Winning Space Race with Data Science

Robert Muigai Ngechu

8/6/2025

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 analysis

Introduction

PROJECT BACKGROUND AND CONTEXT

This project represents the culmination of our learning journey as part of the final course in the IBM Data Science Professional Certificate program. We are working as data scientists for SpaceY, an emerging aerospace company with ambitions to compete in the commercial space launch market alongside industry leaders like SpaceX.

Our mission is to apply data science methodologies to help SpaceY optimize its rocket launch strategies, potentially enabling the company to make more competitive bids against established players in the industry.

PROBLEMS YOU WANT TO FIND ANSWERS

SpaceX has revolutionized space travel by significantly reducing launch costs through the reuse of rocket boosters. Their Falcon 9 launches cost approximately $62 million - a fraction of traditional launch prices. This cost efficiency comes primarily from recovering and reusing the first stage booster, which alone represents a $15+ million value, excluding R&D costs.

However, booster recovery isn't always possible. Heavy payloads or specific orbital requirements may necessitate expending the booster, dramatically increasing costs. Our critical challenge is to predict whether a given launch will successfully recover its first stage booster. Accurate predictions could lead to substantial cost savings - a key factor in maintaining competitive pricing and delivering value to stakeholders.

Section 1

Methodology

Methodology

EXECUTIVE SUMMARY

1 Data collection methodology:

- Utilized SpaceX's open-source REST API to gather launch data
- Performed web scraping of the Wikipedia page "List of Falcon 9 and Falcon Heavy Launches" to supplement our dataset

2 Perform data wrangling:

- Implemented One Hot Encoding to transform categorical variables for machine learning compatibility
- Conducted thorough data cleaning to remove null values and irrelevant features

3Perform exploratory data analysis (EDA) using visualization and SQL

4- Perform interactive visual analytics using Folium and Plotly Dash

5 Perform predictive analysis using classification models:

- Built and evaluated multiple classification models including: Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Tree Classifier
- Compared model performance to identify the most effective prediction approach

Data Collection

1- SpaceX API Integration:

- Collected historical launch records through SpaceX's public REST API
- Executed GET requests to retrieve and process launch data
- Filtered results to include only Falcon 9 missions
- Handled missing payload mass values from classified missions by imputing with average weights

2- Wikipedia Web Scraping:

- Harvested complementary launch data from the Wikipedia page "List of Falcon 9 and Falcon Heavy Launches"
- Accessed the page content through direct URL retrieval
- Identified and extracted all column headers from the HTML table structure
- Converted the scraped table into a structured Pandas DataFrame for analytical processing

Data Collection – SpaceX API

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
[9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets
```

We should see that the request was successfull with the 200 status response code

```
[10]: response=requests.get(static_json_url)

[11]: response.status_code

[11]: 200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
[13]: # Use json_normalize meethod to convert the json result into a dataframe
      data=pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
[15]: # Get the head of the dataframe
      data.head()
```

```
[24]: 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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

```
[25]: # Create a data from launch_dict
      df = pd.DataFrame.from_dict(launch_dict)
```

Show the summary of the dataframe

```
[26]: # Show the head of the dataframe
      df.head()
```

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
[27]: # Hint data['BoosterVersion']!='Falcon 1'
      data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

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

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

Data Wrangling

We can see below that some of the rows are missing values in our dataset.

```
[29]: data_falcon9.isnull().sum()

[29]: FlightNumber       0
      Date               0
      BoosterVersion     0
      PayloadMass        5
      Orbit              0
      LaunchSite         0
      Outcome            0
      Flights            0
      GridFins           0
      Reused             0
      Legs               0
      LandingPad        26
      Block              0
      ReusedCount        0
      Serial             0
      Longitude          0
      Latitude           0
      dtype: int64
```

Before we can continue we must deal with these missing values. The `LandingPad` column will retain None values to represent when landing pads were not used.

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
[30]: # Calculate the mean value of PayloadMass column
      payloadmassavg = data_falcon9['PayloadMass'].mean()
      # Replace the np.nan values with its mean value
      data_falcon9['PayloadMass'].replace(np.nan, payloadmassavg, inplace=True)
```

Data Collection - Scraping

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
[12]: # use requests.get() method with the provided static_url
      # assign the response to a object
      data=requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

```
[14]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
      soup = BeautifulSoup(data, "html.parser")
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
[15]: # Use soup.title attribute
      print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches – Wikipedia</title>
```

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

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
[16]: # Use the find_all function in the BeautifulSoup object, with element type `table`
      # Assign the result to a list called `html_tables`
      html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

```
[17]: # Let's print the third table and check its content
      first_launch_table = html_tables[2]
      print(first_launch_table)
```

```
[18]: column_names = []
      # Apply find_all() function with `th` element on first_launch_table
      # Iterate each th element and apply the provided extract_column_from_header() to get a column name
      # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
      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)
```

Check the extracted column names

```
[19]: print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
[20]: 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']=[]
```

```
[21]: 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
                  # TODO: Append the flight_number into launch_dict with key `Flight No.`
                  #print(flight_number)
                  datatimelist=date_time(row[0])

      ---------------
                  # Date value
```

Data Wrangling

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

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

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

Each launch aims to an dedicated orbit, and here are some common orbit types:

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

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

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

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

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

```
[13]: Outcome
      True ASDS      41
      None None      19
      True RTLS      14
      False ASDS      6
      True Ocean      5
      False Ocean     2
      None ASDS       2
      False RTLS      1
      Name: count, dtype: int64
```

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome` , create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome` ; otherwise, it's one. Then assign it to the variable `landing_class` :

```
[17]: # landing_class = 0 if bad_outcome
      # landing_class = 1 otherwise
      landing_class=[]
      for outcome in df['Outcome']:
          if outcome in bad_outcomes:
              landing_class.append(0)
          else:
              landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

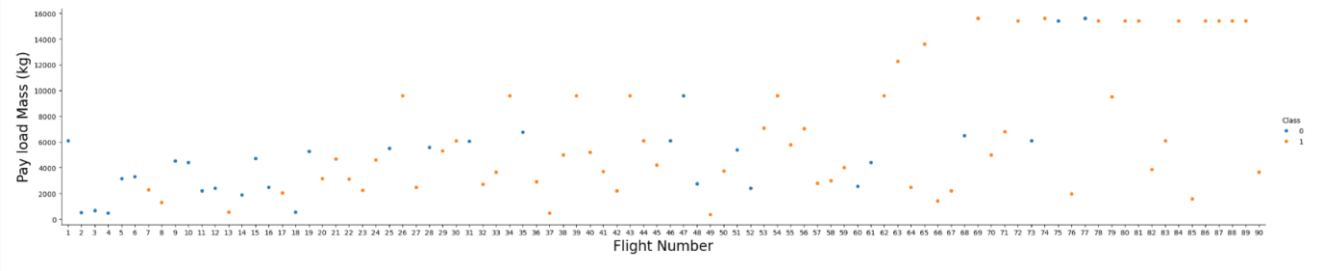
```
[18]: df['Class']=landing_class
      df[['Class']].head(8)
```

