IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ville N
16.10.2025

Outline

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

Executive Summary

- Prediction of spacecraft landing is possible
- Predict whether Falcon 9 first stage lands (binary classification)
- There is data to support business feasibility in this area
- There is not much data regarding launches in general, datasets are relatively small

Introduction

- Launch data for various spacecraft and mission outcome
- The purpose of the study is to evaluate whether spacecraft can be reused based in various parameters

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected from web via API using JSON packet structure
- Perform data wrangling
  - Data was preprocessed and scaled for better analysis performance
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

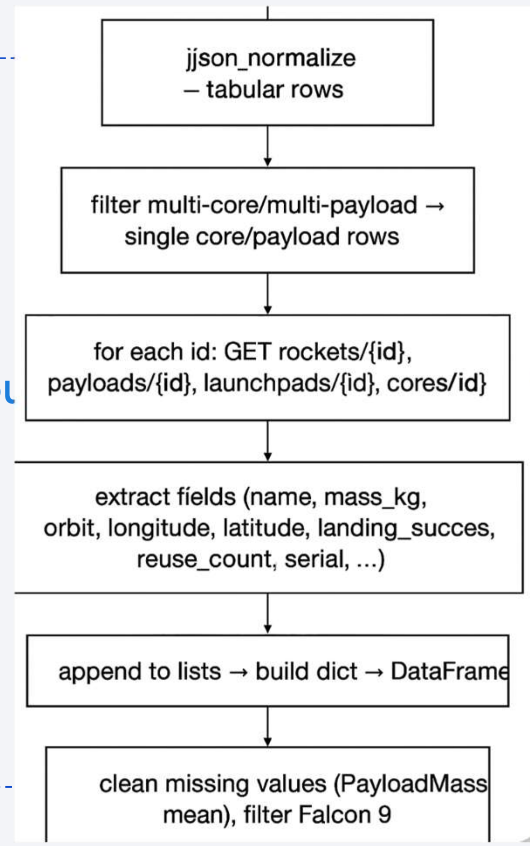
Data Collection

- Collect and clean SpaceX past-launch data via the SpaceX API (static JSON provided) to build a Falcon-9 launch dataset for later modeling.

Data Collection – SpaceX API

- https://github.com/ddali/ville_ibm_data_science/blob/main/1.%20jupyter-labs-spacex-data-collection-api-v2.ipynb

Place you
here



I calls

Data Wrangling

- Lab 2 — Data wrangling for SpaceX Falcon-9 landing prediction: perform EDA and create the binary training label (Class) where 1 = successful first-stage landing, 0 = unsuccessful.
- Inputs / main data

- Loads cleaned launch CSV from previous lab: dataset_part_1.csv (launch metadata with columns such as BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, LandingPad, …).
- https://github.com/ddali/ville_ibm_data_science/blob/main/2.%20labs-jupyter-spacex-Data%20wrangling-v2.ipynb

EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts
- https://github.com/ddali/ville_ibm_data_science/blob/main/4.%20jupyter-labs-eda-dataviz-v2.ipynb
- Chart Type: sns.catplot — PayloadMass vs FlightNumber, hue=Class
  - Purpose: Show how payload mass and launch experience relate to landing success
- Chart Type: sns.catplot (strip) — FlightNumber vs LaunchSite
  - Purpose: Visualize launch distribution across sites over time
- Chart Type: sns.scatterplot — PayloadMass vs LaunchSite, hue=Class
  - Purpose: Compare payload masses and success across launch sites
- Chart Type: Bar chart (groupby Orbit → mean(Class))
  - Purpose: Visualize average landing success by orbit type
- Chart Type: Scatter — FlightNumber vs Orbit, hue=Class
  - Purpose: Explore success trends across orbits and flight experience
- Chart Type: Scatter — PayloadMass vs Orbit, hue=Class
  - Purpose: Analyze payload mass and orbit type impact on success
- Chart Type: Line chart — Year vs average success rate
  - Purpose: Track landing success trend over time

EDA with SQL

- https://github.com/ddali/ville_ibm_data_science/blob/main/3.%20jupyter-labs-eda-sql-edx-sqllite-v2.ipynb
- Queries
  - List unique launch sites (SELECT DISTINCT Launch_Site).
  - Show records with Launch_Site beginning with 'KSC' (WHERE Launch_Site LIKE 'KSC%').
  - Sum payload mass for NASA (CRS) missions grouped by Booster_Version (SUM + GROUP BY).
  - Average payload mass for booster version F9 v1.1 (AVG with WHERE Booster_Version LIKE 'F9 v1.1%').
  - Earliest date for successful ground-pad landing (min(Date) filtered by Landing_Outcome).
  - Booster names that succeeded on ground pad with payload mass between 4000 and 6000 (DISTINCT + WHERE + numeric filters).
  - Count total records with Success/Failure landing outcomes.

Build an Interactive Map with Folium

- folium (Map, Circle, Marker), folium.plugins.MarkerCluster, MousePosition, DivIconExplain object are mainly items on the map.
- https://github.com/ddali/ville_ibm_data_science/blob/main/5.%20lab-jupyter-launch-site-location-v2.ipynb

