IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

This project analyzes SpaceX launch data to identify success factors, launch site, and site utilization trends. Utilizing data from the SpaceX API, this presentation will show the results established from the data wrangling, exploratory data analysis with SQL and visual tools, and predictive modeling. Key findings through these processes reveal patterns in the SpaceX Data that can be confirmed through the use of Decision Tree modeling.

These trends will be useful for identifying the options that best position the launch for success, with regards to launch site, payload mass, and orbit path.

Introduction

- Project Background
  - This presentation analyzes SpaceX launch data to uncover the trends and insights to identify formal relationships between the variables tracked.
- Problems to Address
  - The goal of this presentation is to establish an understanding of the factors influencing launch success rates and site utilization.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using the SpaceX API, followed by the necessary data processing required for wrangling
- Perform data wrangling
  - Data was processed by sending the JSON response from the API through functions from the pandas library to create a data frame that could be wrangled
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

SpaceX Data

- SpaceX tracks and saves every detail of launches and their respective outcomes

SpaceX API

- Access the SpaceX Data through the SpaceX API

Data Scraping

- Convert the JSON response into a workable data frame to prep the data for wrangling

Data Collection – SpaceX API

Call the API

- Import Libraries
- Request and access JSON file

Confirm Status Code

- Check for status code to be `200` before continuing

Access and Normalize Data

- Normalize the data frame
- Save normalized data frame as variable for data wrangling

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

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

Check the content of the response

```
print(response.content)
```

```
response.status_code
```

```
200
```

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


GitHub: https://github.com/washman4133/capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

- Import Libraries
  - requests
  - pandas
  - numpy
  - datetime

Import Libraries

Helper Functions

- getBoosterVersion
- getLaunchSite
- getPayloadData
- getCoreData

- normalize(response.json())

Convert JSON to data frame

Extract data

- data[['rocket','payloads’, ’launchpad','cores’, 'flight_number’, 'date_utc']]

```python
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```


GitHub: https://github.com/washman4133/capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Wrangling

Deal With Missing Values

- Replace missing PayloadMass values with .mean()

Calculations for Context

- Number of launches per site
- Number and occurrence of each orbit
- Number and occurrence of mission outcome of orbits

Create New Labels

- Landing outcome label

Filter Unwanted Data

- Filter by Falcon number and save to new, clean data frame

```
# Apply value_counts() on column LaunchSite
launch_counts = df['LaunchSite'].value_counts()
print(launch_counts)
```

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

```
# Apply value_counts on Orbit column
orbit_counts = df['Orbit'].value_counts()
print(orbit_counts)
```

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

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