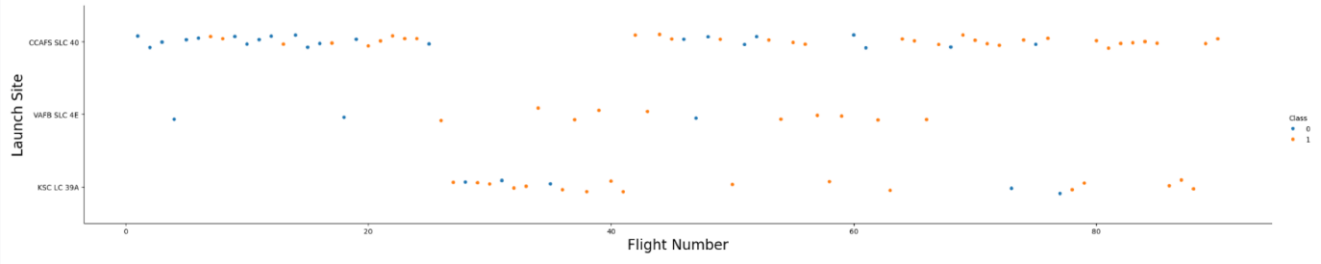
```
[20]: df.tail(5)
```

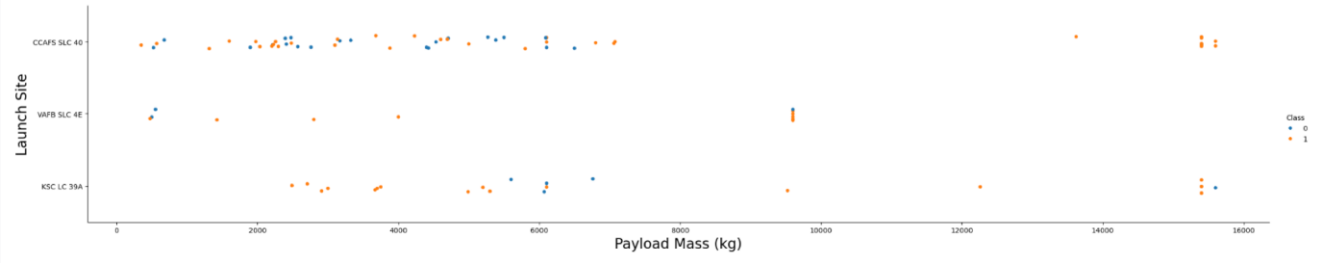
[20]:

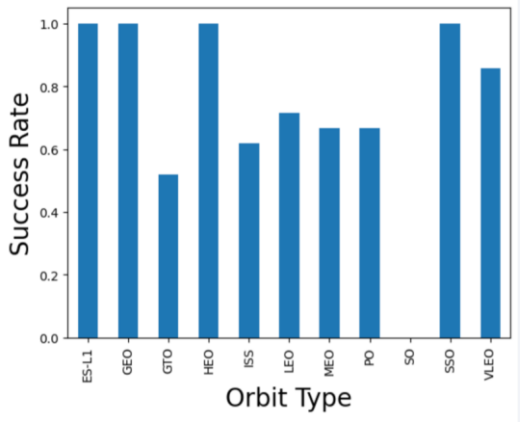
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	Re
85	86	2020-09-03	Falcon 9	15400.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	
86	87	2020-10-06	Falcon 9	15400.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	
87	88	2020-10-18	Falcon 9	15400.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	
88	89	2020-10-24	Falcon 9	15400.0	VLEO	CCAFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	
89	90	2020-11-05	Falcon 9	3681.0	MEO	CCAFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	

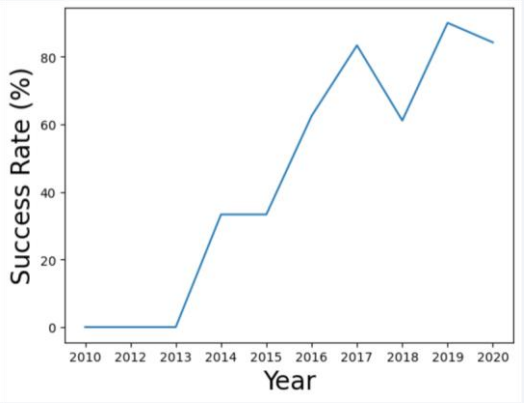
EDA with Data Visualization









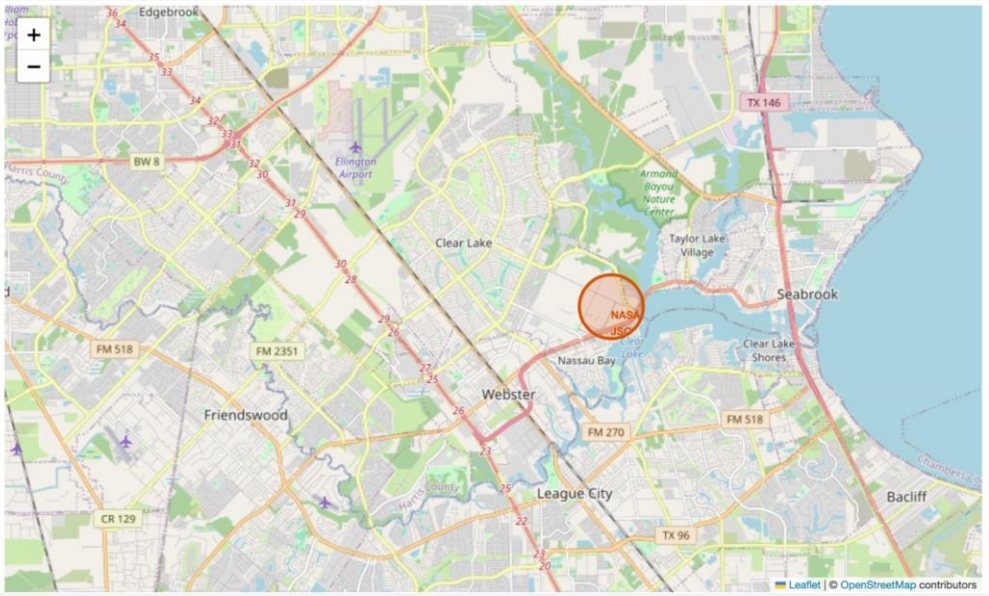


EDA with SQL

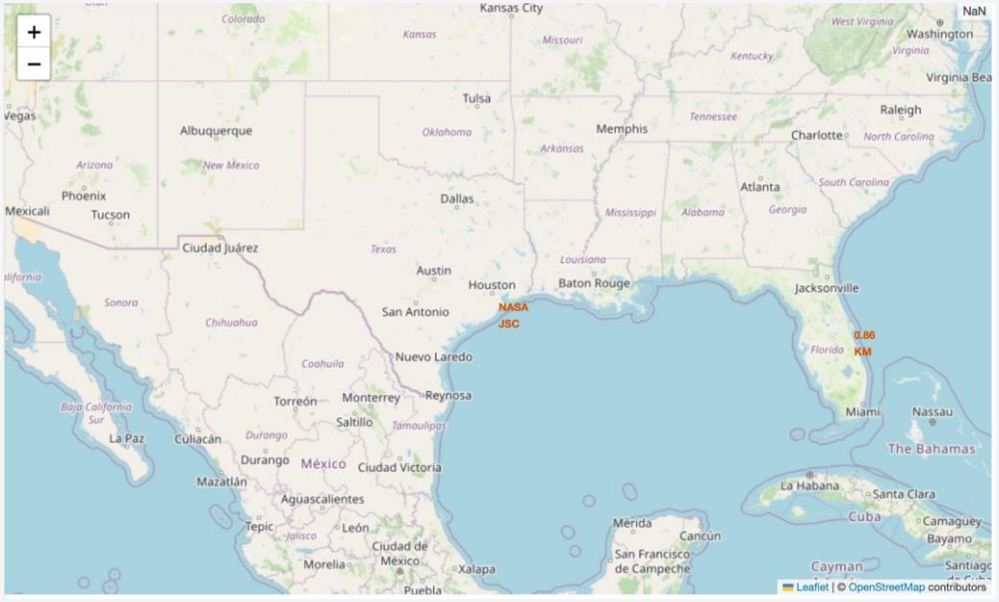
- Display the names of the unique launch sites in the space mission.
- Display 5 records where launch sites begin with the string 'CCA’
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass using 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.