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

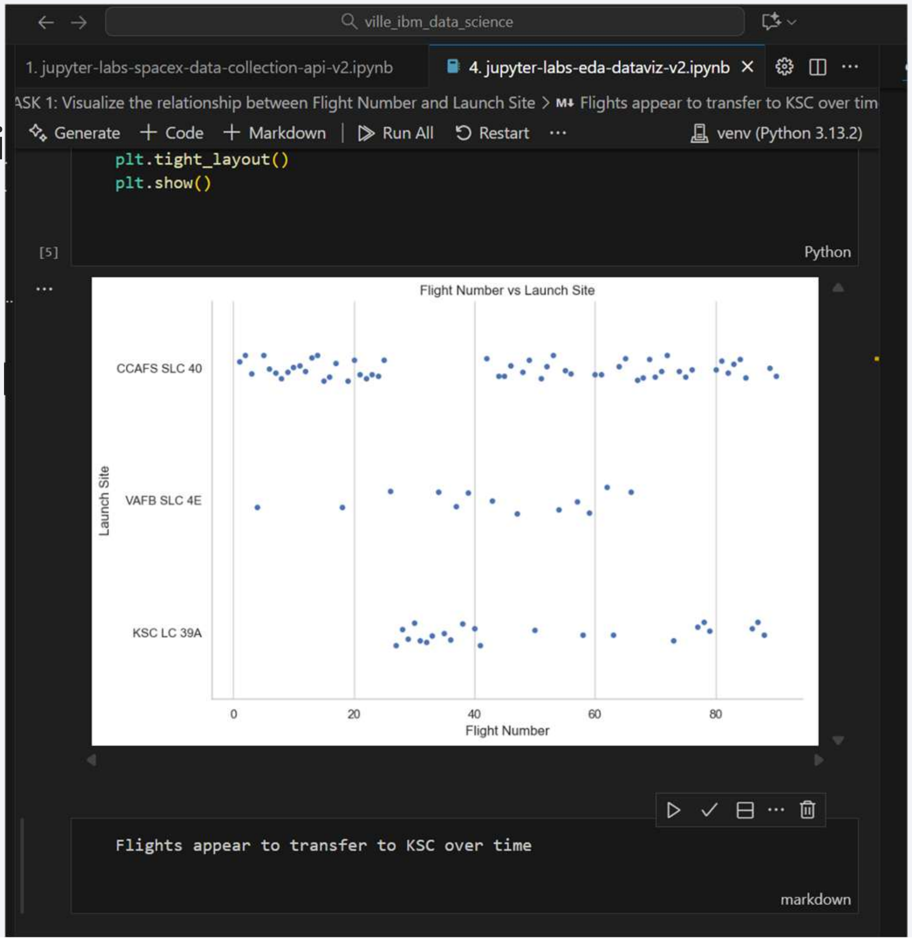
- Best hyperparameters and cross-val accuracy for each model, test accuracies, confusion matrices, and a short conclusion which algorithm performs best on the provided dataset.
- Support Vector Machine (SVC)params: kernel ∈ {linear, rbf, poly, sigmoid}, C and gamma on logspace(-3,3,5)svm_cv is fit on training data and best params / best_score_ printed
- DecisionTreeClassifierparams include criterion, splitter, max_depth, max_features, min_samples_leaf/split
- KNeighborsClassifierparams: n_neighbors, algorithm, p (1 or 2)
- https://github.com/ddali/ville_ibm_data_science/blob/main/6.%20SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

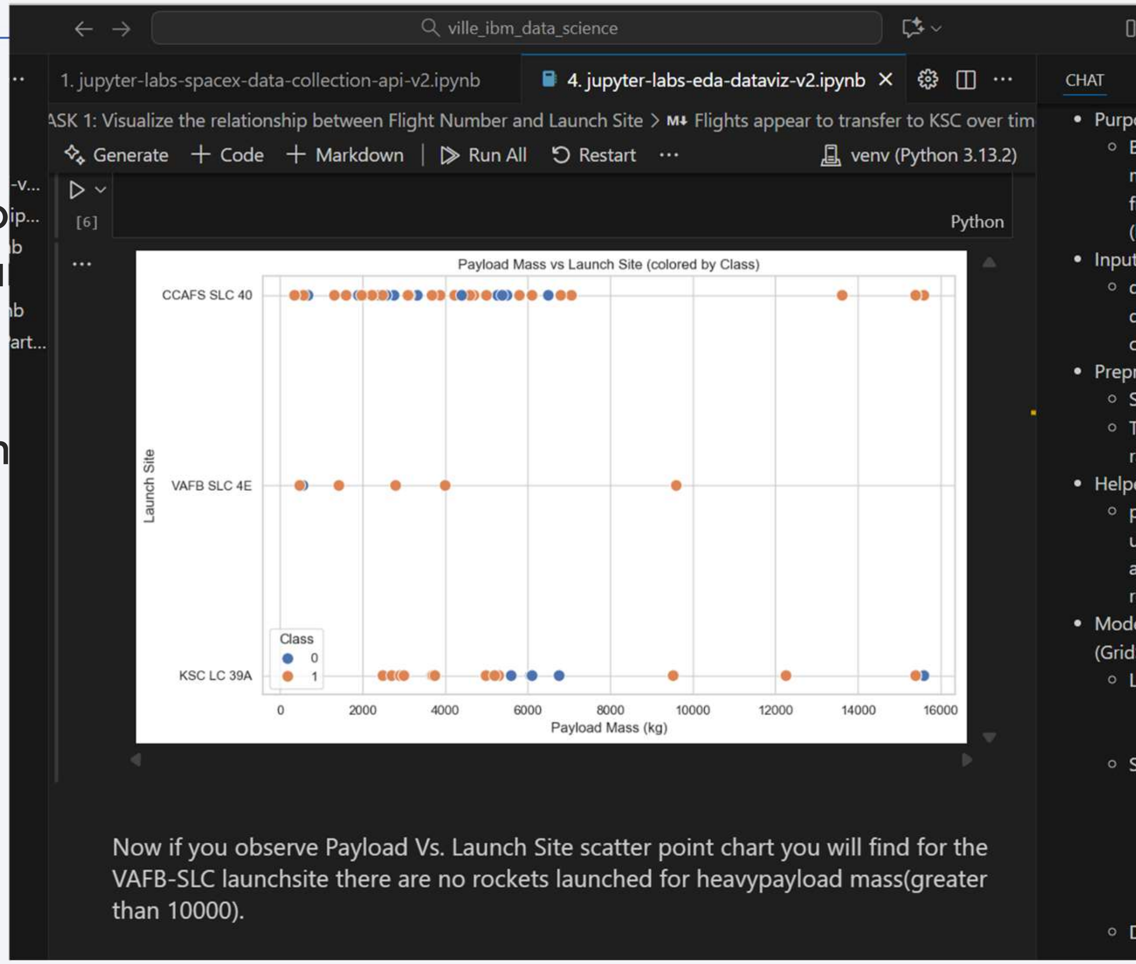
- Show a scatter plot of Flight Number vs. Launch Site
- Show the screenshot of the scatter plot with explanations



