unsuccessful launches (Class 0 and Class 1) across various payload masses.

Success Rate vs Orbit Type



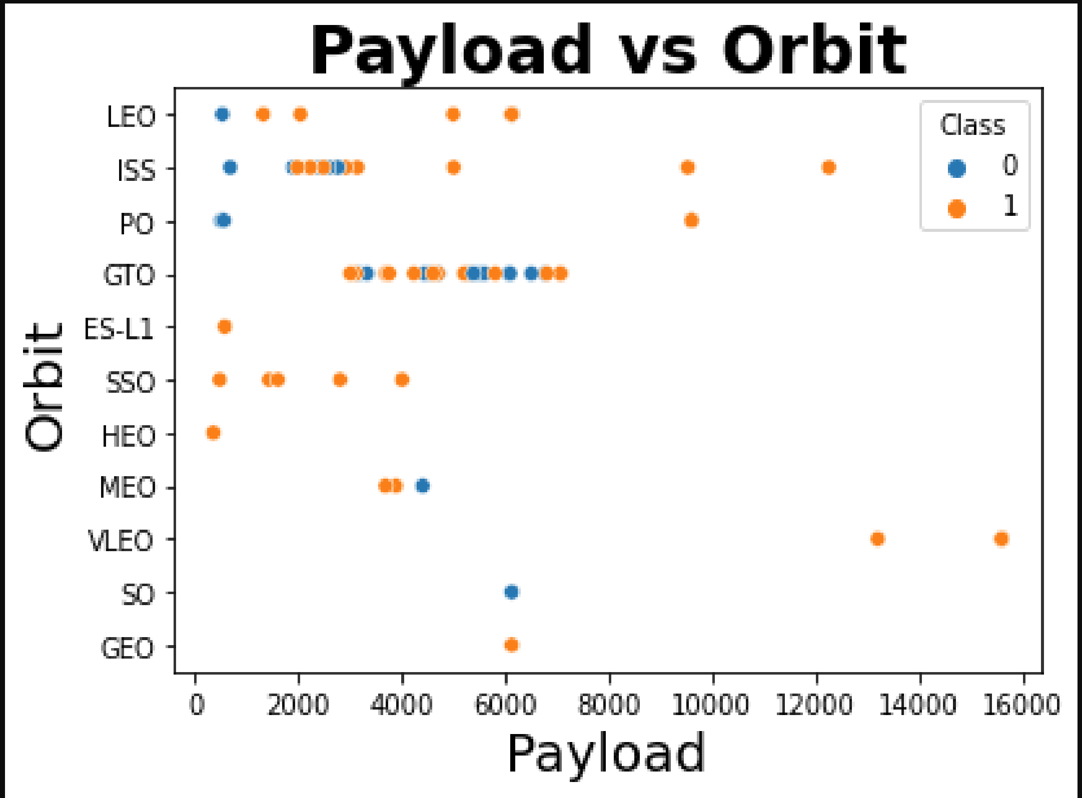
- The chart shows different orbit types have varying levels of launch success rates, with ESI-L1 and GEO achieving almost perfect success, while SSO orbits experience the least success.

Flight Number vs. Orbit



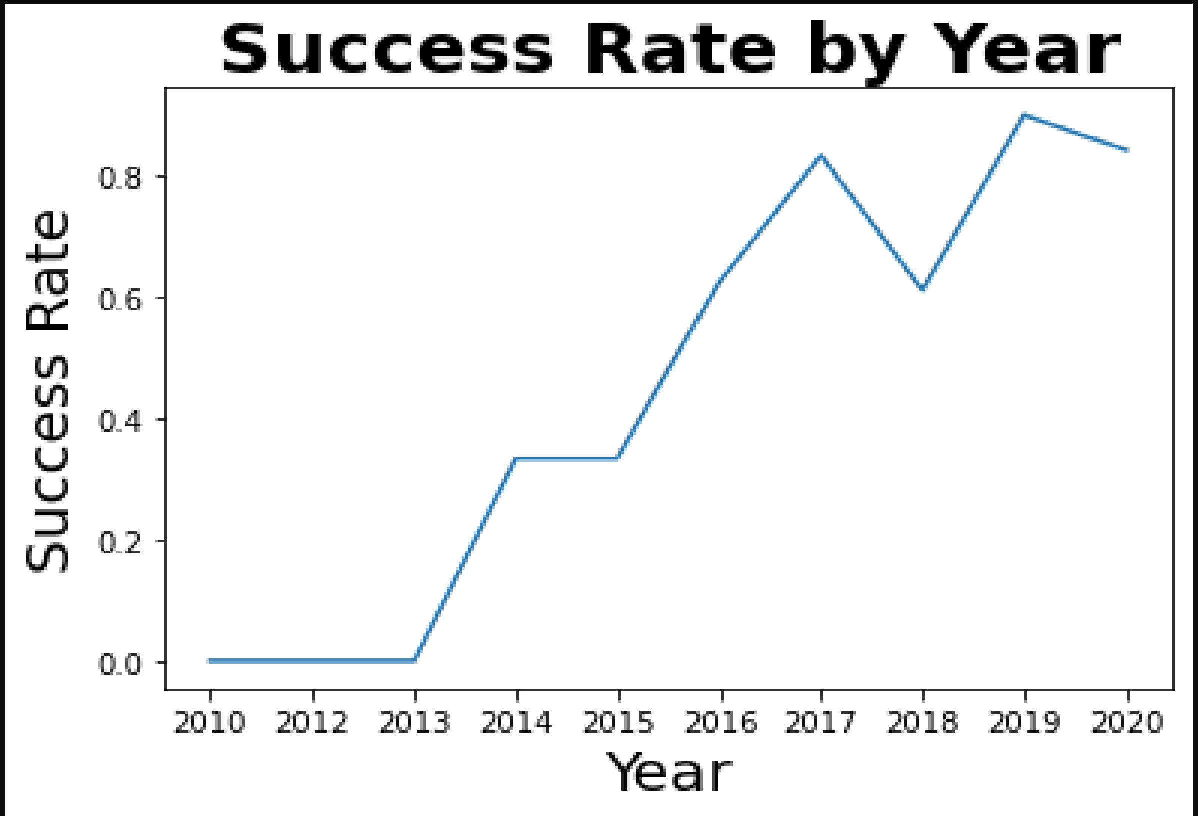
- Orbit types like LEO and ISS have a higher frequency of launches across a broad range of flight numbers, while others like HEO, SO, and GEO have fewer launches.