Build an Interactive Map with Folium

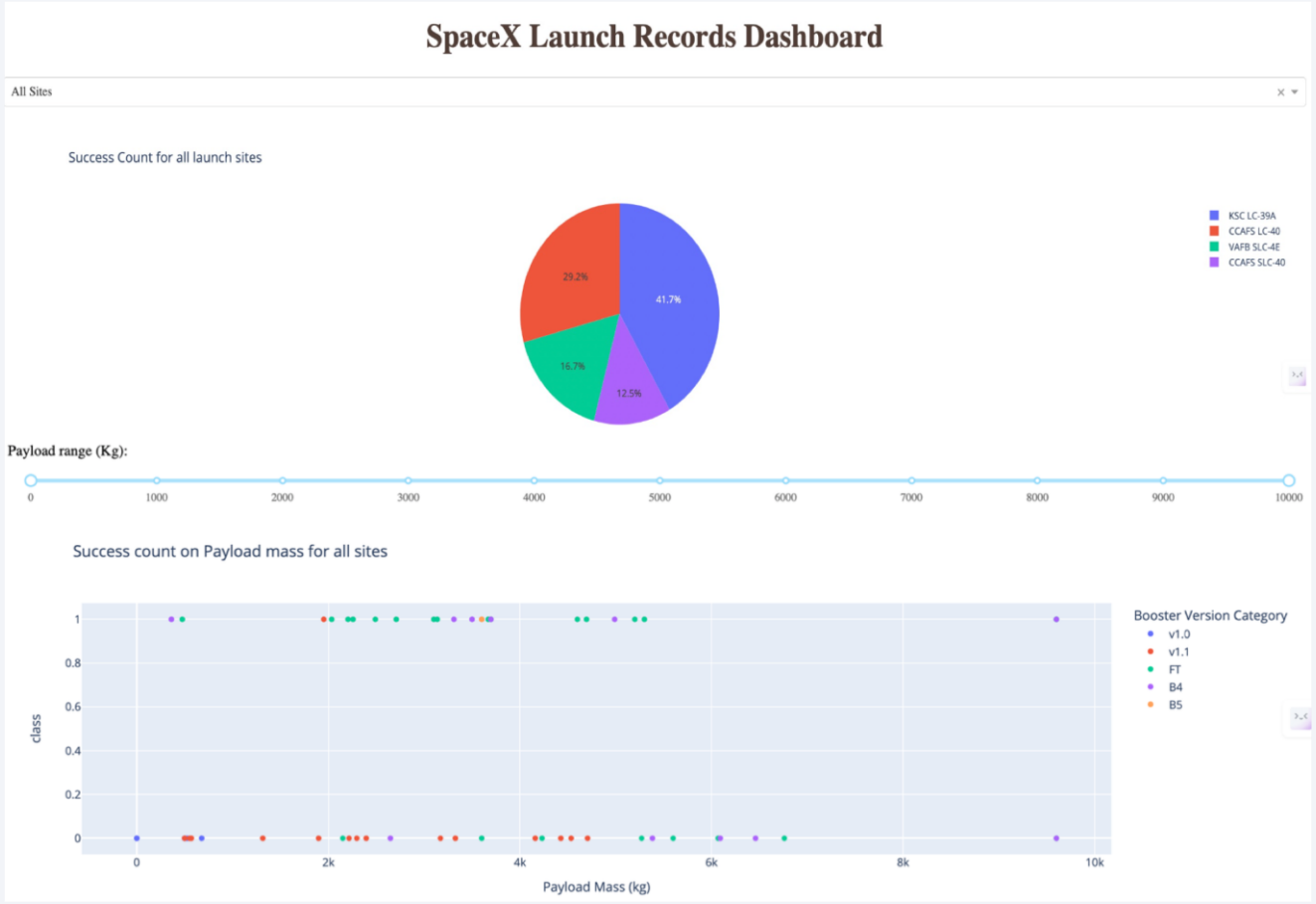
[7]:



[18]:



Build a Dashboard with Plotly Dash



Predictive Analysis (Classification)

1.Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN) models were created and optimized using GridSearchCV to determine the best hyperparameters. Following this, the models were trained on the training dataset.

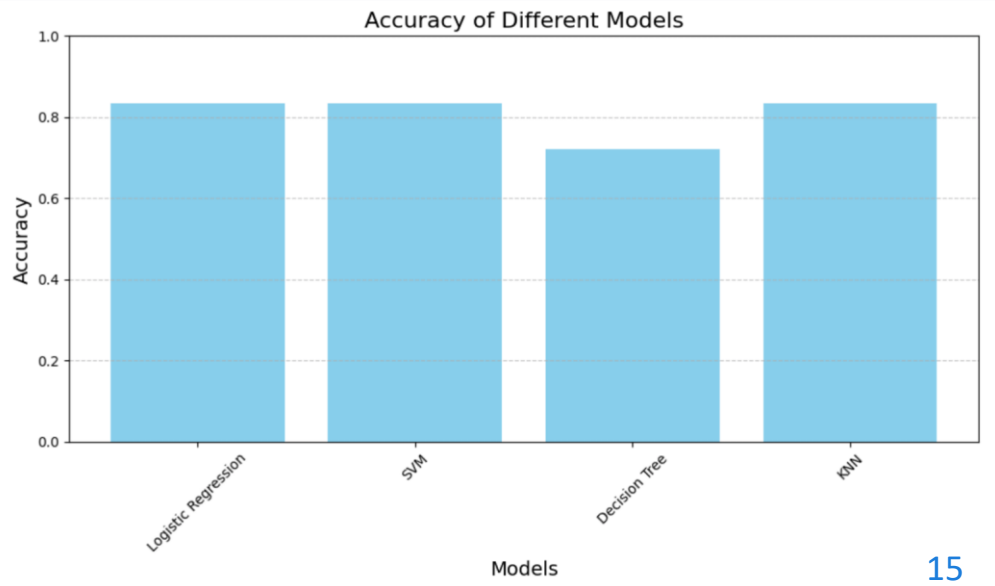
2.The accuracy on the test set was calculated for each machine learning model. It was observed that LR, SVM, and KNN achieved the highest performance, each reaching an accuracy of 83.33%.

TASK 12

Find the method performs best:

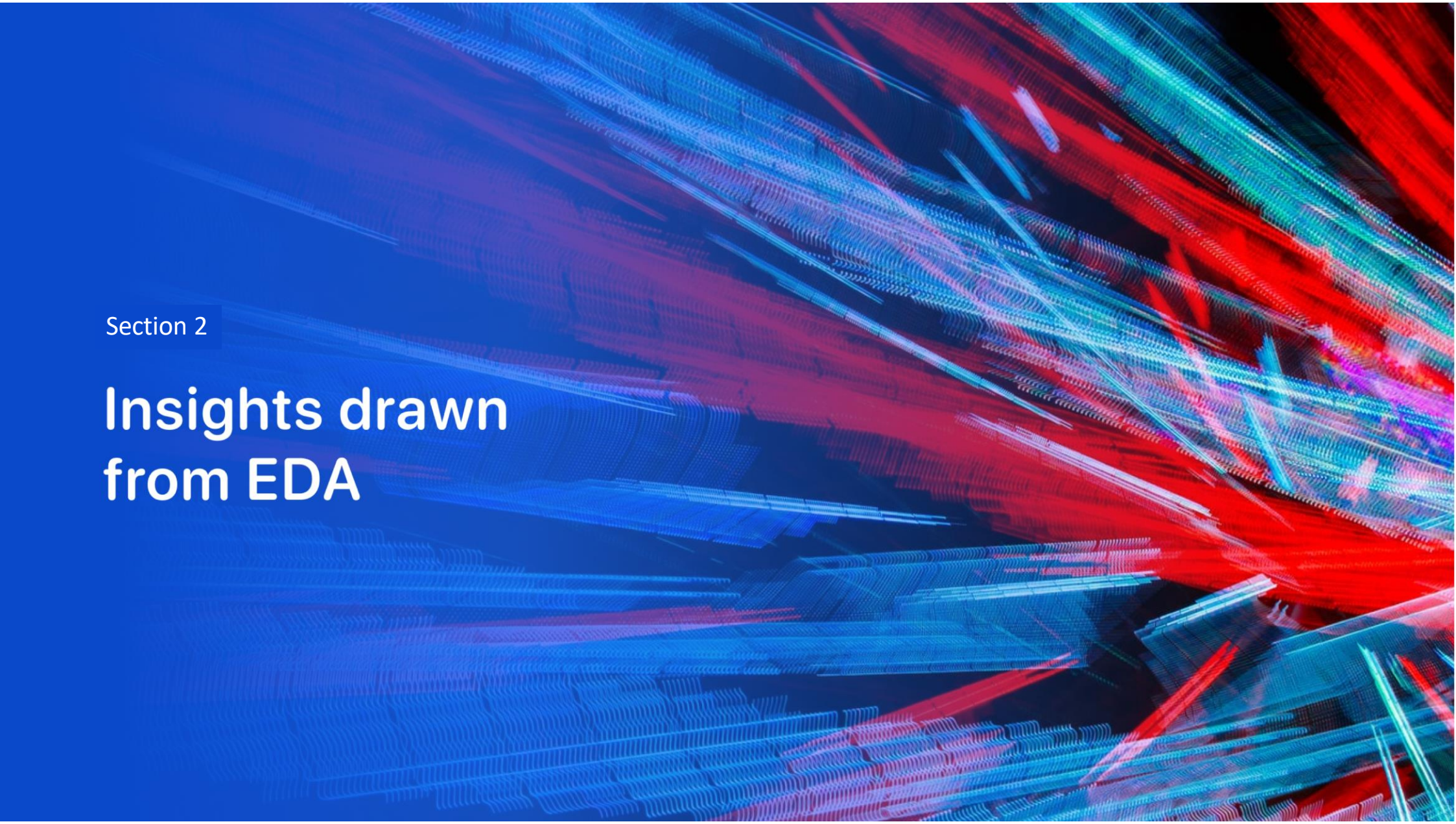
```
In [58]: print('LR Accuracy:', '{:.2%}'.format(logreg_accuracy))
         print( 'SVM Accuracy:', '{:.2%}'.format(svm_accuracy))
         print('Decision Tree Accuracy:', '{:.2%}'.format(tree_accuracy))
         print('KNN Accuracy:', '{:.2%}'.format(knn_accuracy))

LR Accuracy: 83.33%
SVM Accuracy: 83.33%
Decision Tree Accuracy: 72.22%
KNN Accuracy: 83.33%
```

