IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Chigozie Godwin

May 2025

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

The IBM Data Science Professional Certificate capstone project involves the use real-world data to demonstrate expertise in data science and machine learning techniques, then compile the findings into a report.

In this research, SpaceY, a competitor of SpaceX, uses data from the SpaceX Falcon 9 rocket to calculate the cost of a launch as well as the rocket's first stage landing success. Using the information, Space Y bids against SpaceX for a rocket launch. SpaceX claims that the Falcon 9 rocket launch will cost $62 million. For other businesses, however, launching a rocket costs more than 165 million dollars.

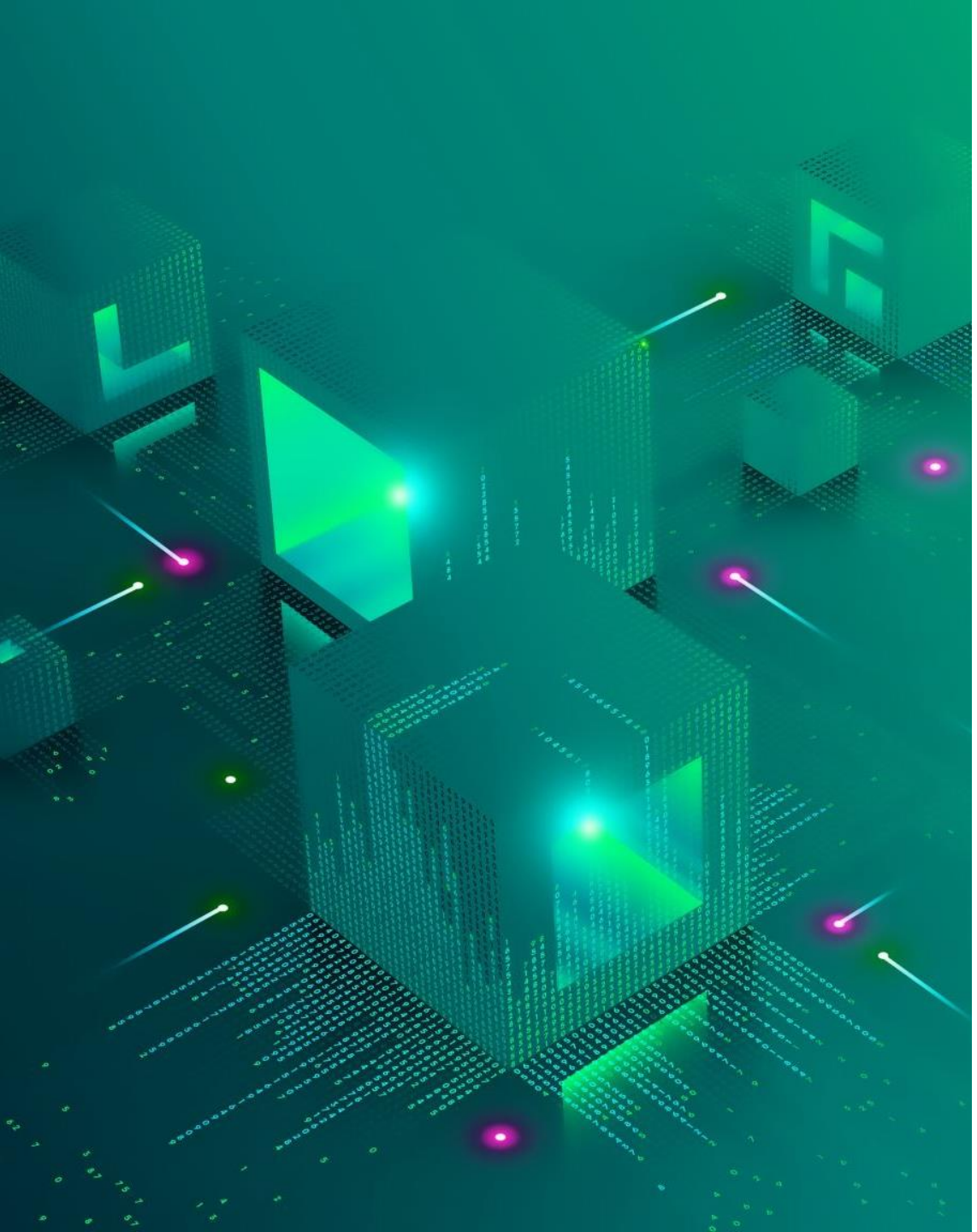
Throughout the project, data collection and analysis are done using Python Jupyter notebooks. The final *.pdf report and these Jupyter notebooks are stored on my GitHub repository page.

Data wrangling, exploratory data analysis (EDA), interactive data visualisation, machine learning (ML) classification model building, and model evaluation are the main components of this research.

Lastly, the precision of various machine learning methods in forecasting the Falcon 9 first stage rocket's future landing is contrasted.

Section 1

Methodology



Methodology

- This project's data was from the Wikipedia launch table and the SpaceX Rest API.

- The data was cleaned, visualised after preparation, and the cleaned data was used in machine learning (ML) predictive models which are logistic regression, support vector machines (SVM), decision trees, and K-nearest neighbours (KNN)

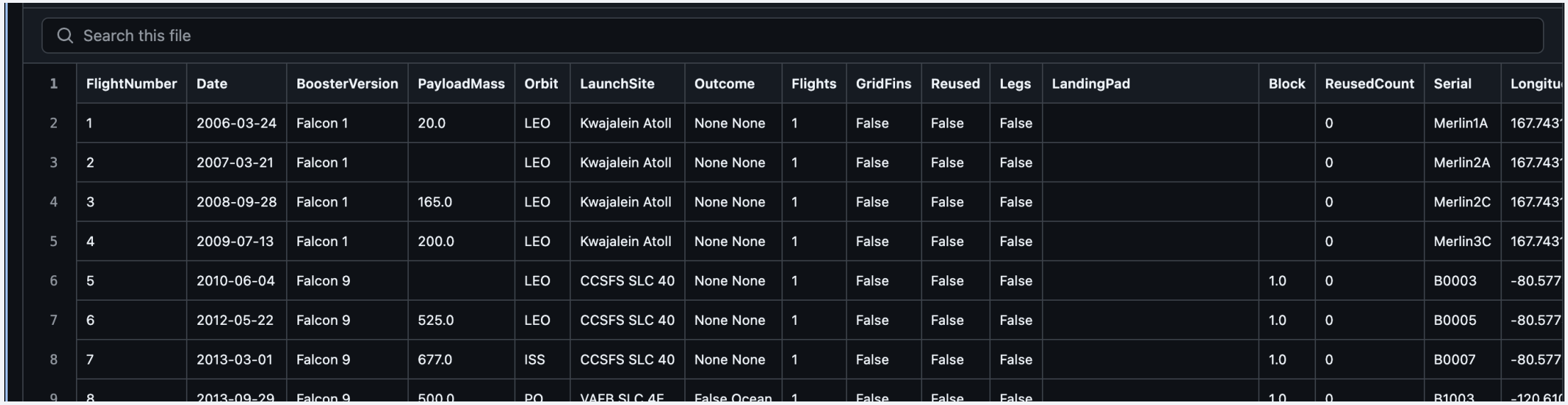
- Additionally, SQL and visualisation were used for exploratory data analysis (EDA).

- Finally, the Python modules Folium and Plotly Dash were used for both interactive visual analytics and data representation.

- Lastly, classification models were used in predictive analysis to determine the correctness of the model and predict if the Falcon 9 rocket's first stage would land successfully using Skikit-learn.

Data Collection

- The data was collected from the SpaceX API and converted to a.json file.
- Assign data to the dataframe and dictionary, scrape and filter data to include Falcon 9 data, then export data to a CSV file.



