IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Joeky Zhou
July 28, 2023

Outline

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

Executive Summary

Our project provides valuable insights and predictive analysis for stakeholders aiming to compete with SpaceX in the space industry. We leverage data collection, data wrangling, exploratory data analysis (EDA), interactive visual analytics, and machine learning to understand factors influencing rocket launch success and cost efficiency.

Key Objectives:

1. Data Collection: We curate extensive rocket launch data, including launch site, payload mass, booster version, success/failure outcomes, and costs, forming the foundation for our analysis.
2. Data Wrangling: Ensuring data quality and consistency, we meticulously clean and transform raw data for further exploration.
3. Exploratory Data Analysis (EDA): By applying statistical techniques and visualization tools, we uncover patterns and correlations in the dataset, providing insights into launch success and cost-effectiveness.
4. Interactive Visual Analytics: Our interactive visualizations allow stakeholders to intuitively explore the data, gaining real-time insights for informed decision-making. These visuals include dashboards, scatter plots, pie charts, and so much more.
5. Predictive Analysis from Machine Learning: Utilizing machine learning algorithms, we build predictive models to forecast rocket launch success, enabling data-driven decisions on launch feasibility and cost optimization.

Key Findings:

1. Launch Site Analysis: The choice of launch site significantly influences success rates, with some sites demonstrating higher reliability for optimal launch site selection.
2. Payload Mass Impact: Payload mass greatly affects launch success, revealing critical thresholds for efficient cost management.
3. Booster Version Effect: Different booster versions exhibit varying success rates, influencing mission success probabilities.
4. Cost Optimization: Our predictive models help stakeholders estimate launch costs based on multiple parameters, enabling cost-efficient and competitive rocket launches.

Introduction

In this project, our goal is to predict the successful landing of the Falcon 9 first stage. SpaceX offers Falcon 9 rocket launches at a significantly lower cost of 62 million dollars compared to other providers, whose costs exceed 165 million dollars per launch. The key to these cost savings lies in SpaceX's ability to reuse the first stage of the rocket. By accurately determining whether the first stage will land successfully, we can estimate the cost of a launch. This information becomes valuable for other companies looking to compete with SpaceX in bidding for rocket launches. Throughout this presentation, we will provide you with an overview of the project and equip you with the necessary tools to address it effectively.

The project presents a set of challenges encompassing data collection, data cleaning, and data forecasting for visual representation in various charts for thorough analysis. The ultimate objective is to identify and employ appropriate predictive analysis methods that would render the information highly valuable for SpaceX stakeholders and potential competitors.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Our data collection methodology involved using API collection tools, web scraping, and Python libraries such as Beautiful Soup, requests, pandas, and NumPy. This powerful combination allowed efficient data extraction from various online sources, including websites and APIs. We manipulated and organized the data using pandas and performed complex calculations with NumPy. This comprehensive approach ensured a robust and automated data collection process, providing a high-quality dataset for subsequent analysis.
- Perform data wrangling
  - In this project, we performed meticulous data wrangling to ensure data quality and consistency. We collected data through APIs and web scraping, then cleaned, transformed, and integrated it into a standardized format. Feature engineering, outlier handling, and validation checks were conducted to enhance the dataset's analytical value. The resulting high-quality dataset laid the foundation for exploratory data analysis, interactive visual analytics, and predictive analysis.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - We fine-tuned and evaluated our classification models by training and testing using machine learning algorithms like classification trees, SVM, and logistic regression.

Data Collection – SpaceX API

- GitHub URL: https://github.com/joekyzhou/SpaceXFinalProject/blob/main/IBM_DataCollection.ipynb

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [12]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [13]: response = requests.get(spacex_url)
```

Check the content of the response

```
In [14]: print(response.content)
```


b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17T00:00:00.000Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":33,"altitude":null,"reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and loss of vehicle","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flight_number":1,"name":"FalconSat","date_utc":"2006-03-24T22:30:00.000Z","date_unix":1143239400,"date_local":"2006-03-25T10:30:00+12:00","date_precision":"hour","upcoming":false,"cores":[{"core":"5e9e289df35918033d3b2623","flight":1,"gridfins":false,"legs":false,"reused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_update":true,"tbd":false,"launch_library_id":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/f9/4a/ZboXReNb_o.png","large":"https://images2.imgbox.com/80/a2/bkWotCIS_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=Lk4zQ2wP-Nc","youtube_id":"Lk4zQ2wP-Nc","article":"https://www.space.com/3590-spacex-falcon-1-rocket-fails-reach-orbit.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":null,"static_fire_date_unix":null,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":301,"altitude":289,"reason":"harmonic oscillation leading to premature engine shutdown"}],"details":"Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

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:

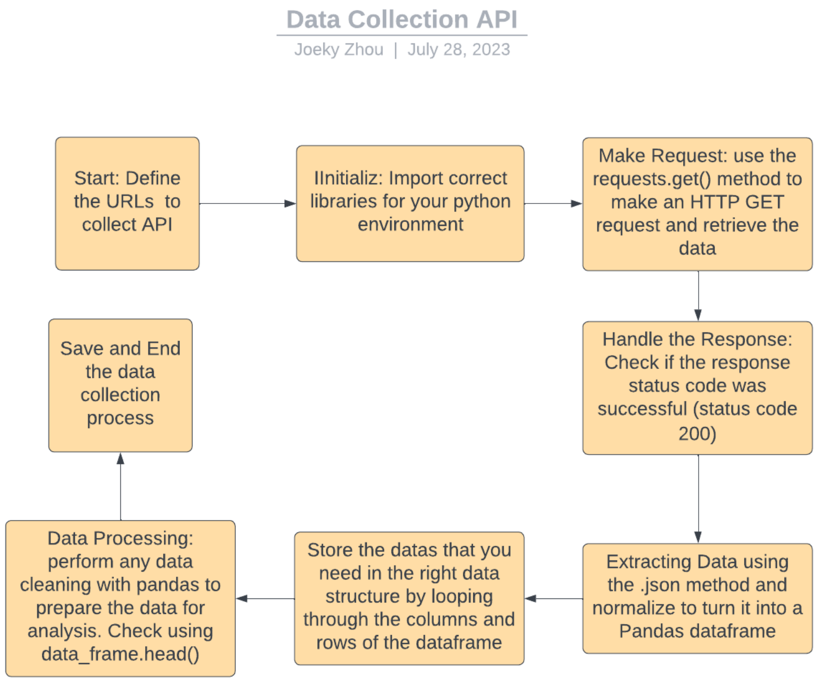
```
In [15]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_
```

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

```
In [16]: response.status_code
Out[16]: 200
```

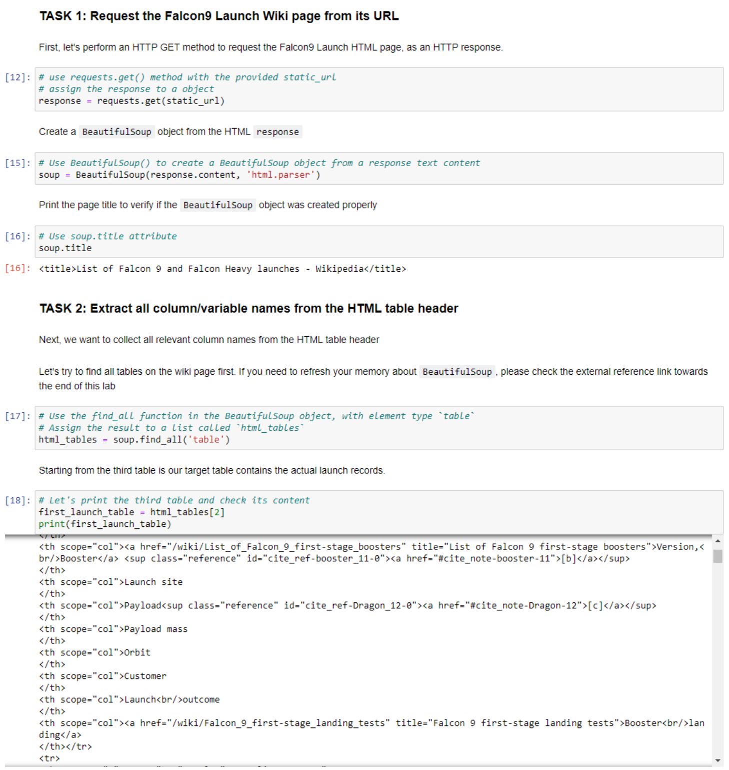
Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
In [17]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```



Data Collection – Scraping

- GitHub URL: https://github.com/joekyzhou/SpaceXFinalProject/blob/main/IBM_Webscraping.ipynb



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
      response = requests.get(static_url)
```