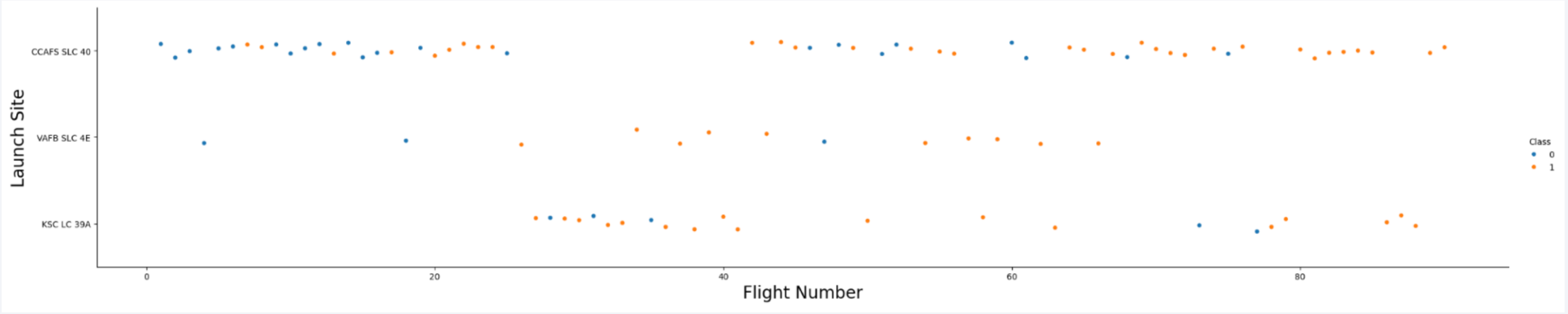


Results

- 1. Among the evaluated models, Logistic Regression (LR), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) demonstrated the highest predictive performance on this dataset.
- 2. Launches with lighter payloads tend to achieve better outcomes than those carrying heavier payloads.
- 3. The probability of a successful SpaceX launch increases with operational experience, indicating a trend toward increasingly reliable launches over time.
- 4. Kennedy Space Center's Launch Complex 39A has recorded the greatest number of successful launches among all the launch sites.
- 5. Orbit types such as GEO, HEO, SSO, and ES L1 are associated with the highest success rates in launch outcomes.

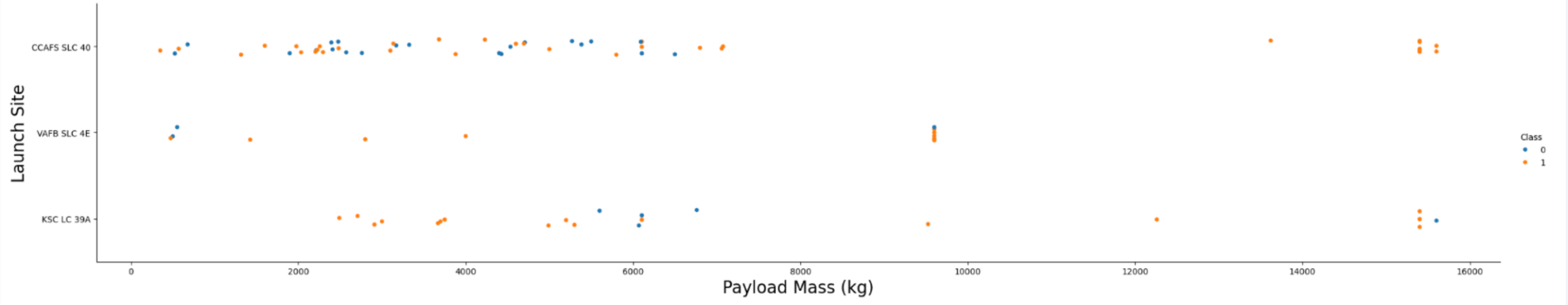

Section 2
Insights drawn
from EDA

Flight Number vs. Launch Site



The total number of launches from the CCAFS SLC 40 launch site is considerably higher than that of other launch locations.