```
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: Outcome, 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`:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
```


GitHub: https://github.com/washman4133/capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Flight Number vs. Launch Site
  - Compare launches over time to look for a relationship between launches and launch site over time
- Payload vs. Launch Site
  - Compare launch sites by payload masses launched
- Success Rate vs. Orbit Type
  - Compare launch outcome success by orbit type to identify the most successful orbit paths
- Flight Number vs. Orbit Type
  - Compare launches over time to look for a relationship between launches and orbit type over time
- Payload vs. Orbit Type
  - Compare payload mass by orbit type to look for a relationship between weight and orbit
- Launch Success Yearly Trend
  - Compare launch outcome success over time

GitHub: https://github.com/washman4133/capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- All Launch Site Names
- Launch Site Names Begin with ‘CCA’
- Total Payload Mass
- Average Payload Mass by F9 v1.1
- First Successful Ground Landing Date
- Successful Drone Ship Landing with Payload between 4,000 and 6,000
- Total Number of Successful and Failure Mission Outcomes
- Boosters Carried Maximum Payload
- 2015 Launch Records
- Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

GitHub: https://github.com/washman4133/capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Launch Site Locations
  - Shows markers for each launch site across the United States, which should show locations in the southern United States, towards the Equator
- Color-Labeled Launch Outcomes
  - Color-labeled markings show the launch outcomes at each launch site
- Proximity PolyLines
  - Lines show the distance to the closest major population area, rail, and highway
  - Distance is also calculated to the major population area, rail, and highway, which should reflect close infrastructure and a far distance from major population centers