Payload vs. Orbit Type



- Some orbit types, like LEO and ISS, mainly deal with lighter payloads, while others, like MEO, handle heavier ones. Both success and failure outcomes are present across the different payload ranges for various orbits.

Launch Success Yearly Trend



- Over the years we can see that the success rate has steadily climbed

All Launch Site Names

SQL Query

```
"SELECT DISTINCT Launch_Site FROM SPACEXTBL;
```

Query Result

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

Explanation

- We are looking for names with unique launch site names in the Spacex table
- Using select distinct statement to retrieve the unique launch site names

Launch Site Names Begin with 'CCA'

SQL Query

```
SELECT * FROM SPACEXTBL
WHERE Launch_Site
LIKE 'CCA%’ limit 5;
```

Query Result

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

Explanation

- We are looking for records that begin with CCA with a limit of 5
- Using the like operator to search a specific pattern and using the limit clause to retrieve on 5 records

Total Payload Mass

SQL Query

```
SELECT Customer, SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

Query Result

total_payload_mass
45596

Explanation

- We are looking for the total payload mass carried by the booster launched by NASA (CRS)
- Using the sum function to return the total and then using the condition customer = NASA CRS to filter the records

Average Payload Mass by F9 v1.1

SQL Query

```
SELECT Booster_Version, AVG(PAYLOAD_MASS__KG_)
AS avg_payload_mass
FROM SPACEXTBL
WHERE Booster_Version = 'F9 v1.1';
```

Query Result

avg_payload_mass
2928

Explanation

- We are looking for the average payload mass of the F9 v1.1 rocket
- Using the avg function to return the mean and then using the condition booster_version = F9 v1.1 to filter the records

First Successful Ground Landing Date

SQL Query

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

Query Result

min_date
2015-12-22

Explanation

- We are looking for the first occurrence of a successful landing on the ground pan
- Using the min function to return the first date and then using the condition Landing_Outcome = Success (ground pad) to filter the records

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query

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

Query Result

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

Explanation

- We are looking for the names of boosters that have success in drone ship and have a payload mass more significant than 4000 but less than 6000
- Using the condition Landing_Outcome = ‘Success (drone ship)’ to filter the records by landing outcome
- Using the condition payload_mass_kg_> 4000 and payload_mass_kg_ < 6000 to filter the records by payload mass

Total Number of Successful and Failure Mission Outcomes

SQL Query

```
SELECT
    CASE
        WHEN Mission_Outcome LIKE 'Success%' THEN 'Success'
        ELSE 'Failure'
    END AS Outcome,
    COUNT(*) AS Total
FROM SPACEXTBL
WHERE Mission_Outcome IN (
    'Success',
    'Success (drone ship)',
    'Success (payload status unclear)',
    'Failure (in flight)'
)
GROUP BY Outcome;
;
```

Query Result

mission_outcomes	qty
Failure	1
Success	99

Explanation

- We are looking for the total number of successful and failure mission outcomes
- Using the count(*) statement to retrieve the number of records
- Using group by statement to group records by missions outcomes

Boosters Carried Maximum Payload

SQL Query

```
SELECT Booster_Version
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL
);
```

Query Result

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

Explanation

- We are looking for the names of booster version which have carried the maximum payload mass
- Using the subquery to get maximum payload mass
- Using the max() function to return the largest value of payload mass

2015 Launch Records

SQL Query