Search this file

1	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitu
2	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin1A	167.743
3	2	2007-03-21	Falcon 1		LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin2A	167.743
4	3	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin2C	167.743
5	4	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin3C	167.743
6	5	2010-06-04	Falcon 9		LEO	CCSFS SLC 40	None None	1	False	False	False		1.0	0	B0003	-80.577
7	6	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False		1.0	0	B0005	-80.577
8	7	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False		1.0	0	B0007	-80.577
9	8	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False		1.0	0	B1003	-120.610

url: https://github.com/godwinchigozie679/Web-scraping-Falcon-9/blob/main/dataset_part_1.csv

Data Collection – SpaceX API

Use the get request and covert data to get a response from the SpaceX API.Jason's file

Clean data and assign data to dictionary and data frame Portion of output

Filter data to include only Falcon 9 launches and export data to a csv file: dataset_part1

```
def getBoosterVersion(data):
    for x in data['rocket']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
            BoosterVersion.append(response['name'])
            print('complete...')
```

Code Snippet

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [3]:
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list

def getLaunchSite(data):
    for x in data['launchpad']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
            Longitude.append(response['longitude'])
            Latitude.append(response['latitude'])
            LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [4]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        if load:
            response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
            PayloadMass.append(response['mass_kg'])
            Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

In [5]:
```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
```

Search this file

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitu
2	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin1A	167.743
3	2	2007-03-21	Falcon 1		LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin2A	167.743
4	3	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin2C	167.743
5	4	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin3C	167.743
6	5	2010-06-04	Falcon 9		LEO	CCSFS SLC 40	None None	1	False	False	False		1.0	0	B0003	-80.577
7	6	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False		1.0	0	B0005	-80.577
8	7	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False		1.0	0	B0007	-80.577
9	8	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False		1.0	0	B1003	-120.610

View of data

GitHub url: https://github.com/godwinchigozie679/Space-X-Falcon-9-First-Stage-Landing-Prediction-Data-Wrangling-Lab2/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

Perform HTTP get to request Falcon 9 HTML page and create Beautiful Soup object from HTML

Extract all column/variable names from the HTML table header

Create a data frame by parsing the launch HTML tables and export data into CSV file As spacex_web_scraped.csv

Code Snippet

```
# use requests.get() method with the provided static_url
# assign the response to a object

response = requests.get(static_url)
print(response)
```

```
<Response [200]>
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
# Get the HTML content
html_content = response.text

# Print the HTML content
# print(html_content)


# Step 3: Create a BeautifulSoup object and specify the parser
soup = BeautifulSoup(html_content, 'html.parser')

# Now you can use 'soup' to parse and extract data from the HTML
# print(soup.prettify())  # This will print the HTML content in a readable format
```

Search this file

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitu
2	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin1A	167.743
3	2	2007-03-21	Falcon 1		LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin2A	167.743
4	3	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin2C	167.743
5	4	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False			0	Merlin3C	167.743
6	5	2010-06-04	Falcon 9		LEO	CCSFS SLC 40	None None	1	False	False	False		1.0	0	B0003	-80.577
7	6	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False		1.0	0	B0005	-80.577
8	7	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False		1.0	0	B0007	-80.577
9	8	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False		1.0	0	B1003	-120.610

View of data

GitHub url: https://github.com/godwinchigozie679/Web-scraping-Falcon-9/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

Load data from dataset_part1.csv file and calculate the number of launches on each site

Calculate the number and the occurrence of each orbit and Calculate the number and occurrence of mission outcome of the orbits

Create a landing outcome label from outcome column and export data into dataset_part2.csv file