GitHub: https://github.com/washman4133/capstone/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Predictive Analysis (Classification)

Standardize Data

- Create an array and fill using .to_numpy()
- Use SciKit to standardize data
  - preprocessing.StandardScaler()
  - .fit_transform()

Split data into training/test sets

- train_test_split() function
- X_train, Y_train
- X_test, Y_test

Apply Models to Data

- Create parameters, apply model functions
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree
  - K Nearest Neighbors

Analyze Models

- Check accuracy using .best_estimator_.score
- Create and analyze Confusion Matrix

GitHub: https://github.com/washman4133/capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

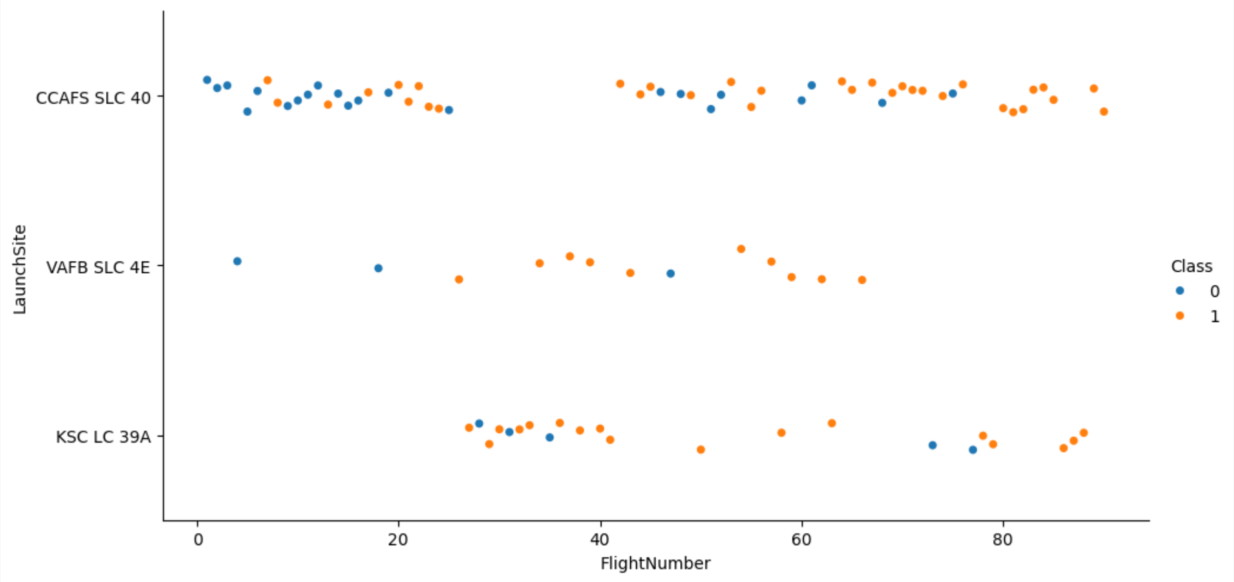
- Exploratory data analysis results
  - Launch Outcomes show rapid improvements between 2013 and 2019, from 0% to 90%+
  - CCAFS SLC 40 is the most utilized Launch Site, KSC LC 39A seems to have been used as a temporary launch site when CCAFS SLC 40 is unusable, and VAFB SLC 4E has rarely been utilized
  - Four orbits have 100% success rates: ES-L1, GEO, HEO, and SSO
- Interactive analytics demo in screenshots
  - CCAFS SLC 40 and KSC LC 39A are closest to the equator, impacting utilization
  - CCAFS SLC 40 and KSC LC 39A are close to infrastructure, including highways and rail
  - CCAFS SLC 40 and KSC LC 39A are almost 30 miles from any major metropolitan area
- Predictive analysis results
  - The Decision Tree is the best predictive model for the dataset

Section 2

Insights drawn from EDA

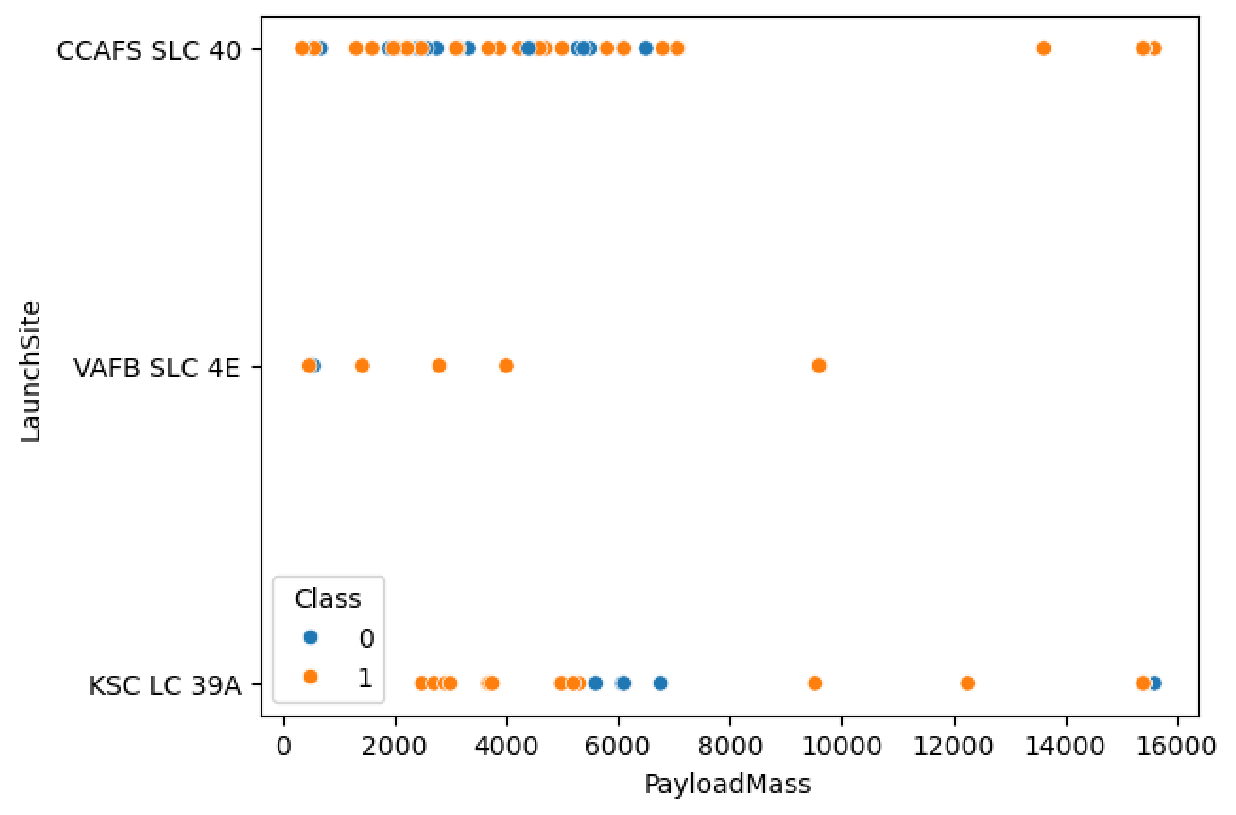
Flight Number vs. Launch Site

- CCAFS SLC 40: The most often used launch site both early on and later in the launches
- VAFB SLC 4E: The least used launch site, located furthest from the equator out of the three launch sites
- KSC LC 39A: Located next to CCAFS SLC 40, KSC LC 39A seems to be a temporary launch site for when CCAFS SLC 40 is not in service



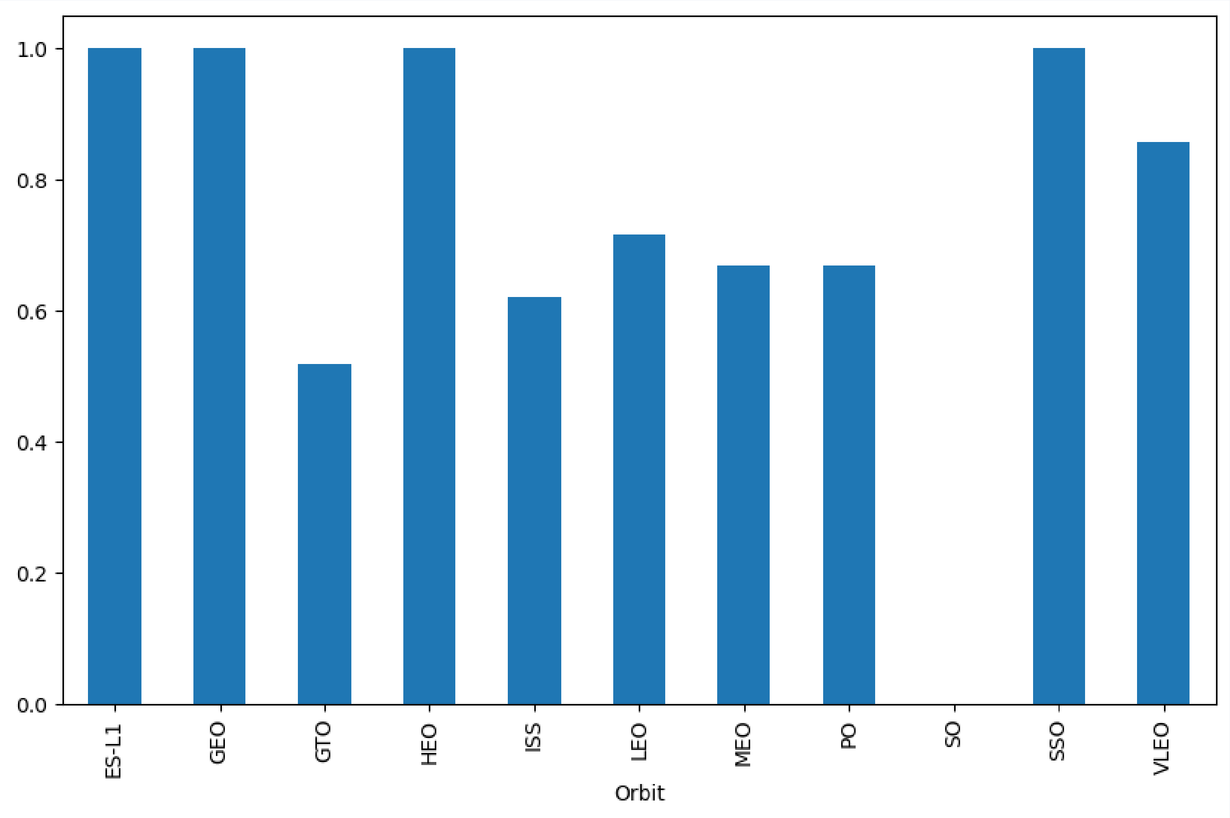
Payload vs. Launch Site