Payload vs. Launch Site

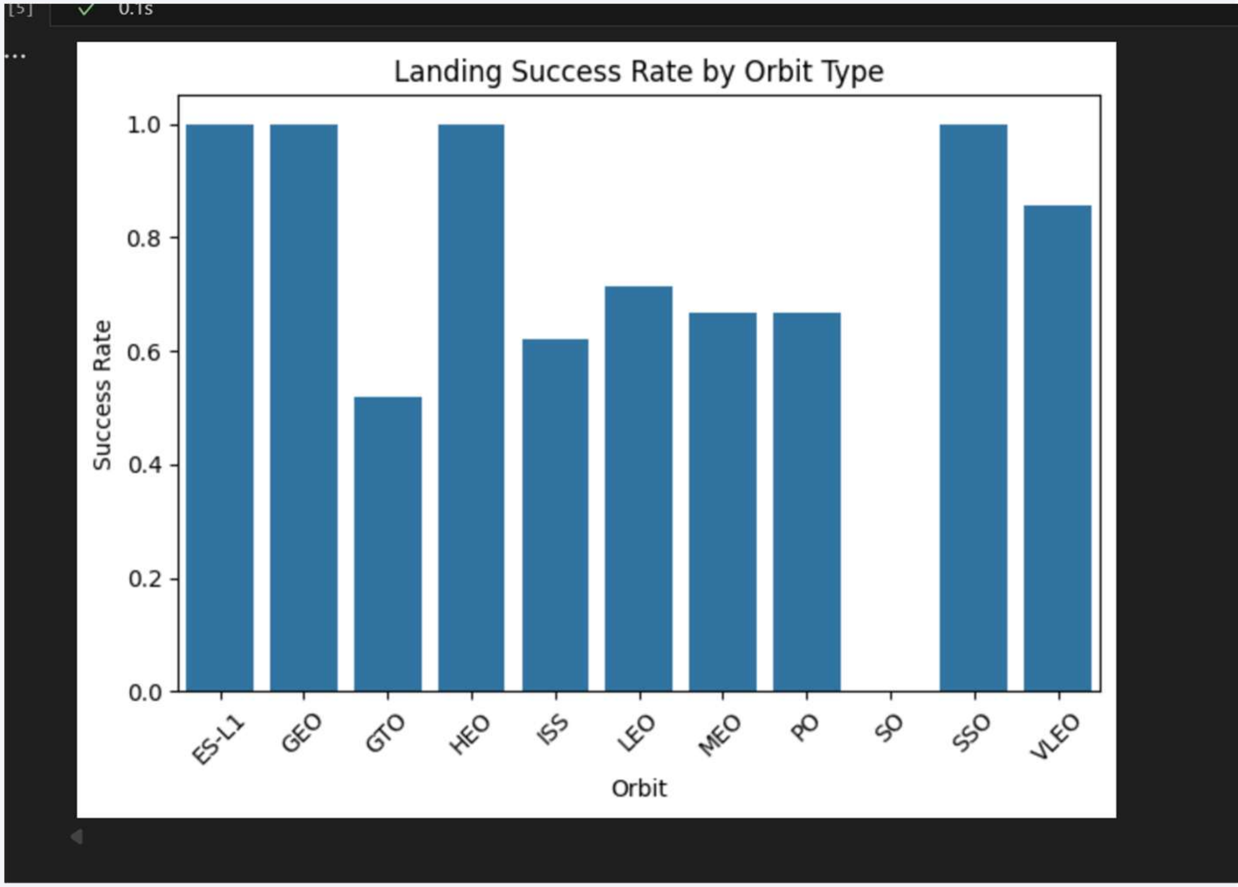
- Show a scatter plo
  of Payload vs. Lau

- Show the screensh
  scatter plot with
  explanations



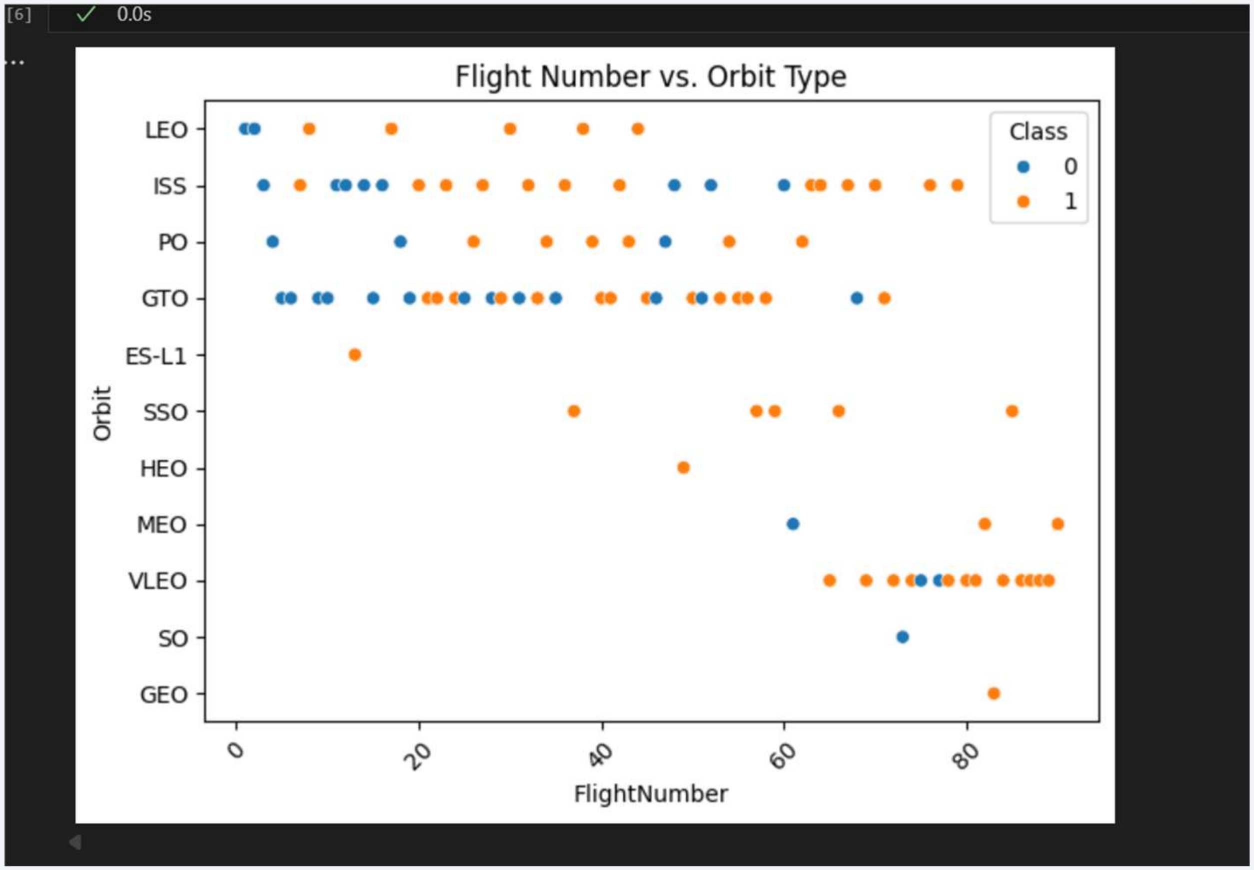
Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type