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

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

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

```
[16]: # Use soup.title attribute
      soup.title

[16]: <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

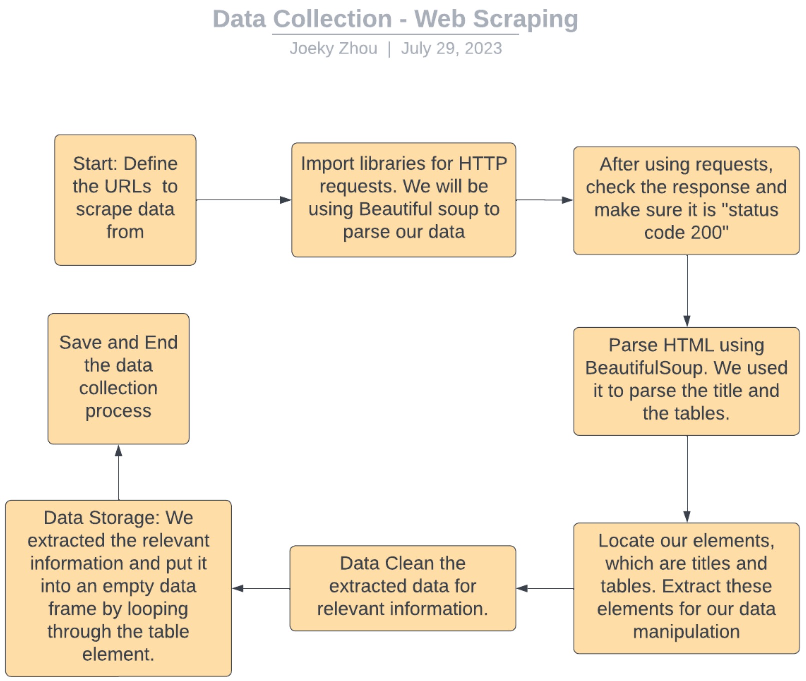
Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
[17]: # 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.

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

<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<
br/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#cite_note-Dragon-12">[c]</a></sup>
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
</th>
<th scope="col">Customer
</th>
<th scope="col">Launch<br/>outcome
</th>
<th scope="col"><a href="/wiki/Falcon_9_first-stage_landing_tests" title="Falcon 9 first-stage landing tests">Booster<br/>lan
ding</a>
</th></tr>
<tr>
```



Data Collection – Data Wrangling

- GitHub URL: https://github.com/joekyzhou/SpaceXFinalProject/blob/main/IBM_DataWrangling.ipynb

Data Analysis

```
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv'
resp = await fetch(URL)
dataset_part_1_csv = io.BytesIO((await resp.arrayBuffer()).to_py())
```

Load Space X dataset, from last section.

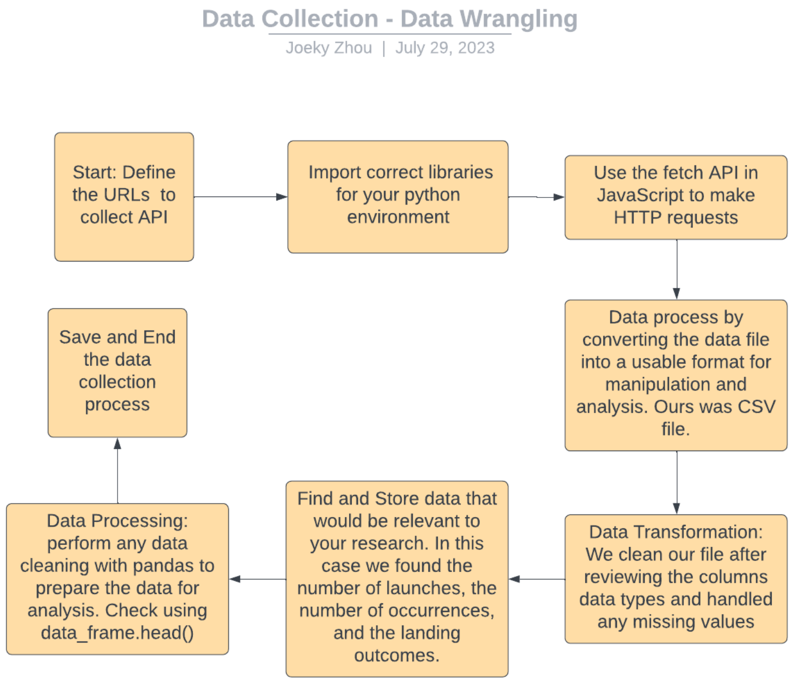
```
df=pd.read_csv(dataset_part_1_csv)
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011

Identify and calculate the percentage of the missing values in each attribute

