IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Yelizaveta Briazkalo
18.06.2025

Outline

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

Executive Summary

- Executive Summary

  Data was collected from the SpaceX API and Wikipedia, then cleaned and filtered to focus on Falcon 9 launches. Key features were analyzed and a binary success label was created. Exploratory data analysis was performed using visualization and SQL. For prediction, data was standardized and split into training and test sets. Four classification models were trained and tuned with cross-validation.

- Summary of all results

  CCAFS SLC 40 is the most used launch site but has a lower success rate. KSC LC 39A has the highest success rate (~77%) and most successful landings. Success rates improved over time, peaking at 90% in 2019. Payloads under 7000 kg are common; heavier payloads have higher success. Some orbits (ES-L1, GEO, HEO, SSO) had 100% success, others less. Launch sites are near coasts and transport routes for safety and logistics. Early missions had more failures, but success increased with technology. Logistic Regression was the best model, with about 83% accuracy and reliable predictions.

Introduction

- Project background and context

The commercial space industry is rapidly evolving, with companies making space travel increasingly accessible and affordable. Among them, SpaceX has emerged as a leader, achieving milestones such as launching spacecraft to the International Space Station, deploying the Starlink satellite constellation, and conducting manned space missions. A key factor behind SpaceX's success is its ability to significantly reduce launch costs by reusing the first stage of its Falcon 9 rockets. While traditional rocket launches may cost upwards of $165 million, SpaceX offers launches at approximately $62 million, largely due to this reusability.

Understanding whether the first stage of a rocket can be successfully recovered is essential for estimating launch costs. In this project, you take on the role of a data scientist at a fictional competitor, Space Y, founded by billionaire industrialist Allon Musk. Your goal is to analyze SpaceX launch data and build tools that help your team understand and predict the likelihood of first-stage reuse without relying on traditional rocket science.

Introduction

- Problems you want to find answers

  Can we predict whether the first stage of a Falcon 9 rocket will successfully land using publicly available data?

  What factors (such as payload, orbit, launch site) influence the success of first-stage landings?

  What insights can we gain from exploratory data analysis to inform model development and business decisions?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected from two main sources: the SpaceX API and Wikipedia.
  - From the SpaceX API, a GET request retrieved launch data in JSON format. The data was converted into a Pandas DataFrame and filtered to include only relevant information, such as rocket, payload, launch site, core, flight number, and date. Additional details (e.g., booster version, payload mass, orbit, core reuse count, and launch site coordinates) were gathered by making further API calls using IDs from the dataset. The data was then cleaned, filtered to include only Falcon 9 launches, and missing values were handled appropriately.
  - From Wikipedia, launch data was scraped from a static page using BeautifulSoup. The correct table was identified, and relevant columns such as flight number, date, and time were extracted and stored in a DataFrame.

Methodology

Executive Summary

- Perform data wrangling
  - The data was loaded from a CSV file and checked for missing values, which were found mainly in the landing pad column. Key columns such as launch site, orbit, and mission outcome were analyzed. A new binary column called Class was created to indicate landing success (1) or failure (0) based on mission outcomes. The overall landing success rate was calculated to be approximately 66.7%.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash

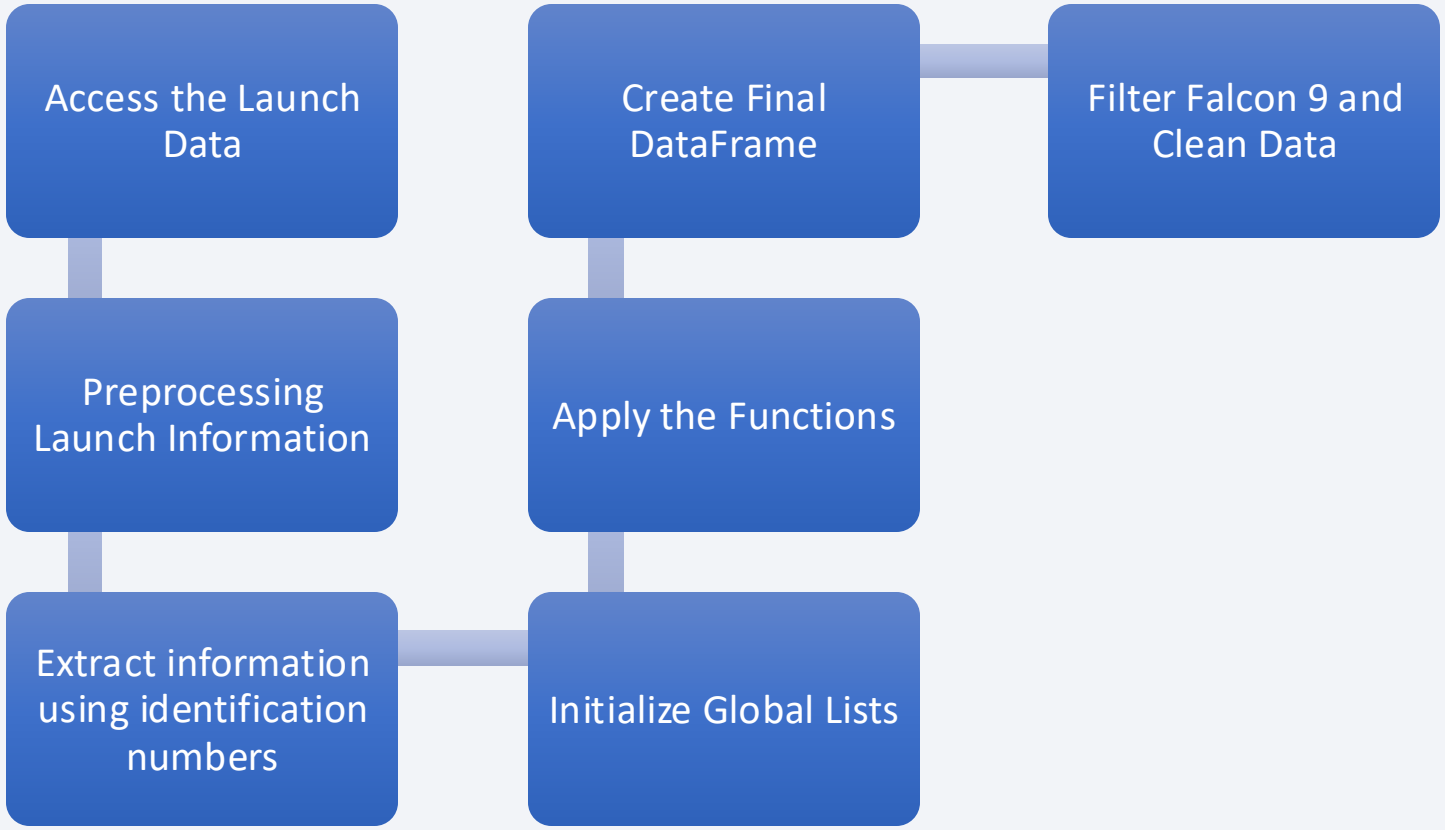
Methodology

Executive Summary

- Perform predictive analysis using classification models
  - The data was standardized and split into training (80%) and testing (20%) sets. Four classifiers—Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbors—were trained and tuned using GridSearchCV with 10-fold cross-validation to find the best hyperparameters.
  - Model performance was compared using accuracy scores, showing similar results around 78–83%. Logistic Regression was chosen for its balanced train and test accuracy, minimizing overfitting risk. Models were further evaluated with confusion matrices, highlighting true positives, false positives, and true negatives to assess prediction quality.

Data Collection – SpaceX API

Flowchart of data collection



Data Collection – SpaceX API

1. Access the Launch Data
   - GET request to retrieve launch data:

```
response=requests.get("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json")
```

   - Parse JSON and convert to DataFrame:

```
data_list=response.json()
data=pd.json_normalize(data_list)
```

Data Collection – SpaceX API

2. Preprocessing Launch Information

- Keep only relevant columns:

```
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
```

- Filter rows:

```
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]
```

- Extract single elements from lists:

```
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])
```

- Convert and filter date:

```
data['date'] = pd.to_datetime(data['date_utc']).dt.date
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Data Collection – SpaceX API

3. Extract information using identification numbers
   - Get booster version from rocket ID:

```
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'])
```

Data Collection – SpaceX API

3. Extract information using identification numbers

- Get payload mass and orbit:

```
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'])Extract single elements from lists:
           data['cores'] = data['cores'].map(lambda x : x[0])
           data['payloads'] = data['payloads'].map(lambda x : x[0])
```

Data Collection – SpaceX API

3. Extract information using identification numbers

- Get launch site name and coordinates:

```
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'])
```

Data Collection – SpaceX API

3. Extract information using identification numbers

- Get core info:

```
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:     .....
```

Data Collection – SpaceX API

4. Initialize Global Lists

BoosterVersion = []

PayloadMass = []

Orbit = []

LaunchSite = []

Outcome = []

Flights = []

GridFins = []

Reused = []

Legs = []

LandingPad = []

Block = []

ReusedCount = []

Serial = []

Longitude = []

Latitude = []

Data Collection – SpaceX API

5. Apply the Functions

getBoosterVersion(data)

getLaunchSite(data)

getPayloadData(data)

getCoreData(data)

Data Collection – SpaceX API

6. Create Final DataFrame

```
launch_dict = {'FlightNumber':
list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

data=pd.DataFrame(launch_dict)
```

Data Collection – SpaceX API

7. Filter Falcon 9 and Clean Data

- Remove Falcon 1 launches and reindex:

```
data_falcon9=data[data['BoosterVersion']!='Falcon 1']
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

- Handle missing values:

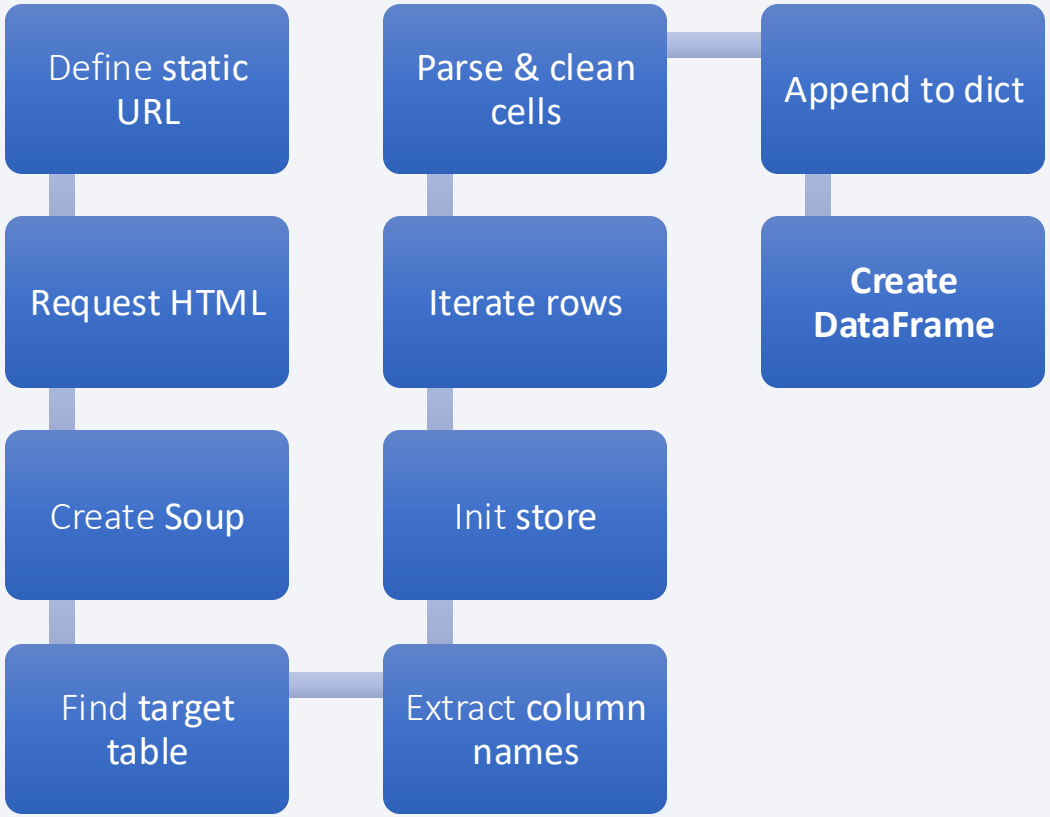
```
payload_mass_mean=data_falcon9['PayloadMass'].mean()
data_falcon9['PayloadMass']=data_falcon9['PayloadMass'].replace(np.nan,
payload_mass_mean)
```

GitHub URL of the completed SpaceX API calls notebook:

SpaceX-API-calls.ipynb

Data Collection - Scraping

Flowchart of web scraping



Data Collection - Scraping

1. Define static URL

   static_url = https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

2. Request HTML

   obj=requests.get(static_url)

3. Create Soup

   soup=BeautifulSoup(obj.text)

Data Collection - Scraping

4. Find target table

```
html_tables=soup.find_all('table')
first_launch_table = html_tables[2] # our target table
```

5. Extract column names

```
column_names = []
for th in first_launch_table.find_all('th'):
    name=extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Data Collection - Scraping

6. Init store

```
launch_dict= dict.fromkeys(column_names)
del launch_dict['Date and time ( )']
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Data Collection - Scraping

7. Iterate rows
8. Parse & clean cells

```
datatimelist=date_time(row[0])
date = datatimelist[0].strip(',')
 time = datatimelist[1]
```

9. Append to dict

```
`Flight No.`launch_dict['Flight No.'].append(flight_number)
launch_dict['Date'].append(date)
launch_dict['Time'].append(time)
```

Data Collection - Scraping

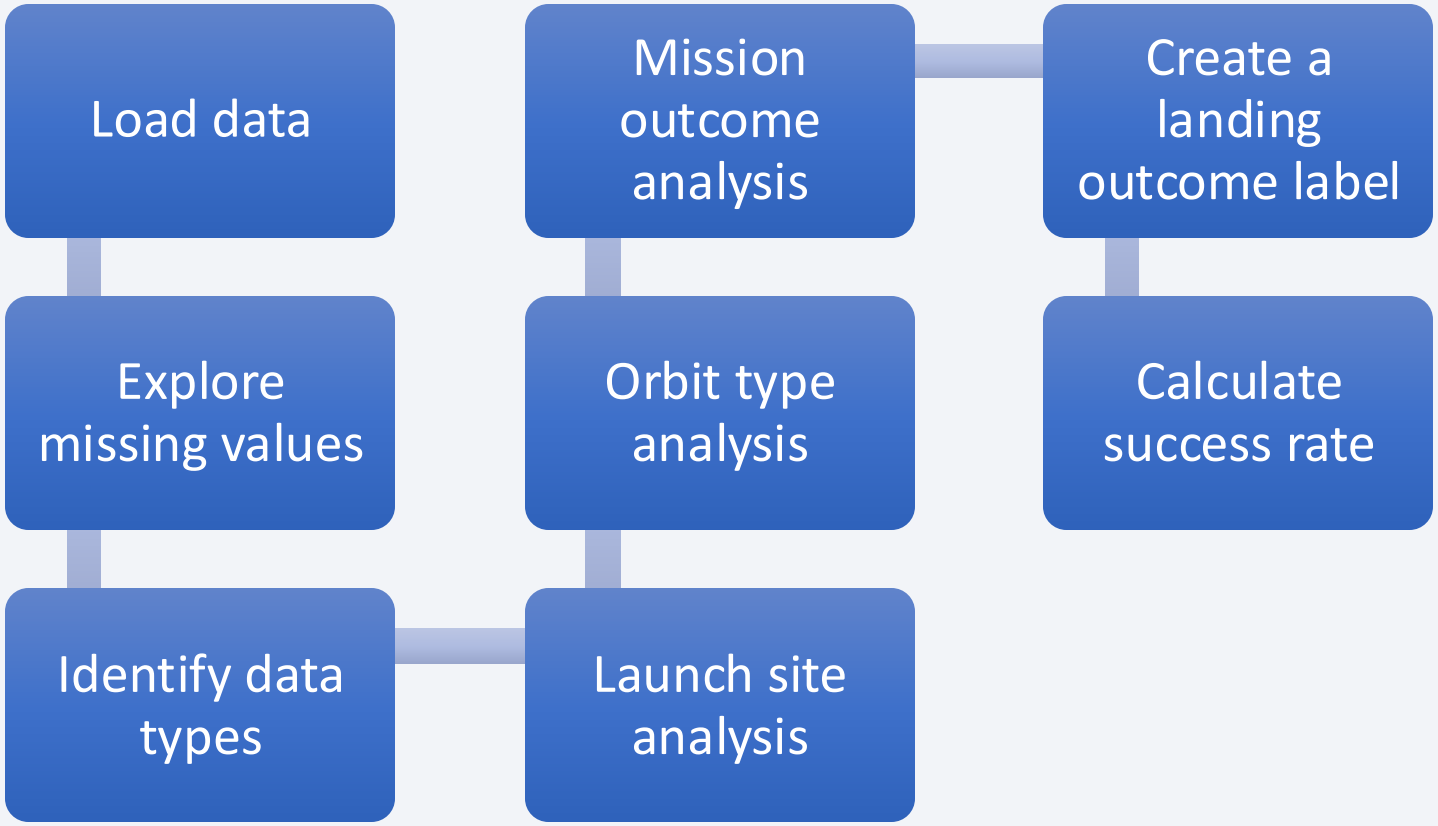
10. Create DataFrame

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

GitHub URL of the completed web scraping notebook:

web-scraping.ipynb

Data Wrangling



Data Wrangling

1. Load Data

df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")

2. Explore missing values

df.isnull().sum()/len(df)*100

Zero in all columns, only in LandingPad column 28.89% of the missing values

3. Identify data types

df.dtypes

Data Wrangling

4. Launch site analysis

df['LaunchSite'].value_counts()

55 launches for CCAFS SLC 40, 22 launches for KSC LC 39A, and 13 launches for VAFB SLC 4E

5. Orbit type analysis

df['Orbit'].value_counts()

Top five orbits: GTO, ISS, VLEO, PO, LEO

Data Wrangling

6. Mission outcome analysis

```
landing_outcomes=df['Outcome'].value_counts()
```

The most frequent outcome in the data is True ASDS=41, means the mission outcome was successfully landed to a drone ship. The second most frequent outcome in the data is True None None=19, means a failure to land

Data Wrangling

7. Create a landing outcome label

```
    bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
landing_class=[]
for outcome in df['Outcome']:
   if outcome in bad_outcomes:
      landing_class.append(0)
   else: landing_class.append(1)
df['Class']=landing_class
```

Data Wrangling

8. Calculate success rate

```
df["Class"].mean()
```

It is 0.667, this means that 2 of 3 landings are successful

GitHub URL of the completed data wrangling related notebooks:

data-wrangling.ipynb

EDA with Data Visualization

In this stage we created some charts:

1. Visualize the relationship between Flight Number and Payload Mass
2. Visualize the relationship between Flight Number and Launch Site
3. Visualize the relationship between Payload Mass and Launch Site
4. Visualize the success rate of each orbit type
5. Visualize the relationship between Flight Number and Orbit type
6. Visualize the relationship between Payload Mass and Orbit type
7. Visualize the launch success yearly trend

GitHub URL of the completed EDA with data visualization notebook:

EDA-with-data-visualization.ipynb

EDA with SQL

The SQL queries that were performed:

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

EDA with SQL

The SQL queries that were performed:

- List the total number of successful and failure mission outcomes
- List all the booster_versions that have carried the maximum payload mass.
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

GitHub URL of the completed EDA with SQL notebook:

EDA-with-SQL.ipynb

Build an Interactive Map with Folium

1. Mark all launch sites on a map
2. Adding the launch outcomes markers for each site
3. Calculate the distances between a launch site to its proximities

Add a MousePosition on the map to get coordinate for a mouse over a point on the map, so we can easily find the coordinates of any points of interests (coastlines, highways, railway and cities), draw a PolyLine between a launch site to the selected point

GitHub URL of the completed interactive map with Folium map:

interactive-map-with-Folium.ipynb

Build a Dashboard with Plotly Dash

1. Add a Launch Site Drop-down Input Component
2. Render success-pie-chart based on selected site dropdown. Visualize total success launches for selected site
3. Add a Range Slider to Select Payload
4. Render the success-payload-scatter-chart scatter plot based on selected site dropdown and selected payload. Visualize correlation between payload mass and success

GitHub URL of the completed Plotly Dash lab:

spacex-dash-app.py

Predictive Analysis (Classification)

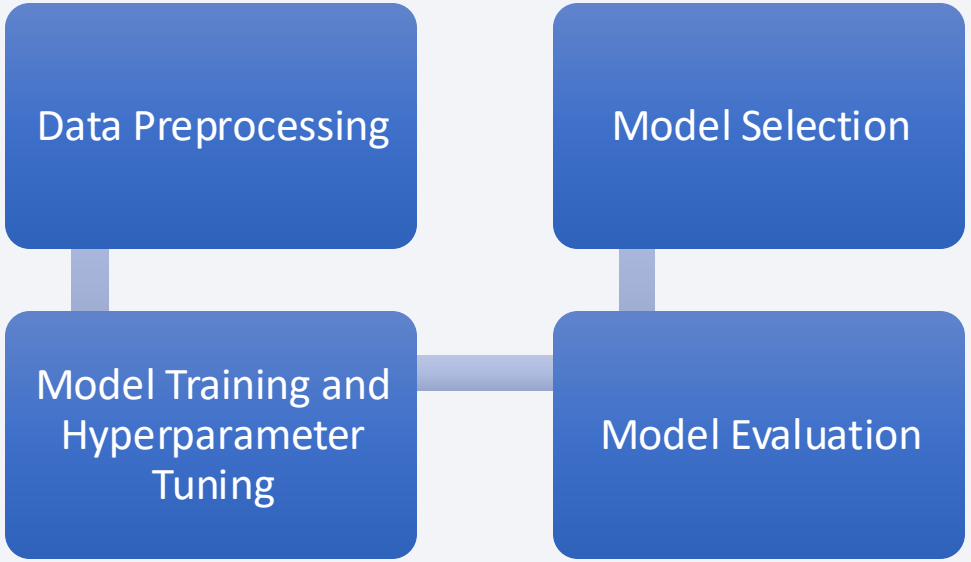
1. Data Preprocessing:

- Standardized the input features using `StandardScaler` to ensure all features contributed equally to the model performance.
- The dataset was then split into training and testing subsets (80/20 split).