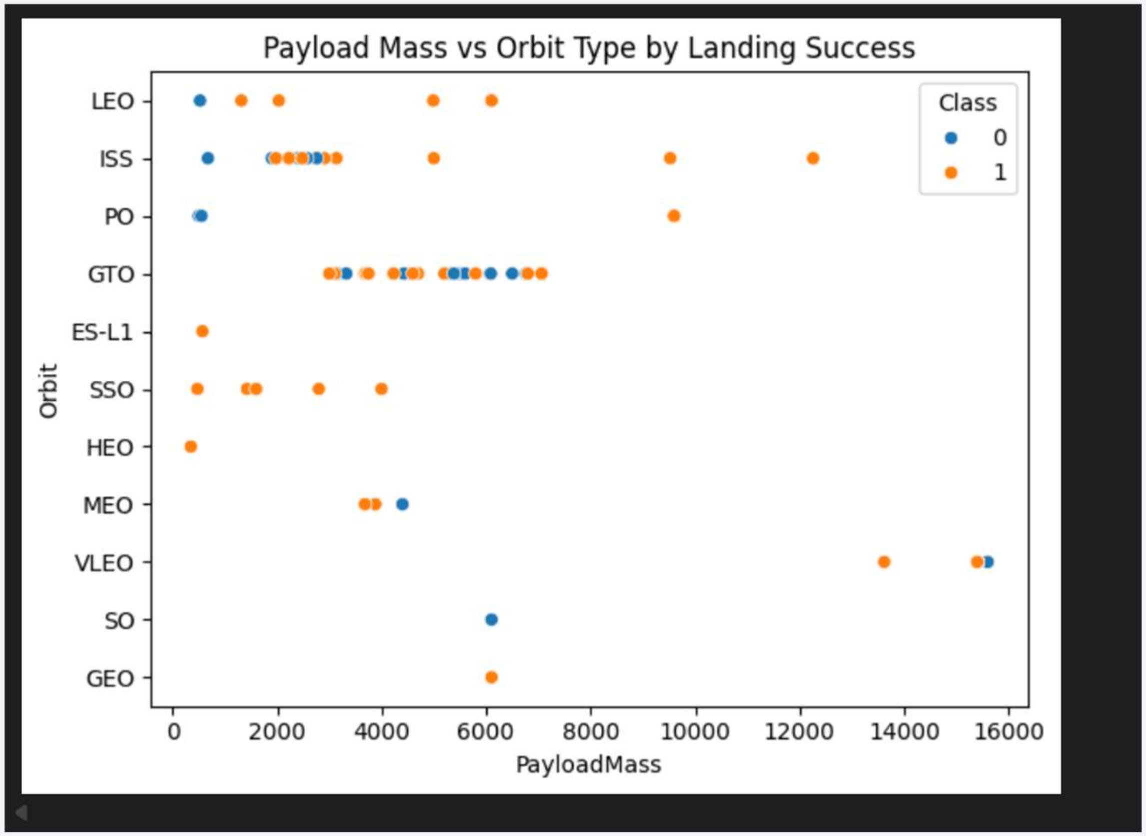
Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type



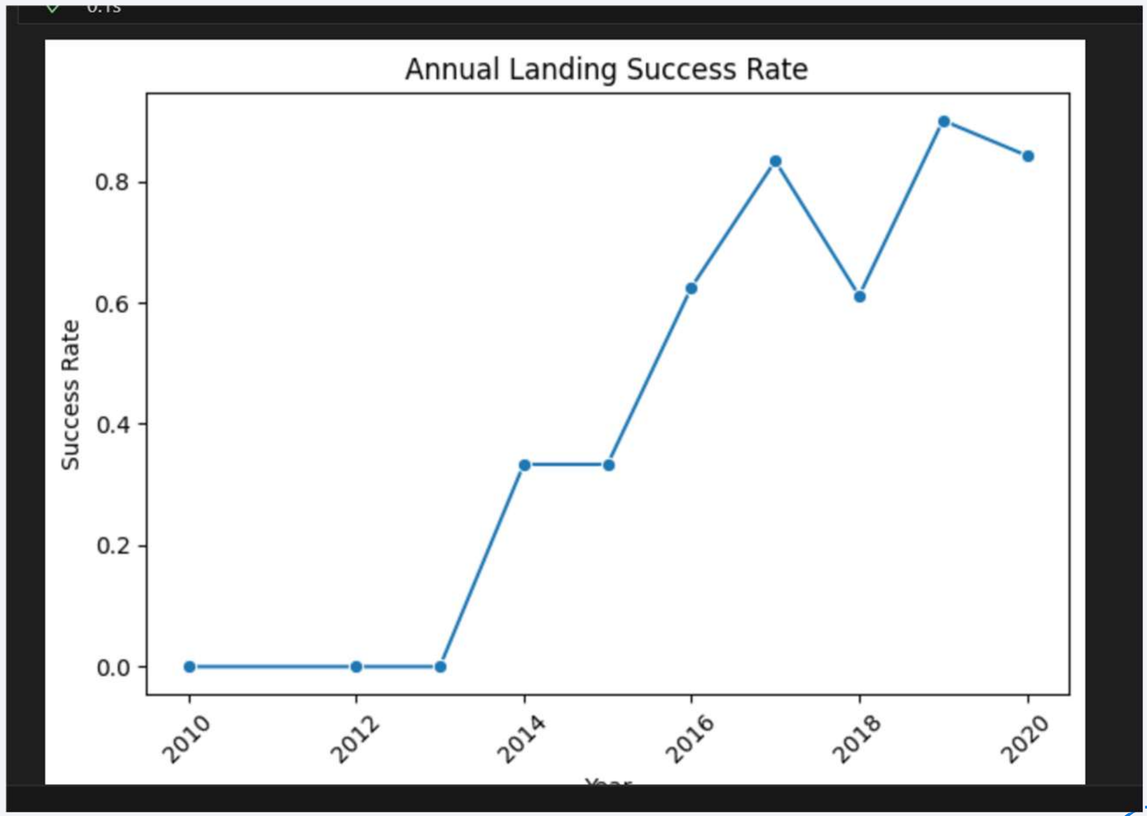
Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type
- Show the screenshot of the scatter plot with explanations



Launch Success Yearly Trend

- Show a line chart of yearly average success rate



All Launch Site Names

- Find the names of the unique launch sites

Now, you can take a look at what are the coordinates for each site.

```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

[5]

	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

Launch Site Names Begin with 'KSC'

- Find 5 records where launch sites' names start with `KSC`

```
ksc_records = spacex_df[spacex_df['Launch Site'].str.startswith('KSC')].head(5)
print(ksc_records)
```

[11] ✓ 0.0s

```
   Launch Site        Lat       Long  class
36  KSC LC-39A  28.573255 -80.646895      1
37  KSC LC-39A  28.573255 -80.646895      0
38  KSC LC-39A  28.573255 -80.646895      1
39  KSC LC-39A  28.573255 -80.646895      1
40  KSC LC-39A  28.573255 -80.646895      0
```

```
query = "SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'KSC%'"
ksc_launches = pd.read_sql(query, con)
print(ksc_launches)
```

[21]

```
         Date Time (UTC) Booster_Version Launch_Site  \
0  2017-02-19   14:39:00   F9 FT B1031.1  KSC LC-39A
1  2017-03-16    6:00:00     F9 FT B1030  KSC LC-39A
2  2017-03-30   22:27:00  F9 FT  B1021.2  KSC LC-39A
3  2017-05-01   11:15:00   F9 FT B1032.1  KSC LC-39A
4  2017-05-15   23:21:00     F9 FT B1034  KSC LC-39A
5  2017-06-03   21:07:00   F9 FT B1035.1  KSC LC-39A
```

Total Payload Mass

- Calculate the total payload carrie
- Present your query result with a s