```
df.isnull().sum()/df.shape[0]*100
```

```
FlightNumber       0.000000
Date               0.000000
BoosterVersion     0.000000
PayloadMass        0.000000
Orbit              0.000000
LaunchSite         0.000000
Outcome            0.000000
Flights            0.000000
GridFins           0.000000
Reused             0.000000
Legs               0.000000
LandingPad        28.888889
Block              0.000000
ReusedCount        0.000000
Serial             0.000000
Longitude          0.000000
```



EDA with Data Visualization

- In the EDA Visualization for SpaceX Falcon 9 First Stage Landing Prediction, catplot, scatterplot, line plot, and bar plot were used to explore and analyze the data after fetching it from a URL. These plots provided valuable insights into launch outcomes, correlations between payload mass and landing success, trends over time, and comparisons of success rates across different launch sites and booster versions. The visualizations supported data-driven decision-making and predictive analysis for first-stage landing prediction.
- Git Hub URL: https://github.com/joekyzhou/SpaceXFinalProject/blob/main/IBM_EDA_Visualize.ipynb

EDA with SQL

- Connect to the data base and load data into a data frame using these libraries:

Import csv, sqlite3, pandas

- Display names of unique launch sites in the space mission using %SQL SELECT DISTINCT
- Display the 5 launch sites record beginning with ‘CCA’ using LIKE ‘CCA%’ LIMIT 5
- Display total payload mass carried by boosters launch by NASA (CRS) using %sql SELECT SUM(PAYLOAD_MASS__KG_), Customer FROM SPACEXTBL WHERE Customer == ‘NASA (CRS)’
- Display the average payload mass carried by F9 v1.1 using AVG(PAYLOAD_MASS__KG_) as well as WHERE BOOSTER_VERSION == ‘F9 v1.1’ in our sql magic code
- List the dates when first success outcomes were achieved by filtering using Landing_Outcome == ‘Success’ in sql magic code
- List the names of the successful boosters with payload mass greater than 4000 and less than 6000 we added these requirements with their respective columns after the WHERE statement in sql magic code
- List the total number of successful vs failure mission outcomes we had to use COUNT(MISSION_Outcome) and GroupBy in the WHERE statement to see both results.
- List the names of the booster versions that had maximum payload mass we used a subquery like this: %sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
- List the records with their months and year displayed along with their failure outcomes and launch sites we used: %sql SELECT substr(Date, 4, 2) as 'Month', substr(Date, 7,4) as 'Year', Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE Landing_Outcome == 'Failure (drone ship)’
- Ranking the count of landing outcomes between certain dates we had to format the dates to make it work like this: %sql SELECT SUBSTRING(Date, 7, 4) || '-' || SUBSTRING(Date, 4, 2) || '-' || SUBSTRING(Date, 1, 2) AS formatted_date, Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTBL WHERE SUBSTRING(Date, 7, 4) || '-' || SUBSTRING(Date, 4, 2) || '-' || SUBSTRING(Date, 1, 2) BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY formatted_date, Landing_Outcome ORDER BY formatted_date DESC

- Git Hub URL: https://github.com/joekyzhou/SpaceXFinalProject/blob/main/IBM_EDA_SQL.ipynb

Build an Interactive Map with Folium

- In the SpaceX Falcon 9 First Stage Landing Prediction project, I leveraged Folium, a Python library for interactive mapping, to visualize the different launch sites. Using Folium's features, I created markers to display the exact locations of the launch sites on the map. Additionally, lines were drawn to illustrate the distances of these sites from the ocean shore, closest city, and railroad stations, enhancing the map's informational value.
- To further enhance the visualization, I used different marker colors to indicate successful and failed launches at each site. This allowed for a quick overview of the success rates at different locations.
- A mouse marker was added to provide real-time information as I hovered over the map, providing additional details or insights as needed.
- For better visualization of each launch site, a Folium circle was created, which acted as a popup on the map. This feature allowed for a clearer representation of the specific site's location and its relevant information.
- Overall, the use of Folium in conjunction with data fetched using the `fetch` function enriched the analysis by providing an interactive and informative map. This allowed for a comprehensive understanding of the distribution of launch sites, their proximity to key landmarks, and the outcomes of Falcon 9 launches at each location.

- Git Hub URL: https://github.com/joekyzhou/SpaceXFinalProject/blob/main/IBM_Folium.ipynb