```
SELECT
    CASE
        WHEN substr(Date, 6, 2) = '01' THEN 'January'
        WHEN substr(Date, 6, 2) = '02' THEN 'February'
        WHEN substr(Date, 6, 2) = '03' THEN 'March'
        WHEN substr(Date, 6, 2) = '04' THEN 'April'
        WHEN substr(Date, 6, 2) = '05' THEN 'May'
        WHEN substr(Date, 6, 2) = '06' THEN 'June'
        WHEN substr(Date, 6, 2) = '07' THEN 'July'
        WHEN substr(Date, 6, 2) = '08' THEN 'August'
        WHEN substr(Date, 6, 2) = '09' THEN 'September'
        WHEN substr(Date, 6, 2) = '10' THEN 'October'
        WHEN substr(Date, 6, 2) = '11' THEN 'November'
        WHEN substr(Date, 6, 2) = '12' THEN 'December'
    END AS MonthName,
    Landing_Outcome AS FailureLandingOutcome,
    Booster_Version,
    Launch_Site
FROM SPACEXTBL
WHERE substr(Date, 1, 4) = '2015'
    AND Landing_Outcome LIKE '%Failure (drone ship)%';
```

Query Result

```
October|Failure (drone ship)|F9 v1.1 B1012|CCAFS LC-40
April|Failure (drone ship)|F9 v1.1 B1015|CCAFS LC-40
```

Explanation

- We are looking for the failed landing outcomes in droneship, their booster versions, and launch site names for the year 2015
- Using the substr function to convert the months
- Then I filtered the outcome with the LIKE clause for Failure (drone ship)

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

SQL Query

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

Query Result