- CCAFS SLC 40 and KSC LC 39A have similar payload use ranges, with the majority of payloads coming in less than 7,000kg
- VAFB SLC 4E has only seen lower mass launches
- CCAFS SLC 40 has had the most success with higher mass launches



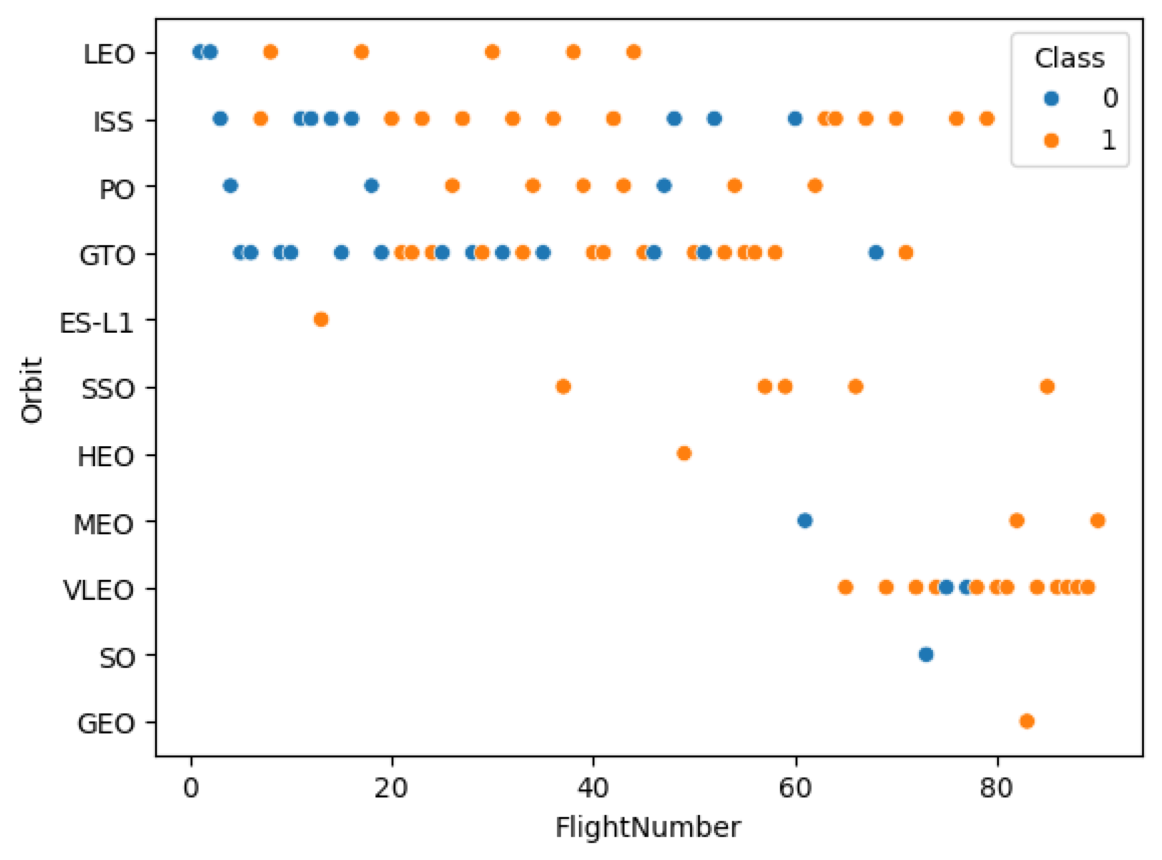
Success Rate vs. Orbit Type

- Orbit types ES-L1, GEO, HEO, and SSO all have perfect success rates
- Orbit type GTO has an approximate 50% success rate
- Success has never been achieved with orbit type SO



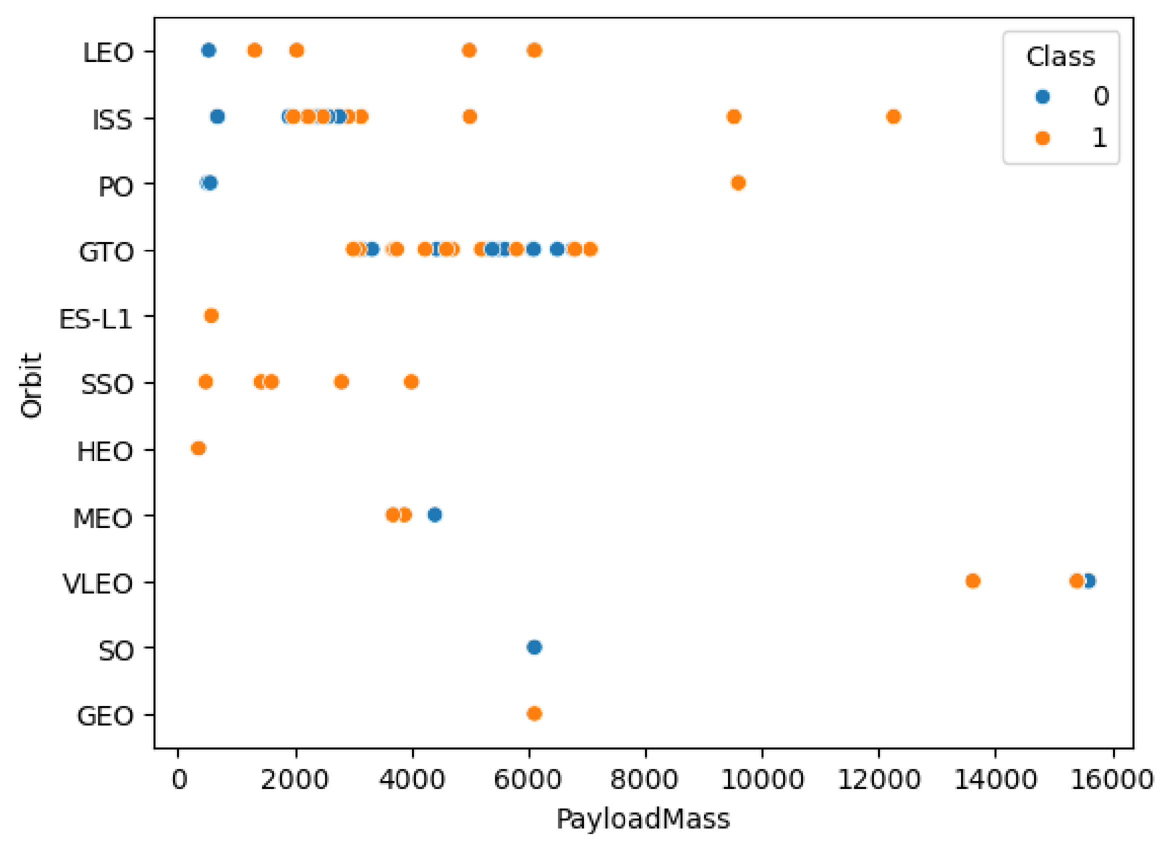
Flight Number vs. Orbit Type

- Orbit Type GTO, which has about a 50% success rate, represents the plurality of orbit type chosen for launch in the first half of launches
- Orbit Type VLEO, which has about a 90% success rate, represents the plurality of orbit type chosen in the last two quintiles of flight numbers



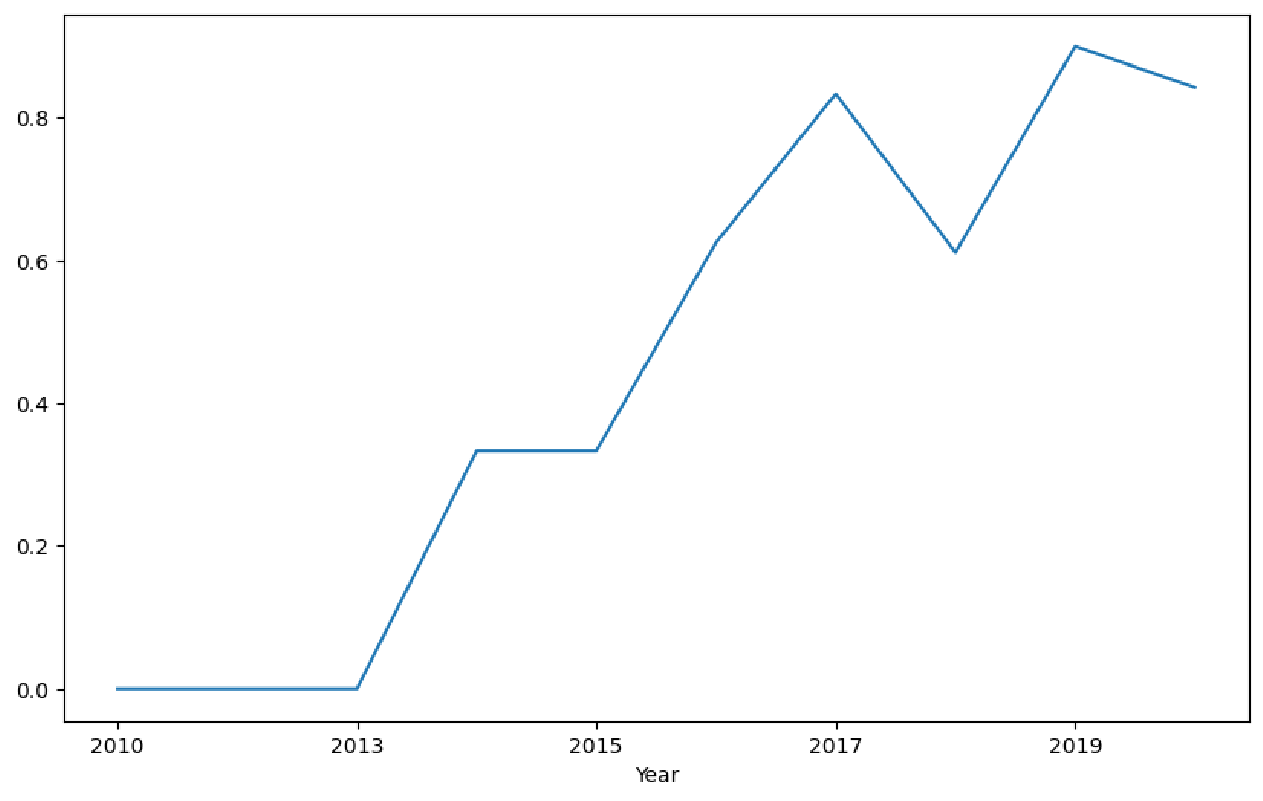
Payload vs. Orbit Type

- The majority of launches with between 2,000 and 4,000 PayloadMass followed the ISS orbit type
- The majority of launches between 4,000 and 10,000 PayloadMass followed the GTO orbit type



Launch Success Yearly Trend

- Early launches found no success
- From 2013 to 2019, there was substantial growth of launch success from 0% to near perfect
- The downward trend since 2019 does not represent a significant decrease in success



All Launch Site Names

- The DISTINCT function in SQL identifies only unique values in a range
- The DISTINCT function is being used on the Launch_Site column in data table SPACEXTABLE to find launch site names

Task 1

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

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

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

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

Launch Site Names Begin with 'CCA'

- The WHERE function in SQL acts as a filter to focus the line of code on a specific, targeted segment of data
- The WHERE function is being used to return all column data in table SPACEXTABLE where launch site names begin with ‘CCA’

Task 2

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