Build a Dashboard with Plotly Dash

- In the Plotly Dash project for SpaceX Falcon 9 First Stage Landing Prediction, I implemented a dropdown menu to select launch sites. Based on the selected launch site, I created two visualizations: a pie chart and a scatter plot.
- The pie chart illustrated the distribution of successful and failed landing outcomes for the chosen launch site, presenting a clear overview of success rates.
- The scatter plot showcased the relationship between payload mass and launch success for the selected launch site. This allowed for a better understanding of how payload mass influenced landing outcomes.
- By combining the dropdown menu, pie chart, and scatter plot in the Plotly Dash dashboard, users could dynamically explore and analyze the data for different launch sites, making informed decisions and gaining valuable insights into Falcon 9 first-stage landing predictions.
- Git Hub URL: https://github.com/joekyzhou/SpaceXFinalProject/blob/main/PlotlyDash.py

Predictive Analysis (Classification)

- The first step to doing any predictive analysis is to import the relevant libraries and collect the relevant data on Falcon 9 first-stage landings. We would then preprocess the data for any missing values and clean them up for analysis. After that we may select a machine learning model for testing. A confusion matrix function was already created so it helped very much in viewing the accuracy of the models. For our models, we had to first train them and then test them.
- We used the GridSearchCV object for logistic regression, SVM, decision tree, and KNN models because it allowed us to perform hyperparameter tuning and find the best combination of hyperparameters for each model.
- Hyperparameters are settings or configurations that we can adjust to control the behavior of a machine learning model. Different hyperparameter values can significantly impact the model's performance and generalization capabilities. Finding the optimal hyperparameters is crucial to achieving the best possible model performance.
- The GridSearchCV object in scikit-learn is a powerful tool that automates the process of hyperparameter tuning by exhaustively searching through a specified hyperparameter grid. It systematically evaluates the model's performance with different combinations of hyperparameters using cross-validation. Cross-validation helps to ensure that the evaluation is more robust and reduces the risk of overfitting.
- By using GridSearchCV, we were able to:

1. Specify a range of hyperparameter values to be tested for each model.
2. Automatically perform cross-validation to assess the model's performance on different folds of the training data.
3. Determine the best combination of hyperparameters that resulted in the highest performance score.

- This process saves us time and effort compared to manually trying out different hyperparameter values. It also ensures that our models are well-tuned and optimized for better predictions on unseen data, leading to more reliable and accurate results.
- Git Hub URL: https://github.com/joekyzhou/SpaceXFinalProject/blob/main/IBM_ML.ipynb

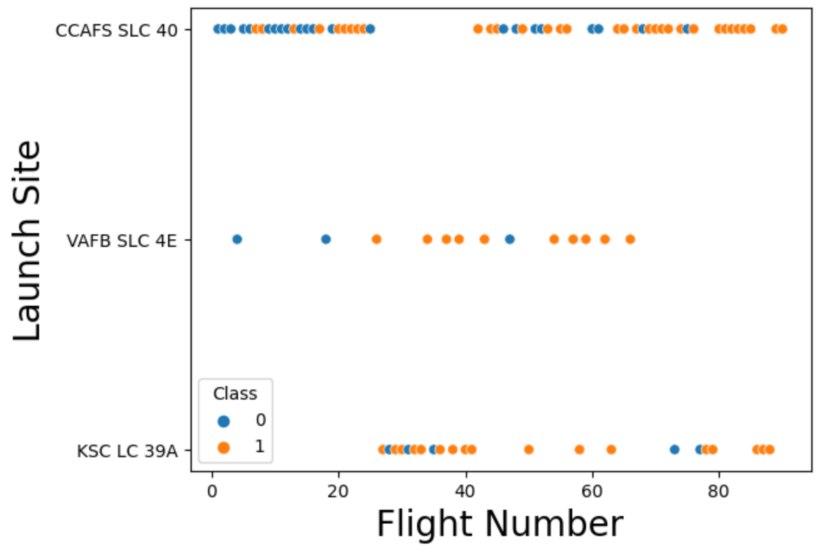
Results

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

Section 2

Insights drawn from EDA

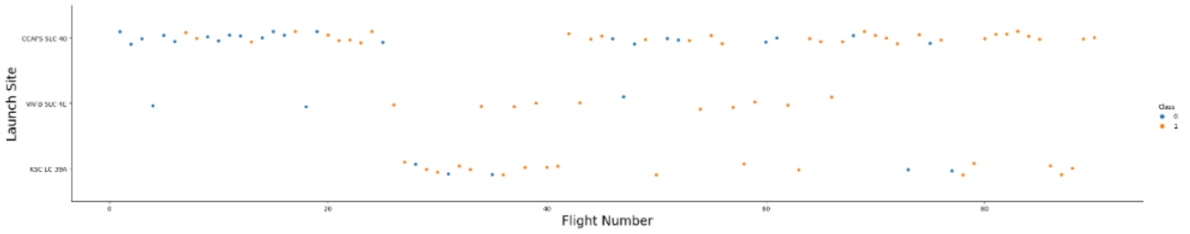
Flight Number vs. Launch Site



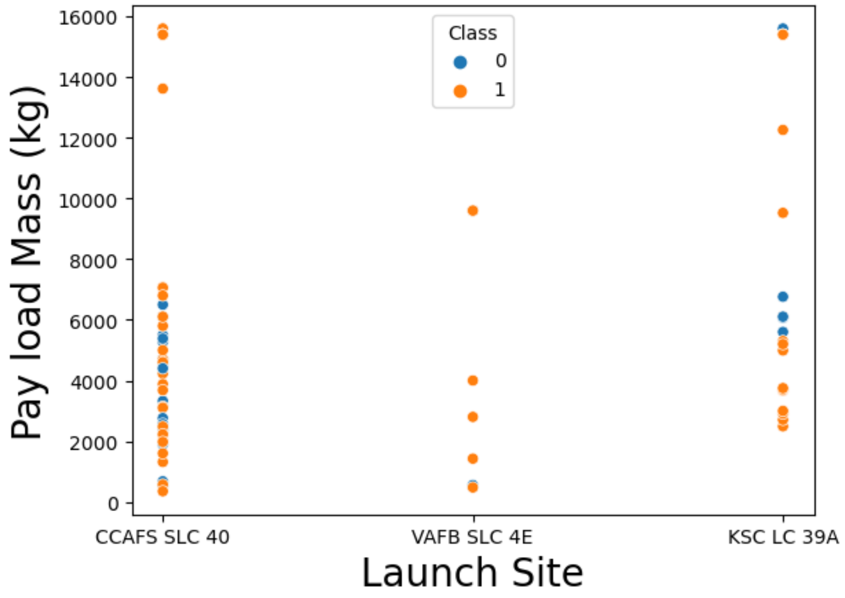
We notice that as we increase the flight numbers for each launch site, the success rate also increases with each launch site having 100% success rate towards the 80th flight number. Site VAFB SLC 4E does show 100% success rate starting at 60th flight nubmer.

Next, let's drill down to each site visualize its detailed launch records.

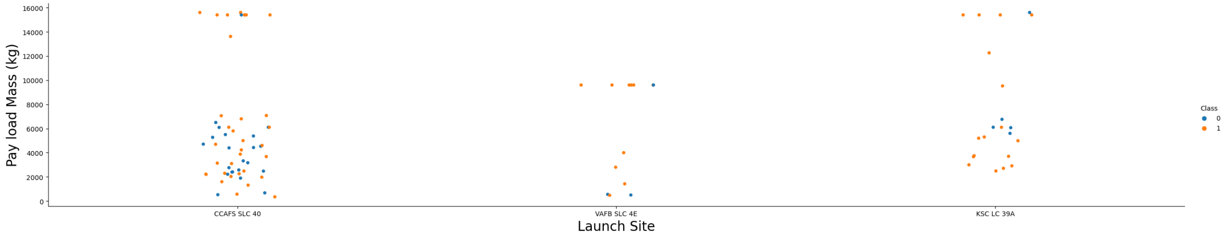
```
]: ### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.catplot(y= "LaunchSite", x ="FlightNumber", hue="Class", data=df, aspect=5)
plt.xlabel('Flight Number', fontsize=20)
plt.ylabel('Launch Site', fontsize=20)
plt.show()
```



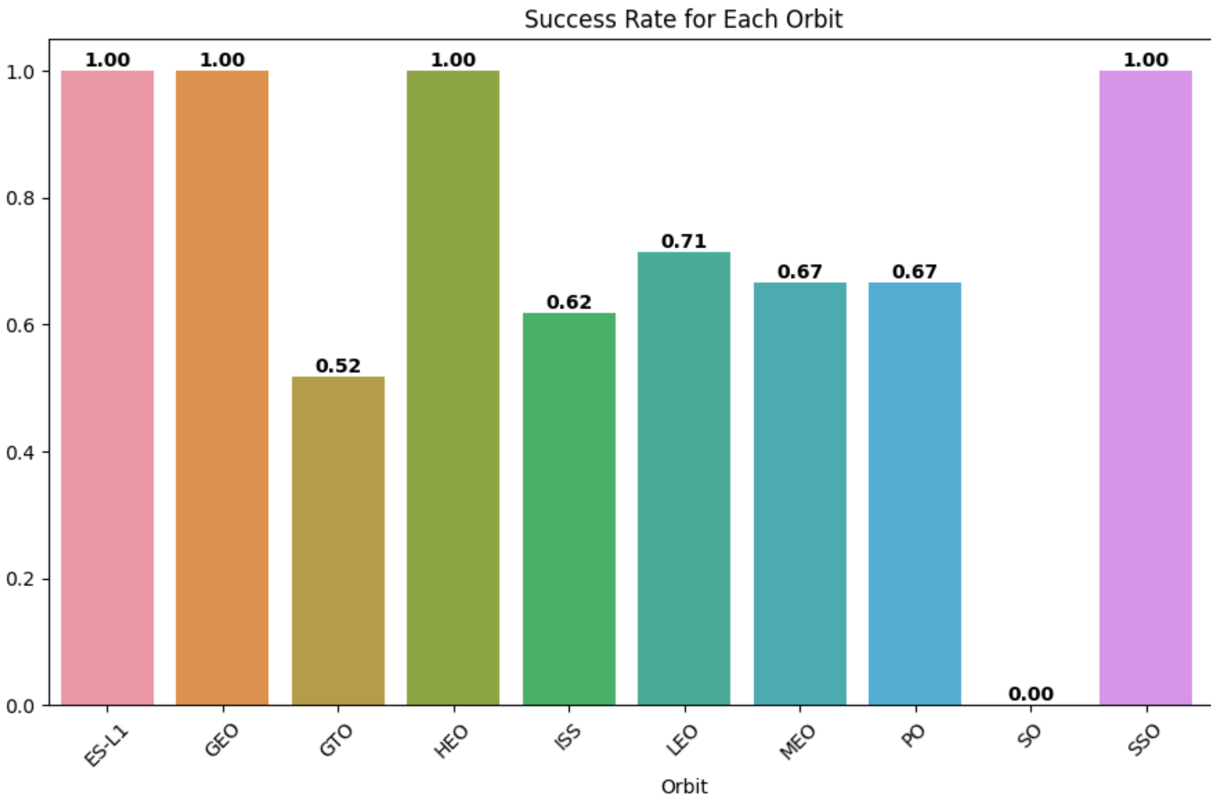
Payload vs. Launch Site



Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rocl mass(greater than 10000).



Success Rate vs. Orbit Type

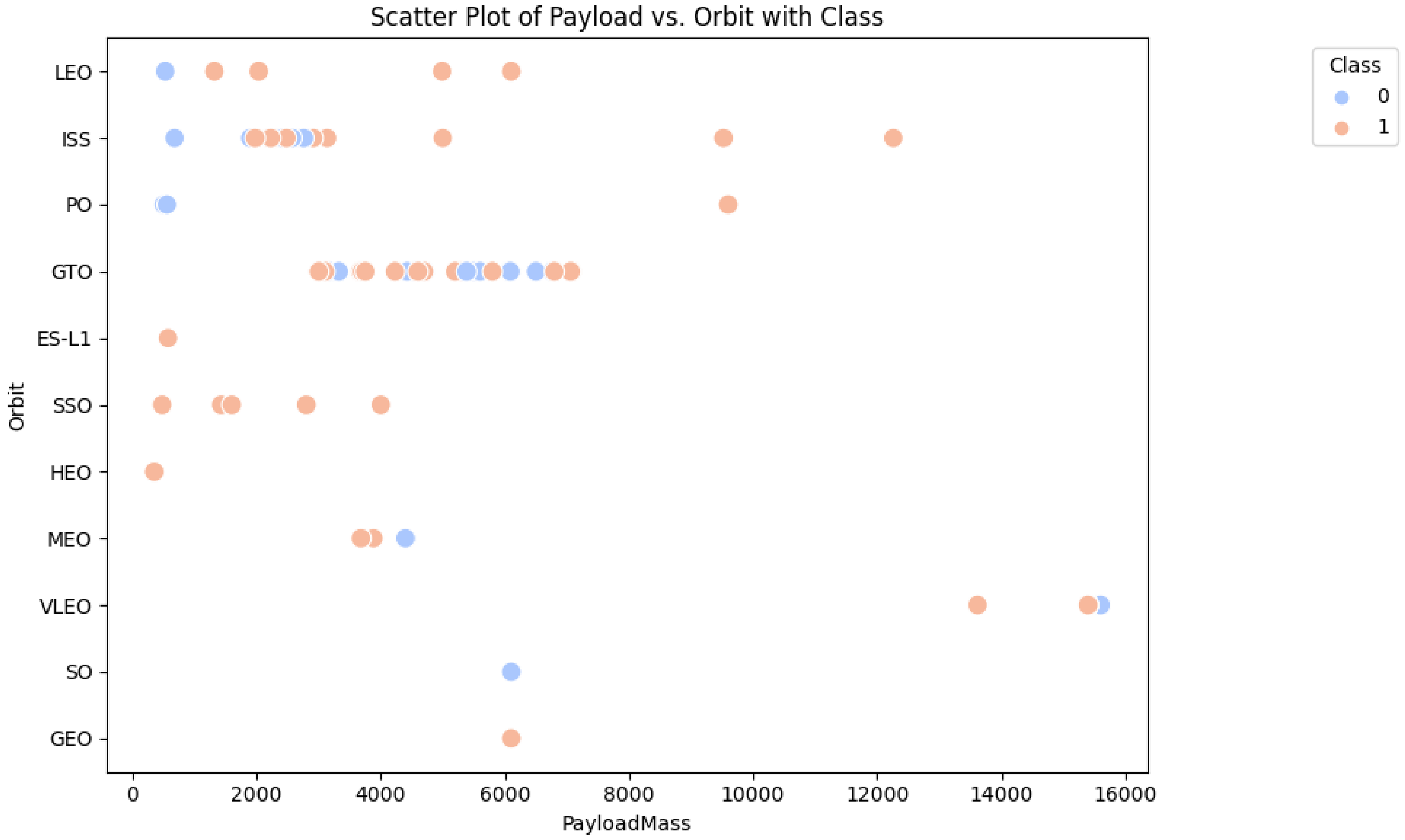


ı should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight nber when in GTO orbit.

Flight Number vs. Orbit Type

You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

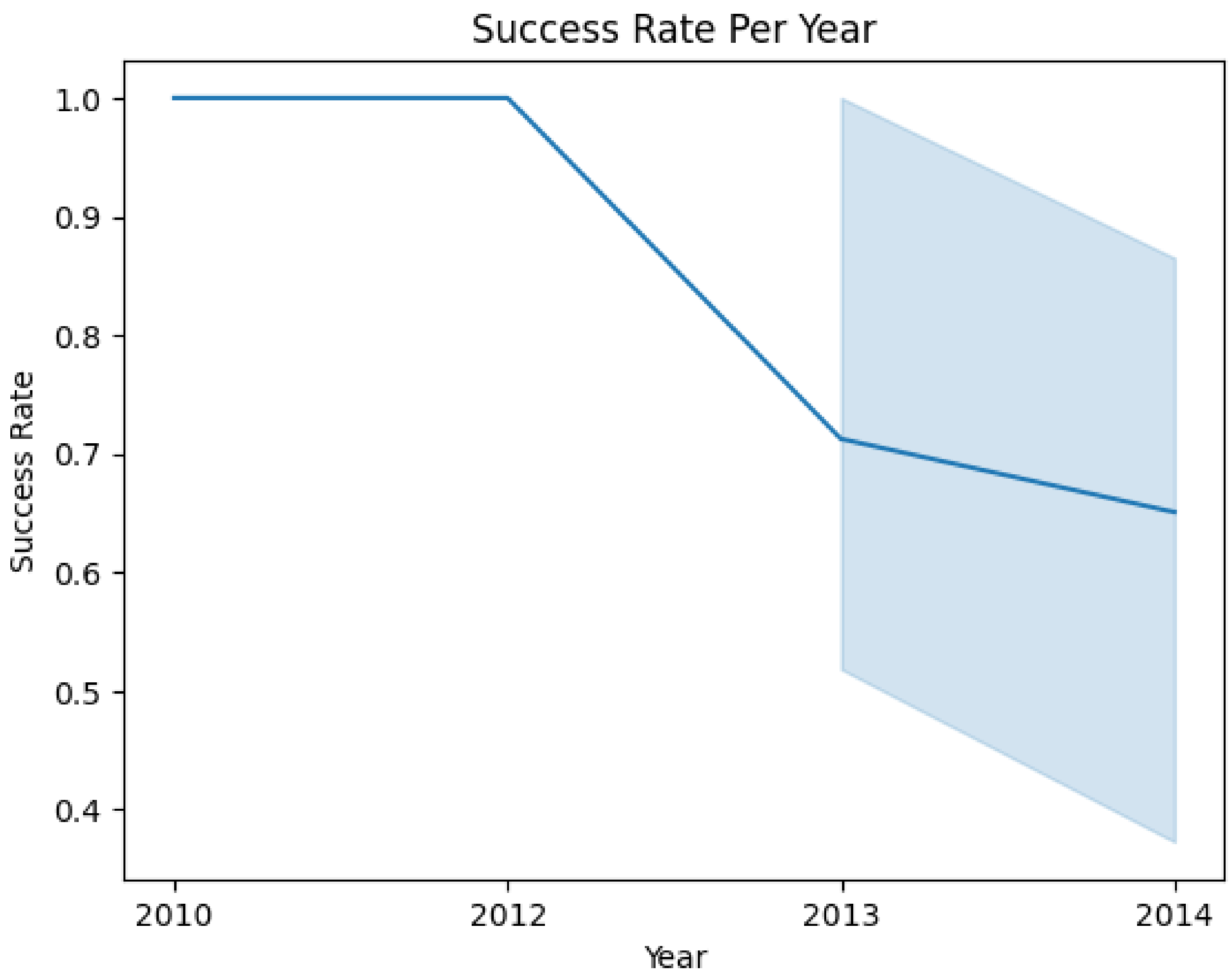
Payload vs. Orbit Type



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

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- The DISTINCT function helped find the unique launch sites in the table and we can see that there are five unique launch sites including “None”