```
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
```

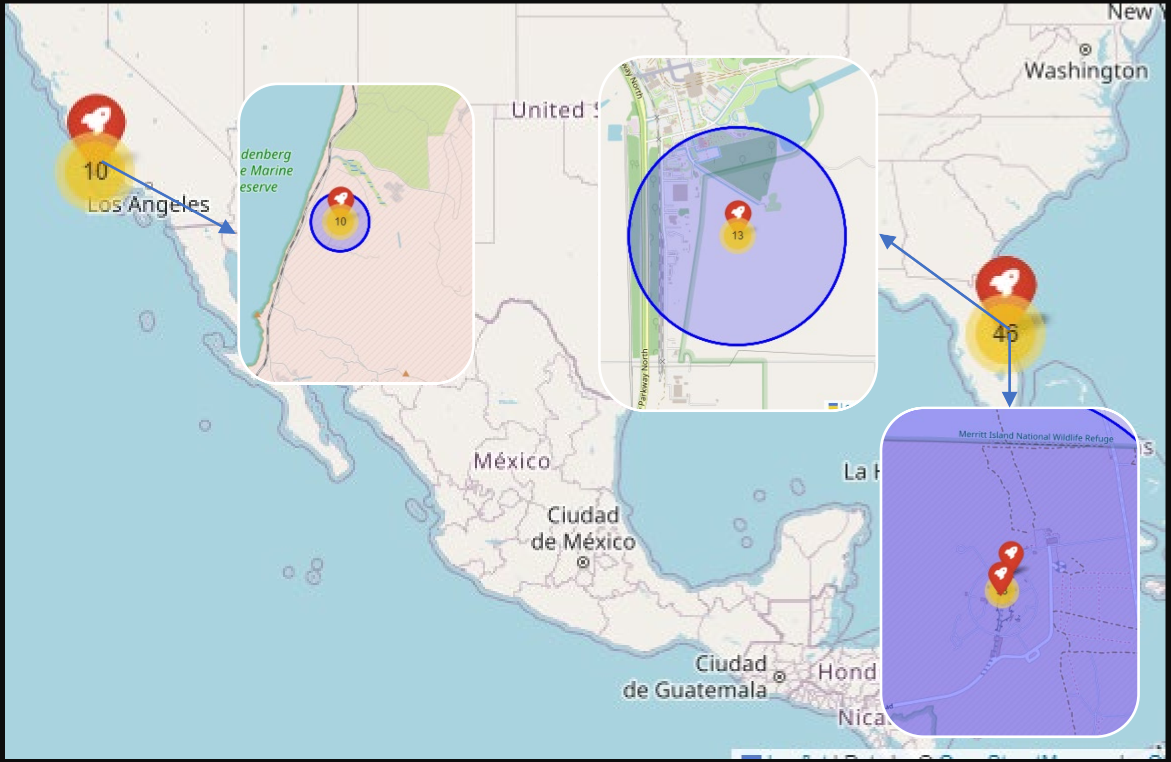
Explanation

- We are ranking the count of landing outcomes between the date 2010-06-04 and 2017-03-20
- Using the group by statement to group records by landing outcomes
- Using order by 2 desc keyword to sort by 2nd column in descending order

Section 3

Launch Sites Proximities Analysis

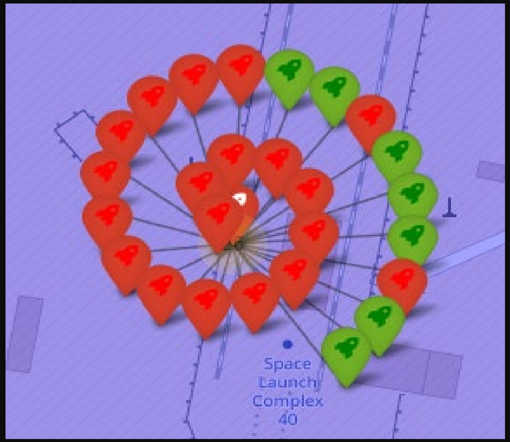
SpaceX Launch Sites Visualization with Folium



- All launch sites are far away from cities and near the coast to minimize risk
- Launch sites are closer to the equator the Earth's surface near the equator is moving at a higher rotational speed than areas near the poles. This increased speed provides rockets an initial velocity boost, making reaching orbit easier and more energy-efficient. This effect is due to the rotation of the Earth, which is fastest at the equator.

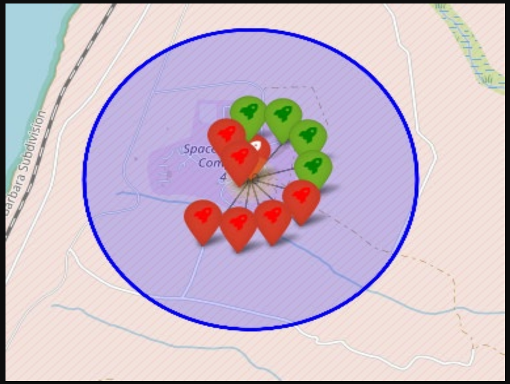
SpaceX Launch Sites Success and Failure using Folium

CCAFS LC-40



CCAFS SLC-40