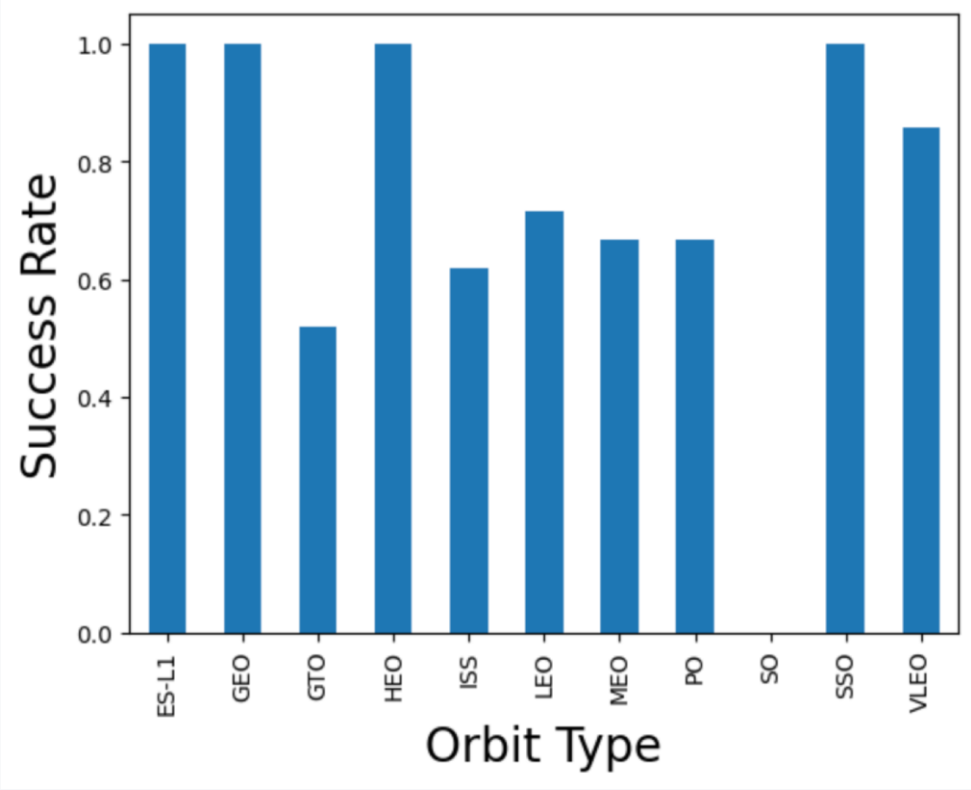
Payload vs. Launch Site



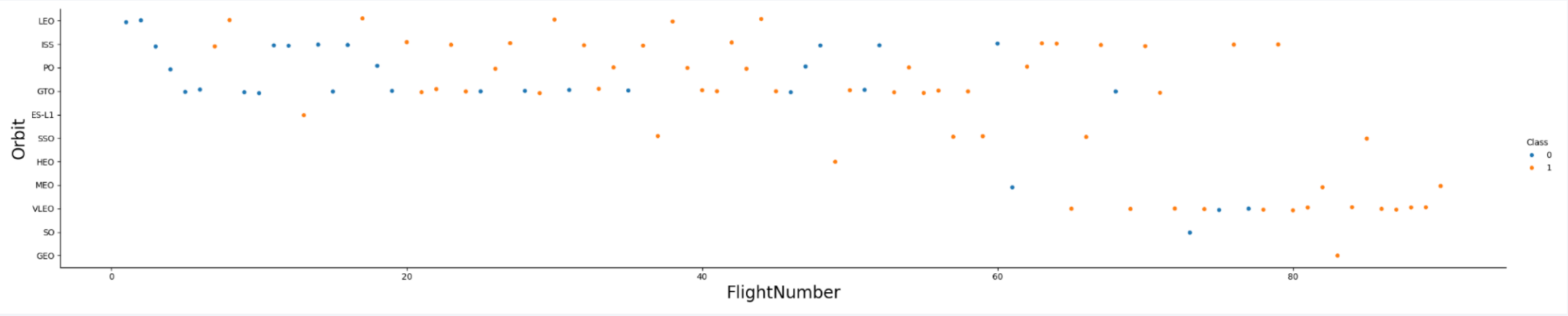
Across all three launch sites, payloads with lower mass have been launched more frequently than those with higher mass.

Success Rate vs. Orbit Type

The orbit types ES-L1, GEO, HEO, and SSO exhibit the highest success rates among all orbit categories.

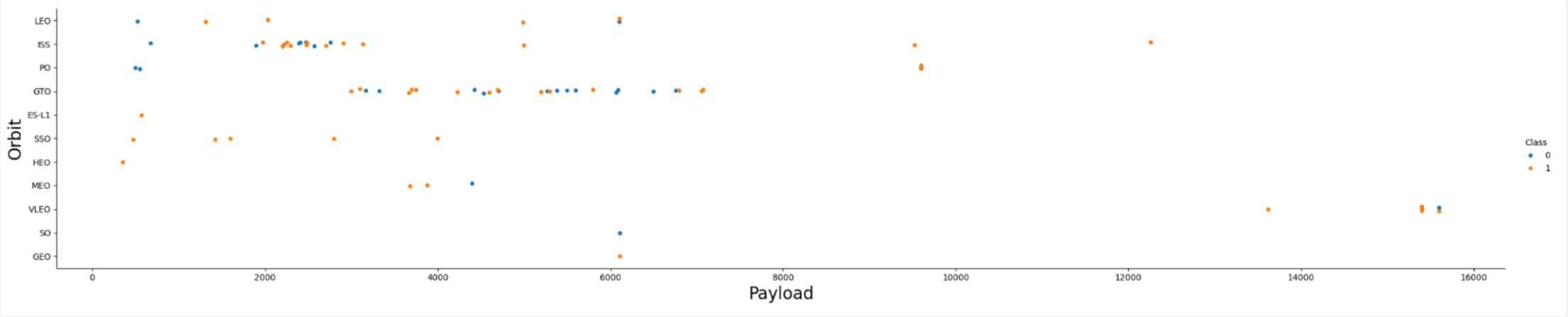


Flight Number vs. Orbit Type



In the earlier years, the majority of launches targeted LEO, ISS, PO, and GTO orbits, but over time, there has been a gradual shift toward VLEO orbit.

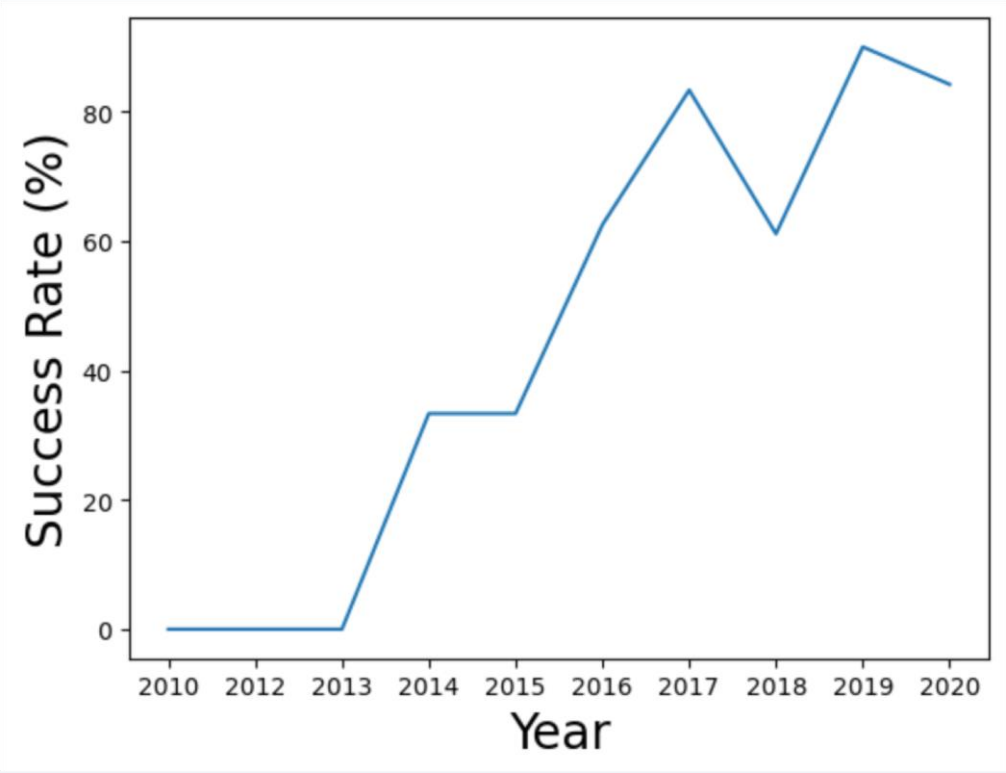
Payload vs. Orbit Type



Heavier payloads generally show higher success rates for landings in LEO and ISS orbits; however, for GTO orbits, the landing outcomes are more uncertain, with successes and failures occurring at nearly equal rates.

Launch Success Yearly Trend

Since 2013, the launch success rate has shown a steady upward trend through 2020, likely reflecting technological progress and accumulated experience.



All Launch Site Names

Executed an SQL query to retrieve all the names of the launch sites.

Task 1

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

```
[18]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

[18]:

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

Launch Site Names Begin with 'CCA'

Ran an SQL query to fetch five launch site names starting with 'CCA'.

Task 2

Display 5 records where launch sites begin with the string 'CCA'

```
[30]: %sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

[30]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (
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 (
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Total Payload Mass

Task 3

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

```
[29]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD FROM SPACEXTBL WHERE Customer LIKE '%CRS%';
 * sqlite:///my_data1.db
Done.
```

[29]:

TOTAL_PAYLOAD
48213

Used an SQL query to extract the total payload mass from boosters launched by NASA (CRS).

Average Payload Mass by F9 v1.1

Task 4

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

```
[33]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD FROM SPACEXTBL WHERE Booster_Version LIKE '%F9 v1.1';
 * sqlite:///my_data1.db
Done.
[33]: AVERAGE_PAYLOAD
      2928.4
```

Executed an SQL command to compute the average payload mass for booster version F9 v1.1.