Code Snippet

```
In [38]: for i,outcome in enumerate(landing_outcomes.keys()):
             print(i,outcome)

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

We create a set of outcomes where the second stage did not land successfully:

In [39]: bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
         bad_outcomes

Out[39]: {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

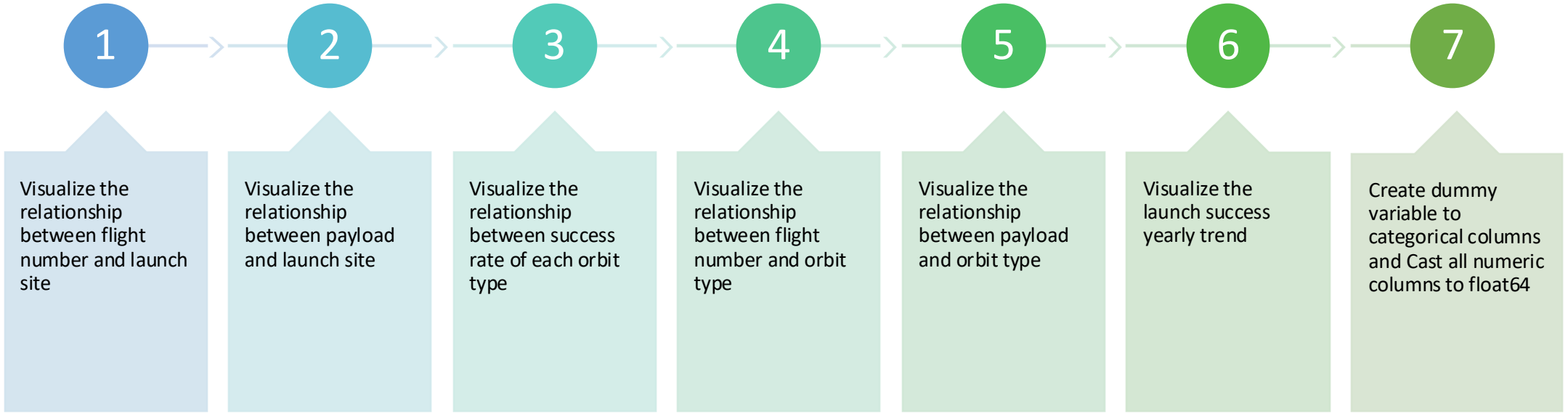
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPac
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN

View of data

GitHub url: https://github.com/godwinchigozie679/Space-X-Falcon-9-First-Stage-Landing-Prediction-Data-Wrangling-Lab2/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Use Matplotlib and Seaborn for data visualization



GitHub url: https://github.com/godwinchigozie679/SpaceX-falcon-9-Exploring-and-Preparing-Data/blob/main/edadataviz%20(1).ipynb

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 average payload mass carried by booster launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the names of the booster_ve which have carried the max payload mass
- List the total number of successful and failure mission outcomes
- List the names of the boosters which have success in drone ship and mass > 4000 & <6000
- List the records which display the month, failure landing, booster version ..etc.
- Rank the count of landing outcomes or success

GitHub url: https://github.com/godwinchigozie679/Assignment-SQL-Notebook-for-Peer-Assignment-/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Mark all launch sites on a map created using Folium by adding markers* with circle , popup label and text label to each site using its longitude and latitude coordinates to show the geographical location approximately to the
- Mark the success/failed launches for each site on the map using colored markers equator
- Calculate the distanc between a launch site to its proximities

GitHub url:
https://github.com/godwinchigozie679/Hands-on-Lab-Interactive-Visual-Analytics-with-Folium/blob/main/lab_jupyter_launch_site_location%20(1).ipynb

Explanation:

- From the visual analysis of the launch site KSC LC-39A we can clearly see that it is:
  - relative close to railway (15.23 km)
  - relative close to highway (20.28 km)
  - relative close to coastline (14.99 km)
- Also the launch site KSC LC-39A is relative close to its closest city Titusville (16.32 km).
- Failed rocket with its high speed can cover distances like 15-20 km in few seconds. It could be potentially dangerous to populated areas.

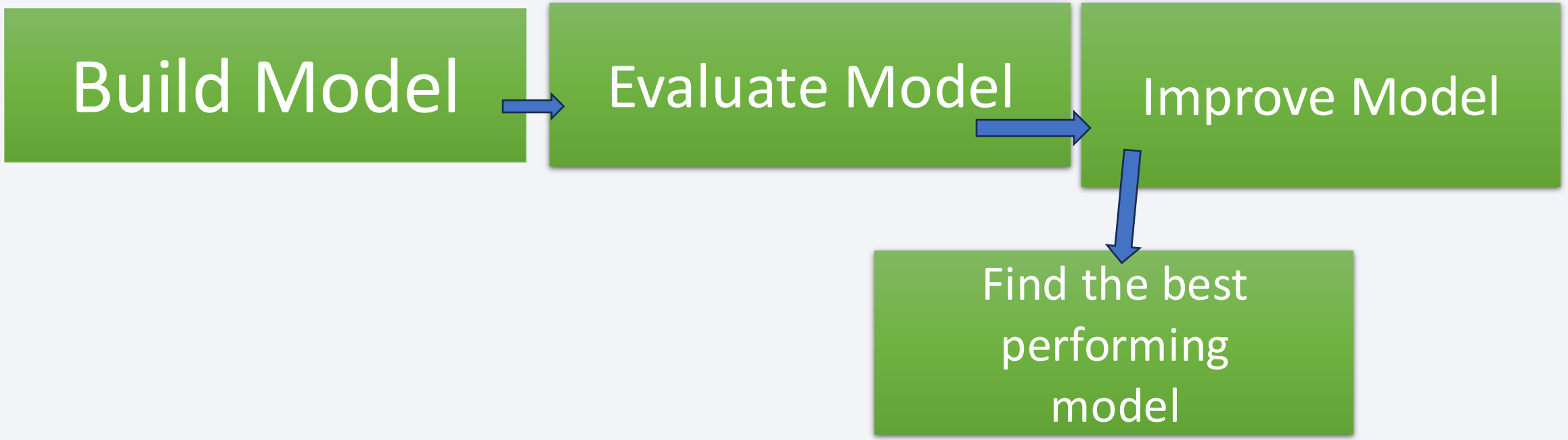
Build a Dashboard with Plotly Dash

GitHub url:
https://github.com/godwinchigozie679/SpaceX-Launch-Records-Dashboard

- Add dropdown list to enable launch site selection
- Add pie chart to show the total successful launches count for all sites and the success vs. failed counts
- Add a range slider to select payload
- Add a scatter chart of payload mass vs. success rate of different booster versions

The dashboard is built using Dash web