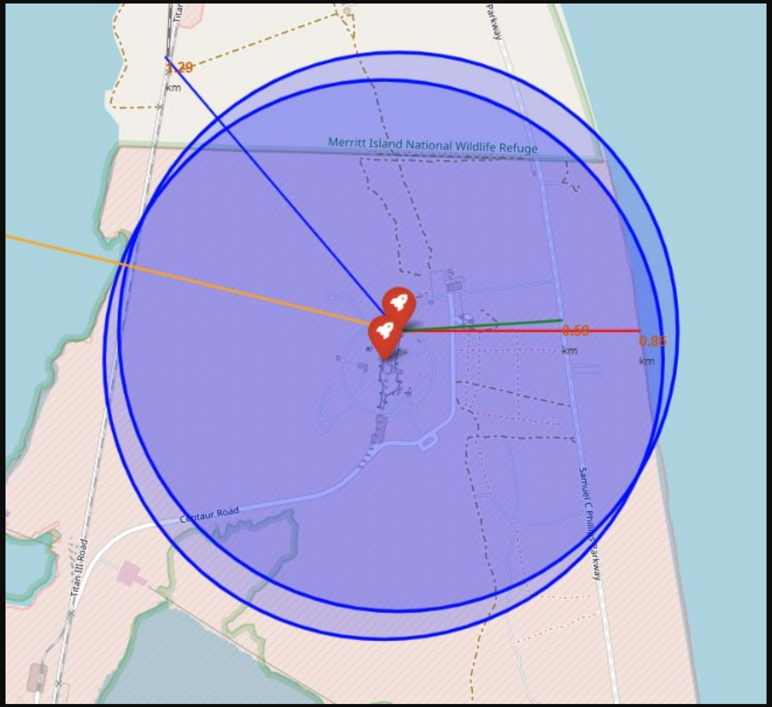
VAFB SLC-4E

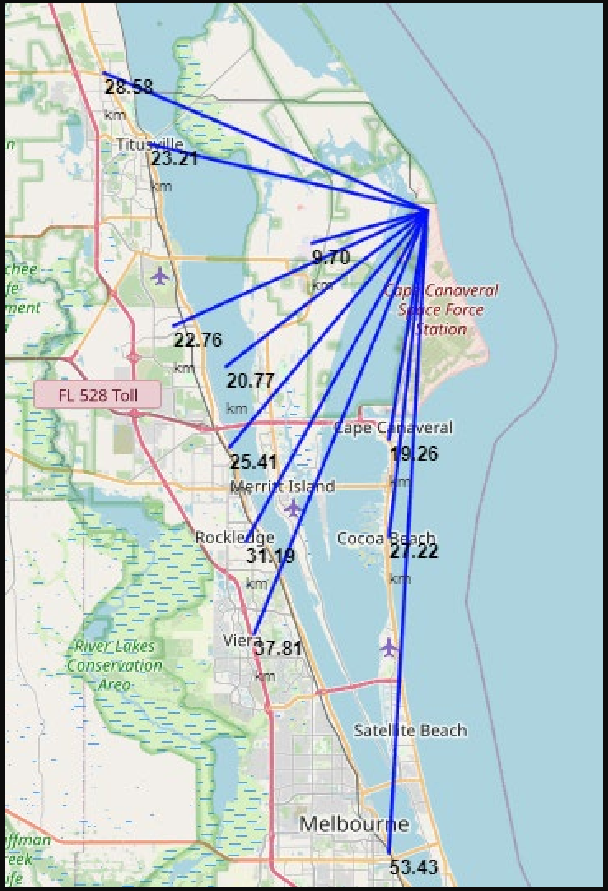


KSC LC-39A

- Green marker: successful landing
- Red marker: failed landing
- KSC LC-39A has the highest success rate

SpaceX Launch Sites Proximity to Cities and other Features



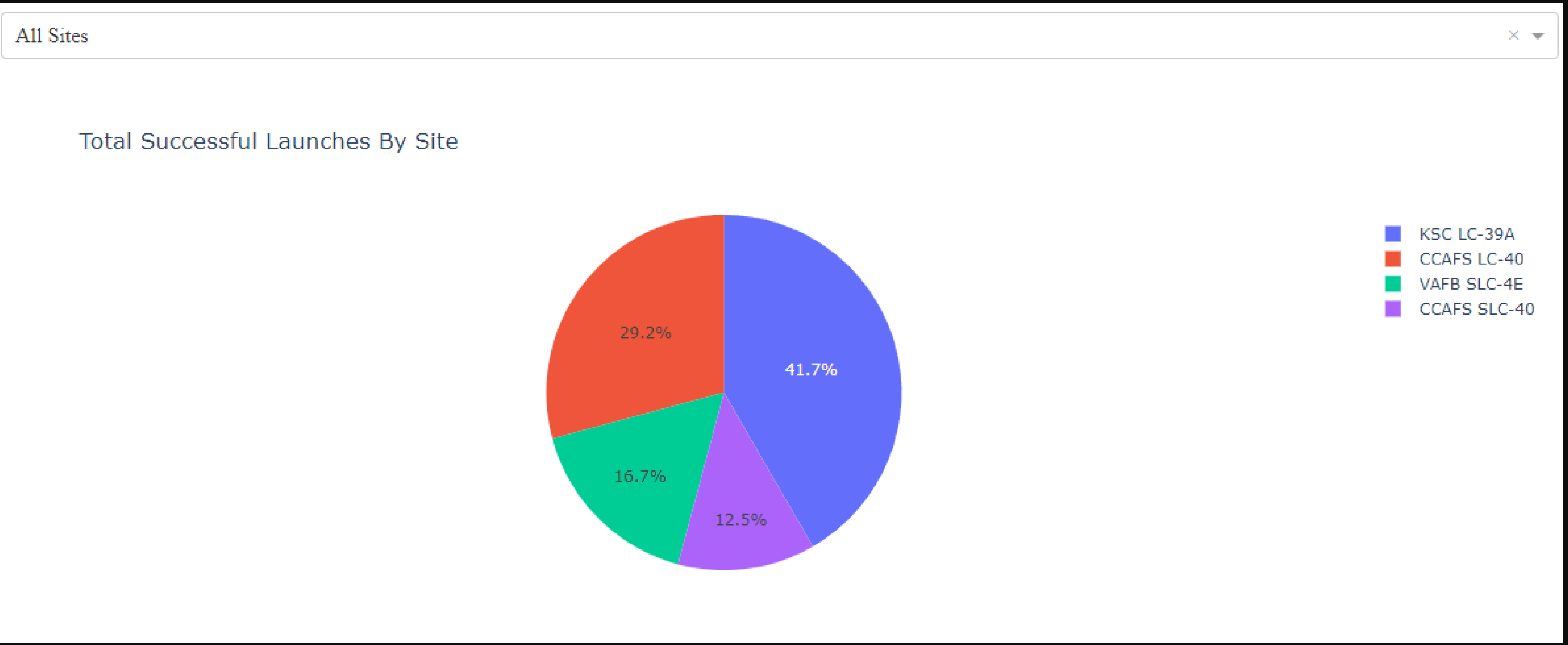


- CCAFS SLC-40 launch is relatively positioned far enough away from populated areas. The closest being the Town of Cape Canaveral, 19.26 km away. This minimizes any risks in the event of an accident.

Section 4

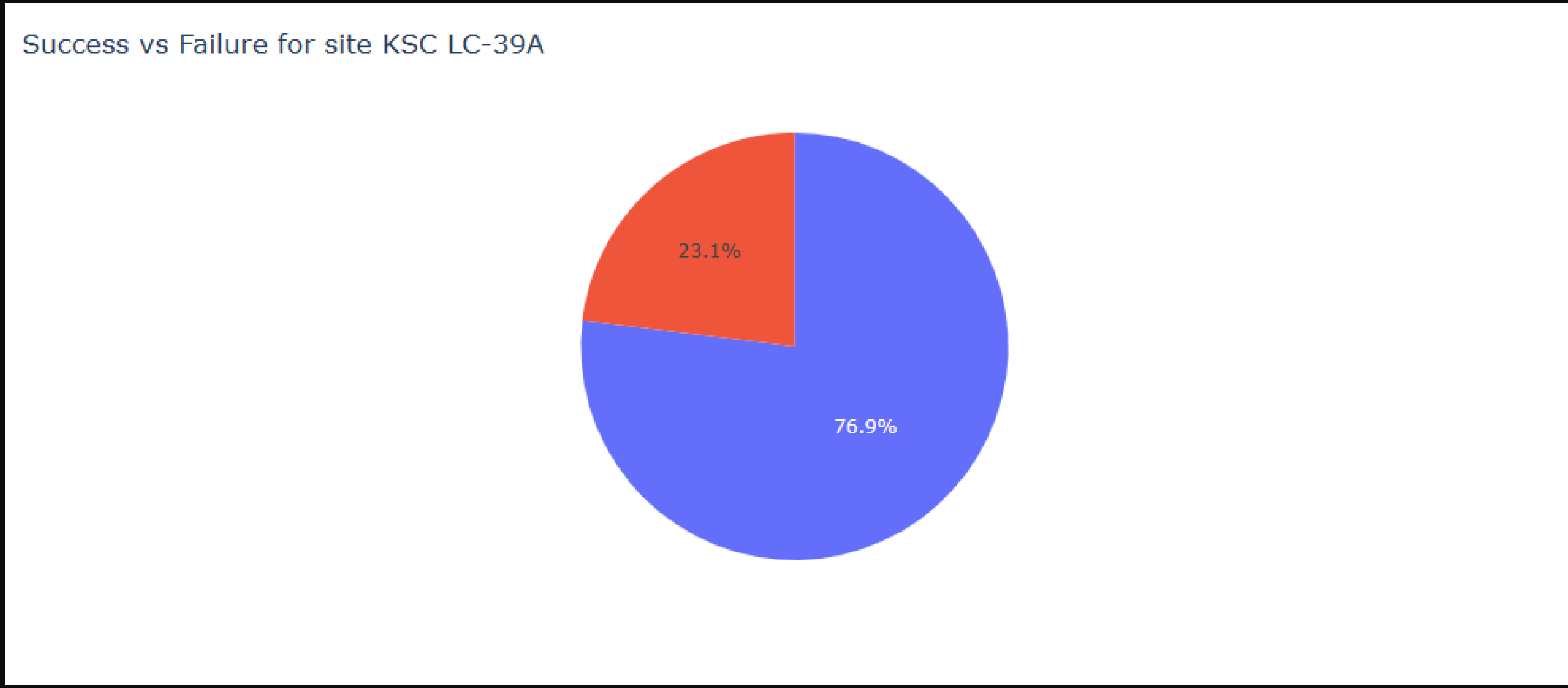
Build a Dashboard with Plotly Dash

Showing Percentages of Successful Launches in Dashboard



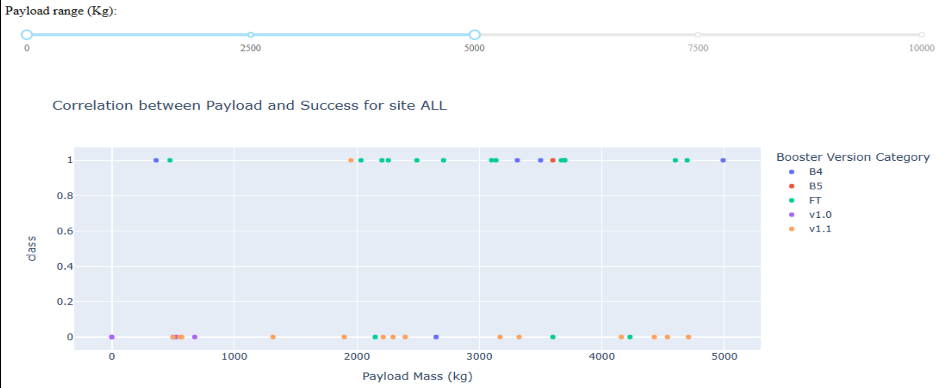
- KSC LC 39A has the most successful launches
- CCAFS SLC-40 has most failures the

Showing Percentages of Successful Launches in Dashboard



- KSC LC 39A has the highest ratio of successful launches

Dashboard Payload vs Launch Outcome for all site



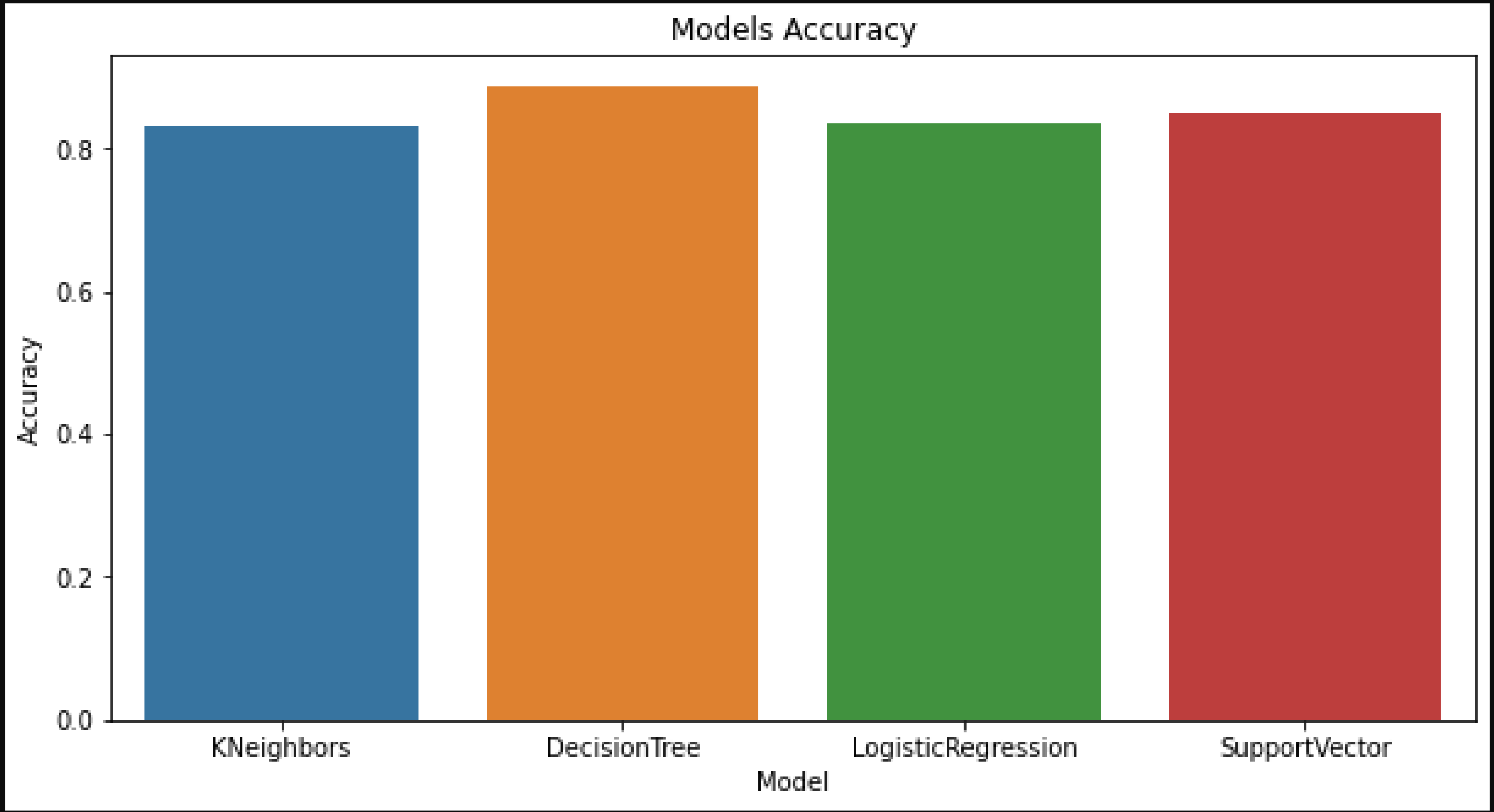