First Successful Ground Landing Date

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
[38]: %sql SELECT MIN(DATE) FROM SPACEXTBL WHERE Landing_Outcome="Success (ground pad)"
```

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

[38]:

MIN(DATE)
2015-12-22

Queried the database with SQL to identify the dates of the first successful landings on a ground pad.

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

```
[44]: %sql SELECT DISTINCT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome="Success (drone ship)" AND PAYLOAD_MASS__KG_
 * sqlite:///my_data1.db
Done.
```

[44]:

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

Ran an SQL query to list booster names that landed successfully on a drone ship and carried payloads between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

[52]:
```
%sql SELECT Mission_Outcome, COUNT(*) AS QTY FROM SPACEXTBL GROUP BY Mission_Outcome ORDER BY QTY DESC
```

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

[52]:

Mission_Outcome	QTY
Success	98
Success (payload status unclear)	1
Success	1
Failure (in flight)	1

Used SQL to determine the total counts of missions that were successful versus those that failed.

Boosters Carried Maximum Payload

Queried with SQL to find the booster name associated with the highest payload mass.

Task 8

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

```
[61]: %sql SELECT DISTINCT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_=(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXT
 * sqlite:///my_data1.db
Done.
```

[61]:

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

Task 9

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[66]:
```
%sql SELECT substr(Date, 6,2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE Landing_Outc
```

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

[66]:

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

Performed an SQL query to list failed drone ship landings in 2015, including booster versions and launch site names.

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

Task 10

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.