Predictive Analysis (Classification)



GitHub url: https://github.com/godwinchigozie679/Space-X-Falcon-9-First-Stage-Landing-Prediction/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5%20(1).ipynb

Predictive Analysis (Classification) Steps

GitHub url: https://github.com/godwinchigozie679/Space-X-Falcon-9-First-Stage-Landing-Prediction/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5%20(1).ipynb

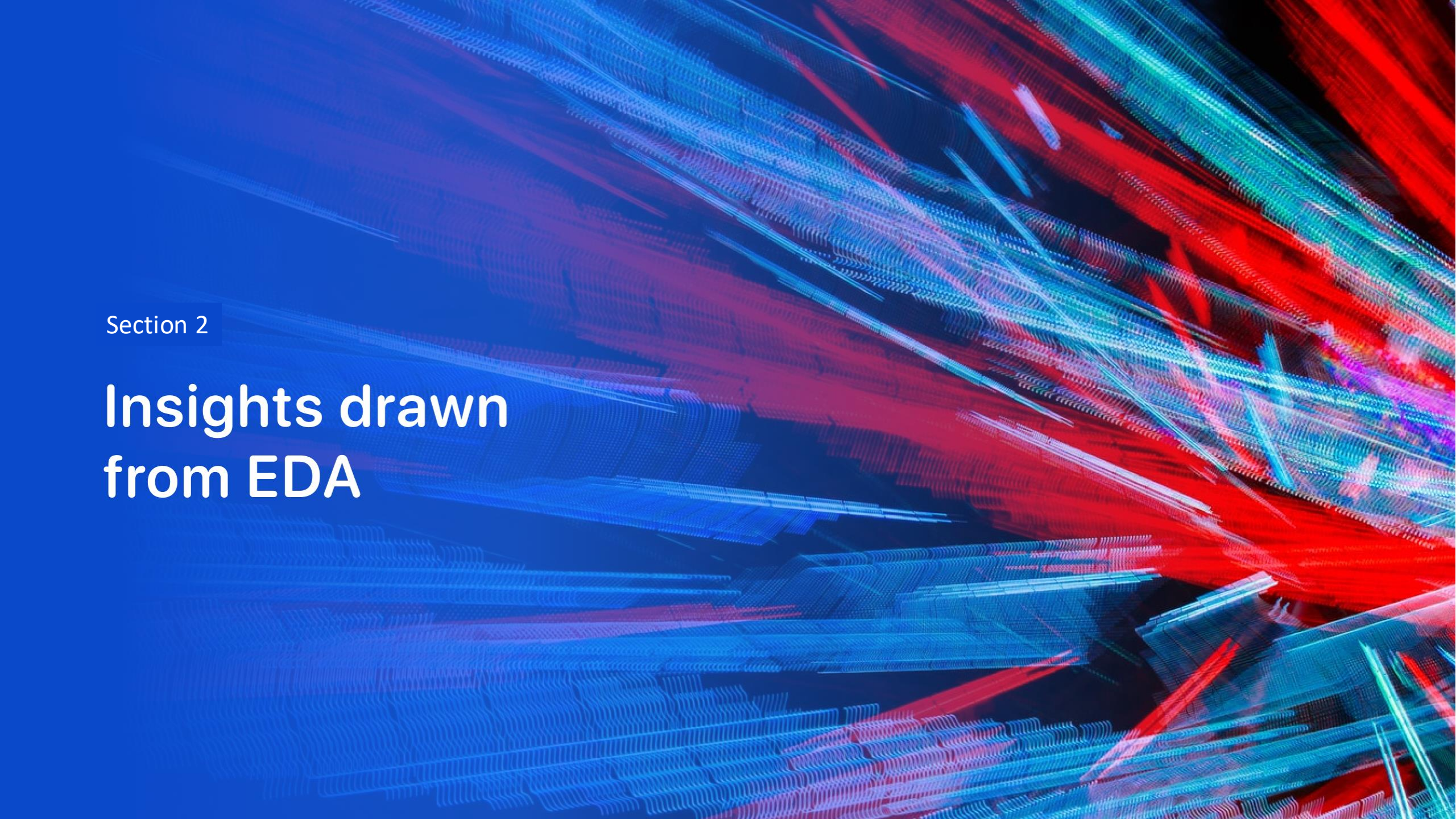
- Load the dataframe and create numpy array from the column class in the data
- Standardize the data in X and assign it variable Y
- Use train_test_split to split the X & Y data into test and training data
- Create a logistic object the create a GridSearchCV object
- Calculate the accuracy on the test data using score method
- Create a support vector machine object then create a GridSearchCV
- Calculate the accuracy on the test data using the method score
- Create a decision tree classifier object then create GridSearchCV Object
- Calculate the accuracy of tree-cv on the test data using the method score
- Create a K nearest neighbors object then GridSearchCV
- Calculate the accuracy of Knn-cv & find the method performs best

Results

EXPLORATORY DATA ANALYSIS RESULTS

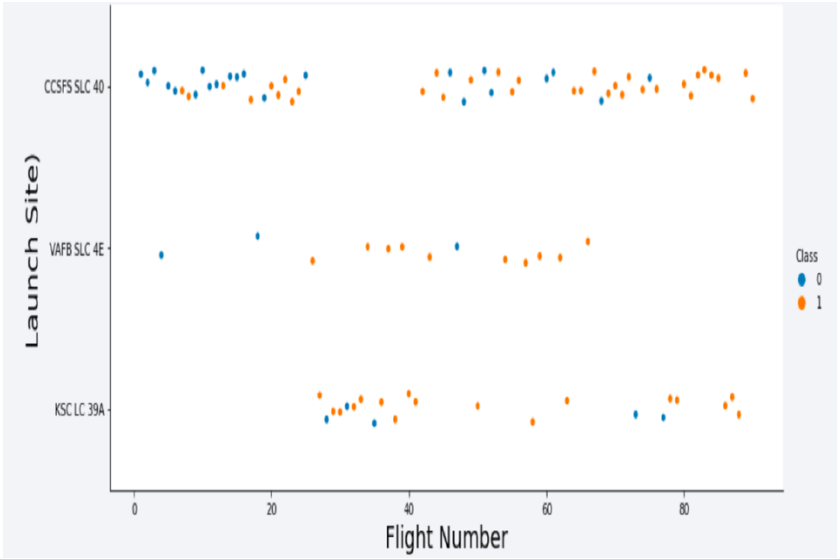
INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS

PREDICTIVE ANALYSIS RESULTS


Section 2
Insights drawn from EDA

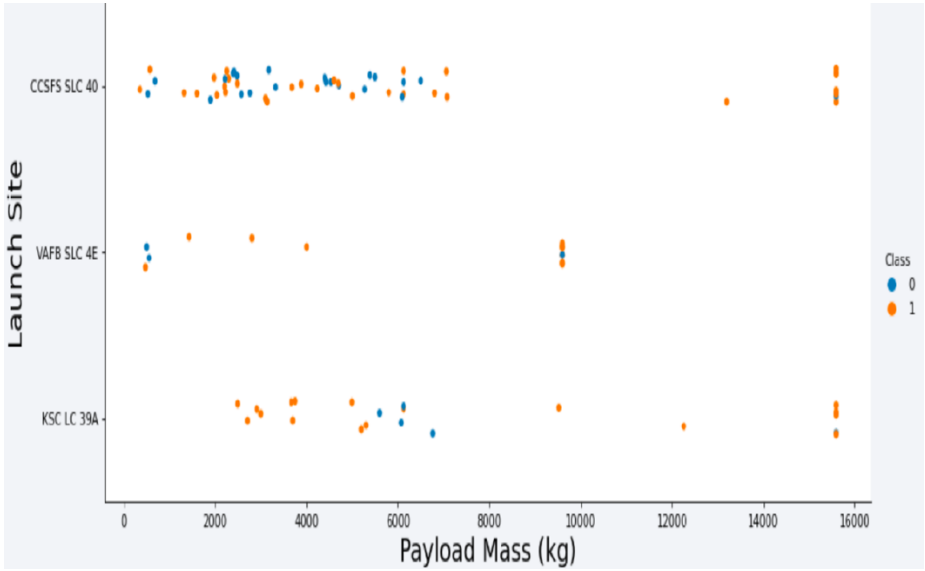
Flight Number vs. Launch Site

- The majority of the flights were launched from the CCAFS SLC 40 sites.
- The VAFB SLC 4E and KSC LC 39A sites have higher success rates than other sites.
- Newer flights have higher success rates than older flights.



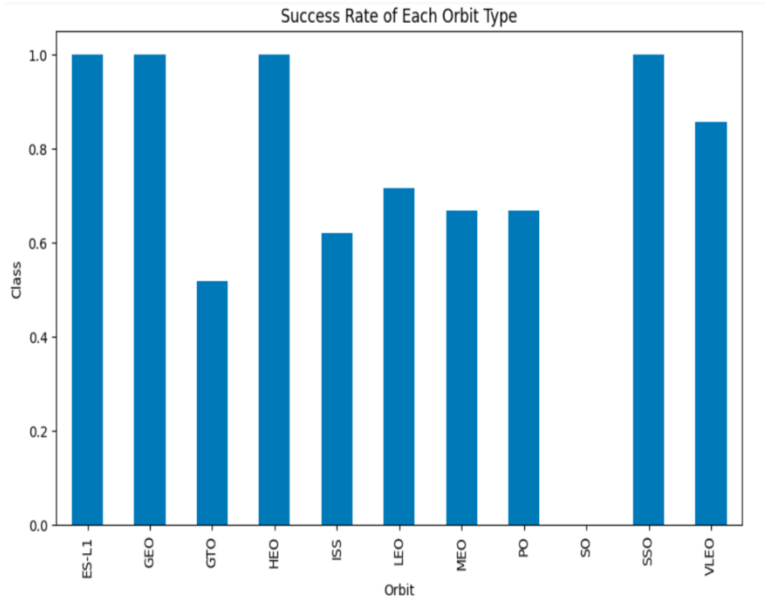
Payload vs. Launch Site

- The majority of the flights with payload mass above 7000 Kg were successful.
- KSC LC 39A success rate for payload mass under 5500 kg is 100%.
- For all launch sites the success rate is proportional to the payload mass.



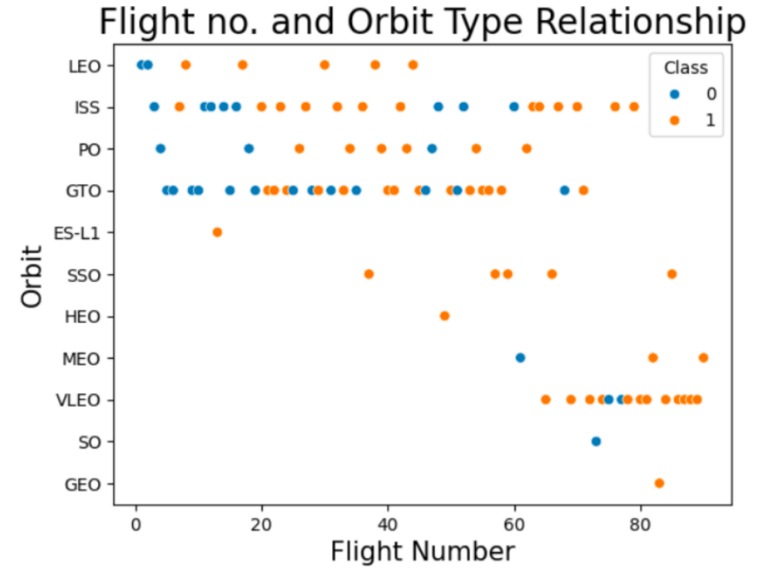
Success Rate vs. Orbit Type

- • The OS orbit has 0% success rate.
- • The ELS-1, GEO, HEO and SSO orbits have 100% success rate.
- • Orbits GTO, ISS, LEO, MEO and PO success rate is higher than 50% and less than 75%.



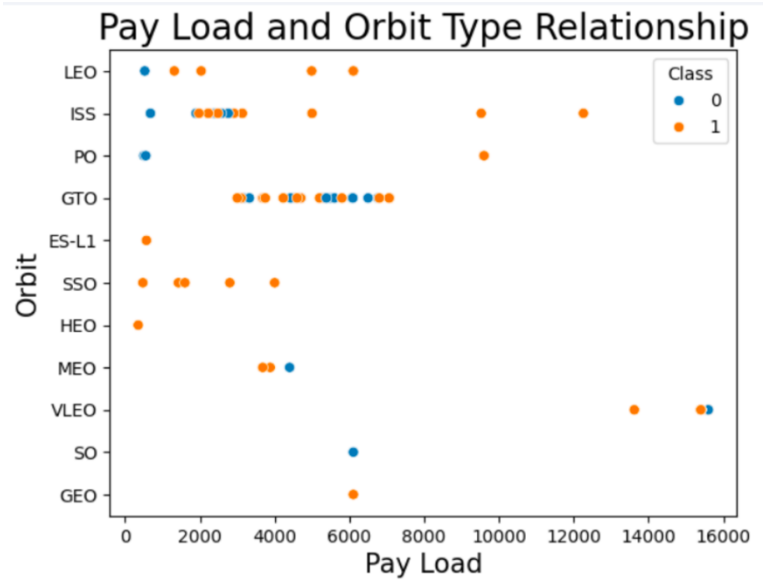
Flight Number vs. Orbit Type

- The majority of the flights were launches to the ISS and GTO orbits.
- The data suggests that there is no relationship between the flight number and the orbit type.



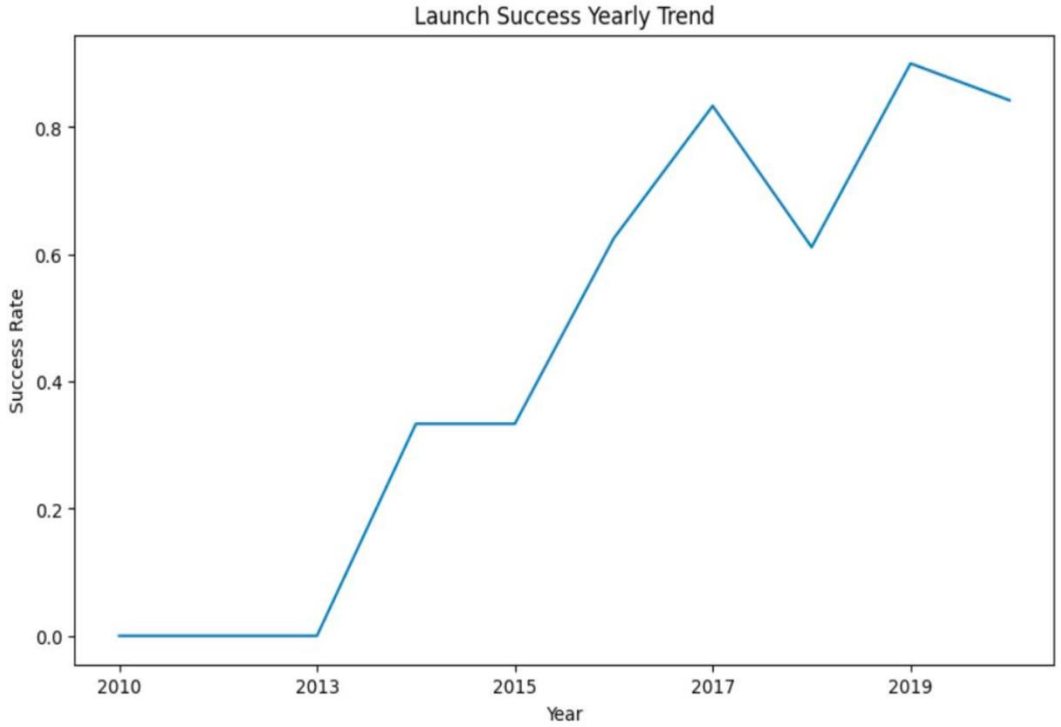
Payload vs. Orbit Type

- Payload masses above 10000 Kg were placed in PO, ISS and LEO orbits.
- Payload masses above 4000 and less than 8000 Kg were placed in the GTO orbit.



Launch Success Yearly Trend

- The launches success rate increased steadily since 2013.
- The increase in the success rate between 2013 and 2017 was linear.
- During 2018 there was a drop in the launches success rate.



All Launch Site Names

- The names of the unique launch sites and the query structure
- for obtaining these sites is shown below.