2. Model Training and Hyperparameter Tuning:

Trained and optimized four different classifiers using `GridSearchCV` with 10-fold cross-validation:

- **Logistic Regression**: Best parameters – `C=0.01, penalty='l2', solver='lbfgs';`
- **Support Vector Machine**: Best parameters – `C=1.0, gamma=0.0316, kernel='sigmoid';`
- **Decision Tree Classifier**: Best parameters – `criterion='gini', max_depth=4, max_features='sqrt', min_samples_leaf=1, min_samples_split=2, splitter='random';`
- **K-Nearest Neighbors**: Best parameters – `n_neighbors=10, algorithm='auto', p=1;`



Predictive Analysis (Classification)

3. Model Evaluation

- **Logistic Regression**: Training accuracy: **0.8464**, Test accuracy: **0.8333**
- **Support Vector Machine**: Training accuracy: **0.8482**, Test accuracy: **0.8333**
- **Decision Tree Classifier**: Training accuracy: **0.871**, Test accuracy: **0.7778**
- **K-Nearest Neighbors**: Training accuracy: **0.8482**, Test accuracy: **0.8333**

Three models achieved the same test accuracy of **0.8333**, and their confusion matrices were identical, indicating similar classification performance. However, there were differences in training accuracy, with Decision Tree performing highest on training data, potentially suggesting overfitting.

4. Model Selection

I selected Logistic Regression as the best model because it had the smallest gap between training and test accuracy, reducing the risk of overfitting. The best parameters for this model were C=0.01, penalty='l2', and solver='lbfgs'.

GitHub URL of the completed predictive analysis lab: predictive-analysis-lab.ipynb

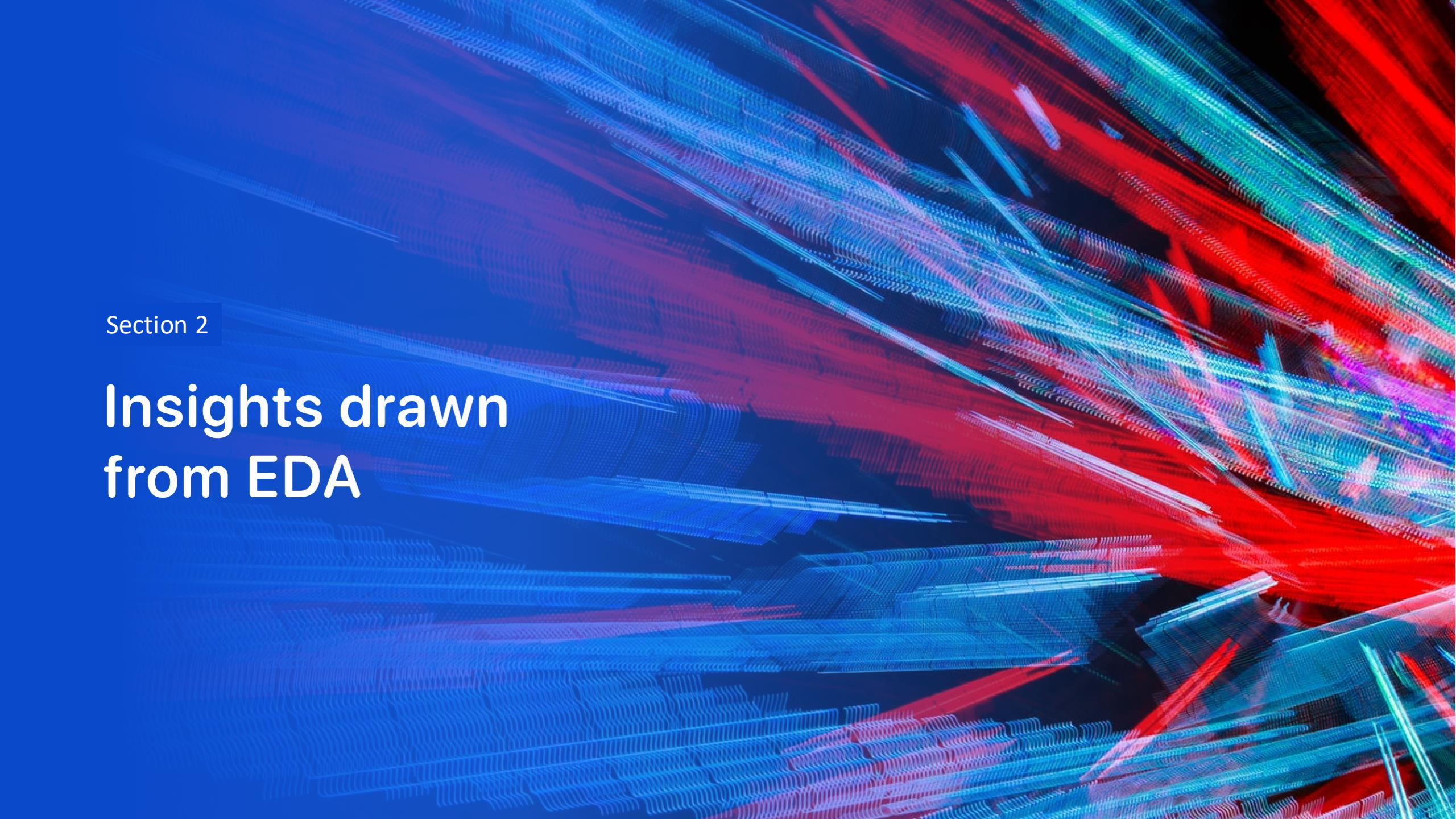
Results

Exploratory data analysis revealed that CCAFS SLC 40 was the most frequently used launch site, especially in early flights, but had a relatively low landing success rate. The KSC LC 39A site showed the highest success rate at approximately 77%, with mostly successful landings after flight 25. Over time, landing failures decreased significantly, peaking at a 90% success rate in 2019.

Payloads up to 7000 kg were most common, with heavier payloads generally achieving higher success rates. Certain orbits like ES-L1, GEO, HEO, and SSO had perfect success rates, while others such as GTO and SO showed lower or no success, partly due to limited flight data.

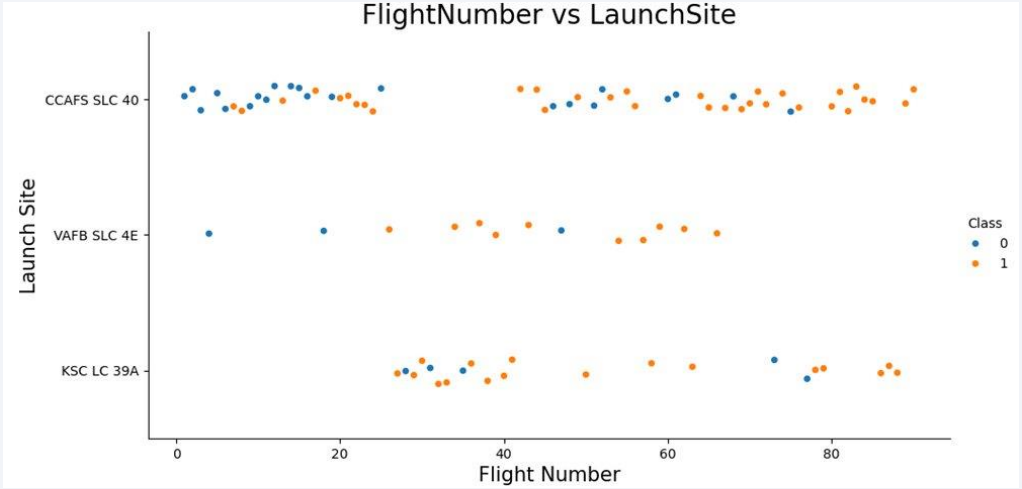
Spatial analysis showed that all launch sites are located close to coastlines, railways, and highways, likely for logistical convenience and safety in case of landing failures. KSC LC-39A led in successful landings count and ratio, whereas CCAFS LC-40 had the lowest success ratio despite a moderate number of landings.

Classification models were trained to predict landing success using standardized features and an 80/20 train-test split. Logistic Regression, SVM, Decision Tree, and K-Nearest Neighbors classifiers achieved similar accuracies around 78-83%. Logistic Regression was selected as the best model due to its balanced train and test accuracy, minimizing overfitting. Confusion matrix analysis showed good prediction quality with 12 true positives, 3 false positives, and 3 true negatives.


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

```
sns.catplot(x='FlightNumber', y='LaunchSite', data=df, hue='Class', aspect = 2)
```



In the graph, we can see that the most popular LaunchSite is CCAFS SLC 40. The first 20 landings mostly took place at CCAFS SLC 40, and they were mostly unsuccessful (blue dots), although there were a few successful ones. During this same period, there were also two failed landings at VAFB SLC 4E.