```
query = """
SELECT Booster_Version, SUM(PAYLOAD_MASS__KG_) AS Total_Mass_KG
FROM SPACEXTBL
WHERE Payload LIKE '%CRS%'
GROUP BY Booster_Version

UNION ALL

SELECT 'TOTAL' AS Booster_Version, SUM(PAYLOAD_MASS__KG_) AS Total_Mass_KG
FROM SPACEXTBL
WHERE Payload LIKE '%CRS%'
"""

crs_mass_by_booster = pd.read_sql(query, con)
print(crs_mass_by_booster)
```

[6] ✓ 0.0s

```
   Booster_Version  Total_Mass_KG
0   F9 B4  B1039.2           2647
1    F9 B4 B1039.1           3310
2    F9 B4 B1045.2           2697
3    F9 B5 B1048.4          15600
4   F9 B5 B1049.7           15600
5   F9 B5 B1051.2            4200
6   F9 B5 B1056.2            2268
7    F9 B5 B1056.4          15600
8   F9 B5 B1058.4            2972
9    F9 B5 B1059.2           1977
10      F9 B5B1050           2500
11    F9 B5B1051.1          12055
12   F9 B5B1056.1            2495
13    F9 B5B1059.1           2617
14  F9 FT  B1035.2           2205
15   F9 FT B1021.1           3136
16   F9 FT B1025.1           2257
17   F9 FT B1031.1           2490
18   F9 FT B1035.1           2708
19  F9 v1.0  B0006            500
20  F9 v1.0  B0007            677
21          F9 v1.1           2296
22   F9 v1.1 B1010           2216
23   F9 v1.1 B1012           2395
24   F9 v1.1 B1015           1898
25   F9 v1.1 B1018           1952
26           TOTAL         111268
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

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

```
query = """
SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Mass_KG
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.1%'
"""
avg_mass_df = pd.read_sql(query, con)
print(f"Average payload mass for F9 v1.1: {avg_mass_df['Average_Mass_KG'][0]:.2f} kg")
```

[23]

```
Average payload mass for F9 v1.1: 2534.67 kg
```

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on drone ship.Present your query result with a short explanation here

Task 5

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

```
first_date = df[df['Landing_Outcome'] == 'Success (ground pad)']['Date'].min()
print(f"First ground pad landing: {first_date}")
```

```
First ground pad landing: 2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but l

- Present your query result with a short expl

Task 6

List the names of the boosters which have success in ground pad

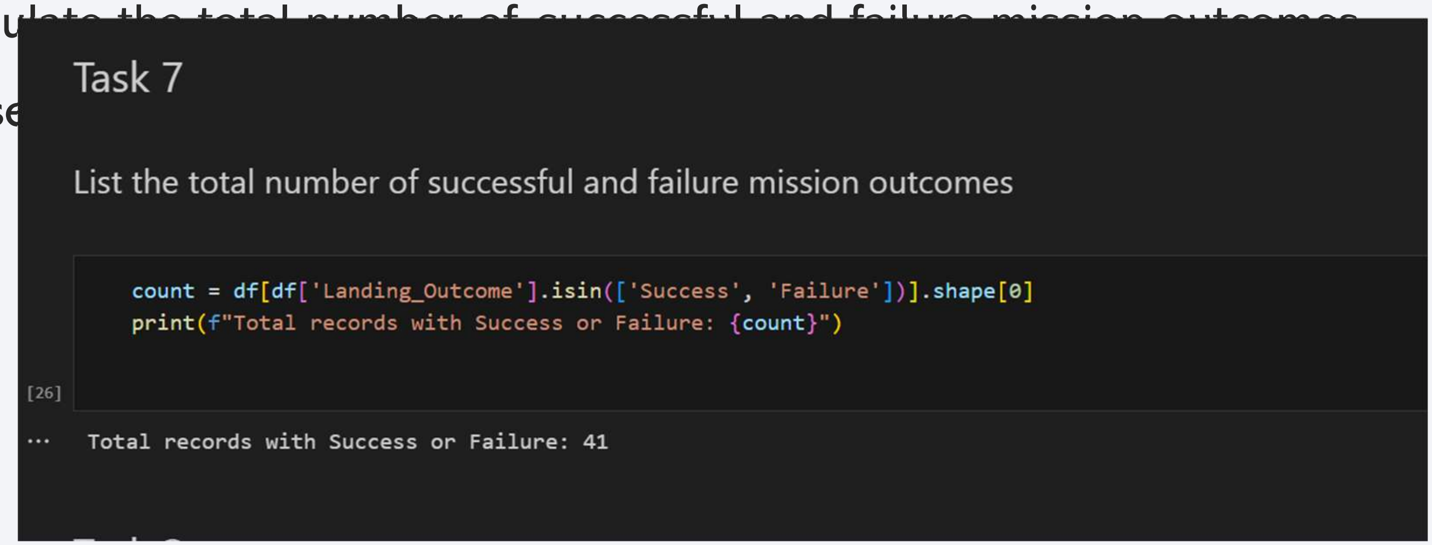
```
query = """
SELECT DISTINCT Booster_Version
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (ground pad)'
  AND PAYLOAD_MASS__KG_ > 4000
  AND PAYLOAD_MASS__KG_ < 6000
"""
filtered_boosters_df = pd.read_sql(query, con)
print(filtered_boosters_df)
```

[25]

```
...   Booster_Version
   0   F9 FT B1032.1
   1   F9 B4 B1040.1
   2   F9 B4 B1043.1
```

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Prese



Boosters Carried Maximum

- List the names of the booster which have c
- Present your query result with a short expl

Task 8

List all the booster_versions that have carried the maximum payload mass. Use a subquery.