```
1 features= df[['LaunchSite']]
2 features.head()
```

	LaunchSite
0	CCAFS SLC 40
1	CCAFS SLC 40
2	CCAFS SLC 40
3	VAFB SLC 4E
4	CCAFS SLC 40

Launch Site Names Begin with 'CCA'

- 5 records for launch sites begin with the string 'CCA' and the query used for obtaining the information is shown below.

```
[ ]  1 %sql SELECT * FROM SPACEXTABLE 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
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The calculated total payload mass carried by boosters from NASA site =45596 Kg.
- The query for obtaining the total payload mass is shown below.

```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEXTABLE where "Customer" like 'NASA (CRS)%'
```

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

SUM(PAYLOAD_MASS__KG_)
48213

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1=2534.7 Kg.
- Furthermore, the query used to calculate the average payload mass carried by booster F9 v1.1 is shown below.

```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTABLE where "Booster_Version" LIKE 'F9 v1.1%'
```

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

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

- The first successful landing outcome on a ground pad was in 2015-12-22.
- The query for obtaining this result is shown below.

```
[ ]  1 %sql SELECT MIN(Date) AS first_successful_landing_date FROM SPACEXTABLE  WHERE landing_outcome = 'Success (ground pad)';
```

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

first_successful_landing_date
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 less than 6000

- The query used in obtaining this information is shown below.

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

```
[ ]  1 %sql SELECT Booster_Version FROM SPACEXTABLE WHERE landing_outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_
     2
```

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

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

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failed missions is as follows:

  Failure (in flight)= 1

  Successful number of flights= 98

- The query result is shown below.

```
[ ]  1 %sql SELECT mission_outcome, COUNT(*) AS total_count FROM SPACEXTABLE WHERE  mission_outcome IN ('Success', 'Failure (in flight)') GROUP BY mis
     2
```

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

Mission_Outcome	total_count
Failure (in flight)	1
Success	98

Boosters Carried Maximum Payload

- List of the boosters which have carried the maximum payload mass are shown below.
- The query used in obtaining the booster names is shown below.

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

```
[ ]   1 %sql SELECT booster_version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX (PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
      2
```

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

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

- List of the failed”landing_outcomes” in drone ship, their booster version, and the launch site name during year 2015 is shown below.
- The query used in obtaining the information is shown below.

```
[ ]  1 %sql SELECT CASE WHEN substr(Date, 6, 2) = '01' THEN 'January' WHEN substr(Date, 6, 2) = '02' THEN 'February' WHEN substr(Date, 6, 2) = '03' TH
     2
```

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

month_name	Booster_Version	Launch_Site	Landing_Outcome
January	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
April	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- 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

- The query used to obtain the results is shown below.

```
[ ]  1 %sql SELECT landing_outcome, COUNT(*) AS outcome_count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing_outcc
     2
```

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

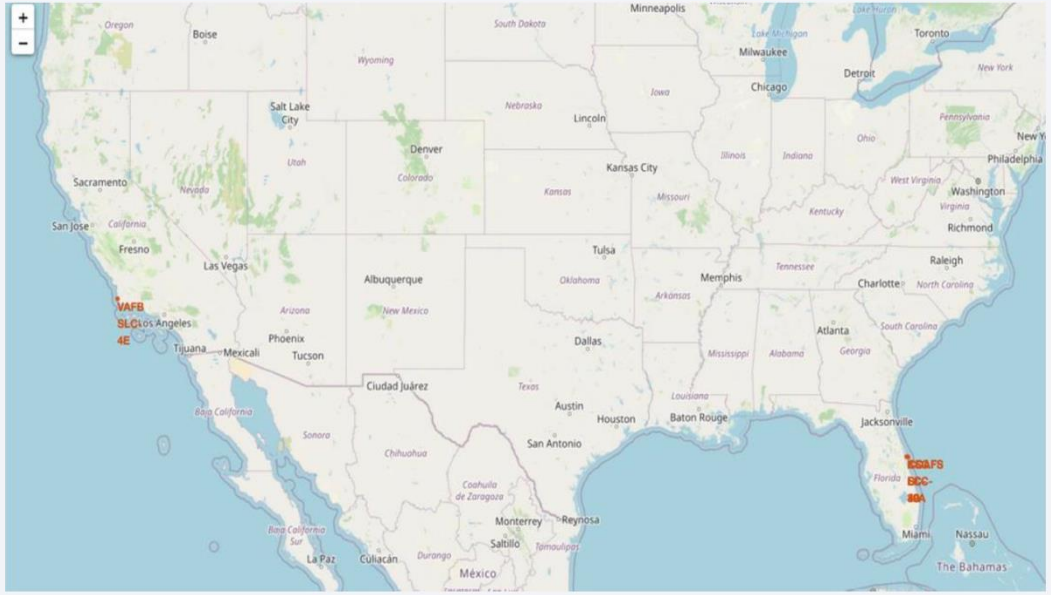
Landing_Outcome	outcome_count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

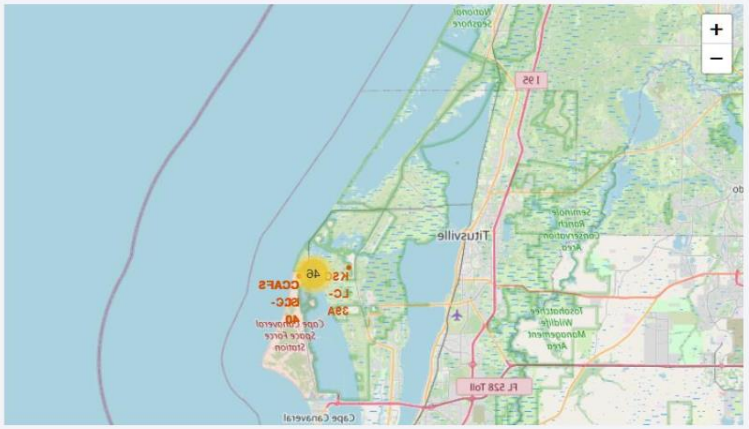
Section 3

Launch Sites Proximities Analysis

USA Launch Sites in California and Florida

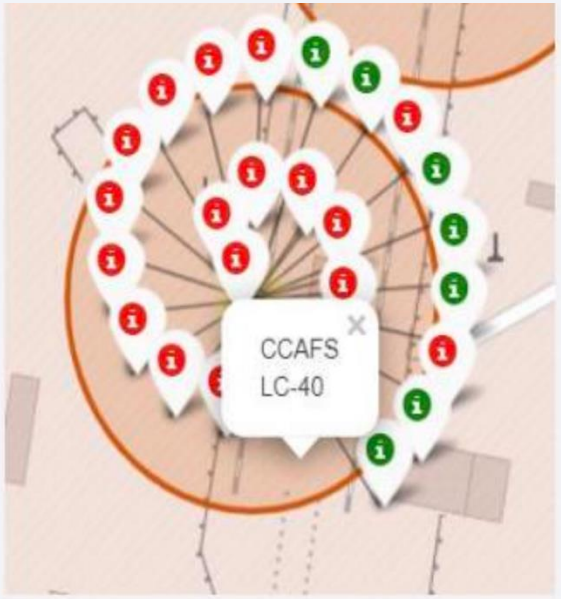
- Most of Launch sites considered in this project are in proximity to the Equator line. Launch sites are made at the closest point possible to Equator line, because anything on the surface of the Earth at the equator is already moving at the maximum speed (1670 kilometers per hour). For example launching from the equator makes the spacecraft move almost 500 km/hour faster once it is launched compared half-way to north pole.
- All launch sites considered in this project are in very close proximity to the coast While starting rockets towards the ocean we minimize the risk of having any debris dropping or exploding near people.