- B4 stands out as the most versatile booster, capable of handling a wide range of payloads, from very light to heavy (up to 10,000 kg), and maintaining a high success rate.
- Other booster versions, such as **B5, FT, v1.0, and v1.1**, have been used for varying payload masses but seem more concentrated in the 0 to 6,000 kg range.
- All booster versions consistently exhibit a high success rate across different payload masses, indicating reliable performance.
- While present in the higher payload range, the FT booster does not handle payloads as heavy as the B4.
- None of the other booster versions (except B4) are shown to manage the heaviest payloads (close to 10,000 kg).

- In summary, while all booster versions demonstrate reliable performance across different payload masses, the B4 booster stands out in terms of versatility and capability to manage a broader spectrum of payloads, including the heaviest ones.

Section 5

Predictive Analysis (Classification)

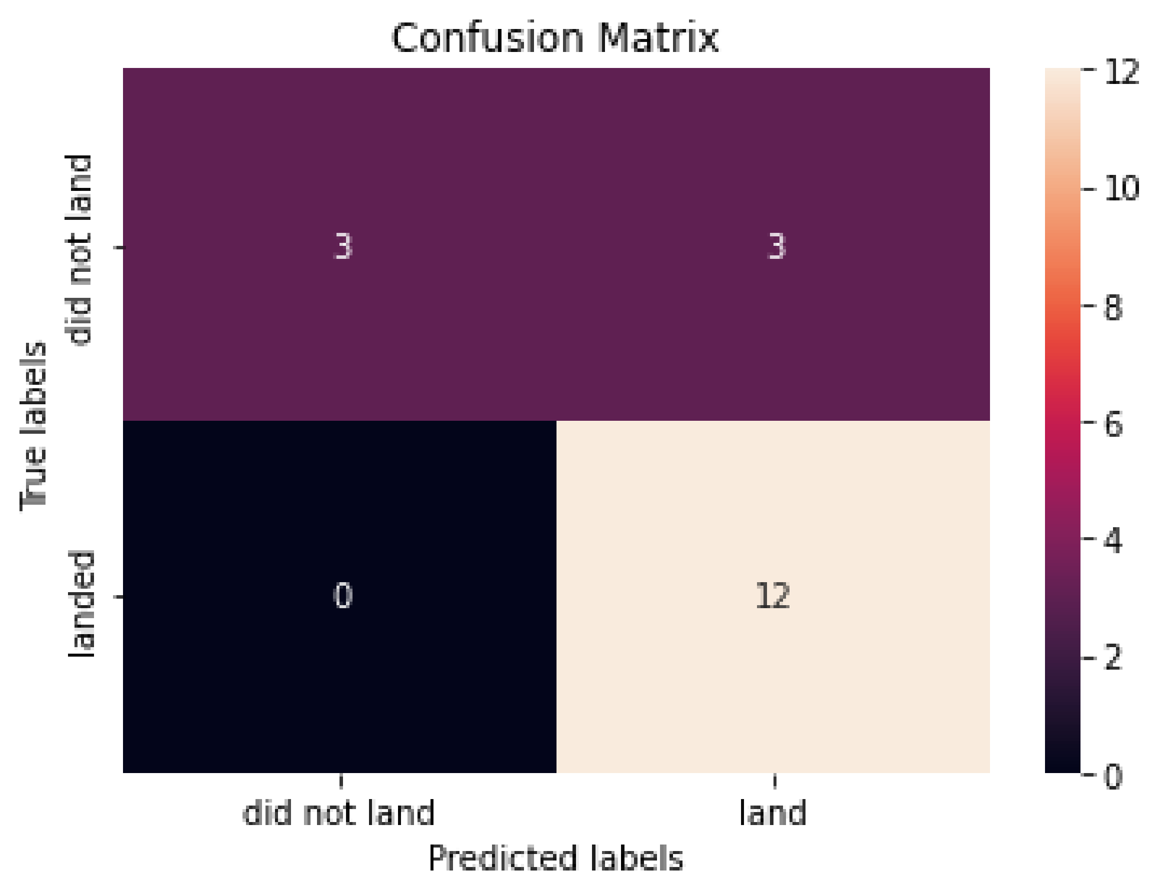
Classification Accuracy



- All models have similar accuracy. The Decision Tree performs the best with a score of .876786