```
%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 SUM function in SQL adds up the values of the range given in the parenthesis
- The SUM function is being used to return the sum of data in table SPACEXTABLE where the customer name begin equals ‘NASA (CRS)’

Task 3

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)';
```

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

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

- The AVG function in SQL finds the average of values of the range given in the parenthesis
- The AVG function is being used to return the average Payload Mass (kg) where the Booster rocket used was ‘F9 v1.1’

Task 4

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';
```

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

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

- The MIN function in SQL finds the minimum value of the range given in the parenthesis
- The MIN function is being used to return the earliest date (minimum date value) where the Landing Outcome was a success

Task 5

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

Hint:Use min function

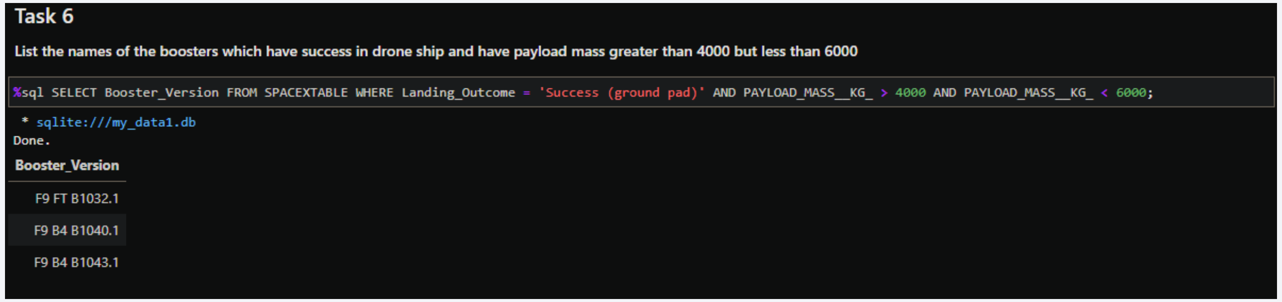
```
%sql SELECT MIN(Date) as First_Successful_Landing FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';
```

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

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Booster Variants that have successfully landed with between 4,000 and 6,000 Payload
  - F9 FT B1032.1
  - F9 B4 B1040.1
  - F9 B4 B1043.1



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

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

- Combining COUNT with GROUP BY allows for the created output to create rows based on the GROUP BY (Mission Outcome) and count the values associated with those Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT Mission_Outcome, COUNT(*) as Number_of_Missions FROM SPACEXTABLE GROUP BY Mission_Outcome
```

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

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

Boosters Carried Maximum Payload

- Using SELECT MAX() allows for us to see the boosters that have carried max payload

F9 B5 1048.4	F9 B5 1049.4
F9 B5 1051.3	F9 B5 1056.4
F9 B5 1048.5	F9 B5 1051.4
F9 B5 1049.5	F9 B5 1060.2
F9 B5 1058.3	F9 B5 1051.6
F9 B5 1060.3	F9 B5 1049.7

Task 8

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

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