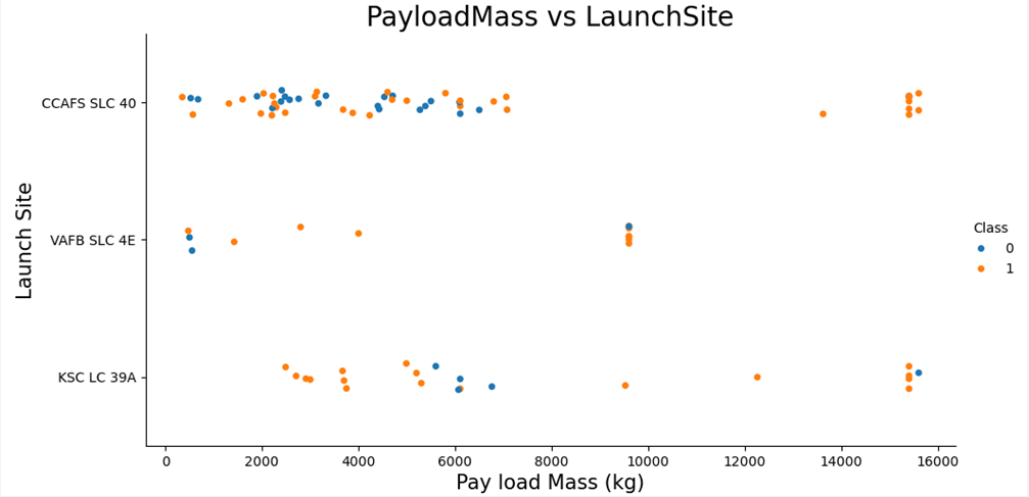
Starting from flight number 20, the number of failed landings begins to decrease. In the range between flight 25 and 40, a new landing site, KSC LC 39A, starts being used, and the results are mostly positive (orange dots). We also see two successful landings at VAFB SLC 4E in this period. Additionally, there is a gap in the use of CCAFS SLC 40 during this time.

From flight 40 to 70, all three landing sites are in use. CCAFS SLC 40 has significantly more landings than the other two, but it also has more failed landings. VAFB SLC 4E has only one failed landing, and KSC LC 39A had only successful landings.

Starting around flight 70, VAFB SLC 4E stops being used. The number of failed landings at CCAFS SLC 40 and KSC LC 39A remains low, with only 3 cases.

Payload vs. Launch Site

```
sns.catplot(x='PayloadMass', y='LaunchSite', data=df, hue='Class', aspect = 2)
```



In the graph, we can see that payloads up to 7000 kg are used most often, and rockets with this PayloadMass most frequently land at CCAFS SLC 40. However, the other two landing sites are also used, except that KSC LC 39A was not used for rockets with a PayloadMass below 2000 kg.

We also see that in this lower mass range, there are many more failed landings. I believe this is because these were earlier flights.

For heavier payloads, there is less data, but based on what we can see, landings with higher PayloadMass are almost always successful (only two failed cases).

We can also notice that VAFB SLC 4E is not used for payloads over 10,000 kg. Another interesting point is that for PayloadMass values up to 8000 kg, the values vary continuously — we see a horizontal spread of points on the graph. But after 10,000 kg, the values look more fixed or predetermined — the same numbers are used in many landings. This appears in the graph as vertical lines of dots.

Success Rate vs. Orbit Type

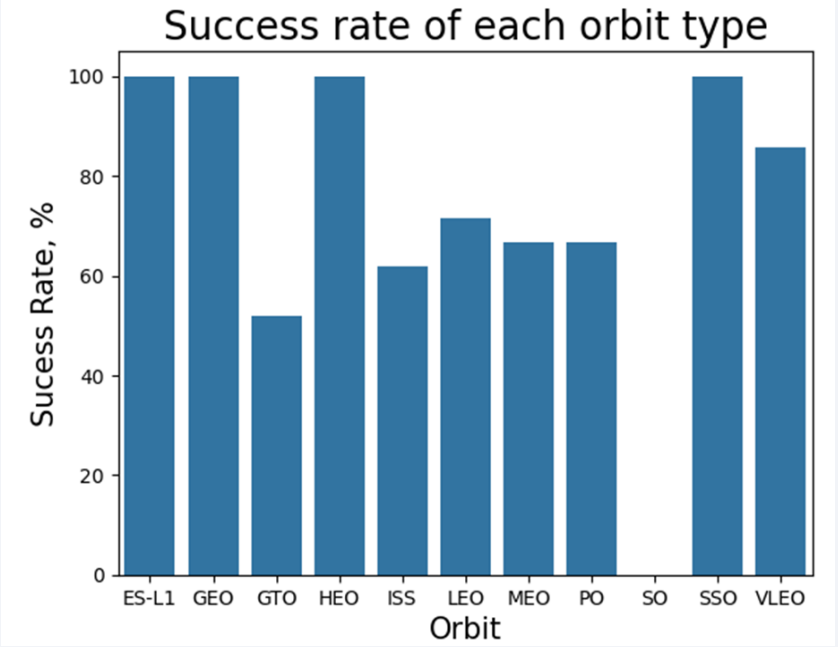
```
success_rate=df[df['Outcome'].isin(['True Ocean', 'True RTLS', 'True ASDS'])].groupby(['Orbit']).count()/df[['Orbit','Outcome']].groupby(['Orbit']).count()

success_rate=success_rate[['Outcome']].reset_index()

success_rate=success_rate.fillna(0)

success_rate[['Outcome']]=success_rate[['Outcome']]*100

sns.barplot(y='Outcome', x='Orbit', data=success_rate)
```



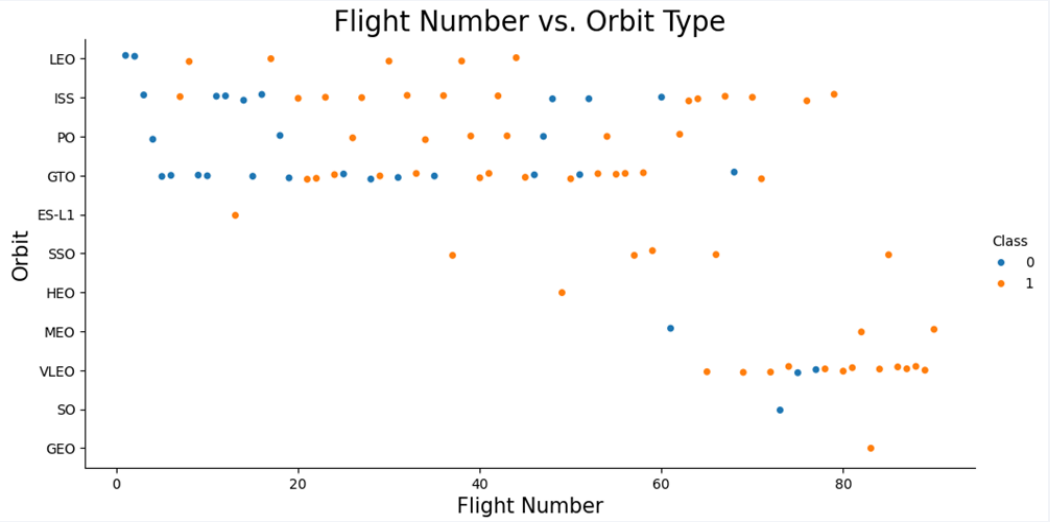
In the graph, we see that the most successful landings came from the ES-L1, GEO, HEO, and SSO orbits — all with a 100% success rate.

For the GTO, ISS, LEO, MEO, and PO orbits, the success rates are lower. The lowest among these is for GTO, with 50% successful landings, while the highest is for LEO, with 70%.

The worst result is for the SO orbit — 0% success, meaning all landings from this orbit failed. However, this is based on only one flight, which was unsuccessful, so the data may be misleading.

Flight Number vs. Orbit Type

```
sns.catplot(x='FlightNumber', y='Orbit', data=df, hue='Class', aspect=2)
```



In the graph, we can see that there were only a few flights to the ES-L1, GEO, HEO, and SSO orbits, but all of them were successful without exception.

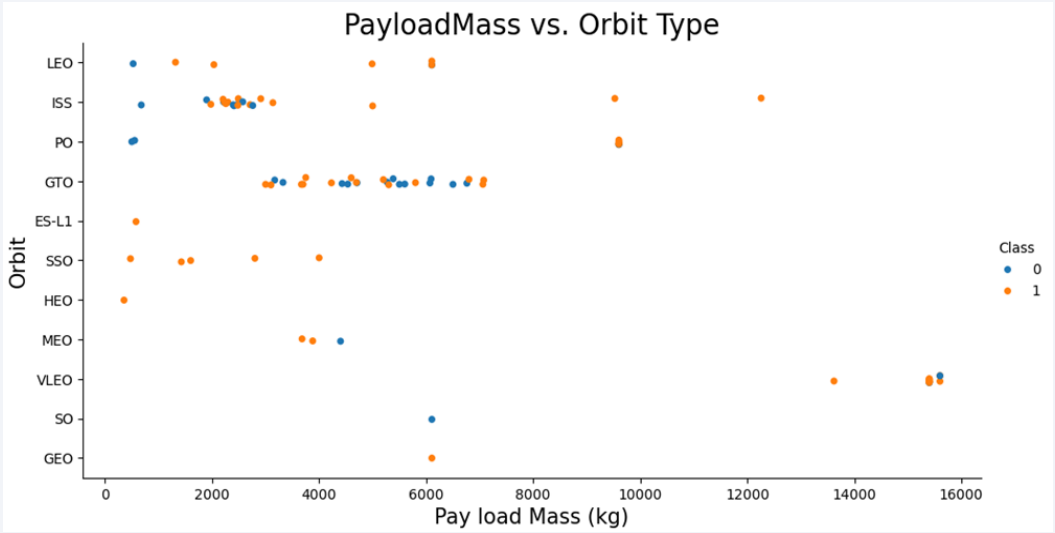
Most of the flights were to the LEO, ISS, PO, GTO, and VLEO orbits. Many of the early flights to these orbits ended in failure, which explains the lower success rates shown in the previous graph. Still, even as the number of flights increased, failures continued to happen — although not as often as in the beginning

The LEO orbit stopped being used after around flight number 40, while VLEO started to be used more actively after flight 60.

As for the SO orbit, as mentioned on the previous slide, there was only one recorded flight to this orbit, and it was unsuccessful. This happened around flight number 60.

Payload vs. Orbit Type

```
sns.catplot(x='PayloadMass', y='Orbit', data=df, hue='Class', aspect=2)
```



In the graph, we can see two clusters of points — one on the ISS orbit in the 2000–3000 kg range, and another on the GTO orbit between 3000–7000 kg.

If we divide the X-axis into two equal parts, we'll notice that there are more flights in the first half, and all orbits are represented there. In contrast, in the second half of the axis (i.e., above 8000 kg), there are significantly fewer flights, and they are limited to only three orbits: ISS, PO, and VLEO.

We can also note that there were only a few flights with a payload mass close to zero — just seven, and half of them failed. These flights were on the LEO, ISS, PO, ES-L1, SSO, and HEO orbits.