```
%sql SELECT DISTINCT(Launch_Site) as Uniuqe_Launch_Sites FROM SPACEXTBL
```

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

Uniuqe_Launch_Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

Launch Site Names Begin with 'CCA'

- Utilizing the LIKE function we were able to query out the records that began with the letters CCA in the table.

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

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Utilizing the SUM function and the WHERE function, we were able to find the total payload mass carried by boosters launched by NASA (CRS).

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_)as 'Total_Payload_Mass_Carried', Customer FROM SPACEXTBL WHERE Customer == 'NASA (CRS)'
```

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

Total_Payload_Mass_Carried	Customer
45596.0	NASA (CRS)

Average Payload Mass by F9 v1.1

- Utilizing the AVG function and WHERE function, we were able to find the average payload mass carried by booster version F9 v1.1.

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_), Booster_Version FROM SPACEXTBL WHERE Booster_Version == 'F9 v1.1'
```

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

AVG(PAYLOAD_MASS__KG_)	Booster_Version
2928.4	F9 v1.1

First Successful Ground Landing Date

- As the dates were already ordered in the format dd/mm/yy, we were able to see that the first successful landing outcome was on July 22, 2018.

```
%sql SELECT Date, Landing_Outcome FROM SPACEXTBL WHERE Landing_Outcome == 'Success' LIMIT 5
```

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

Date	Landing_Outcome
22/07/2018	Success
25/07/2018	Success
08/07/2018	Success
09/10/2018	Success
10/08/2018	Success

Successful Drone Ship Landing with Payload between 4000 and 6000

- We would have to use the WHERE function and provide all the restrictions to find the list of boosters which have success in drone ship while having payload mass (KG) between 4000 ~ 6000.

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

```
%sql SELECT * FROM SPACEXTBL WHERE (PAYLOAD_MASS__KG_ > 4000) & (PAYLOAD_MASS__KG_ < 6000) & (Landing_Outcome == 'Success (drone
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
05/06/2016	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696.0	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
14/08/2016	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600.0	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
30/03/2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300.0	GTO	SES	Success	Success (drone ship)
10/11/2017	22:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200.0	GTO	SES EchoStar	Success	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- The best way to query out the total number of successful and failure mission outcomes is by using the GROUP BY Function and COUNT function.

List the total number of successful and failure mission outcomes

```
: ON_Outcome FROM SPACEXTBL WHERE MISSION_Outcome == 'Success' or MISSION_Outcome == 'Failure (in flight)' GROUP BY MISSION_Outcome
```

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

Total Number	Mission_Outcome
1	Failure (in flight)
98	Success

Boosters Carried Maximum Payload

- By performing subquery after the WHERE function, we were able to select only the Maximum Payloads, which came out to be 15,600 KG.

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM S
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600.0
F9 B5 B1049.4	15600.0
F9 B5 B1051.3	15600.0
F9 B5 B1056.4	15600.0
F9 B5 B1048.5	15600.0
F9 B5 B1051.4	15600.0
F9 B5 B1049.5	15600.0
F9 B5 B1060.2	15600.0
F9 B5 B1058.3	15600.0
F9 B5 B1051.6	15600.0
F9 B5 B1060.3	15600.0
F9 B5 B1049.7	15600.0

2015 Launch Records

- In order to find the 2015 launch records for failure (drone ship) landing outcomes, we had to format the date as substrings first and two results were found for the year 2015.

```
%sql SELECT substr(Date, 4, 2 ) as 'Month', substr(Date, 7,4) as 'Year', Landing_Outcome, Booster_Version, Launch_Site FROM
```

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

Month	Year	Landing_Outcome	Booster_Version	Launch_Site
10	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
01	2016	Failure (drone ship)	F9 v1.1 B1017	VAFB SLC-4E
04	2016	Failure (drone ship)	F9 FT B1020	CCAFS LC-40
06	2016	Failure (drone ship)	F9 FT B1024	CCAFS LC-40

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

- We first had to format the dates by using the substring method before we can group these newly formatted dates in descending order from 2010-06-04 to 2017-03-20.

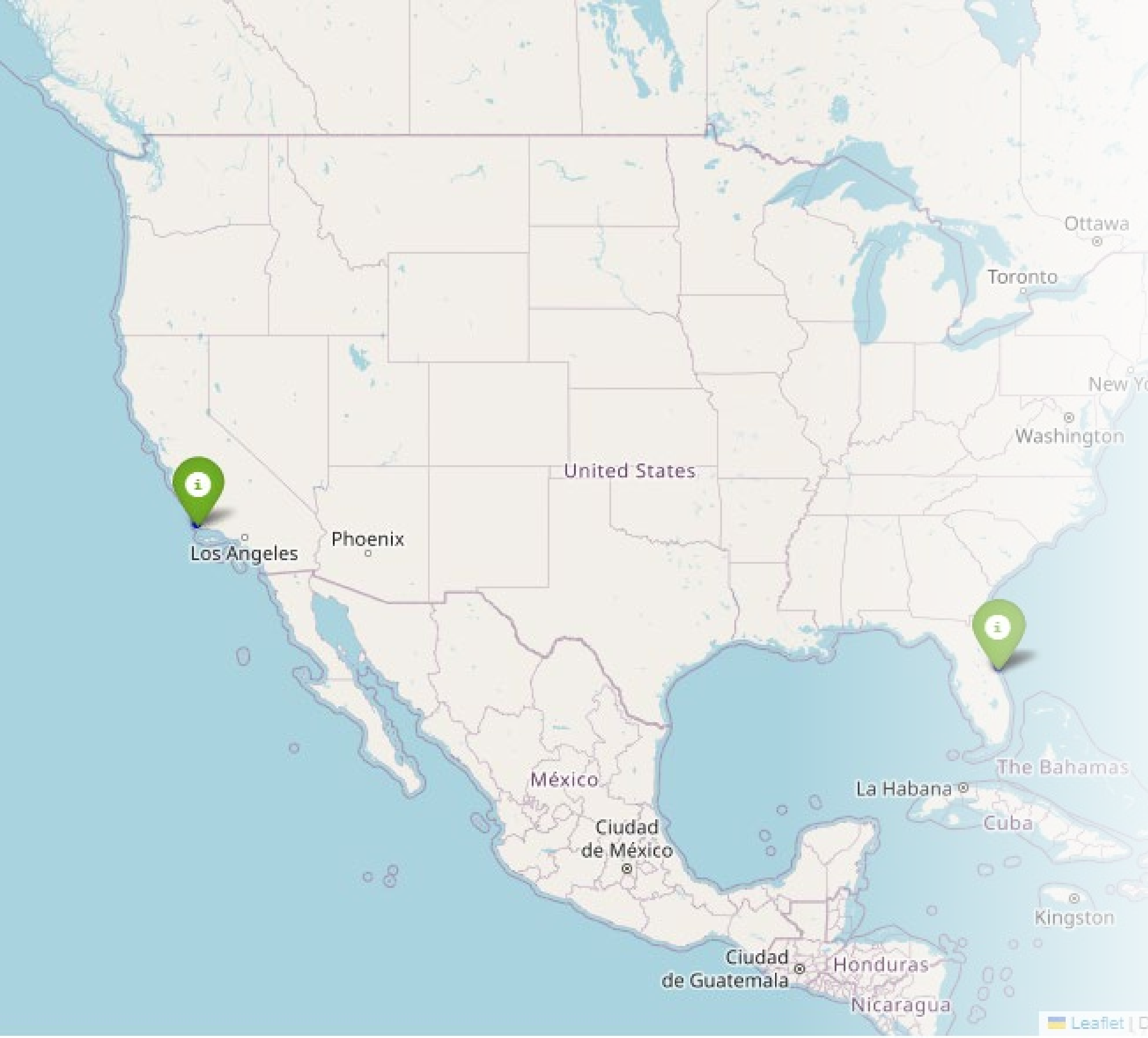
```
%sql SELECT SUBSTRING(Date, 7, 4) || '-' || SUBSTRING(Date, 4, 2) || '-' || SUBSTRING(Date, 1, 2) AS formatted_date, Landing_Outc
```

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

formatted_date	Landing_Outcome	Outcome_Count
2017-03-16	No attempt	1
2017-03-06	Success (ground pad)	1
2017-02-19	Success (ground pad)	1
2017-01-14	Success (drone ship)	1
2017-01-05	Success (ground pad)	1
2016-08-14	Success (drone ship)	1
2016-08-04	Success (drone ship)	1
2016-07-18	Success (ground pad)	1
2016-06-15	Failure (drone ship)	1
2016-06-05	Success (drone ship)	1
2016-05-27	Success (drone ship)	1
2016-04-03	Failure (drone ship)	1
2016-01-17	Failure (drone ship)	1
2015-12-22	Success (ground pad)	1
2015-11-02	Controlled (ocean)	1
2015-10-01	Failure (drone ship)	1
2015-06-28	Precluded (drone ship)	1
2015-04-27	No attempt	1
2015-04-14	Failure (drone ship)	1
2015-02-03	No attempt	1
2014-09-21	Uncontrolled (ocean)	1
2014-07-14	Controlled (ocean)	1
2014-07-09	No attempt	1
2014-06-01	No attempt	1
2014-05-08	No attempt	1
2014-04-18	Controlled (ocean)	1
2013-09-29	Uncontrolled (ocean)	1
2013-03-12	No attempt	1
2013-01-03	No attempt	1
2012-08-10	No attempt	1
2012-05-22	No attempt	1
2010-08-12	Failure (parachute)	1