```
 * 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

- This query checks for launches in 2015 by Booster Version F9 v1.1
- Both F9 v1.1 boosters used CCAFS LC-40 Launch Site
- Both F9 v1.1 boosters failed

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.

```
%sql SELECT substr(Date, 6, 2) AS Month, Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTABLE
```

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

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

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

- This query provides the count for Landing Outcomes between the dates of June 4th, 2010 and March 20th, 2017, in descending order

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.

```
%sql SELECT Landing_Outcome, COUNT(*) as Outcome_Count FROM SPACEXTABLE WHERE Date >= '2010-06-04' AND
```

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

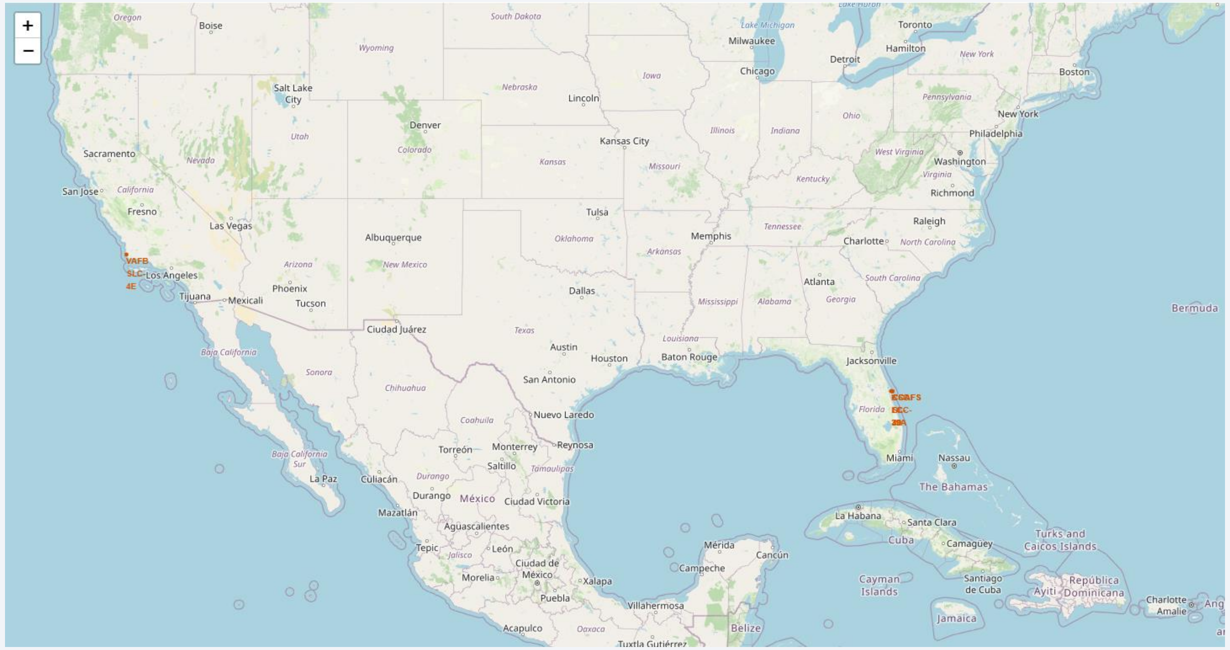
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

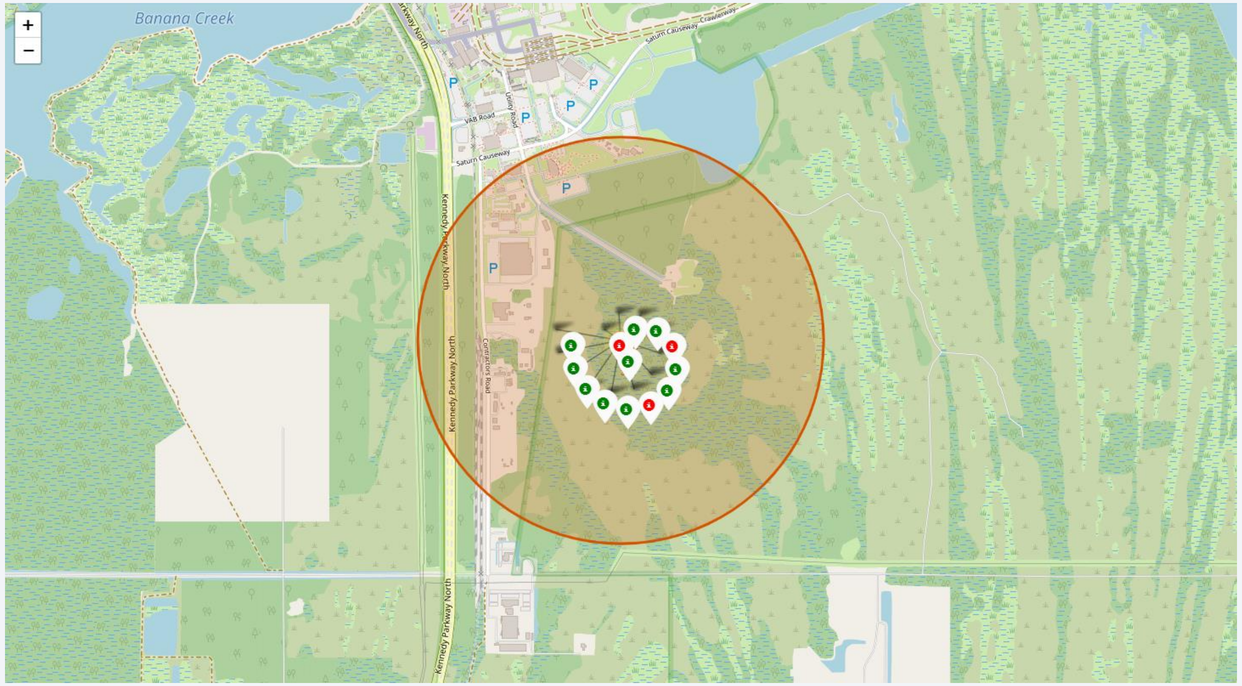
Folium: Launch Site Locations

- This map shows the three Launch Sites (VAFB SLC 4E, CCAFS SLC 40, KSC LC 39A) with markers.
- VAFB SLC 4E is located in Southern California
- CCAFS SLC 40 and KSC LC 39A are located next to each other in Florida