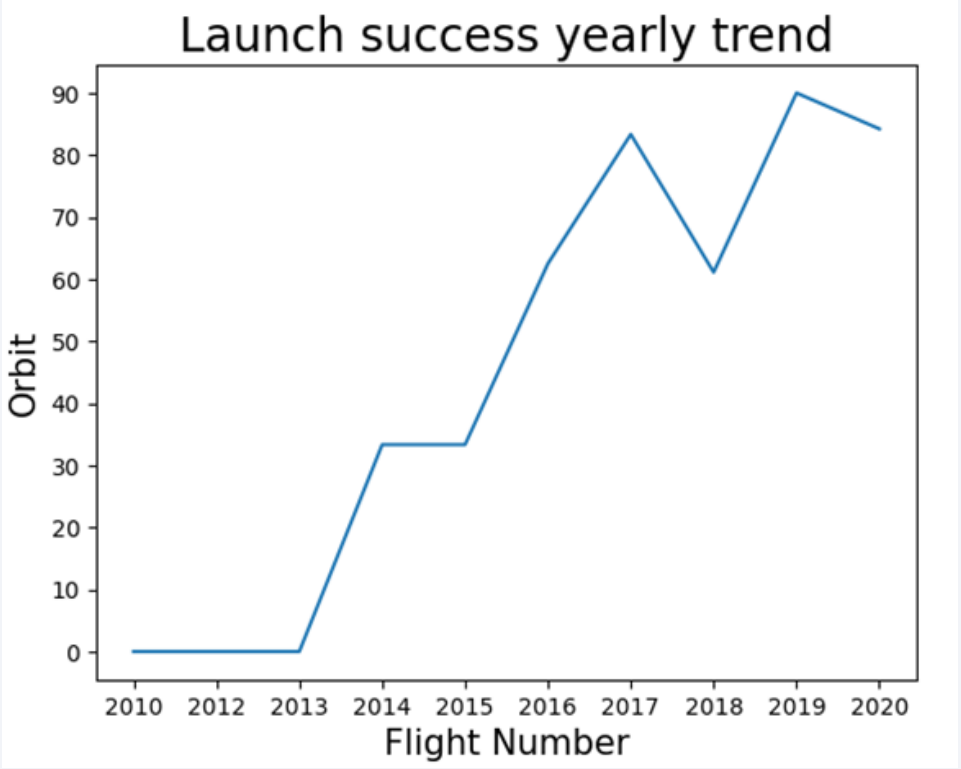
The most common flights are to the ISS and GTO orbits, carrying payloads between 2000 and 7000 kg.

Launch Success Yearly Trend

```
sns.lineplot(x='Date', y='Outcome', data=success_rate)
```

In the graph, we can see that the landing success rate was zero during the first three years. From 2013 to 2017, the success rate gradually increased, reaching around 80%.

In 2018, this rate suddenly dropped to 60%, but the following year, 2019, it rose sharply to 90%, which was the peak value. A year later, it dropped again slightly to around 80%, which is still a very good result.



All Launch Site Names

```
select distinct Launch_Site

from SPACEXTABLE
```

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

In the query, we used the DISTINCT function, which returns only unique names.

As a result, we see four launch sites:
CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, and CCAFS SLC-40.

Launch Site Names Begin with 'CCA'

```
select *  from SPACEXTABLE  where Launch_Site like 'CCA%'  limit 5
```

In the query, we used the LIKE function, which searches for similar records. The % symbol represents any number of any characters. We also used LIMIT to restrict the number of results to 5.

As a result, we get 5 records where the Launch_Site starts with 'CCA'. As we can see, all of them have the LEO orbit, and the mission outcome is successful.

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

```
select sum(PAYLOAD_MASS__KG_)
from SPACEXTABLE
where Customer='NASA (CRS)'
```

sum(PAYLOAD_MASS__KG_)
45596

The task was to display the total payload mass carried by boosters launched by NASA (CRS).
So, we selected all rows where the Customer is 'NASA (CRS)' and used the SUM function to calculate the total mass.
The result is 45,596 kg.

Average Payload Mass by F9 v1.1

```
select avg(PAYLOAD_MASS__KG_)
from SPACEXTABLE
where Booster_Version ='F9 v1.1'
```

avg(PAYLOAD_MASS__KG_)
2928.4

We selected all rows where Booster_Version is F9 v1.1 and used the AVG function to calculate the average payload mass.
The result is 2928.4 kg.

First Successful Ground Landing Date

```
select min(Date)
 from SPACEXTABLE
where Landing_Outcome ='Success (ground pad)'
```

min(Date)
2015-12-22

We selected all rows where Landing_Outcome is Success (ground pad) and used the MIN function to find the earliest date.
The result is December 22, 2015.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
select Booster_Version
from SPACEXTABLE
where Landing_Outcome ='Success (drone ship)'
and PAYLOAD_MASS__KG_ between 4000 and 6000We selected all rows
```

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

where Landing_Outcome is Success (drone ship) and the PAYLOAD_MASS__KG_ is greater than 4000 and less than 6000.
The results can be seen in the image.

Total Number of Successful and Failure Mission Outcomes

```
%sql select Mission_Outcome, count(*)
from SPACEXTABLE
group by Mission_Outcome
```

Mission_Outcome	count(*)
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

In this query, we used GROUP BY to group the data by Mission_Outcome, and then used the COUNT function to count the number of entries for each value.
As a result, we see that Mission_Outcome was mostly successful, with only one failure and one ambiguous case, which was also counted as successful.

Boosters Carried Maximum Payload

```
select Booster_Version
from SPACEXTABLE
where PAYLOAD_MASS__KG_=
    (select max(PAYLOAD_MASS__KG_)
    from SPACEXTABLE)
group by Booster_Version
```

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

In this query, we used a subquery with the MAX function to find the highest PAYLOAD_MASS__KG_ in the database.
This allowed us to filter and sort by the largest payload mass.
The results, shown in the image, are all F9 B5 boosters of various modifications.

2015 Launch Records

```
select substr(Date, 6,2) as 'Month',
Landing_Outcome, Booster_Version, Launch_Site

from SPACEXTABLE

where substr(Date,0,5)='2015'

and Landing_Outcome like 'Failure%
```

In this query, we used SUBSTR to extract the month and year from the date, filtered the records for the year 2015, and used the LIKE function to find all failed Landing_Outcome entries.
According to the results, there were only two failures, both on the drone ship.

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

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