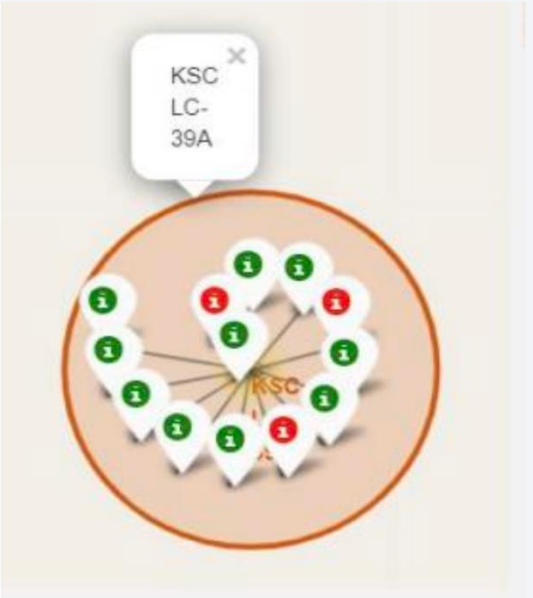


Color Labels Showing the Launch Sites on a Map

	Launch Site	Lat	Long	marker_color
0	CCAFS LC-40	28.562302	-80.577356	red
1	CCAFS LC-40	28.562302	-80.577356	red
2	CCAFS LC-40	28.562302	-80.577356	red
3	CCAFS LC-40	28.562302	-80.577356	red
4	CCAFS LC-40	28.562302	-80.577356	red
5	CCAFS LC-40	28.562302	-80.577356	red
6	CCAFS LC-40	28.562302	-80.577356	red
7	CCAFS LC-40	28.562302	-80.577356	red
8	CCAFS LC-40	28.562302	-80.577356	red
9	CCAFS LC-40	28.562302	-80.577356	red
10	CCAFS LC-40	28.562302	-80.577356	red
11	CCAFS LC-40	28.562302	-80.577356	red
12	CCAFS LC-40	28.562302	-80.577356	red
13	CCAFS LC-40	28.562302	-80.577356	red
14	CCAFS LC-40	28.562302	-80.577356	red
15	CCAFS LC-40	28.562302	-80.577356	red
16	CCAFS LC-40	28.562302	-80.577356	red
17	CCAFS LC-40	28.562302	-80.577356	green
18	CCAFS LC-40	28.562302	-80.577356	green
19	CCAFS LC-40	28.562302	-80.577356	red
20	CCAFS LC-40	28.562302	-80.577356	green



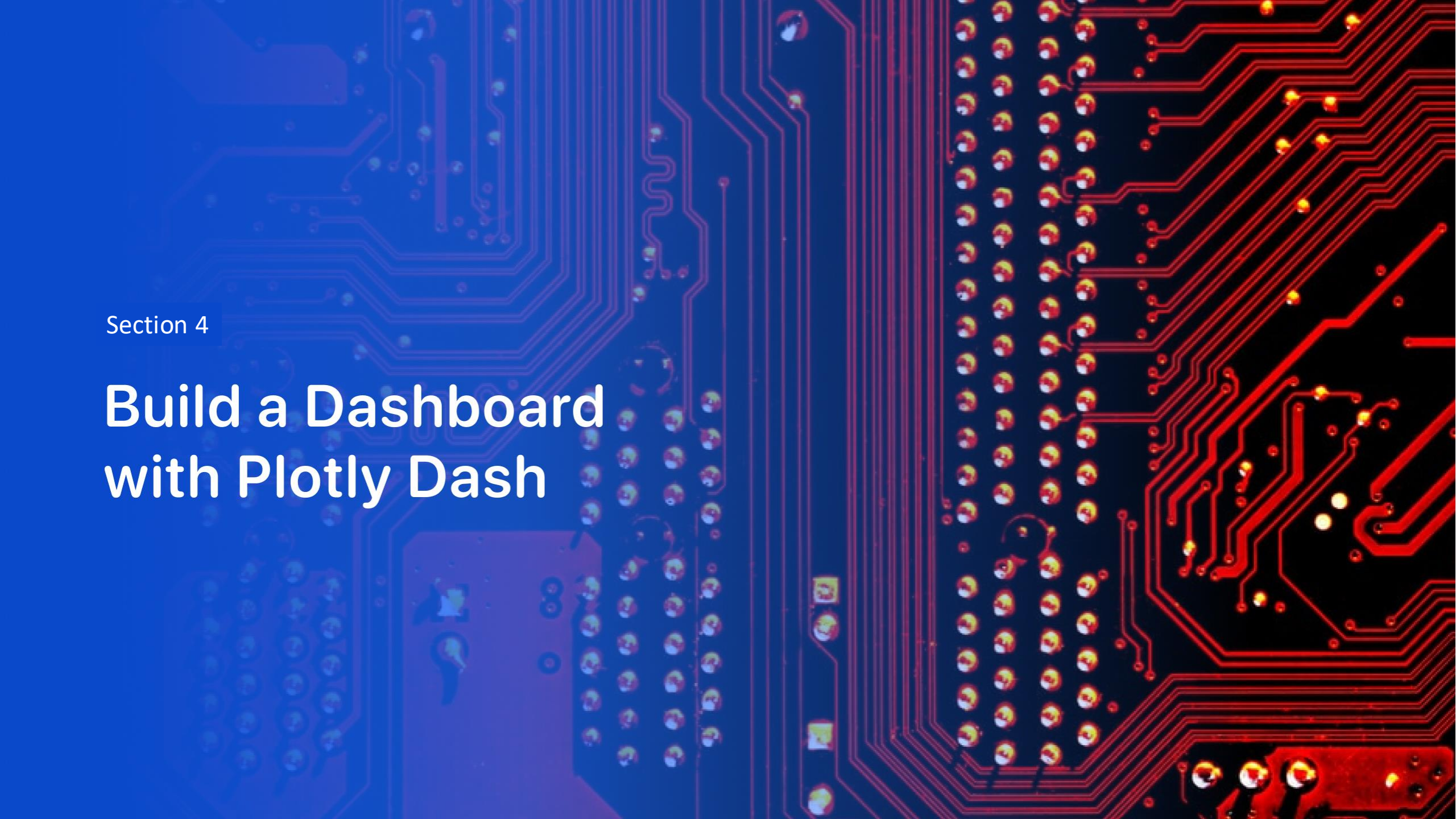
36	KSC LC-39A	28.573255	-80.646895	green
37	KSC LC-39A	28.573255	-80.646895	red
38	KSC LC-39A	28.573255	-80.646895	green
39	KSC LC-39A	28.573255	-80.646895	green
40	KSC LC-39A	28.573255	-80.646895	red
41	KSC LC-39A	28.573255	-80.646895	green
42	KSC LC-39A	28.573255	-80.646895	green
43	KSC LC-39A	28.573255	-80.646895	red
44	KSC LC-39A	28.573255	-80.646895	green
45	KSC LC-39A	28.573255	-80.646895	green
46	KSC LC-39A	28.573255	-80.646895	green
47	KSC LC-39A	28.573255	-80.646895	green
48	KSC LC-39A	28.573255	-80.646895	green



Green= Successful Launch
Red= Failed Launch

Safe Distance to Launch Site

The obtained results indicate that all launch sites are at safe distance from railway lines and cities.

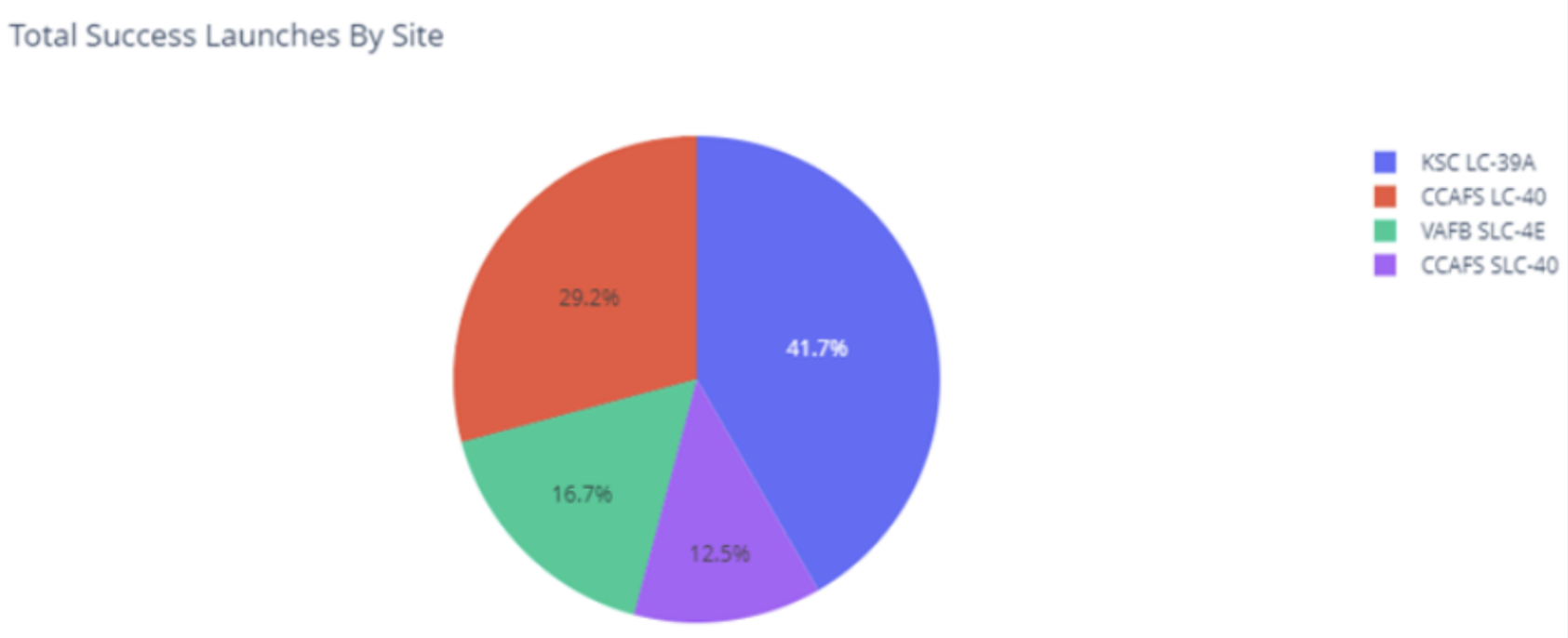

Section 4
Build a Dashboard with Plotly Dash

Total Launch Success for All Sites

- The highest success launch rates were recorded at these sites :

