

# Winning Space Race with Data Science

Tobias Grassani  
August 27, 2024



# Outline

---

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

# Executive Summary

---

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - EDA results
  - Interactive analytics
  - Predictive analysis

# Introduction

---

## Background

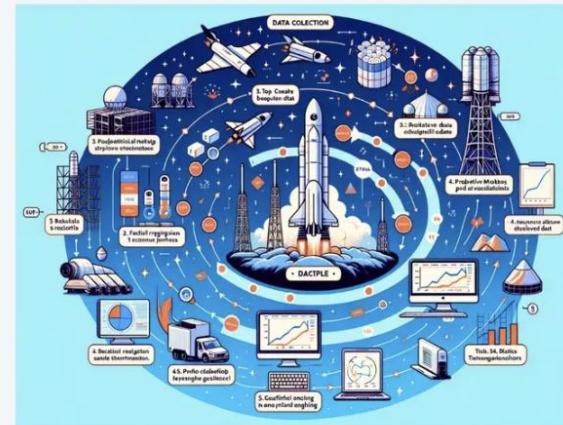
Our project is part of a big learning journey – the final course of the IBM Data Science series. Think of us as the brainy folks at SpaceY, a new company aiming to launch rockets just like SpaceX.

In this project, we will use the cool methodologies from data science to help SpaceY make smart choices when it comes to sending rockets into space, and maybe even outbid the big guys at SpaceX.

## Business Problem

Here's the scoop: SpaceX is the big name in space because they figured out how to send rockets up without breaking the bank. They say it costs about 62 million dollars to launch their Falcon 9, which is way cheaper than what other companies charge. They save a bunch by reusing the first part of the rocket. Normally, that part alone could set you back over 15 million dollars, not even counting the money spent developing it.

But sometimes, SpaceX has to let that first part go – maybe the satellite they're carrying is super heavy, or they need to send it to a special place in space. When that happens, it's bye-bye savings. So, we're on a mission to guess if that rocket part is going to make it back safely or not. Getting this right could mean big savings, and that's a number any shareholder would like to see.





Section 1

# Methodology

# Methodology – Executive Summary

---

- Data collection methodology:
  - SpaceX Open Source Rest API
  - Web Scraping from Wikipedia page ‘List of Falcon 9 and Falcon Heavy Launches’
- Perform data wrangling
  - Transforming categorical data using One Hot Encoding for machine learning algorithms and removing any empty or unnecessary information from the dataset.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression, K-Nearest Neighbors, Support Vector Machines, and Decision Tree models have been developed to determine the most effective classification method.

# Data Collection

---

The data sets are collected using 2 methods:

## 1) Request to the SpaceX API

- Gathered SpaceX's past launch data via their open-source API.
- Retrieved and processed this data with GET request.
- Ensured the data included only Falcon 9 launches.
- Filled in missing payload weights from secret missions with average values.

## 2) Web Scraping

- Requested past Falcon 9 and Falcon Heavy launch data from Wikipedia's relevant page.
- Accessed the Falcon 9 Launch page via its direct Wikipedia link.
- Extracted all the column names from the HTML table.
- Parsed and transformed the table into a Pandas data frame suitable for analysis.

# Data Collection – SpaceX API

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

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

```
In [29]: static_json_url="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/
```

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

```
In [30]: response.status_code
```

```
Out[30]: 200
```

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

```
In [31]: # Use json_normalize method to convert the json result into a dataframe  
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
In [32]: # Get the head of the dataframe  
data.head()
```

```
Out[32]: static_fir_date_utc  static_fir_date_unix
```

Requested and parsed the data from SpaceX API using GET request

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
In [41]: launch_dict = {"FlightNumber": list(data['flight_number']),  
'Date': list(data['date']),  
'BoosterVersion':BoosterVersion,  
'PayloadMass':PayloadMass,  
'Orbit':Orbit,  
'LaunchSite':LaunchSite,  
'Outcome':Outcome,  
'Flights':Flights,  
'GridFins':GridFins,  
'Reused':Reused,  
'Legs':Legs,  
'LandingPad':LandingPad,  
'Block':Block,  
'ReusedCount':ReusedCount,  
'Serial':Serial,  
'Longitude': Longitude,  
'Latitude': Latitude}
```

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

```
In [42]: # Create a data from launch_dict  
df = pd.DataFrame.from_dict(launch_dict)
```

Show the summary of the dataframe

```
In [43]: # Show the head of the dataframe  
df.head()
```

Stored the data and constructed the dataset into a new dictionary with relative columns

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

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

```
In [44]: # Hint data['BoosterVersion']!='Falcon 1'  
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

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

```
In [45]: data_falcon9.loc[:, 'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))  
data_falcon9
```

Filtered the dataframe to only include Falcon 9 launches needed for the project analysis

## Task 3: Dealing with Missing Values

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

```
In [62]: # Calculate the mean value of PayloadMass column  
# Replace the np.nan values with its mean value  
payloadmass_avg = data_falcon9['PayloadMass'].mean()  
data_falcon9['PayloadMass'].replace(np.nan, payloadmass_avg, inplace=True)
```

```
In [63]: data_falcon9.isnull().sum()
```

```
Out[63]: FlightNumber      0  
Date          0  
BoosterVersion 0  
PayloadMass     0  
Orbit          0  
Launchsite      0  
Outcome         0  
Flights         0  
GridFins        0  
Reused          0  
Legs            0  
LandingPad      26  
Block           0  
ReusedCount     0  
Serial          0  
Longitude        0  
Latitude         0  
dtype: int64
```

You should see the number of missing values of the `PayloadMass` change to zero.