```
select count(*) as 'total', Landing_Outcome from SPACEXTABLE
where Date between '2010-06-04' and '2017-03-20'
group by Landing_Outcome order by count(*) desc
```

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

In this query, we grouped the records by Landing_Outcome, filtered those between 2010-06-04 and 2017-03-20, counted the number of entries for each Landing_Outcome using the COUNT function, and sorted the results in descending order with ORDER BY DESC.

So, the most frequent Landing_Outcome is no attempt, and the rarest is Precluded (drone ship).

Section 3

Launch Sites Proximities Analysis

All Launch Sites on a Map

Launch Sites CCAFS SLC-40 and CCAFS LC-40 are located very close to each other.
KSC LC-39A is a bit farther away, but VAFB SLC-4E is on the opposite side of America.
So, it's difficult to show all four launch sites on the same map without overlap.

From their locations, we can draw some conclusions:

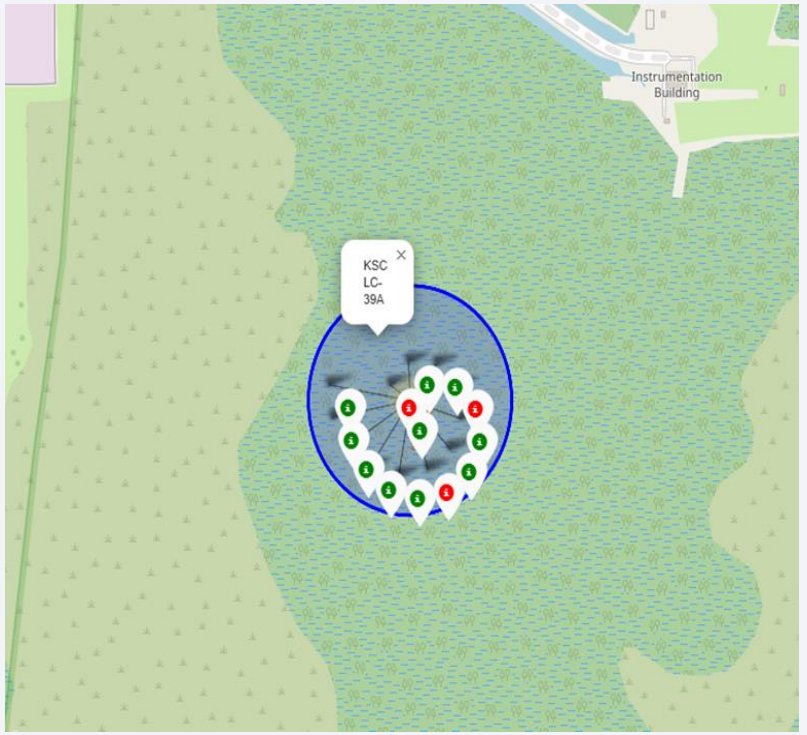
- All four launch sites are located in the southern United States — in California and Florida, relatively close to the Equator. This may be because gravitational force is weaker at the Equator and stronger at the poles.
- All four launch sites are located very close to the coast. This likely helps reduce the risk of damage in case of a failed landing by allowing the rocket to land in the ocean.

Add the launch outcomes for each site

We added markers to indicate landings and their outcomes.
If the landing was successful, the marker is green; if it failed, the marker is red.

VAFB SLC-4E

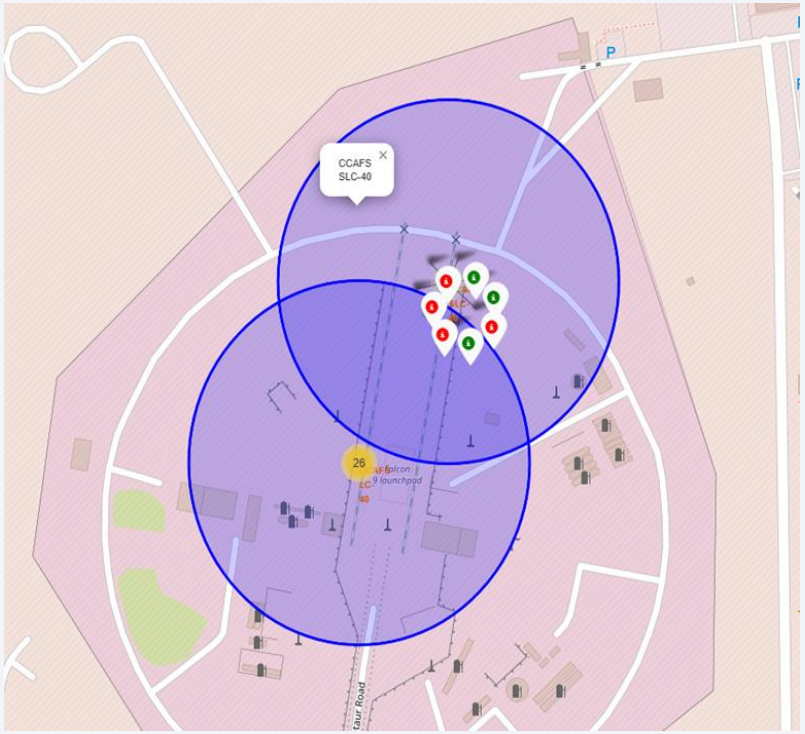
In the first screenshot, we see the VAFB SLC-4E launch site. There are 10 markers in total, 4 of them green, which means this launch site has a success rate of only 40% — a poor result.
In the second screenshot, we see the KSC LC-39A launch site. There are 13 markers in total, 10 of them green, giving this site a success rate of 77%.



KSC LC-39A

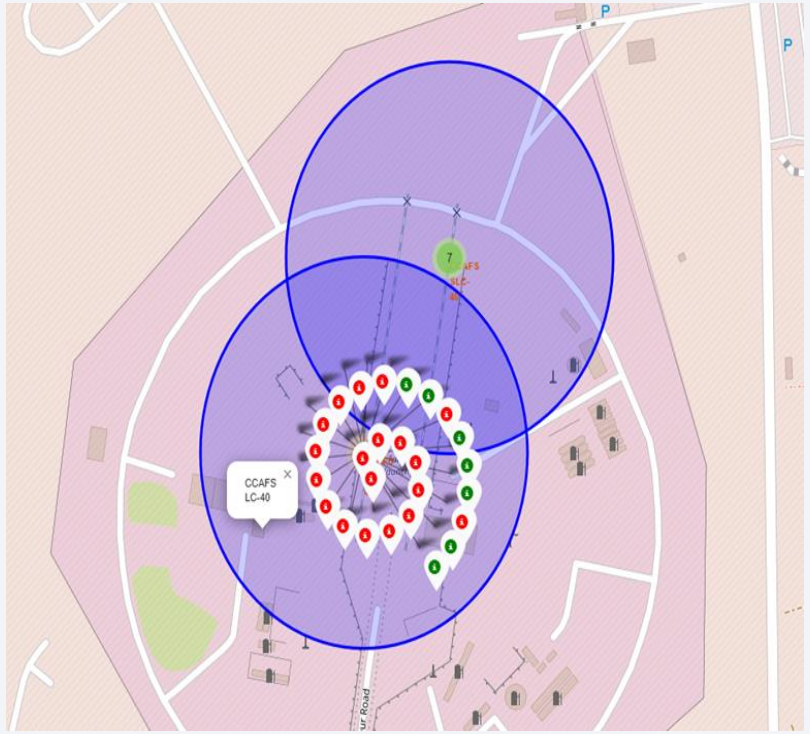
Add the launch outcomes for each site

CCAFS SLC-40 and CCAFS LC-40 are located very close to each other, but there were far fewer landings at CCAFS SLC-40 compared to CCAFS LC-40.



CCAFS SLC-40

In the first screenshot, we can see that out of 7 landings at CCAFS SLC-40, only 3 were successful, which means the success rate is below 50%.
In the second screenshot, we see the markers for CCAFS LC-40 — there are 26 in total, with only 7 green ones. This gives CCAFS LC-40 a success rate of 27%, which is the lowest among all the launch sites.

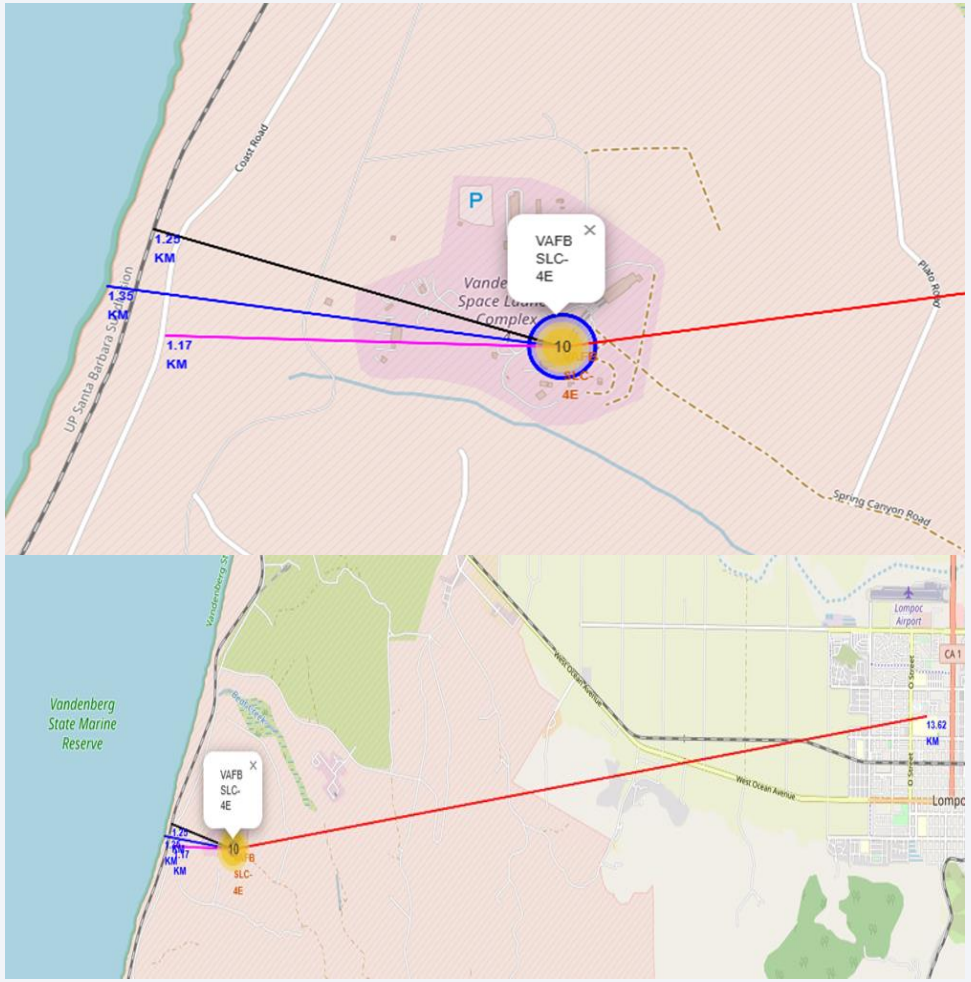


CCAFS LC-40

The distances between a launch site to its proximities

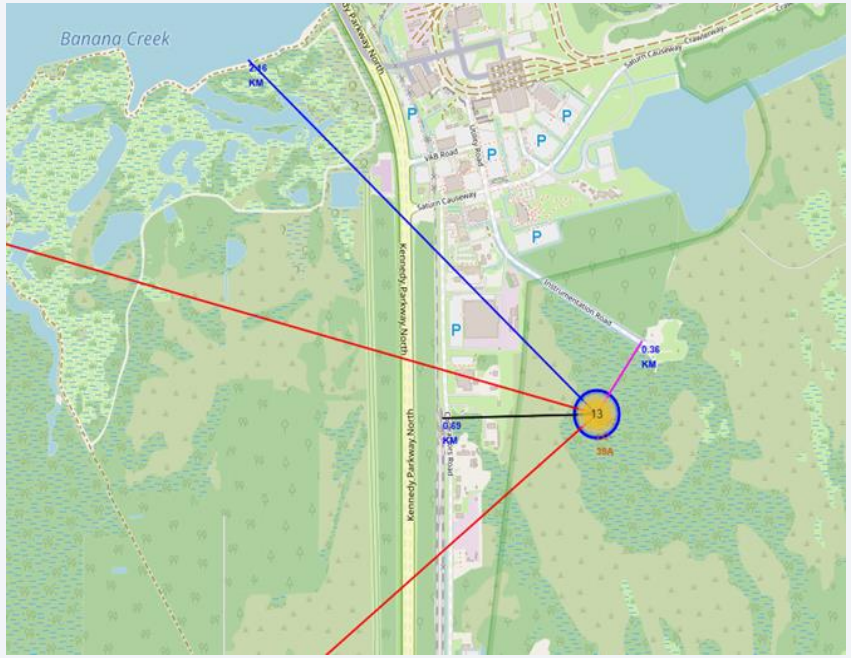