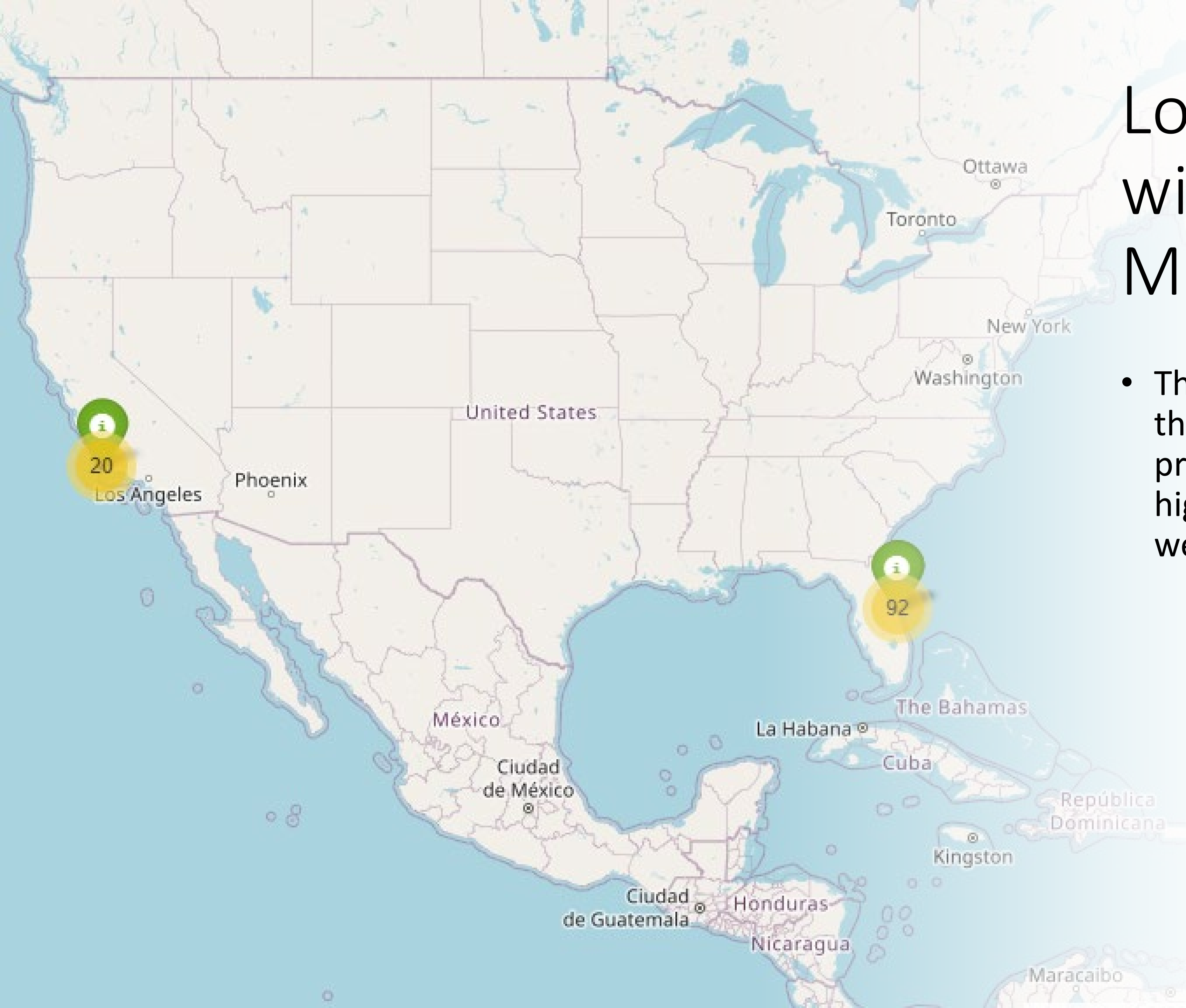
Section 3

Launch Sites Proximities Analysis



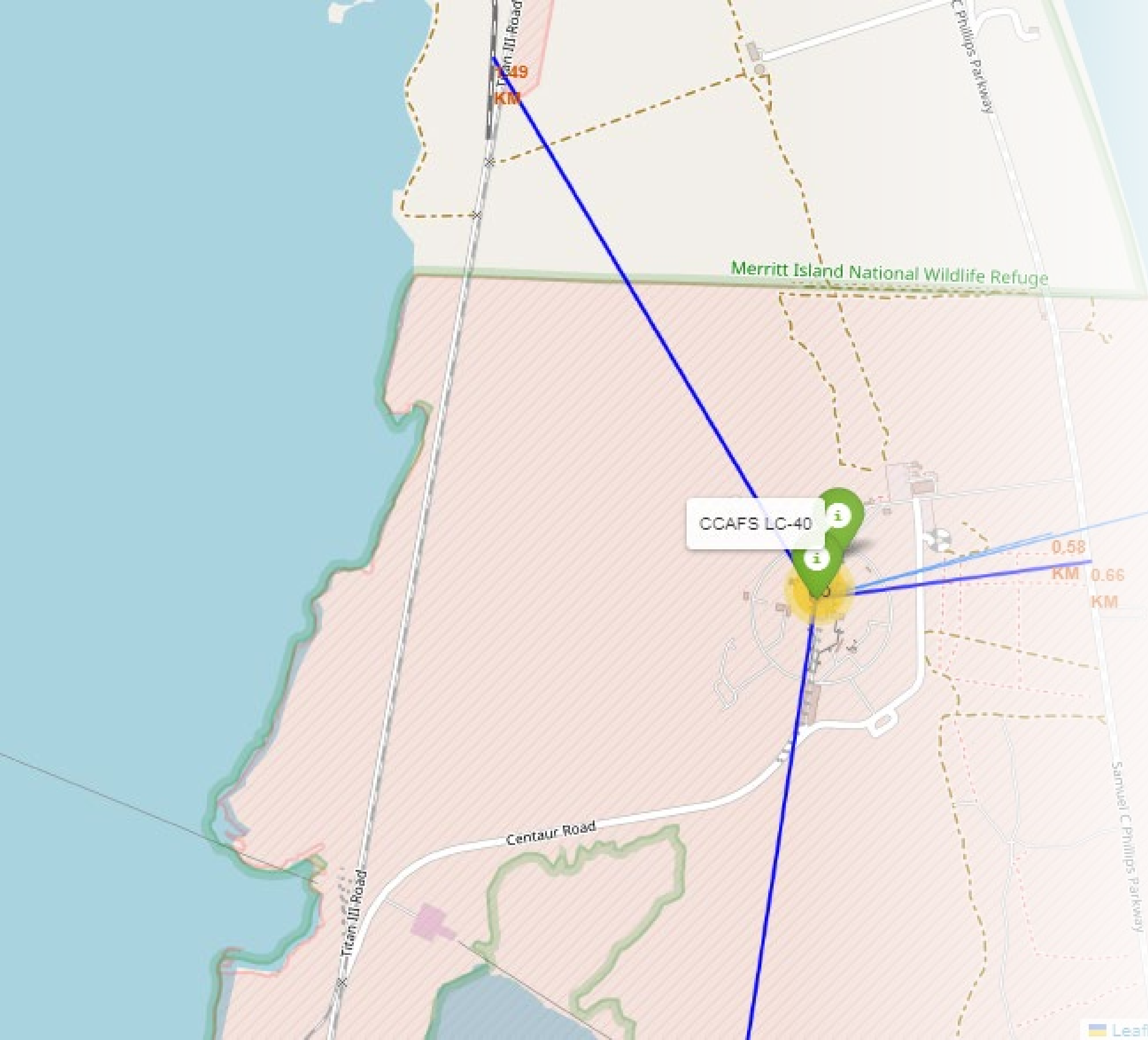
Location Analysis with Folium

- Our folium map clearly distinguishes the launch sites owned by SpaceX to either be on the far west coast or far east coast of the united states, which makes perfect sense as rocket launching may be a dangerous activity to initiate. We can also see that they are no where near the Equator line.



Location Analysis with Folium Marker Color

- The marker outcome shows that the east coast shows more of a promising result with a relatively higher success rate than the west coast launch sites.



Site CCAFS LC-40 Proximities

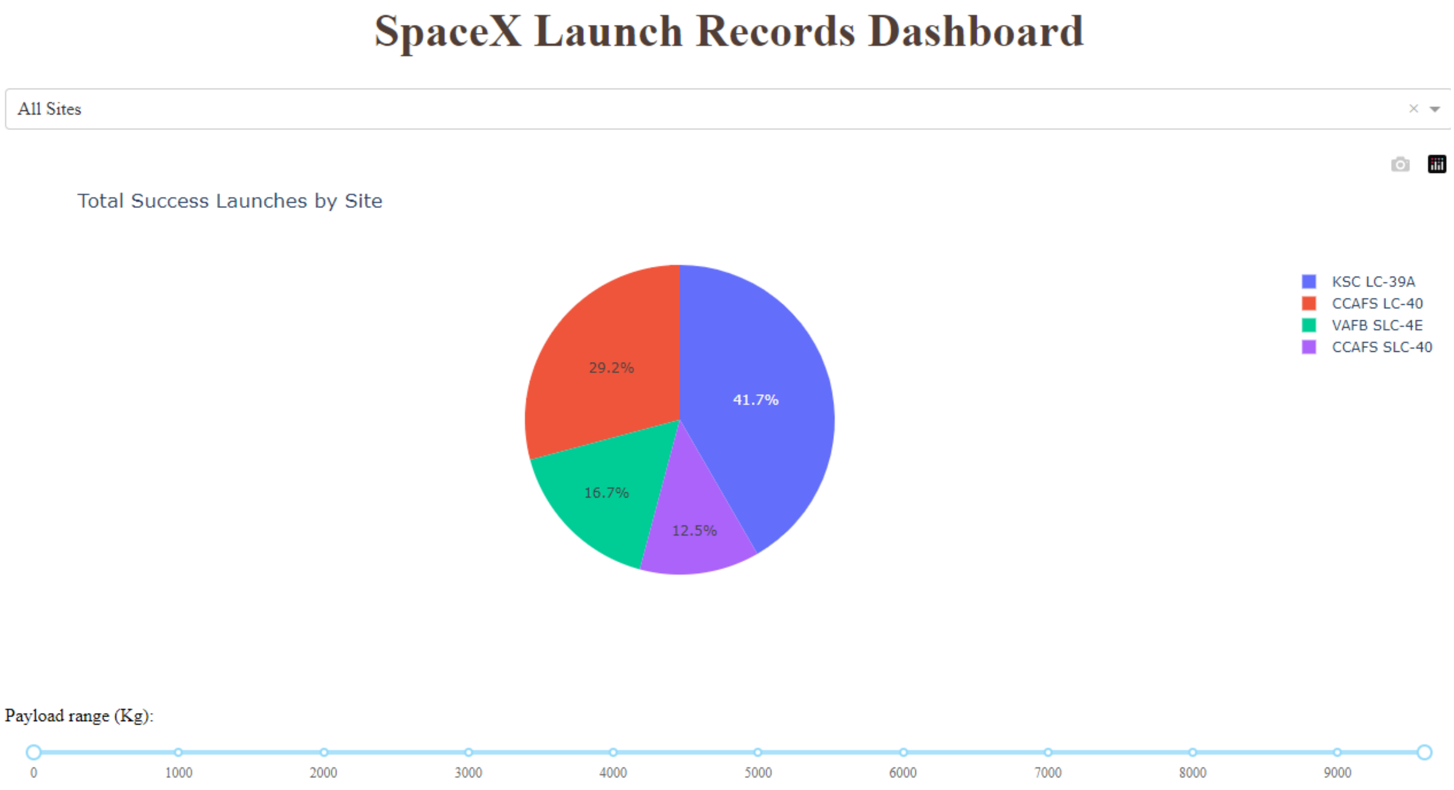
- Further investigating site CCAFS LC-40, we were ablet o find that its proximity to the east coastline is approximately 0.58m, to the railroad is 1.49m and to the city Cape Canaveral is 18.17m.