Folium: Color-Labeled Launch Outcomes

- This map shows the Launch Outcomes at KSC LC-39A
- Green represents successful launch outcomes
- Red represents failed launch outcomes



Folium: Proximity PolyLines

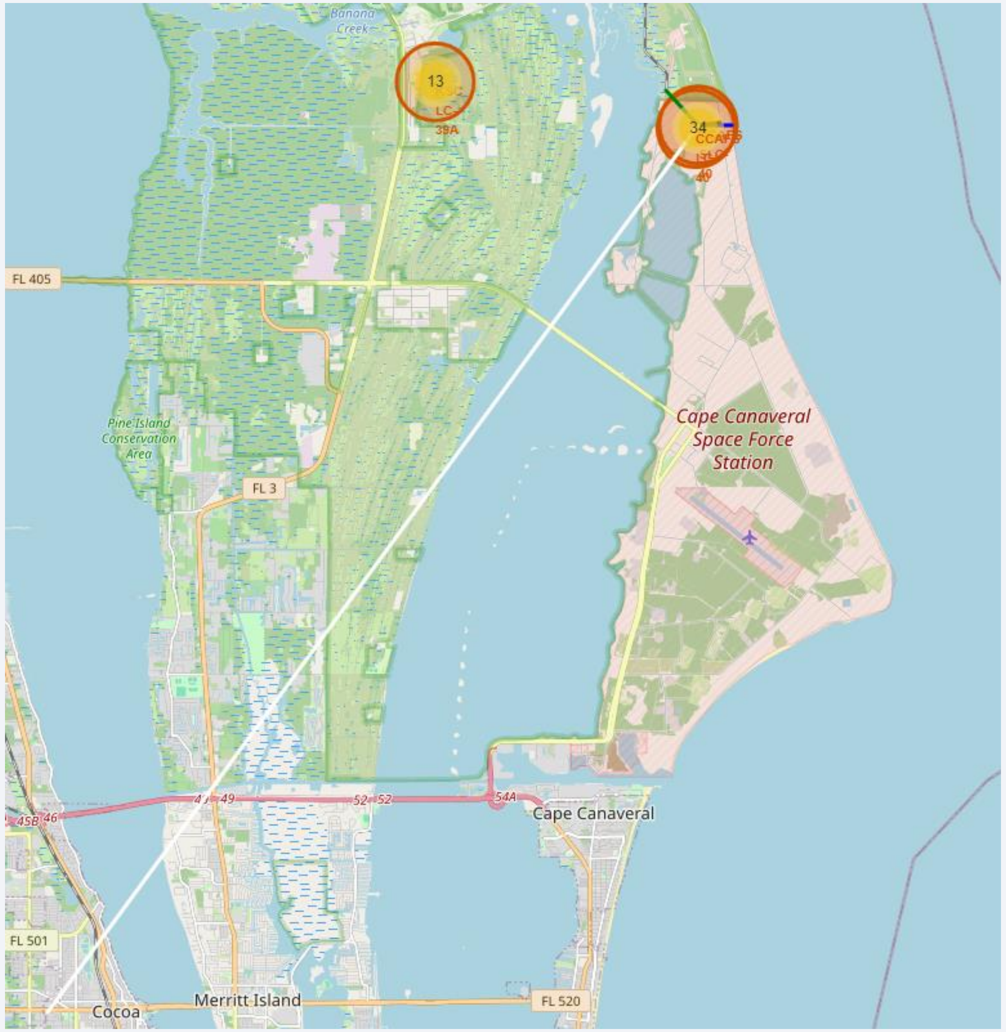
- This map shows the PolyLines from CCAFS SLC 40 to a city, rail, and highway
  - White – Cocoa, Florida
  - Green - Rail
  - Blue - Highway

```
distance_rail = calculate_distance(launch_lat, launch_long, rail_coord_lat, rail_coord_long)
print(distance_rail)

distance_high = calculate_distance(launch_lat, launch_long, high_coord_lat, high_coord_long)
print(distance_high)

distance_city = calculate_distance(launch_lat, launch_long, city_map_coord_lat, city_map_coord_long)
print(distance_city)
```

```
1.2324077761382506
0.5873408207094963
28.692187370293524
```