Add a MousePosition on the map to get coordinate for a mouse over a point on the map, so we can easily find the coordinates of any points of interests (coastlines, railways, highways, cities), draw a PolyLine between a launch site to the selected point

This slide shows the VAFB SLC-4E launch site and lines leading to the nearest points of interest.
The blue line leads to the coastline, and we can see that the distance is 1.35 km.
The black line leads to the railway, with a distance of 1.25 km.
The pink line goes to the nearest highway, which is 1.17 km away.
The red line points to the nearest city, with a distance of 13.62 km.

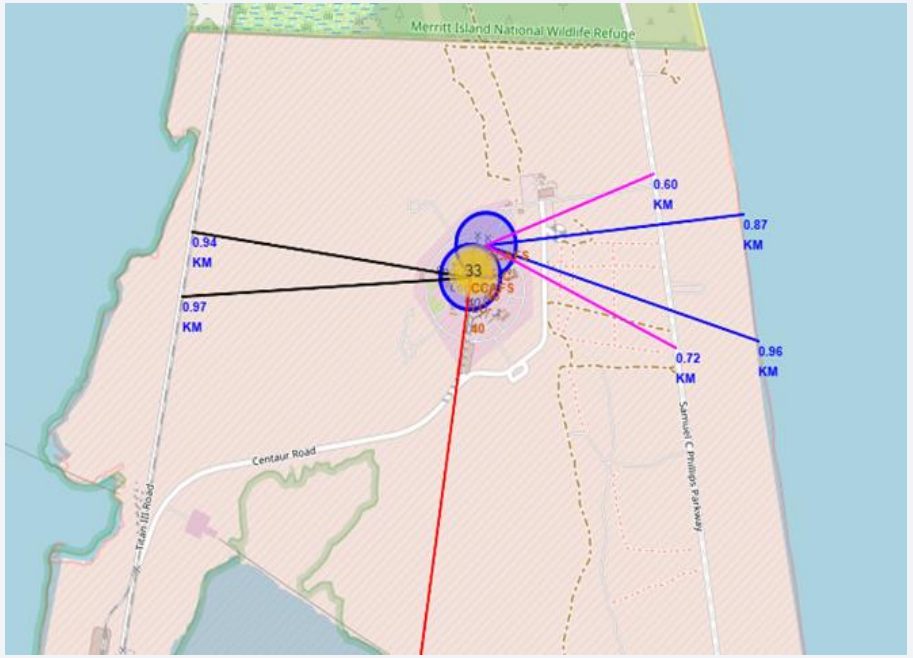


The distances between a launch site to its proximities

In the first image, we see the KSC LC-39A launch site and lines leading to the nearest points of interest: The blue line leads to the coastline, with a distance of 2.46 km. The black line leads to the railway, which is 0.69 km away. The pink line leads to the nearest highway, with a distance of 0.36 km.



In the second image, we see the CCAFS SLC-40 and CCAFS LC-40 launch sites.
The distance to the coastline is 0.87 km and 0.96 km.
The distance to the railway is 0.94 km and 0.97 km.
The distance to the highway is 0.60 km and 0.72 km.



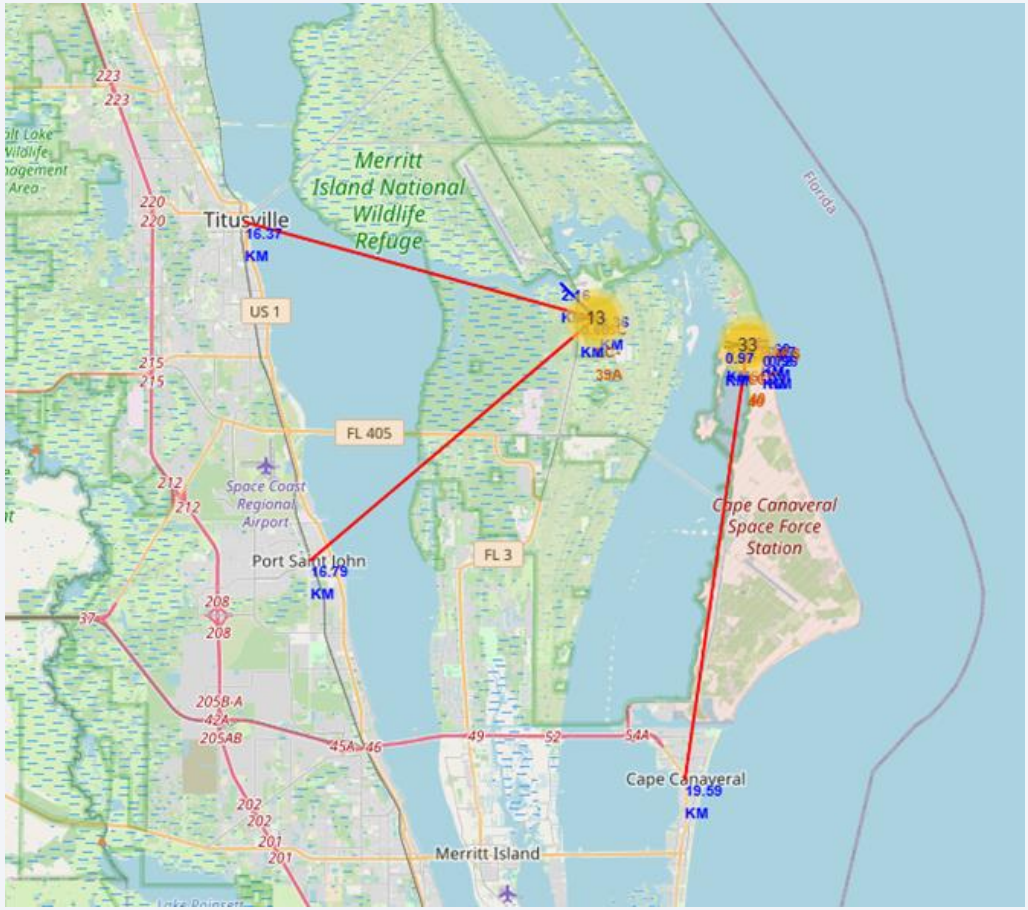
KSC LC-39A

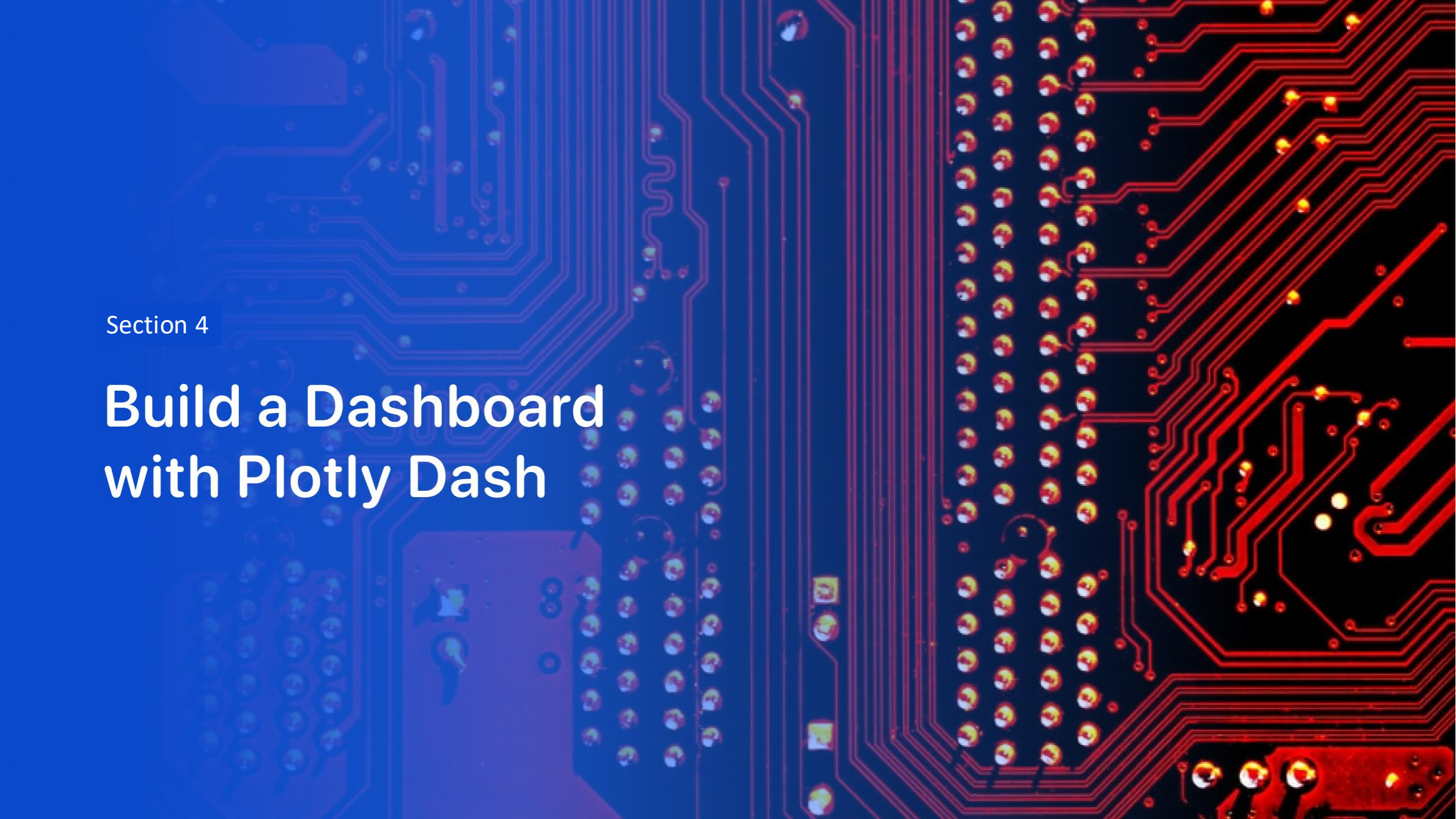
CCAFS SLC-40, CCAFS LC-40

The distances between a launch site to its proximities

This slide shows a screenshot with three launch sites: KSC LC-39A, CCAFS SLC-40, and CCAFS LC-40. Red lines are drawn to the nearest cities, with distances of 16.37 km, 16.79 km, and 19.59 km.

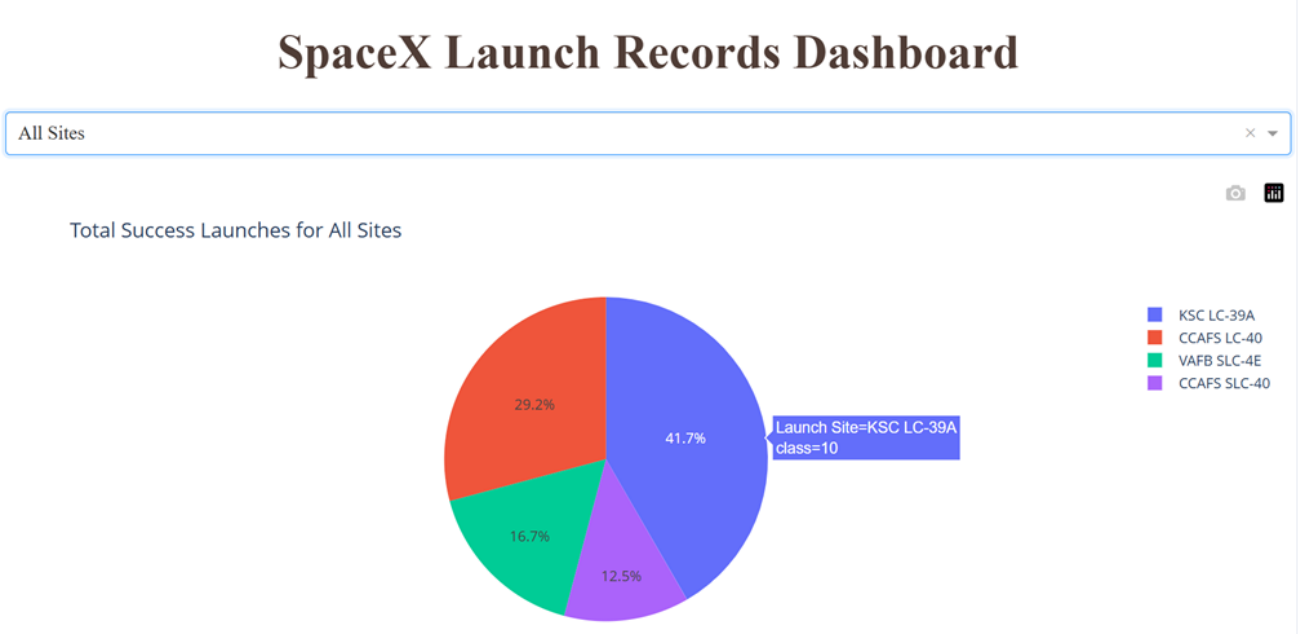
In summary, the launch sites are located quite close to the coastlines, with an average distance of 1.335 km, which helps minimize damage in case of a failed landing by allowing the rocket to land in the ocean. They are also located close to railways and highways, with average distances of 0.963 km and 0.713 km respectively — most likely due to logistics.
As expected, the launch sites are positioned at a safe distance from cities, averaging around 17 km, which helps reduce risk to people.




Section 4
Build a Dashboard with Plotly Dash

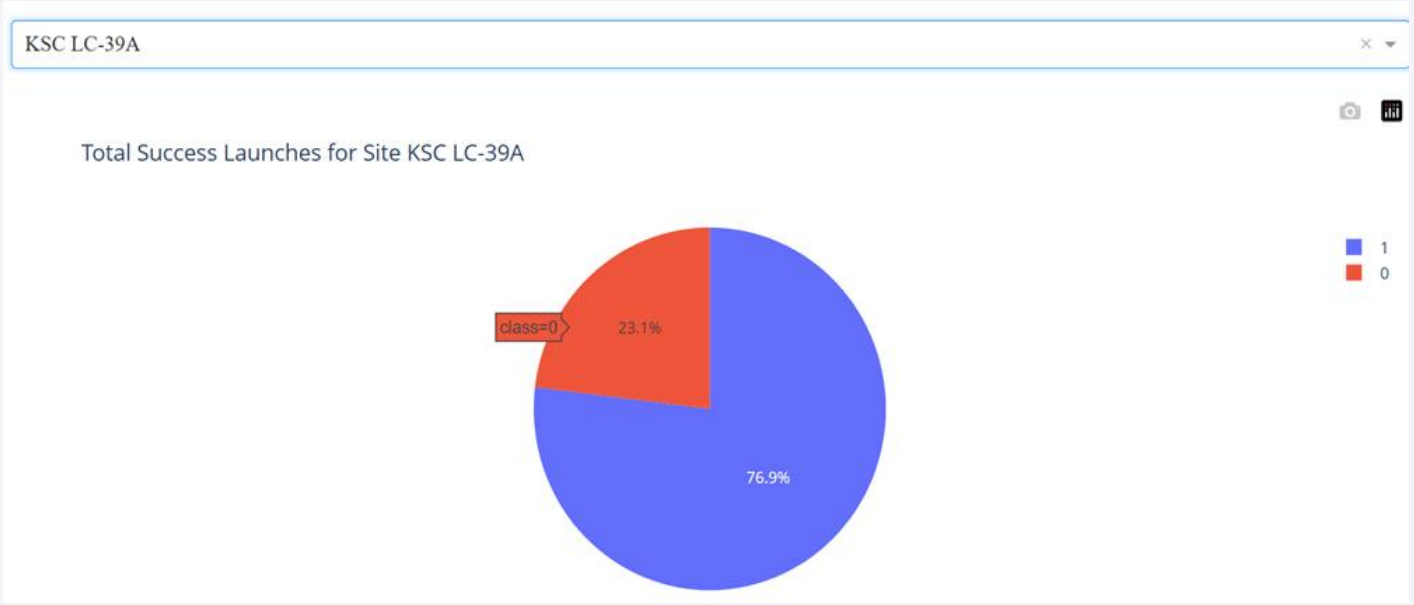
Visualize total success launches for all sites

The highest number of successful landings was at the KSC LC-39A launch site — 10 successes, which makes up 41.7% of all successful landings.
In second place is CCAFS LC-40 with 7 successful landings, accounting for 29.2%.
The lowest value is for CCAFS SLC-40, with 12.5%.



The launch site with highest launch success ratio

The highest success ratio among all launch sites is at KSC LC-39A — 76.9%. There were 13 landings in total, and 10 of them were successful, which is a strong result.
From the previous slide, we remember that CCAFS LC-40 ranks second in terms of the number of successful landings (7), right after KSC LC-39A.