Section 4

Build a Dashboard with Plotly Dash

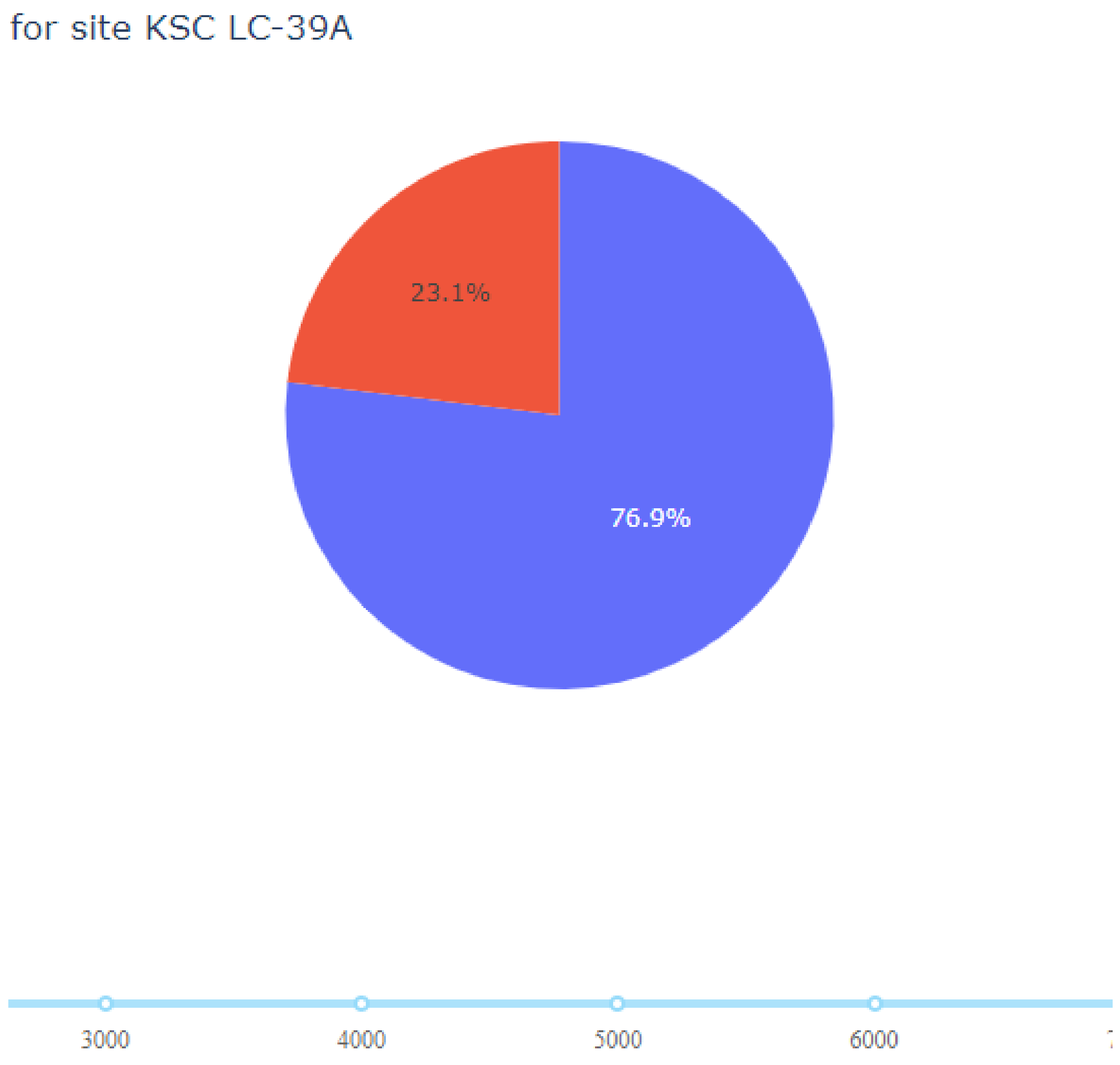
SpaceX Success Launch by Site

- From our pie chart, we can conclude that Site KSC LC-39A has a higher success rate than the other sites with a percentage of success rate of 41.7%.



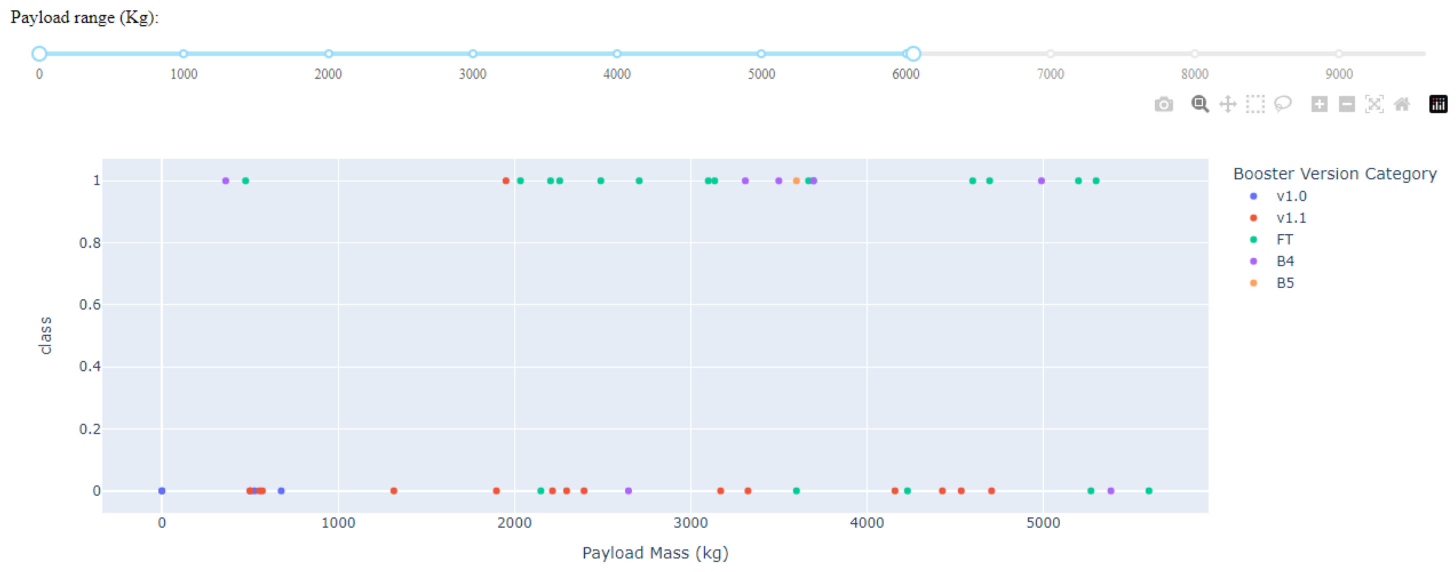
Site KSC LC-39A Success vs Failure Launches

- According to our pie charts, we can see that Site KSC LC-39A has a success rate of 76.9% to failure rate of 23.1% making it the launch site with the highest success launches compared to the other launch sites.



Payload Mass vs Booster Versions

- As we increase our payload mass range from 0 to 6000, we notice that the booster version FT significantly do better than the other booster versions. On the other hand, booster version v1.1 show significantly bad results with class of 0 representing failures rather than successes.