```
query = """
SELECT Booster_Version, SUM(PAYLOAD_MASS__KG_) AS Total_Mass_KG
FROM SPACEXTBL
WHERE Payload LIKE '%CRS%'
GROUP BY Booster_Version

UNION ALL

SELECT 'TOTAL' AS Booster_Version, SUM(PAYLOAD_MASS__KG_) AS Total_Mass_KG
FROM SPACEXTBL
WHERE Payload LIKE '%CRS%'
"""

crs_mass_by_booster = pd.read_sql(query, con)
print(crs_mass_by_booster)
```

[6] ✓ 0.0s

```
   Booster_Version  Total_Mass_KG
0   F9 B4  B1039.2           2647
1    F9 B4 B1039.1           3310
2    F9 B4 B1045.2           2697
3    F9 B5 B1048.4          15600
4   F9 B5 B1049.7           15600
5   F9 B5 B1051.2            4200
6   F9 B5 B1056.2            2268
7    F9 B5 B1056.4          15600
8   F9 B5 B1058.4            2972
9    F9 B5 B1059.2           1977
10      F9 B5B1050           2500
11    F9 B5B1051.1          12055
12   F9 B5B1056.1            2495
13    F9 B5B1059.1           2617
14  F9 FT  B1035.2           2205
15   F9 FT B1021.1           3136
16   F9 FT B1025.1           2257
17   F9 FT B1031.1           2490
18   F9 FT B1035.1           2708
19  F9 v1.0  B0006            500
20  F9 v1.0  B0007            677
21         F9 v1.1           2296
22   F9 v1.1 B1010           2216
23   F9 v1.1 B1012           2395
24   F9 v1.1 B1015           1898
25   F9 v1.1 B1018           1952
26           TOTAL         111268
```

2015 Launch Records

- List the records which will display the mon landing_outcomes in ground pad ,booster months in year 2017

- Present your query result with a short expl

```
query = """
SELECT
  strftime('%Y', Date) AS Year,
  strftime('%m', Date) AS MonthNumber,
  CASE strftime('%m', Date)
    WHEN '01' THEN 'January'
    WHEN '02' THEN 'February'
    WHEN '03' THEN 'March'
    WHEN '04' THEN 'April'
    WHEN '05' THEN 'May'
    WHEN '06' THEN 'June'
    WHEN '07' THEN 'July'
    WHEN '08' THEN 'August'
    WHEN '09' THEN 'September'
    WHEN '10' THEN 'October'
    WHEN '11' THEN 'November'
    WHEN '12' THEN 'December'
  END AS Month,
  Booster_Version,
  Launch_Site,
  Landing_Outcome
FROM SPACEXTBL
WHERE
  strftime('%Y', Date) = '2017' AND
  Landing_Outcome LIKE '%Ground Pad%'
"""

result = pd.read_sql(query, con)
print(result)
```

[12] ✓ 0.0s

```
   Year MonthNumber      Month Booster_Version   Launch_Site  \
0  2017          02   February    F9 FT B1031.1   KSC LC-39A
1  2017          05        May    F9 FT B1032.1   KSC LC-39A
2  2017          06       June    F9 FT B1035.1   KSC LC-39A
3  2017          08     August    F9 B4 B1039.1   KSC LC-39A
4  2017          09  September    F9 B4 B1040.1   KSC LC-39A
5  2017          12   December   F9 FT  B1035.2  CCAFS SLC-40

        Landing_Outcome
0  Success (ground pad)
1  Success (ground pad)
2  Success (ground pad)
3  Success (ground pad)
4  Success (ground pad)
5  Success (ground pad)
```

Rank Landing Outcomes B

- Rank the count of landing outc
(ground pad)) between the date
descending order

- Present your query result with a

Rank the count of landing outcomes (such as Failure (drone ship) or Succe

```python
query = """
SELECT
  Landing_Outcome,
  COUNT(*) AS Outcome_Count
FROM SPACEXTBL
WHERE
  Date >= '2010-06-04' AND
  Date <= '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Outcome_Count DESC
"""