[71]:
```
%sql SELECT Landing_Outcome, COUNT(*) AS COUNT_OF_LANDING FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-2
```

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

[71]:

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

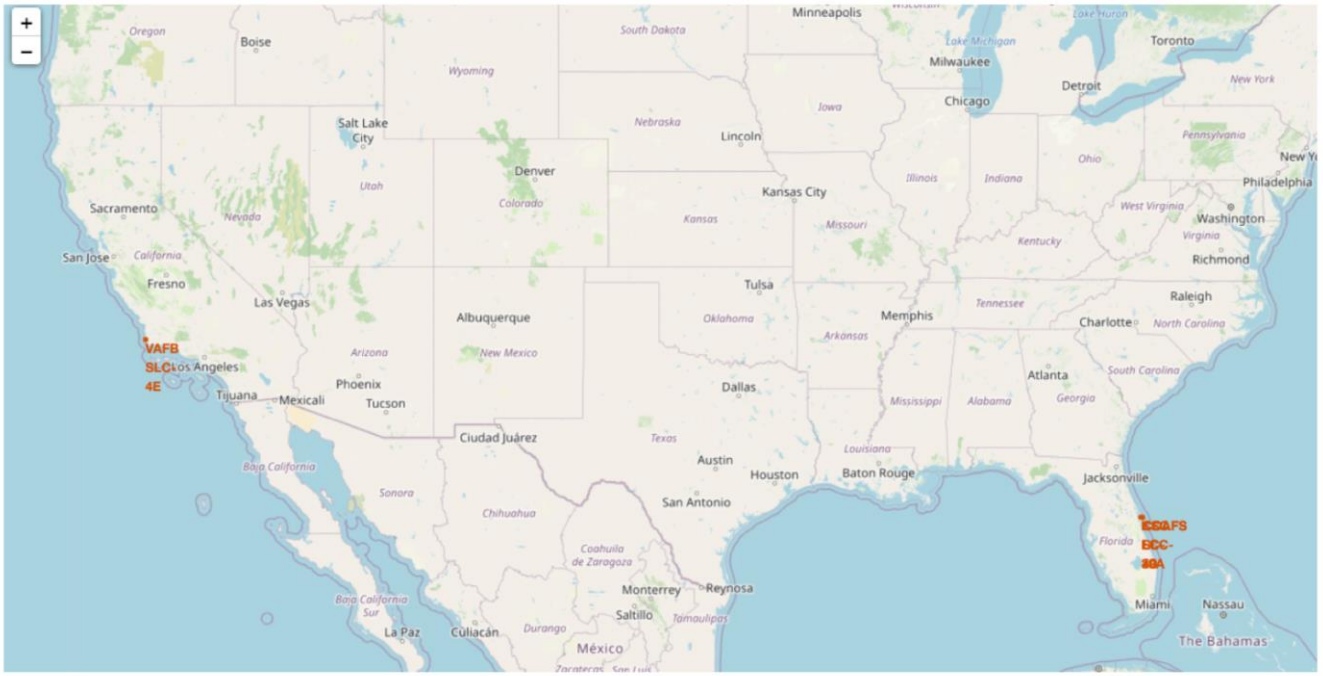
Ran an SQL query to rank the number of landing outcomes (e.g., Failure on drone ship, Success on ground pad) between 2010-06-04 and 2017-03-20 in descending order.


Section 3
Launch Sites
Proximities Analysis

All launch sites on a map

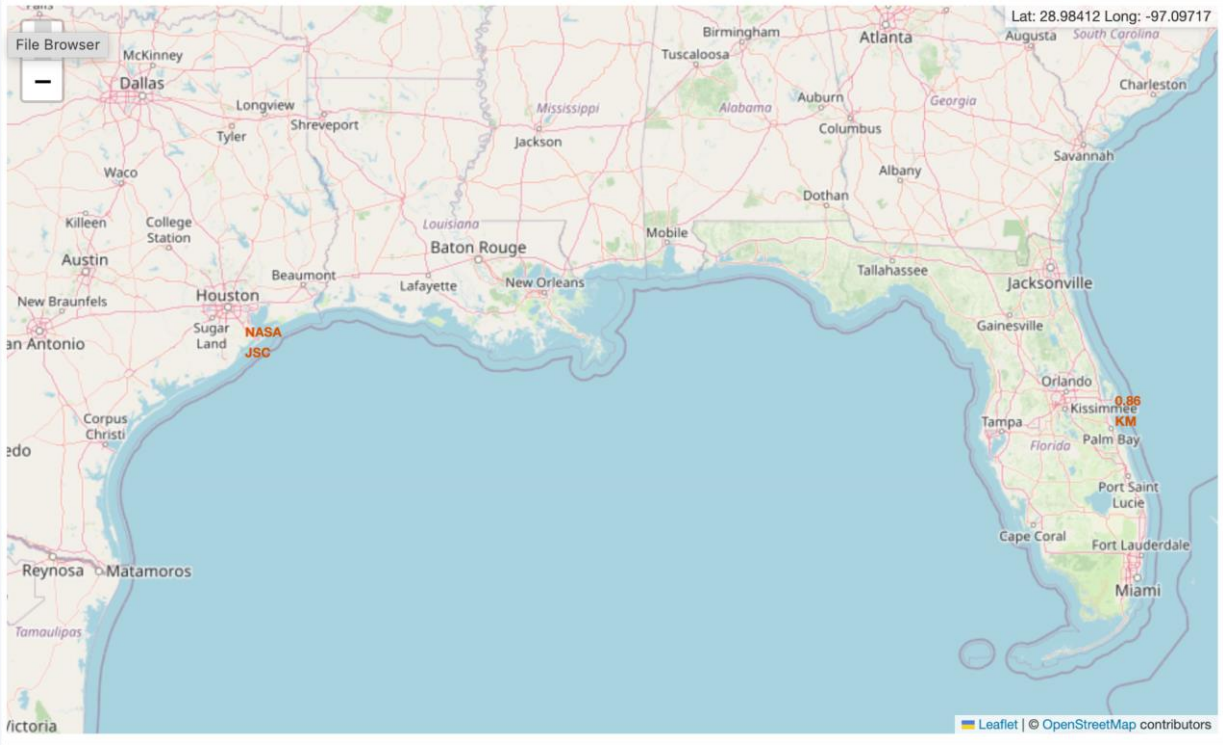
The launch site locations are indicated on the map using markers that display their respective names.

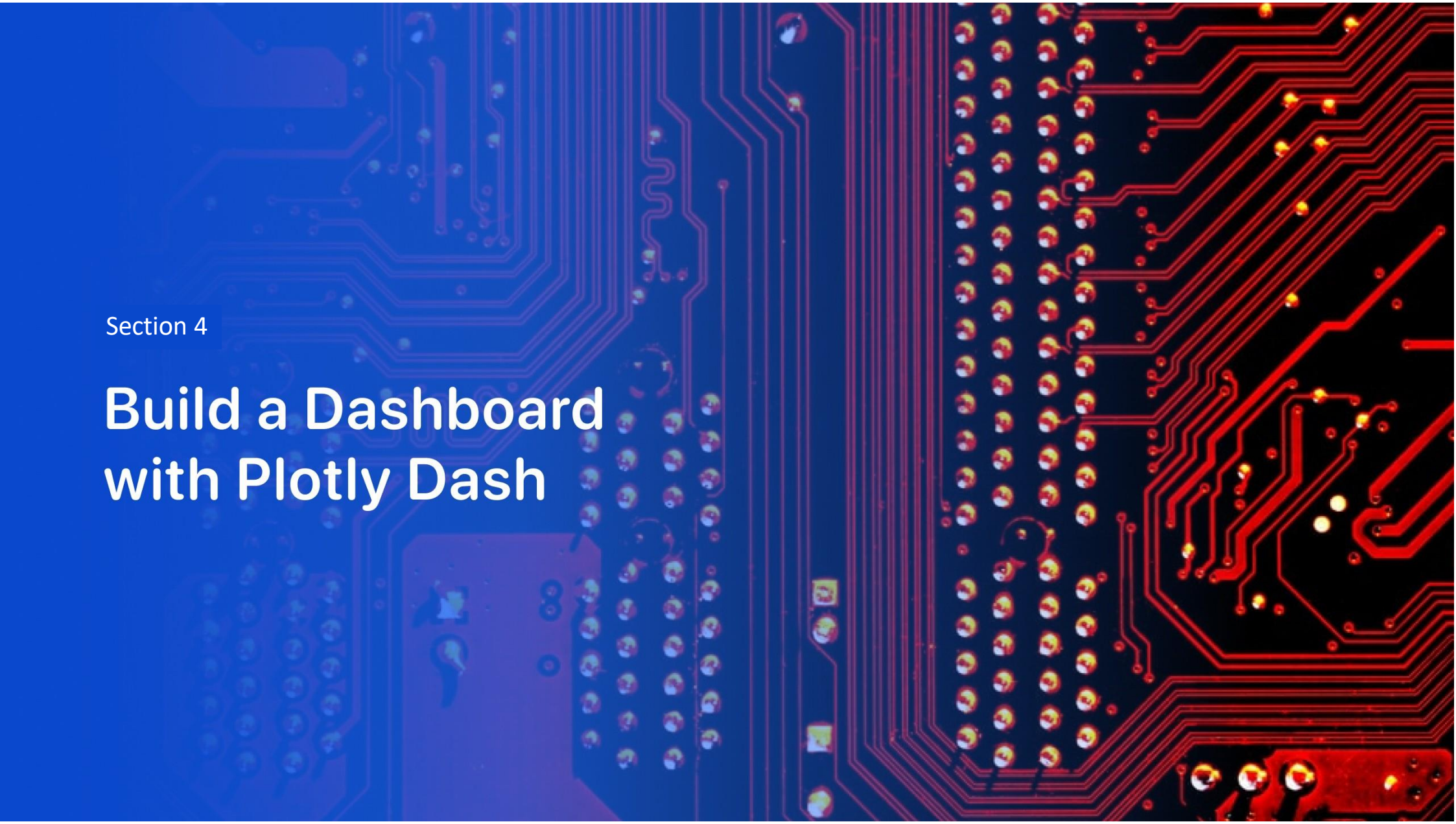
The generated map with marked launch sites should look similar to the following:



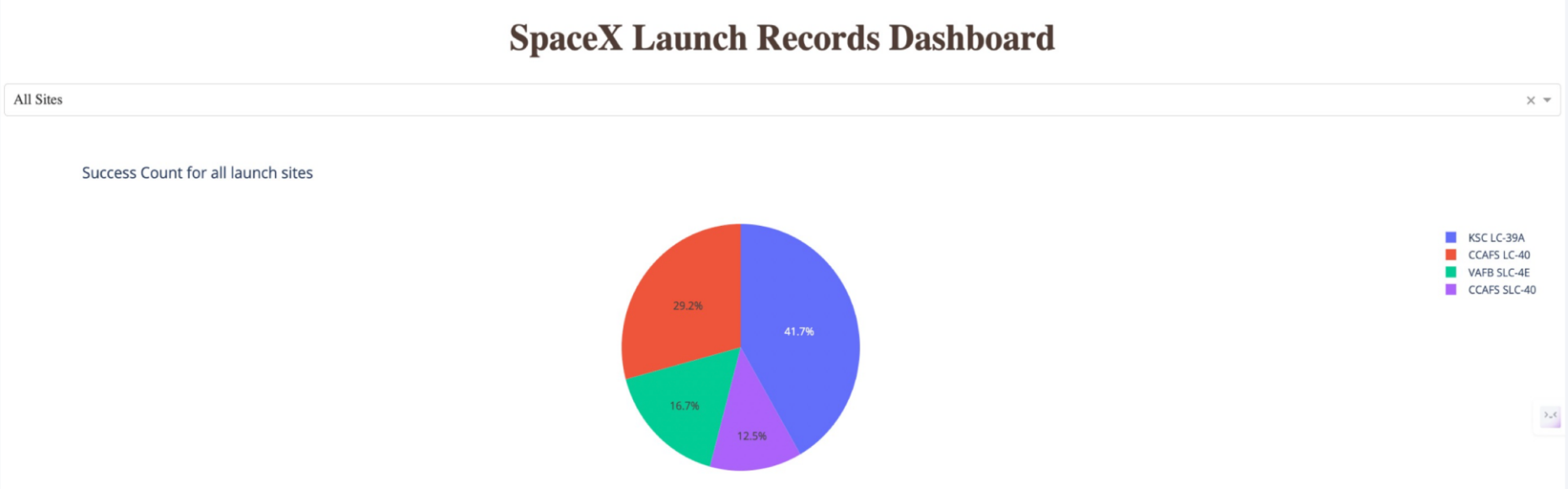
Distances between a launch site to its proximities

The nearest coastline to NASA JSC is identified as a point using the MousePosition tool, and the distance between this point and the launch site is calculated to be around 0.86 km.



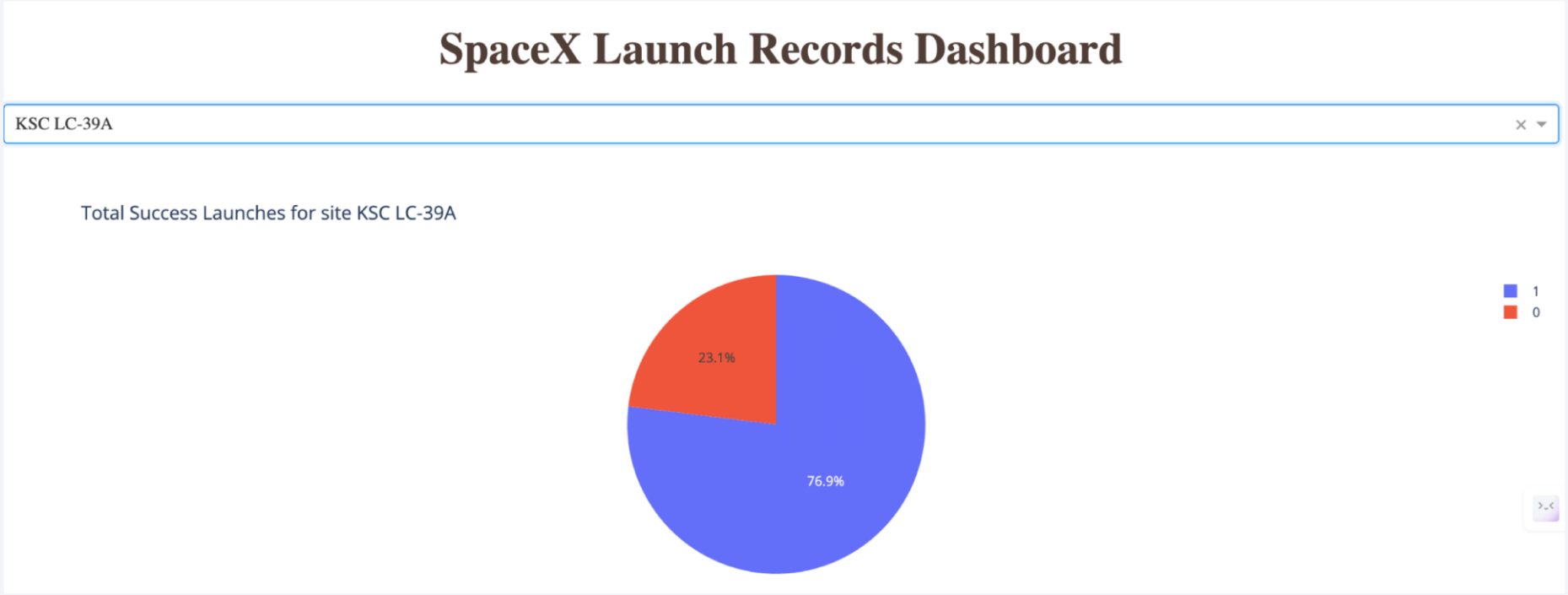

Section 4
Build a Dashboard with Plotly Dash

Total success launches for all sites



Among all recorded launches, KSC LC-39A accounts for the highest proportion of successful launches at 41.7%, while CCAFS SLC-40 has the lowest with just 12.5%.

Success ratio with the highest success launches



KSC LC-39A, the launch site with the greatest number of successful missions, has a success rate of 76.9% and a failure rate of 23.1% for launches conducted from this site.

Payload vs. launch outcome

The most successful launches are concentrated in the 2,000–4,000 kg payload range, with the 4,000–6,000 kg range following closely behind. Among booster versions, FT shows the highest number of successful launches.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

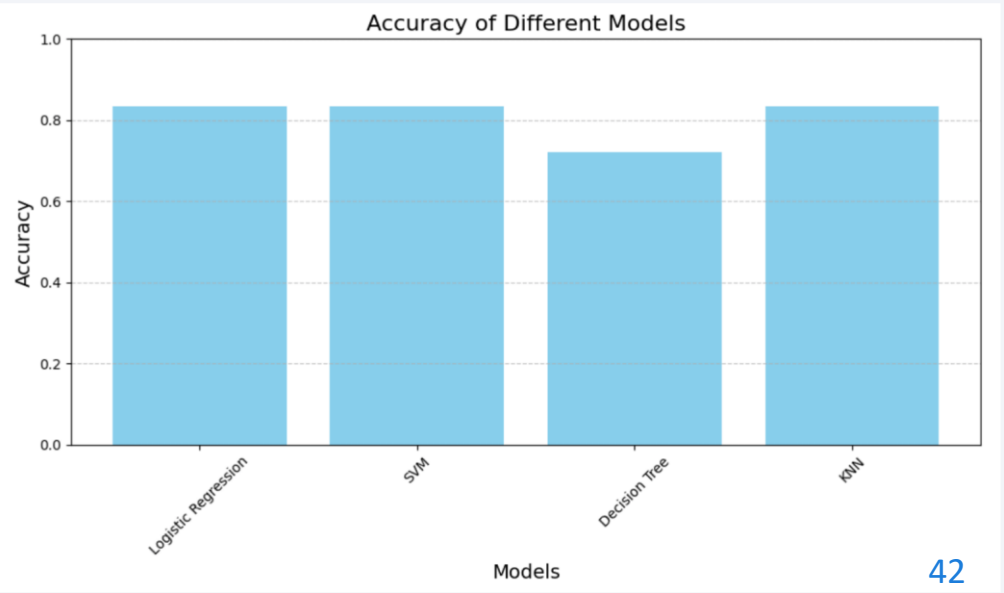
- LR, SVM, and KNN models achieved the highest accuracy of 83.33%, making them the top-performing models.

TASK 12

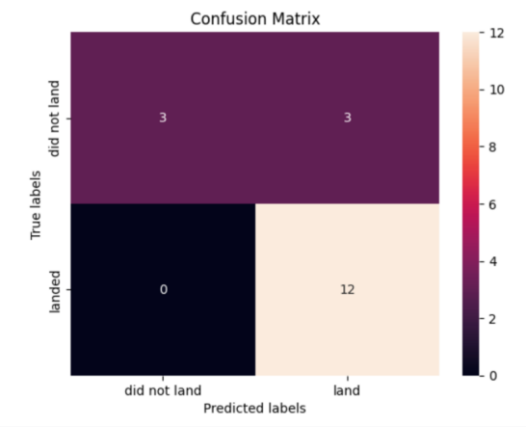
Find the method performs best:

```
In [58]: print('LR Accuracy:', '{:.2%}'.format(logreg_accuracy))
         print( 'SVM Accuracy:', '{:.2%}'.format(svm_accuracy))
         print('Decision Tree Accuracy:', '{:.2%}'.format(tree_accuracy))
         print('KNN Accuracy:', '{:.2%}'.format(knn_accuracy))

LR Accuracy: 83.33%
SVM Accuracy: 83.33%
Decision Tree Accuracy: 72.22%
KNN Accuracy: 83.33%
```

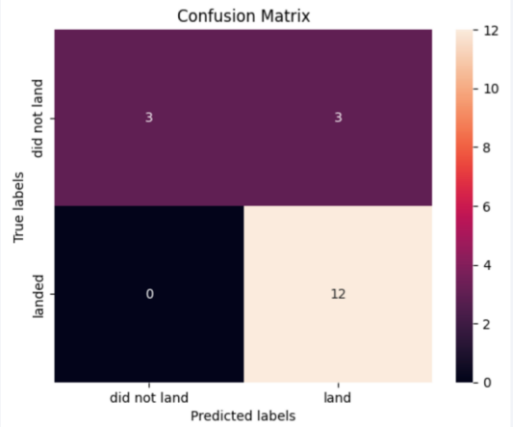


Confusion Matrix



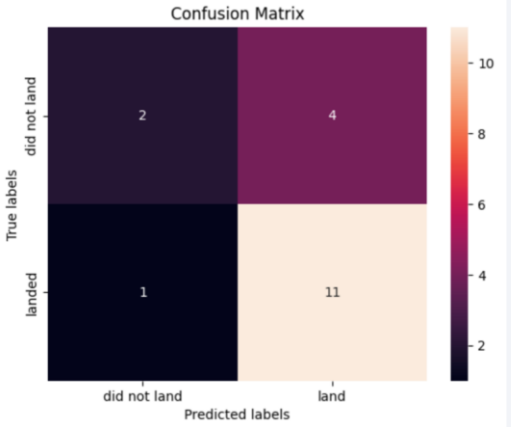
LR Test Accuracy:

0.83



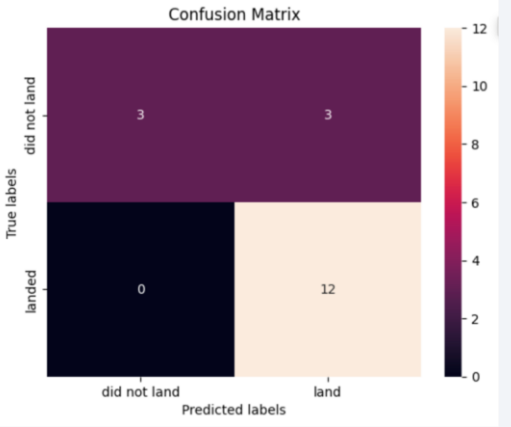
SVM Test Accuracy:

0.83



DT Test Accuracy:

0.72



KNN Test Accuracy:

0.83

Conclusions

- LR, SVM, and KNN models show the best performance in predicting outcomes within this dataset.
- The orbit types GEO, HEO, SSO, and ES L1 demonstrate the highest success rates among all orbit categories.
- Launches carrying lighter payloads tend to result in better performance than those with heavier payloads.
- The probability of a successful SpaceX launch improves with increased years of experience, indicating a trend toward more reliable launches over time.
- Kennedy Space Center’s Launch Complex 39A leads in the number of successful launches when compared to other launch locations.

Thank you!