	Model	Accuracy	Test Accuracy	Best Params
0	Logistic Regression	0.821429	0.833333	{'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
1	SVM	0.848214	0.833333	{'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
2	Decision Tree	0.876786	0.833333	{'criterion': 'entropy', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
3	KNN	0.833929	0.777778	{'algorithm': 'auto', 'n_neighbors': 3, 'p': 1}

Confusion Matrix



- The model of the Decision Tree correctly predicted 15 out of 18 events.
- The model made 3 incorrect predictions (false negatives).
- The accuracy of the model can be calculated as (TP + TN) / (TP + TN + FP + FN) = (12 + 3) / 18 = 15/18 = 0.8333 or 83.33%.

Conclusions

Trend Analysis:
SpaceX has been riding a wave of success over the past few years. The data indicates a consistent upward trajectory in their launch successes, reflecting technology, processes, and expertise improvements.

Orbital Landings:
A significant achievement has been successfully landing all first stages of rockets designated for ES-L1, GEO, HEO, and SSO orbits. This showcases SpaceX's reusable rocket technology's efficiency and reliability across various mission types.

Versatility of the B4 Booster:
The B4 booster has proven to be a workhorse for SpaceX. Its adaptability is noteworthy, with the capability to handle a vast spectrum of payloads. Whether it's light cargo or heavy equipment weighing up to 10,000 kg, the B4 maintains a commendable success rate, highlighting its engineering excellence.

Launch Site Success Rates:
Among all the launch sites SpaceX uses, the KSC LC-39A has the highest success rate. This could be attributed to multiple factors such as location advantages, infrastructure, or the specific missions launched from this site.

Machine Learning Insights:
After analyzing various algorithms for predicting the success of a landing, the Decision Tree classifier emerged as the best fit. This suggests that the decision-making process for landings has identifiable patterns and rules, which the Decision Tree algorithm can effectively map out.

Thank you!