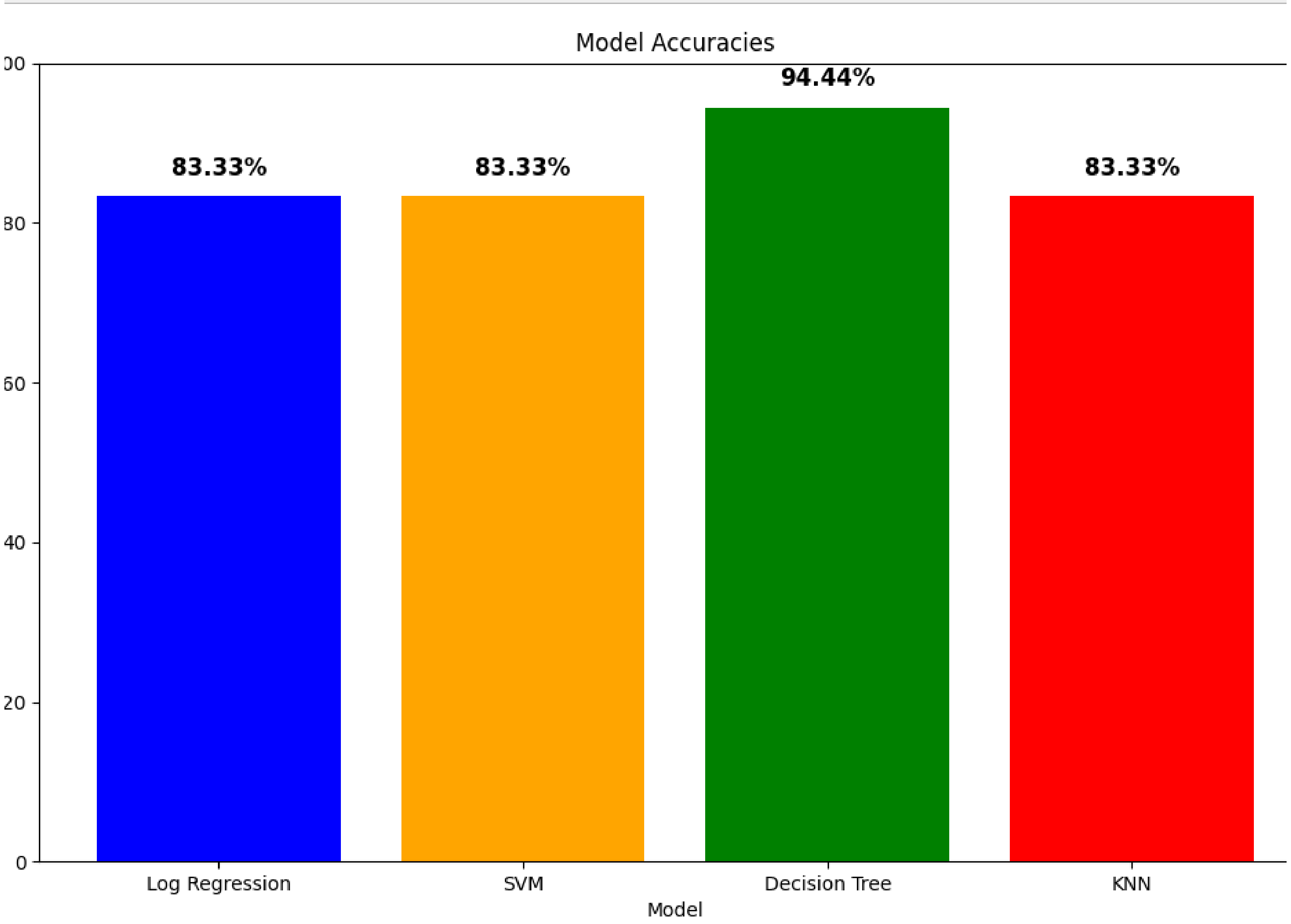


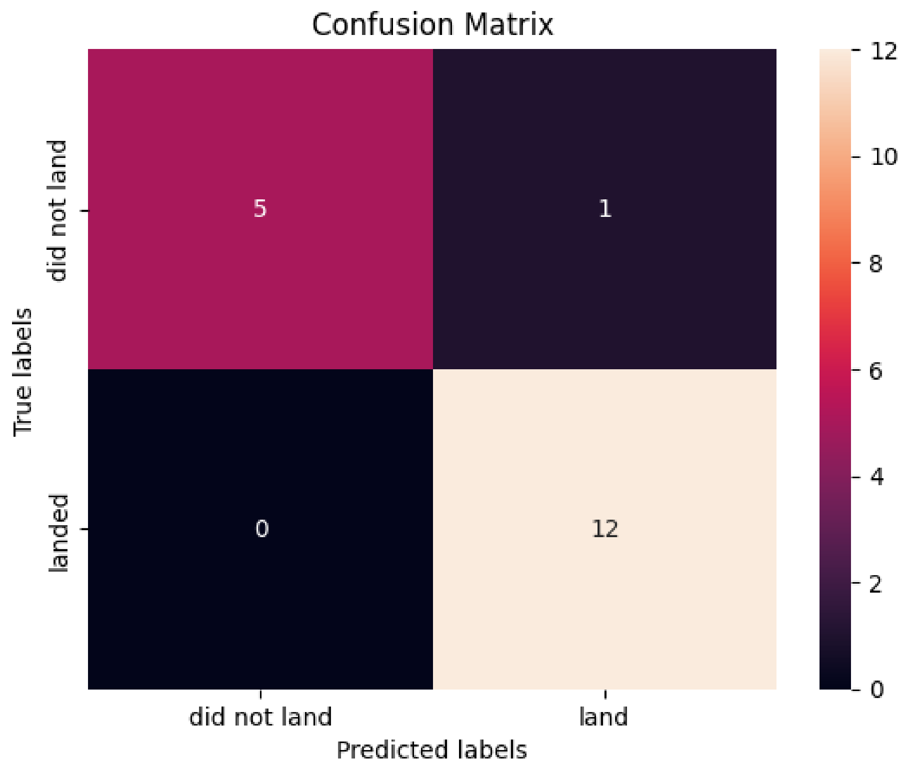
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- According to the bar chart, we can conclude that the Decision Tree Model provides a more accurate prediction.





Confusion Matrix

- The confusion matrix with the best accuracy is demonstrated by the decision tree classifier.
- From the confusion matrix we have a True Positive of 5, which means that the decision tree correctly predicted 5 instances when boosters did not land.
- False Negative of 1, which means the decision incorrectly predicted 1 instance when a booster did land.
- False Positive of 0, which means the decision tree did not make any mistakes on predicting the negative instances.
- True Negative of 12, which means the decision tree correctly predicted 12 instances when the boosters did indeed land.
- We can then see that we have an accuracy of 94.44%, Precision of 100%, True Positive Rate or Recall of 83.33%, Specificity of 100%, and F1-Score of around 90.91% using the founded TP, FN, FP, and TN into formulas.
- These findings show that the decision tree is making correct predictions for both positive and negative cases.

Conclusions

- Decision Tree model achieved the highest accuracy of 94.44%, outperforming other models.
- Logistic Regression, SVM, and KNN models showed similar accuracies of 83.33%, which is interesting.
- The Decision Tree model's high accuracy makes it a promising choice for predicting Falcon 9 First-Stage Landing Outcomes.
- Precision, Recall, and F1-score metrics should be considered alongside accuracy to get a comprehensive evaluation of model performance.
- Hyperparameter tuning and feature engineering significantly impact their model performances.
- Visualizations, such as confusion matrices and decision boundaries, provide valuable insights into model behavior.
- Visualizations like Folium and Plotly Dash also help us better understand booster performances significantly.
- The dataset's restructuring using SQL magic was an important step to challenging the data structure effectively.
- The analysis successfully identified factors influencing landing outcomes, aiding data driven decision making for SpaceX launches.

Appendix

1. Data Sources:

- SpaceX Falcon 9 Launch Data: Retrieved from SpaceX official website (URL: www.spacex.com/launches)

2. Data Preprocessing Details:

- Date Restructuring: Utilized SQL magic to convert date format

3. Model Details:

- Logistic Regression: Solver: ‘lbfgs’, C: [0.01, 0.1, 1]
- SVM: Kernel: [‘linear’, ‘poly’, ‘sigmoid’, ‘rbf’], C: [0.01, 0.1, 1]
- Decision Tree Classifier: Max Depth: [2*n for n in range(1,10)], Min Samples Leaf: [1, 2, 4]
- KNN: Number of Neighbors: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], Algorithm: ['auto', 'ball_tree', 'kd_tree', 'brute’]

4. Confusion Matrix:

- Decision Tree: TP = 5, FP = 0, TN = 12, FN = 1
- Other Models: TP= 3, FP = 0, TN = 12, FN =3

5. Visualization Details:

- Scatter Plots, Bar Charts, Pie Charts, Line Plots, Plotly Dash, Confusion Matrix, Folium Maps

6. Additional Performance Metrics:

- Precision, Recall, and F1-Score for all models

7. References:

- IBM Data Visualization with Python Course, Final Assignment: Part 2 – Create Dashboard with Plotly and Dash Skills Network Labs
- IBM Data Visualization with Python Course, Final Assignment: Part 1 – Create Visualizations using Matplotlib, Seaborn & Folium
- Aryan Gupta (April 25, 20213).Date Format in SQL – SQL Date Time Format: How to Change It? Retrieved from https://www.simplilearn.com/tutorials/sql-tutorial/sql-date-format
- IBM Machine Learning with Python Labs: SVM (Support Vector Machines)
- IBM Machine Learning with Python Labs: KNN
- IBM Machine Learning with Python Labs: Decision Trees
- Enes Polat (5Y Ago).Grid Search with Logistic Regression Retrieved from https://www.kaggle.com/code/enespolat/grid-search-with-logistic-regression

Thank you!