However, its success ratio is the lowest — only 27%.
Even though this launch site has a relatively high number of successful landings, it also has as many as 19 failed ones.



Visualize correlation between payload mass and success

The highest number of successful landings is observed in the 2000–4000 kg payload range, while the most failures occur in the 6000–8000 kg range.

In the graph, we can also see that the highest success rate is achieved when the FT Booster version is used.

Section 5

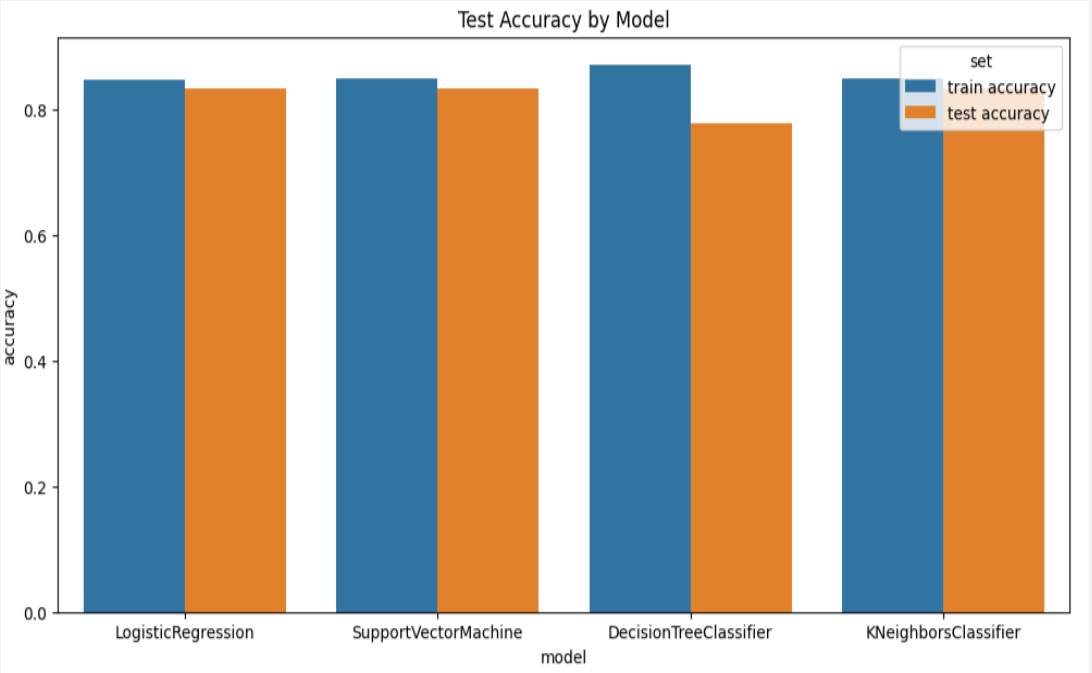
Predictive Analysis (Classification)

Classification Accuracy

As we can see from the bar plot showing the accuracy of all four models we used, the results are quite similar:
Three models have a test accuracy of 0.83, while the Decision Tree Classifier has a test accuracy of 0.78.

The train accuracy varies slightly between models — the highest is in the Decision Tree Classifier, and the lowest is in Logistic Regression.

Based on these results, I would choose the Logistic Regression model because it has the smallest gap between test accuracy and train accuracy, which helps reduce the risk of overfitting.

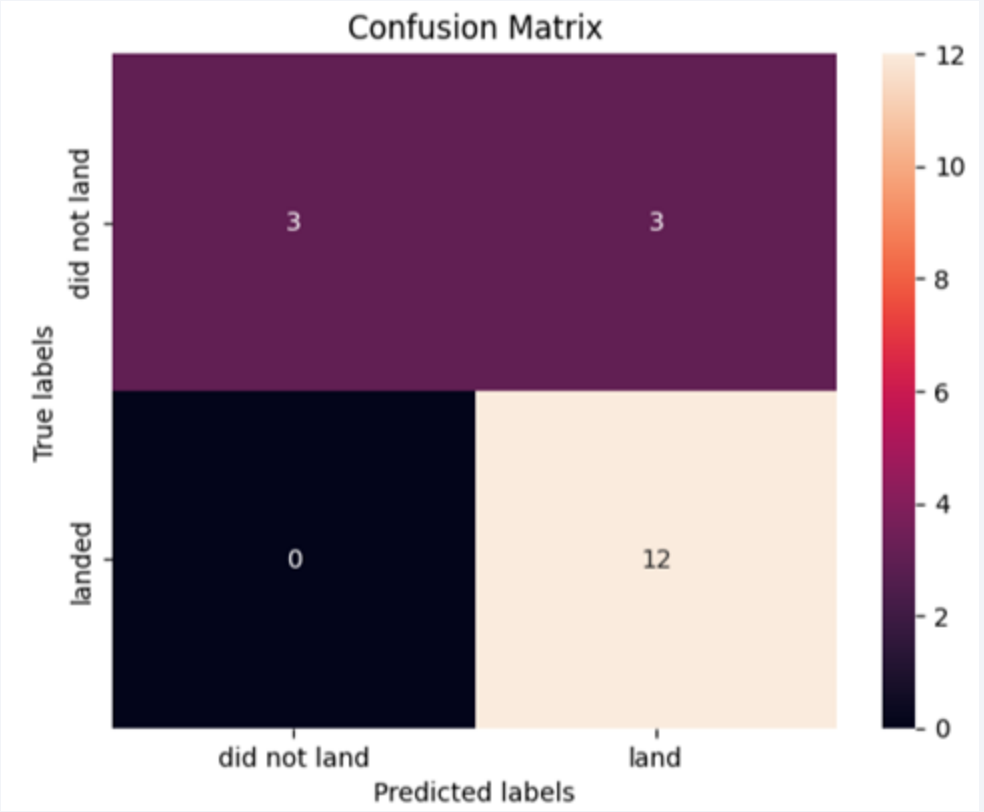


Confusion Matrix

The image shows a confusion matrix, which is the same for all models except the Decision Tree Classifier.

So, what do we see here:

- True Positive – 12: the model predicted that the rocket would land successfully, and it actually did.
- False Positive – 3: the model predicted a successful landing, but the rocket did not land.
- True Negative – 3: the model predicted that the rocket would not land, and that was correct.



Conclusions

- The most used launch site is CCAFS SLC 40, but it has a relatively low landing success rate compared to others.
- KSC LC 39A shows the highest landing success rate (about 77%) and accounts for the largest share of successful landings.
- Landing success rates improved over time, reaching a peak of 90% in 2019.
- Payloads up to 7000 kg are most common; heavier payloads tend to have higher success rates.
- Certain orbits (ES-L1, GEO, HEO, SSO) have 100% landing success, while others like GTO and SO have lower or no success.
- Launch sites are located close to coastlines, railways, and highways to minimize risk and facilitate logistics.
- Early missions had more failed landings, but failure rates decreased with experience and technology improvements.
- Logistic Regression was the preferred classification model due to its balanced accuracy and lower overfitting risk.
- Classification results showed around 83% test accuracy, with confusion matrix metrics indicating good prediction of successful and unsuccessful landings.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!