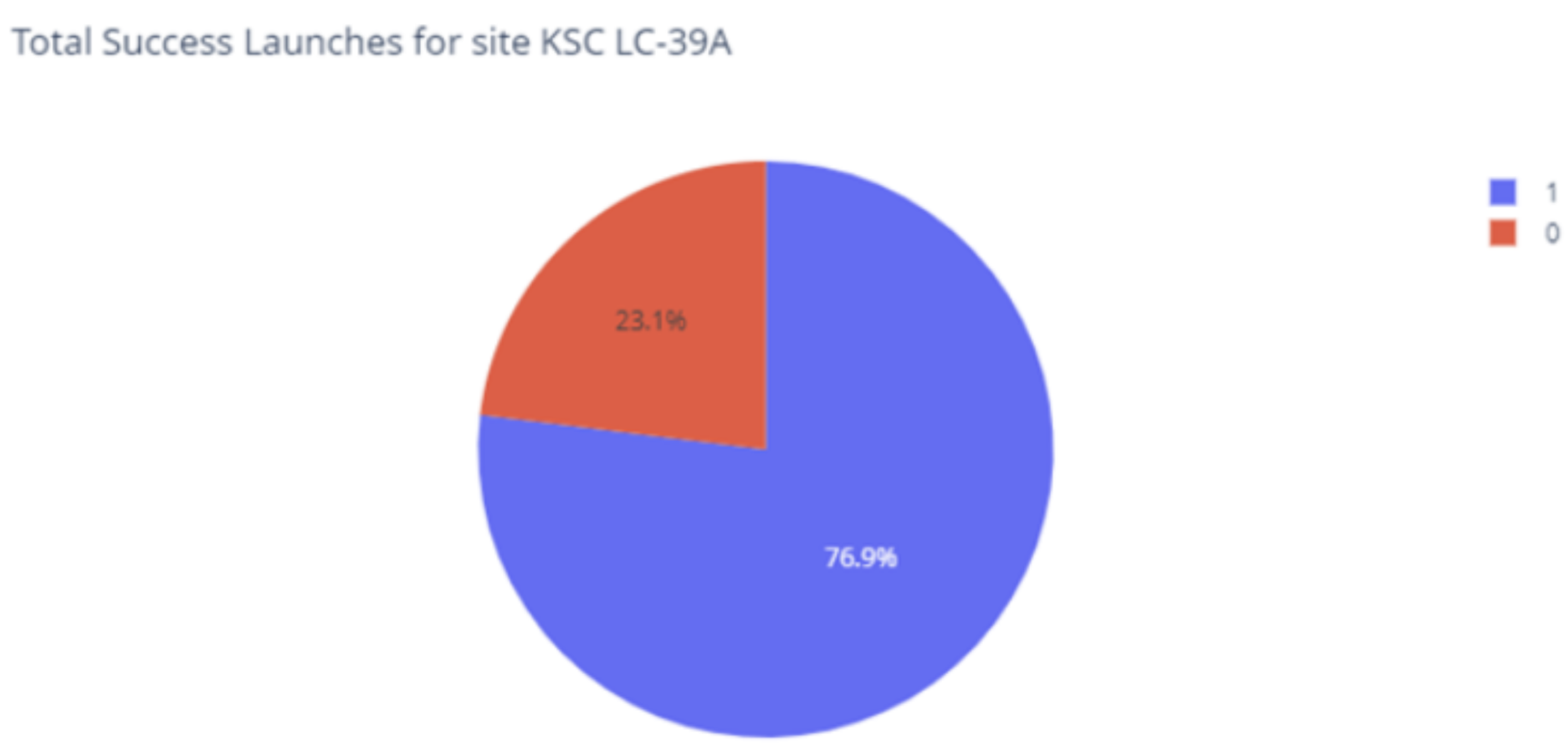
➢KSC LC-39A (41.7%)

➢CCAFS LC-40 (29.2%)



KSC LC-39 Launch Site Success Rate

- Site KSC LC-39 success rate is 76.9%



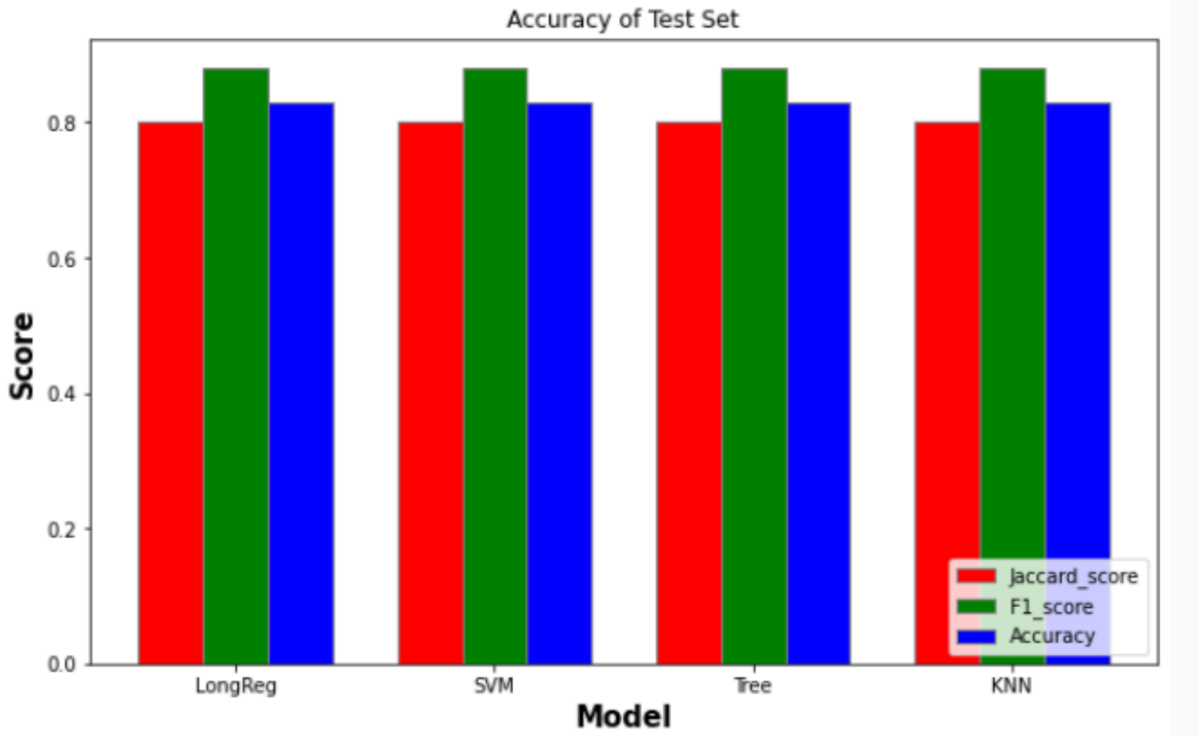
Payload vs. Launch Outcome for All Sites