ranked_outcomes = pd.read_sql(query, con)
print(ranked_outcomes)
```

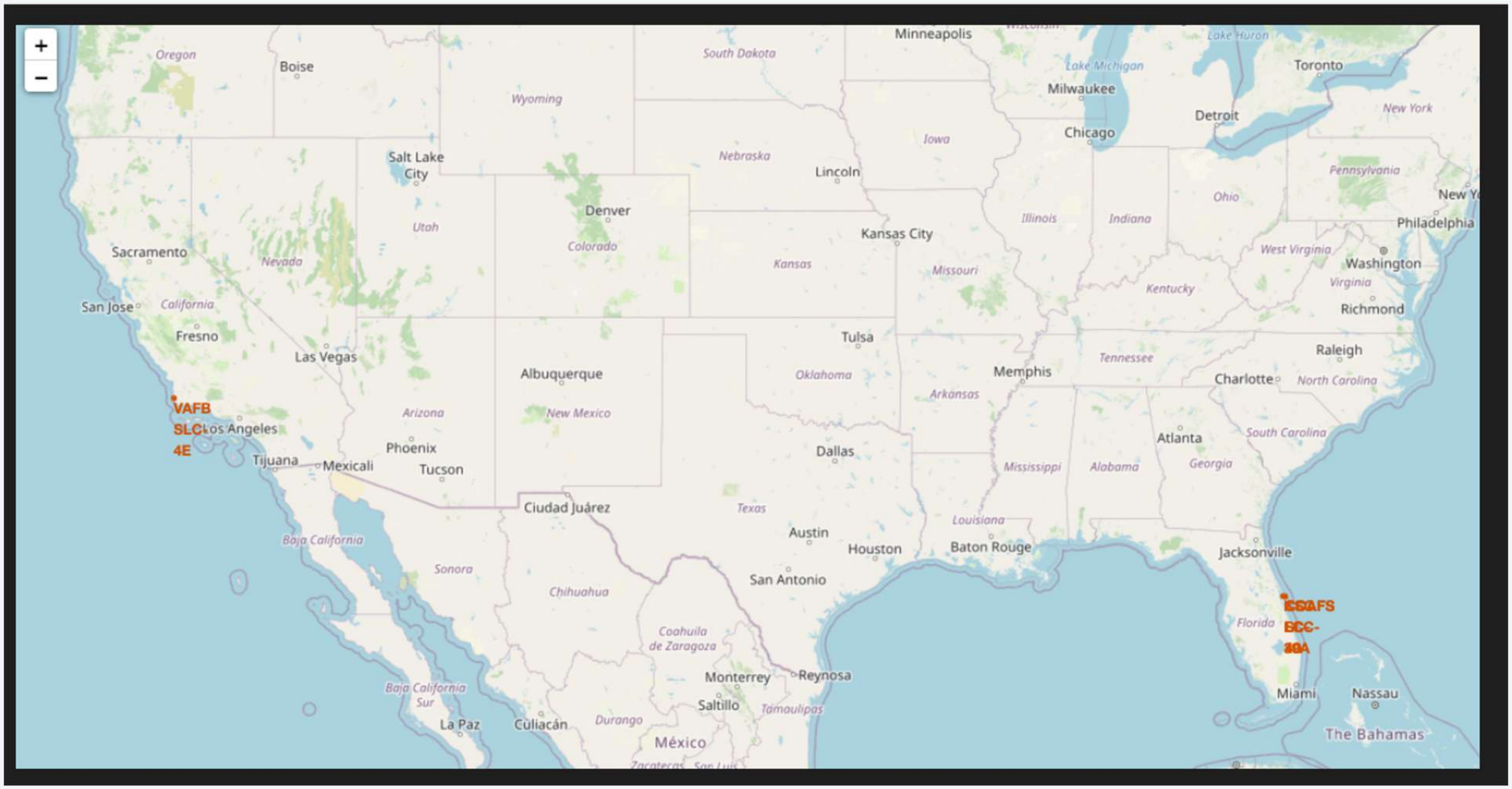
[13] ✓ 0.0s

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

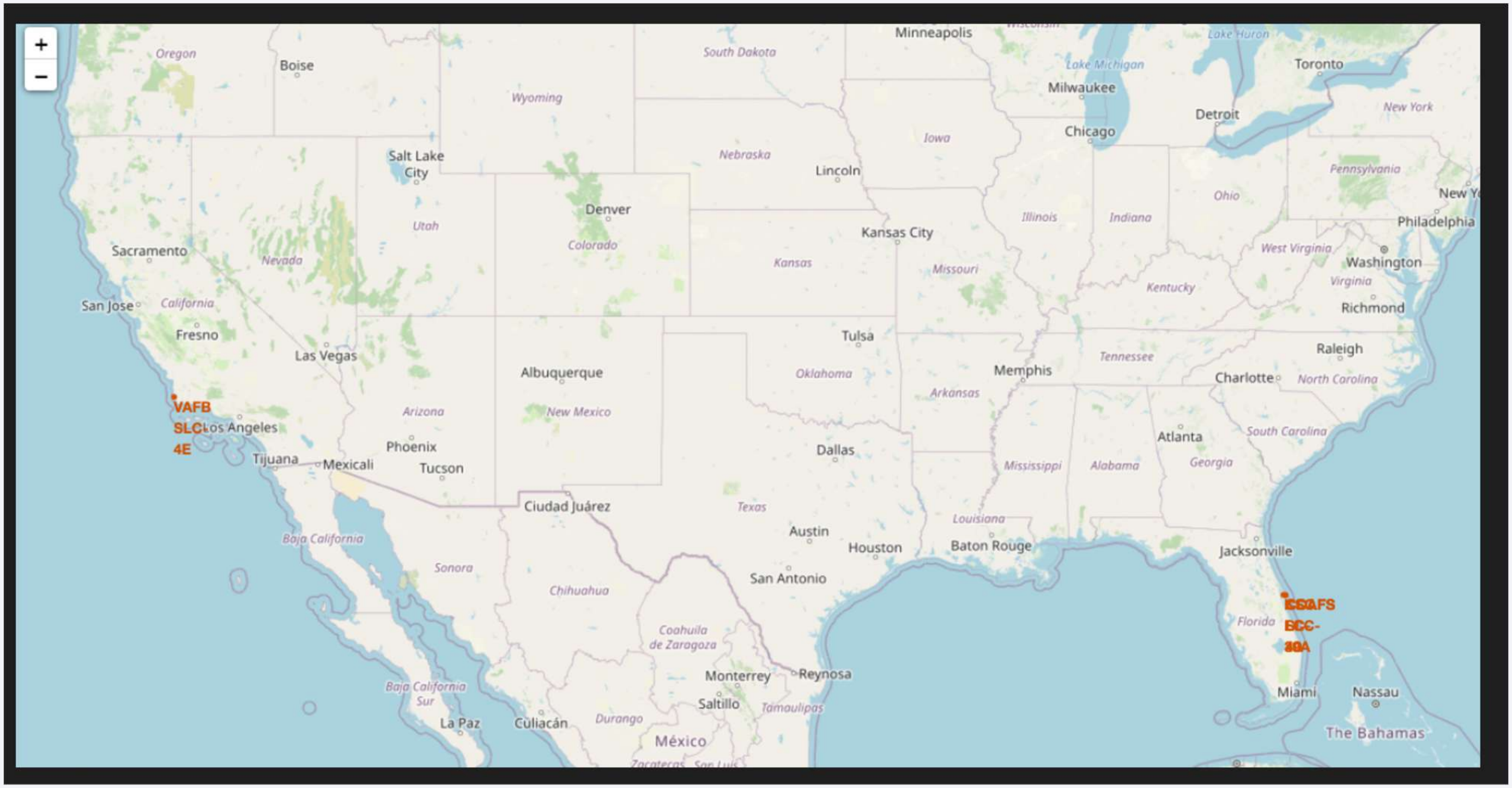
Section 3

Launch Sites Proximities Analysis

Launch site locations

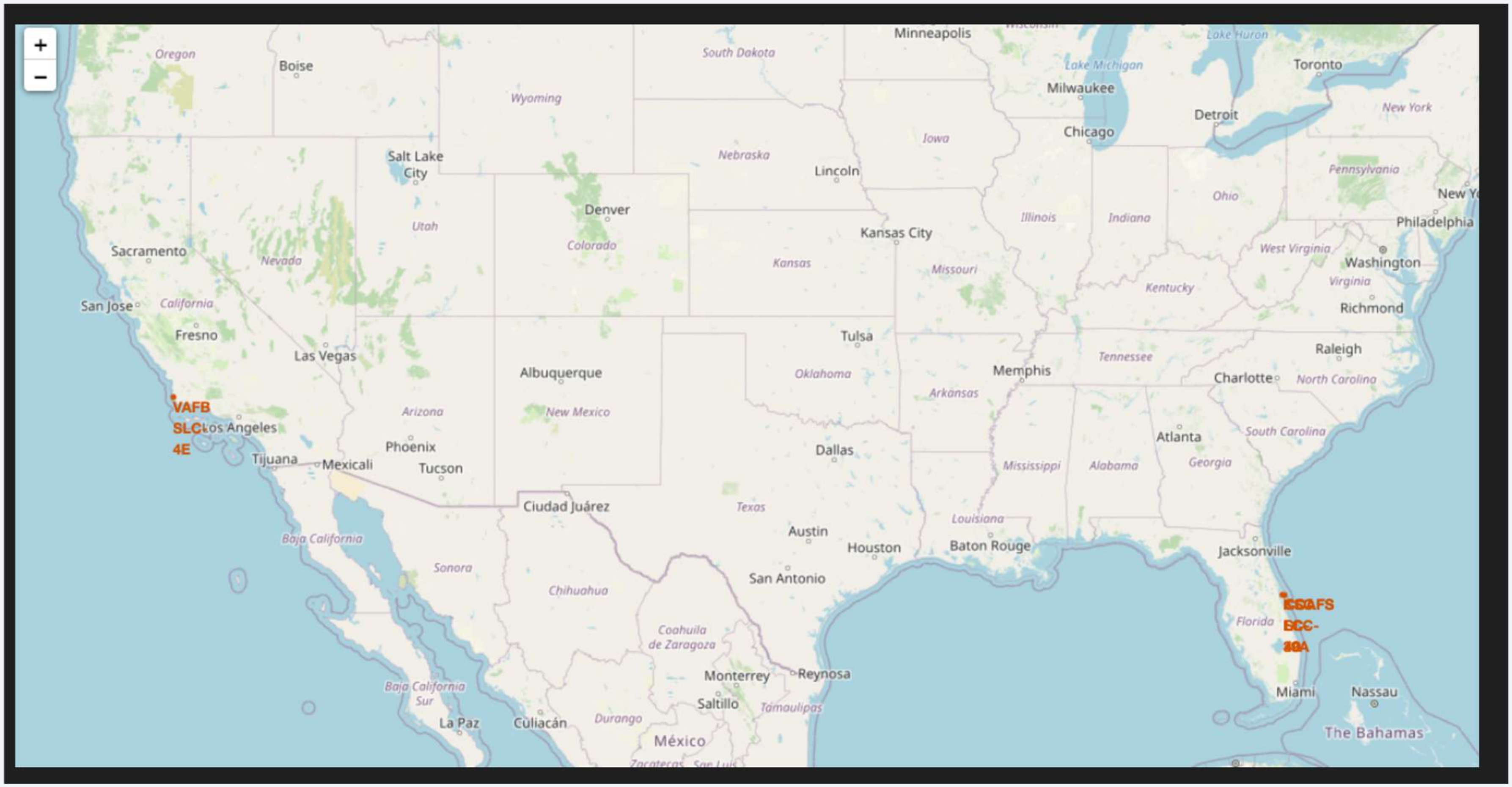


outcomes



VAFB
SLC-
4E
CCAFS
Los Angeles
Sacramento
San Jose
Fresno
Las Vegas
Boise
Salt Lake City
Denver
Albuquerque
Phoenix
Tucson
Tijuana
Mexicali
Ciudad Juárez
Minneapolis
Milwaukee
Chicago
Detroit
Toronto
Lincoln
Kansas City
Tulsa
Memphis
Dallas
Austin
Houston
San Antonio
Baton Rouge
Atlanta
Charlotte
Raleigh
Richmond
Washington
Philadelphia
Jacksonville
Miami
Nassau
The Bahamas
Monterrey
Saltillo
Reynosa
La Paz
Culiacán
México

Outcomes 2



VAFB
4E
Oregon
Boise
South Dakota
Minneapolis
Toronto
Wyoming
Lake Michigan
Milwaukee
Detroit
New York
Chicago
Iowa
Nebraska
Salt Lake City
Lincoln
Pennsylvania
Ohio
Philadelphia
Denver
Kansas City
Illinois
Indiana
Utah
Colorado
Sacramento
Nevada
Kansas
Missouri
West Virginia
Washington
Virginia
Kentucky
San Jose
California
Richmond
Fresno
Tulsa
Raleigh
Las Vegas
Albuquerque
Oklahoma
Memphis
Tennessee
Charlotte
North Carolina
Arkansas
Arizona
New Mexico
Atlanta