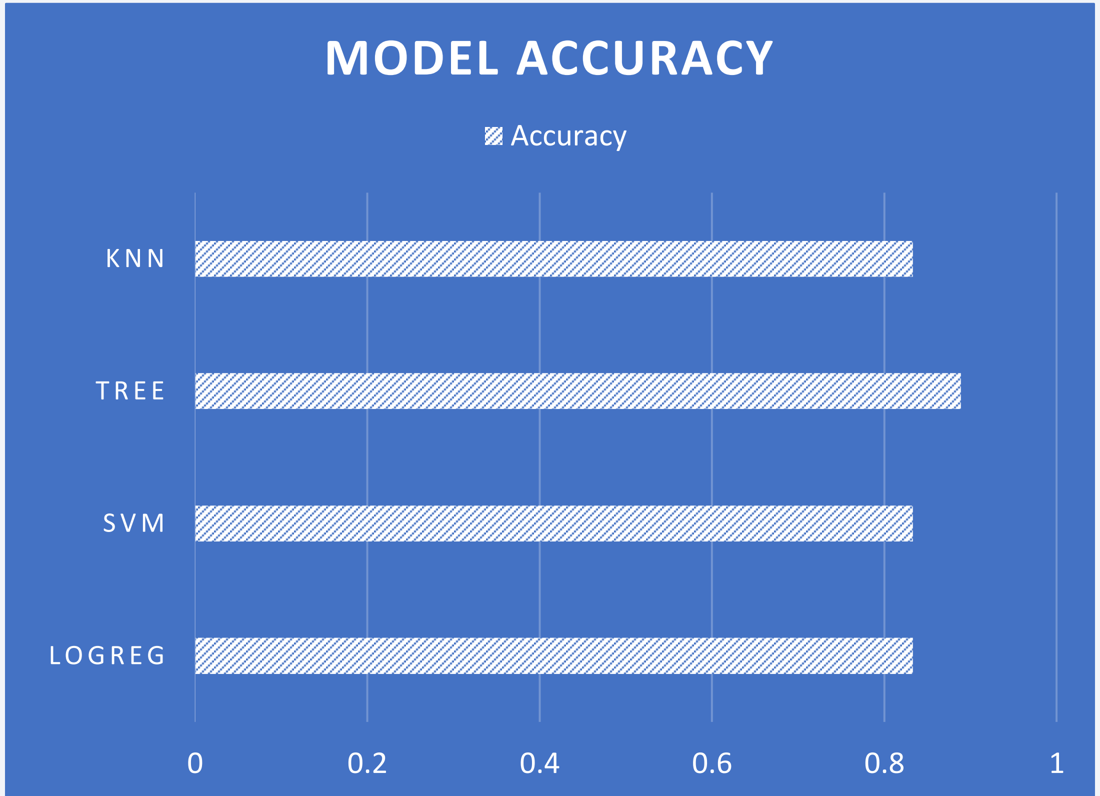


Section 4

Predictive Analysis (Classification)

Classification Accuracy

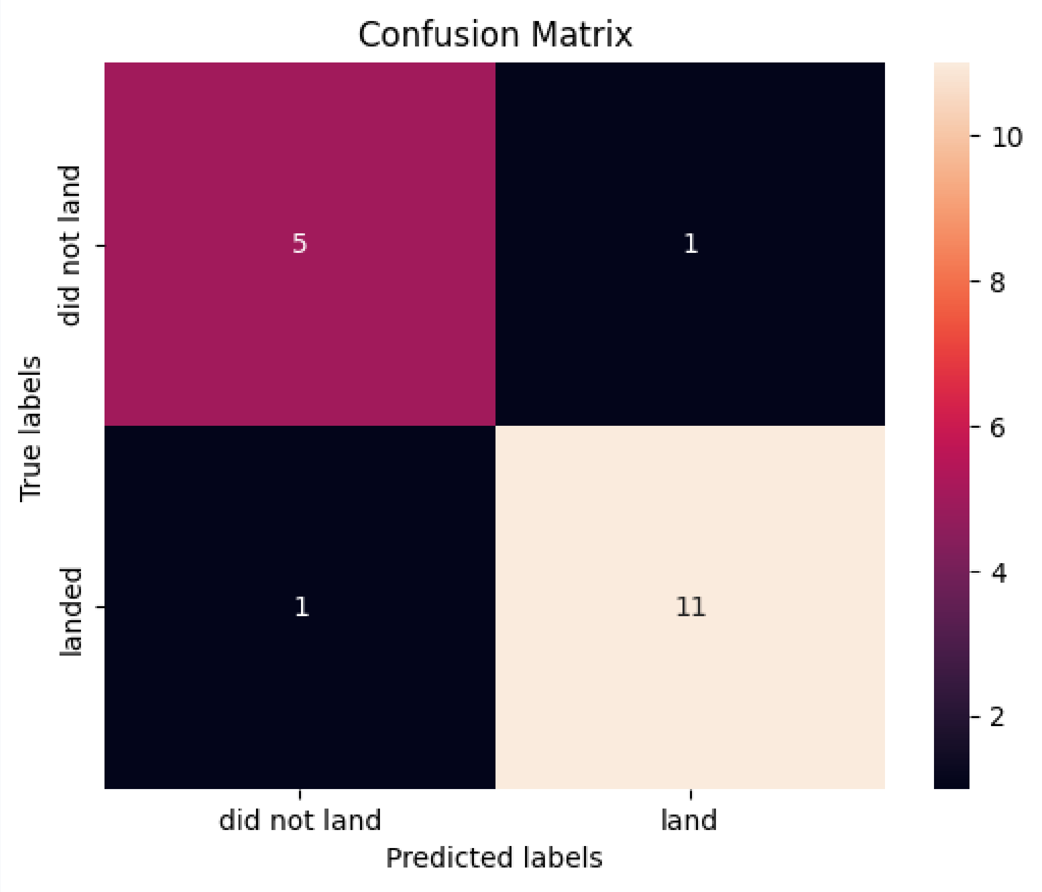
- The Decision Tree had the highest model classification accuracy
- KNN, SVM, and LogReg all returned the same classification accuracy value



	LogReg	SVM	Tree	KNN
Accuracy	0.833333333	0.833333333	0.888888889	0.833333333

Confusion Matrix

- This confusion matrix shows Predicted Lables on the X-axis and the True Labels on the Y-axis
- Absolute accuracy would show a value of 0 in the bottom left and top right
- The Decision Tree was able to get 16 out of a possible 18 correct, as shown on the previous slide with a 88.8888889% accuracy rating



Conclusions

- Launch success rates have risen drastically to near perfection since 2013
- The most utilized launch site, CCAFS SLC 40, is closest to the Equator, with the second most utilized launch site being a backup site directly next to CCAFS SLC 40
- The two most utilized launch sites sit very close to key infrastructure, such as rail or highways, but are relatively distant from major population areas
- The majority of launches have a payload mass between 2,000kg and 6,000kg
- The VLEO orbit, most commonly found in many recent launches, is the best orbit when focusing on orbits with substantial, comparative use

GitHub Hyperlink

https://github.com/washman4133/capstone

Thank you!