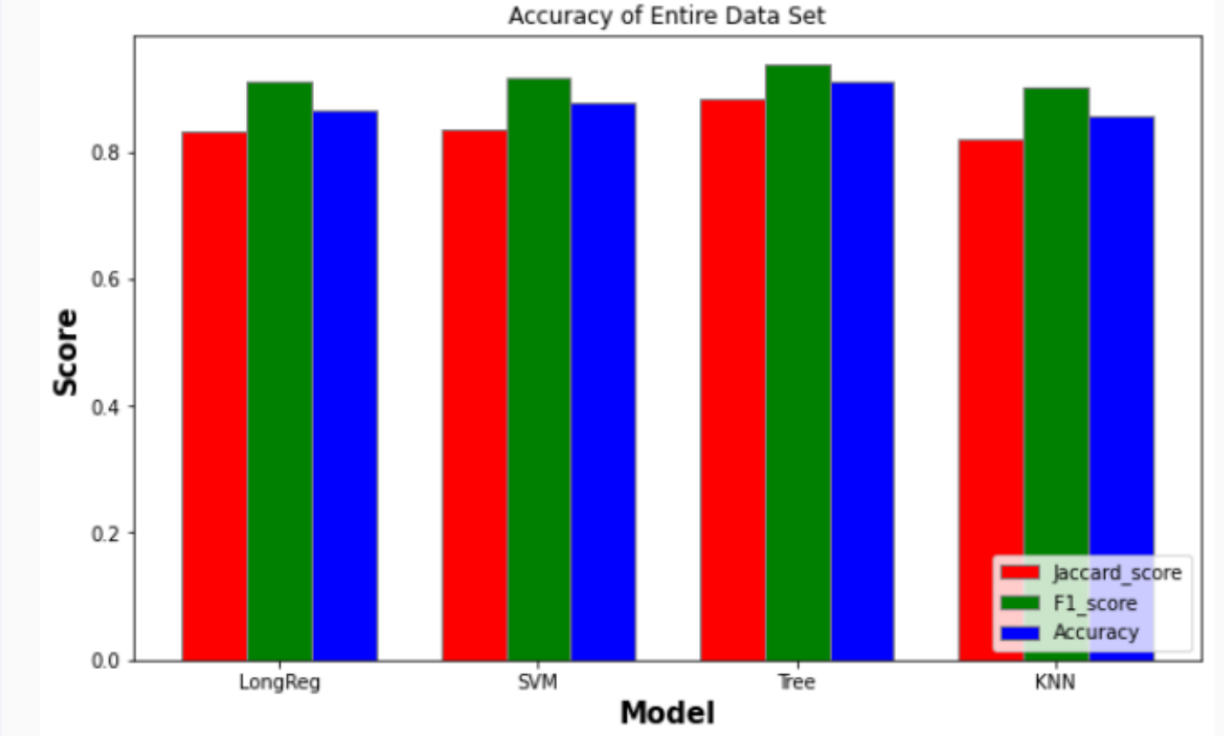
Highest success rate for payloads is between 2000 and 5500 Kgs

Section 5

Predictive Analysis (Classification)

Classification Accuracy

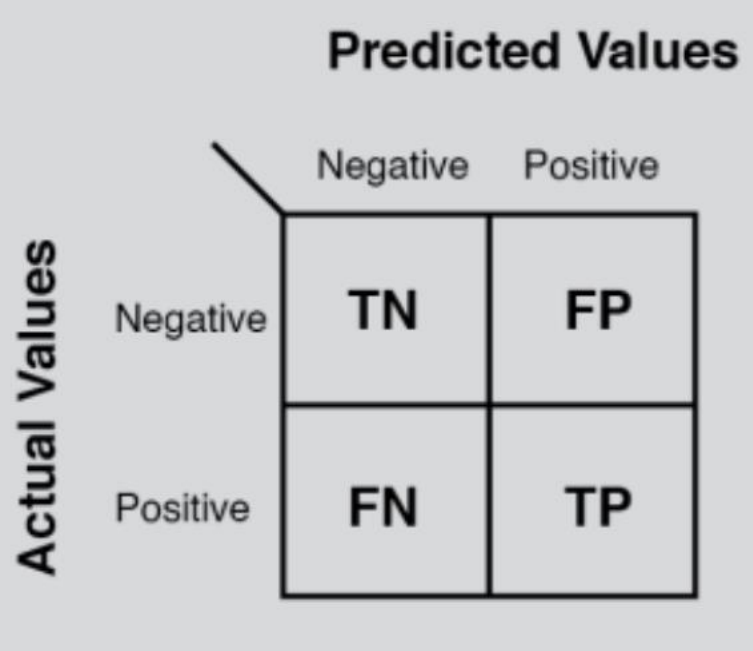


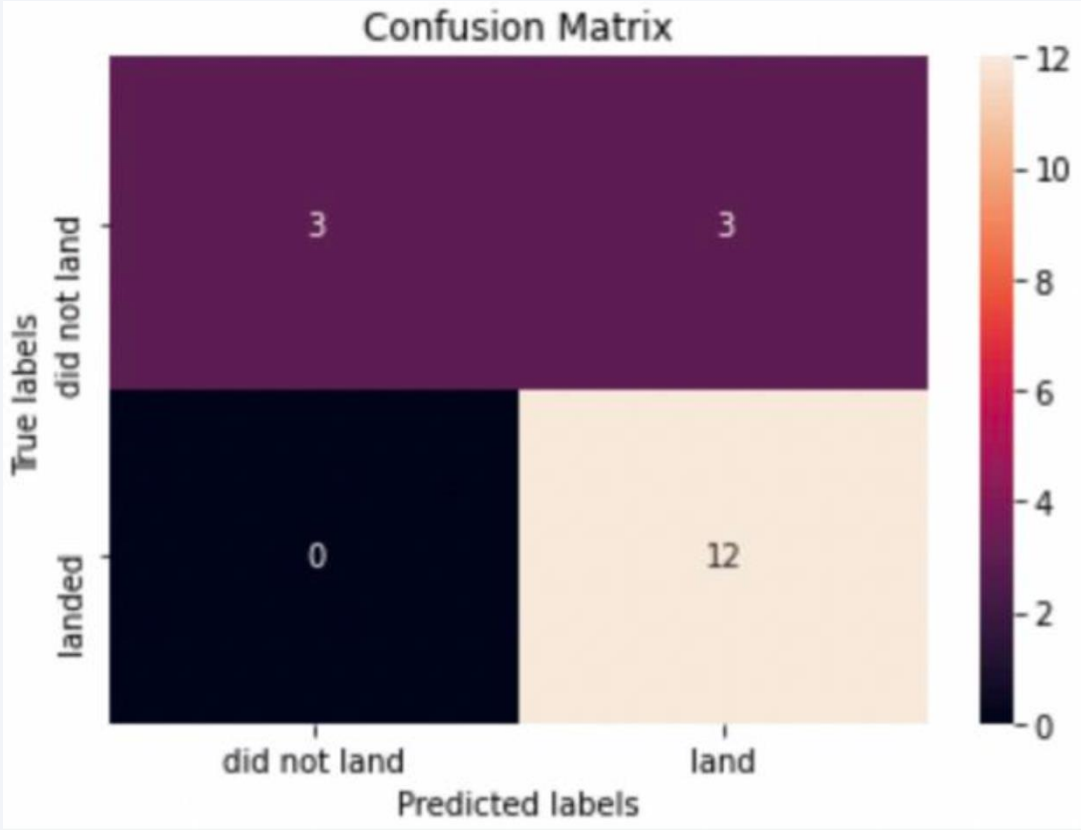


- Using the test set the same accuracy results were obtained from the four models.
- The Tree Model provided the best accuracy results for the entire data set.

Confusion Matrix

- The confusion matrix analysis suggests that the best performing model is the Logistic Regression model.
- The confusion matrix predicts 13 true positives, 3 false positives, 3 true positive, and 0 false negative.





Conclusions

- The success rate for the rocket launches increased after 2013.
- Orbits GEO, HEO, ES-L1 and SSO have 100% launch success rate.
- Launch site KSC LC-39A has the highest success rate.
- The Decision Tree model is the best ML algorithm for analyzing the SpaceX data set and provided the best accuracy results.

Appendix

https://github.com/godwinchigozie679/SpaceX-falcon-9-Exploring-and-Preparing-Data

https://github.com/godwinchigozie679/Assignment-SQL-Notebook-for-Peer-Assignment-

https://github.com/godwinchigozie679/SpaceX-Falcon-9-First-Stage-Landing-Prediction-Assignment-Exploring-and-Preparing-Data

https://github.com/godwinchigozie679/Hands-on-Lab-Interactive-Visual-Analytics-with-Folium

https://github.com/godwinchigozie679/SpaceX-Launch-Records-Dashboard

https://github.com/godwinchigozie679/Space-X-Falcon-9-First-Stage-Landing-Prediction

https://github.com/godwinchigozie679/Space-X-Falcon-9-First-Stage-Landing-Prediction-Data-Wrangling-Lab2

https://github.com/godwinchigozie679/Web-scraping-Falcon-9

Thank you!