South Carolina
Phoenix
Dallas
Tijuana
Mexicali
Tucson
Mississippi
Alabama
Georgia
Ciudad Juárez
Texas
Louisiana
Austin
Houston
Baton Rouge
Jacksonville
Baja California
Sonora
San Antonio
Chihuahua
Florida
Coahuila de Zaragoza
Monterrey
Reynosa
Miami
Nassau
Baja California Sur
Saltillo
Tamaulipas
La Paz
Culiacán
Durango
México
The Bahamas

Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title

- Show the screenshot of the piechart for the launch site with highest launch success ratio

- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

GridSearchCV

best_estimator_: SVC

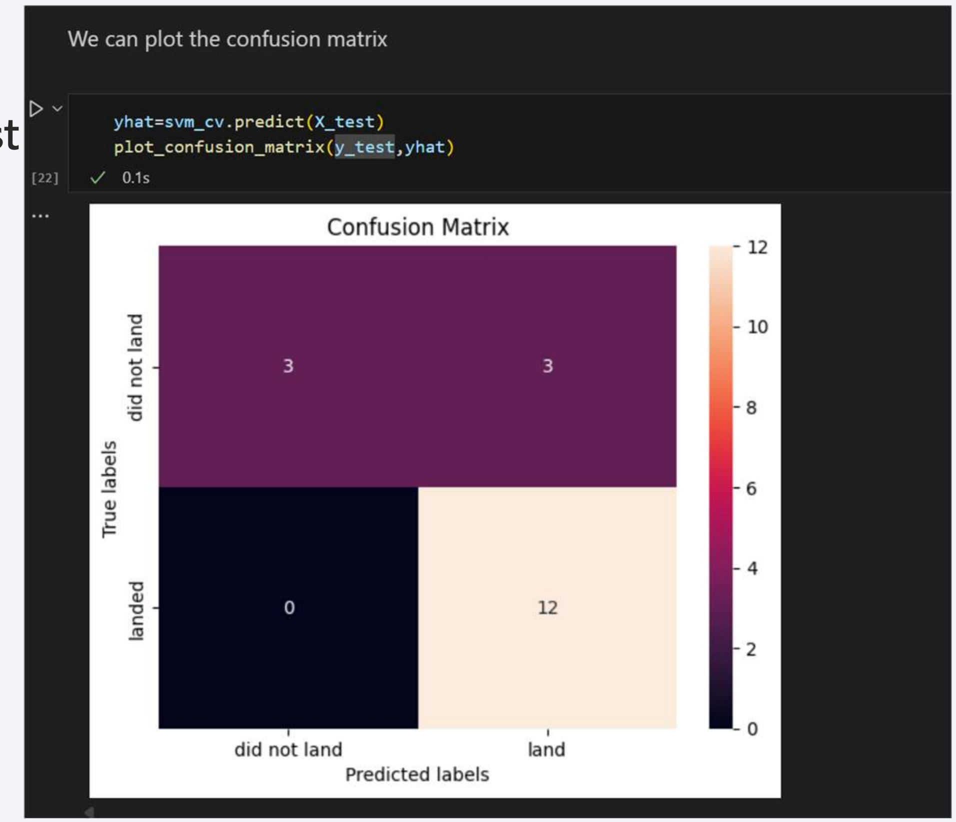
SVC

▼ Parameters

C	np.float64(1.0)
kernel	'sigmoid'
degree	3
gamma	np.float64(0....2277660168379)
coef0	0.0
shrinking	True
probability	False
tol	0.001
cache_size	200
class_weight	None
verbose	False
max_iter	-1
decision_function_shape	'ovr'
break_ties	False
random_state	None

Confusion Matrix

- Show the confusion matrix of the best explanation



Conclusions

- Decision tree is the best
- Tuned hyperparameters (best parameters): {'C': np.float64(1.0), 'gamma': np.float64(0.03162277660168379), 'kernel': 'sigmoid'}
- Cross-validated accuracy: 0.8482142857142856
- Test set accuracy: 0.8333333333333334

Thank you!