Now we should have no missing values in our dataset except for in `LandingPad`.

Cleaned the data and removed missing values of PayloadMass

My GitHub reference: [1. Hands-on Lab Complete the Data Collection API Lab.ipynb](#)

# Data Collection – Web Scraping from Wikipedia

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

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

```
In [5]:  
# 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
```

```
In [6]:  
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content  
soup = BeautifulSoup(response.text)  
  
Print the page title to verify if the BeautifulSoup object was created properly
```

```
In [7]:  
# Use soup.title attribute  
page_title = soup.title.string  
print(page_title)  
  
List of Falcon 9 and Falcon Heavy launch
```

Requested data from Wikipedia using HTTP GET request and BeautifulSoup

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

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

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

```
In [8]:  
# Use the find_all function in the BeautifulSoup object, with element type 'table'  
# Assign the result to a list called 'html_tables'  
  
html_tables = soup.find_all('table')
```

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

```
In [9]:  
# Let's print the third table and check its content  
first_launch_table = html_tables[2]  
print(first_launch_table)  
  
<table class="wikitable plainrowheaders collapsible" style="width: 100%;>  
  <tbody><tr>  
    <th scope="col">Flight No.  
    <th scope="col">Date and<br/>time (<a href="
```

Extracted all column/variable names from the HTML table header

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

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

```
In [13]:  
launch_dict= dict.fromkeys(column_names)  
  
# Remove an irrelevant column  
del launch_dict['Date and time ( )']  
  
# Let's initial the launch_dict with each value to be an empty list  
launch_dict['Flight No.']= []  
launch_dict['Launch site']= []  
launch_dict['Payload']= []  
launch_dict['Orbit']= []  
launch_dict['Customer']= []  
launch_dict['Launch outcome']= []  
# Added some new columns  
launch_dict['Booster booster']= []  
launch_dict['Booster landing']= []  
launch_dict['Date']= []  
launch_dict['Time']= []
```

Cleaned the column data, created an empty dictionary with extracted columns and appended the column names

```
# Booster landing  
# TODO: Append the launch_outcome into launch_dict with key 'Booster landing'  
booster_landing = landing_status[row[8]]  
launch_dict['Booster landing'].append(booster_landing) #TODO-11  
print(booster_landing)
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
In [16]:  
df=pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })  
df=pd.DataFrame(launch_dict)  
df
```

Out [16]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Booster booster	Booster landing	Booster attempt	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	NASA (COTS)\NROL	Success	v1.0B0004.1	F9	Failure	December 2010	8:15:43
1	2	CCAFS	Dragon	0 LEO	NASA (COTS)\NROL	Success	v1.0B0004.1	F9	Failure	No attempt\	22 May 2012	15:43
2	3	CCAFS	Dragon	525 kg LEO	NASA (COTS)	Success	v1.0B0005.1	F9	No attempt\	attempt\	22 May 2012	07:44

Created a data frame by parsing the launch HTML tables

My GitHub reference: [2. Hands-on Lab Complete the Data Collection with Web Scraping lab.ipynb](#)

# Data Wrangling

## TASK 1: Calculate the number of launches on each site

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

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

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

```
In [5]: # Apply value_counts() on column LaunchSite
launch_site_count = df['LaunchSite'].value_counts()
print(launch_site_count)
```

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

Calculated the number of launches of each site

## TASK 2: Calculate the number and occurrence of each orbit

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

```
In [6]: orbit_count = df['Orbit'].value_counts()
print(orbit_count)
```

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	

## TASK 3: Calculate the number and occurrence of mission outcome of the orbits

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

```
In [7]: # landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
print(landing_outcomes)
```

True ASDS	41
None None	19
True None	14
False ASDS	6
True Ocean	5
False Ocean	2
None ASDS	2

Calculated the number and occurrence of each orbit and its mission outcome

## TASK 4: Create a landing outcome label from Outcome column

Using the `bad_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`:

```
In [10]: # landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

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

```
In [11]: df['Class']=landing_class
df[['Class']].head(8)
```

Class	0	0
-------	---	---

Created a landing outcome label form Outcome column

```
In [12]: df.head(5)
```

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

We can use the following line of code to determine the success rate:

```
In [13]: df['Class'].mean()
```

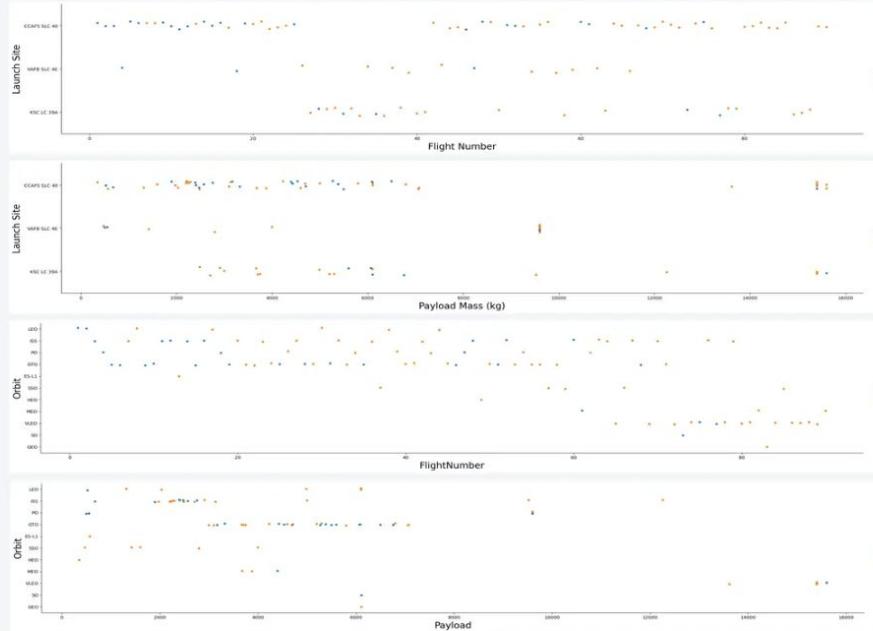
0.6666666666666666

```
df.to_csv("dataset_part_2.csv", index=False)
```

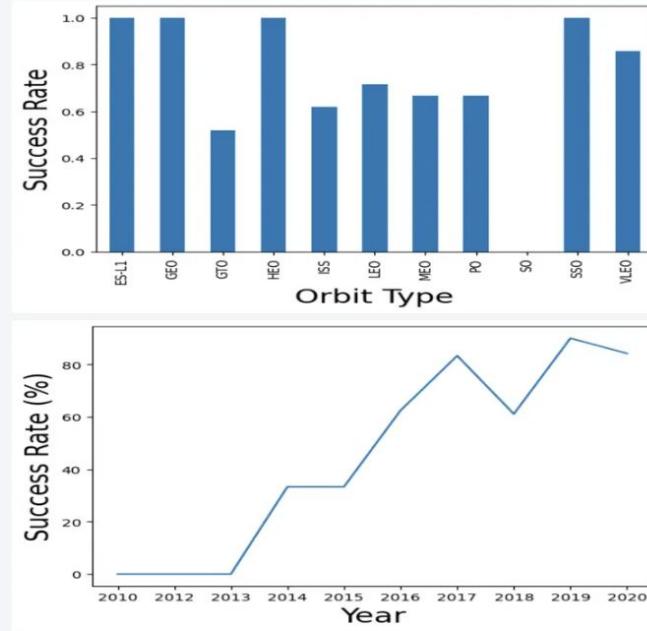
Calculated the average success rate

# EDA with Data Visualization

Scatter point charts are used to visualize the relationship between (1) Flight Number and Launch Site, (2) Payload and Launch Site, (3) Flight Number and Orbit type, (4) Payload and Orbit type



Bar chart was used to visualize the relationship between success rate of each orbit type; Line chart was used to visualize the launch success yearly trend



# EDA with SQL

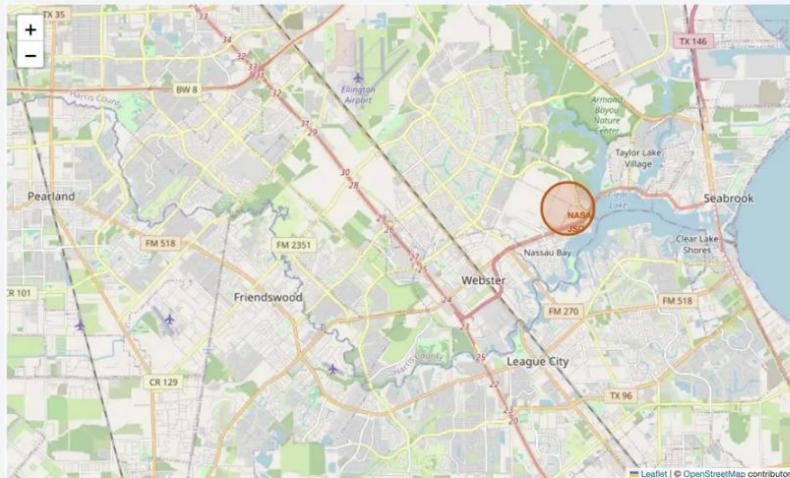
---

## SQL queries are performed to:

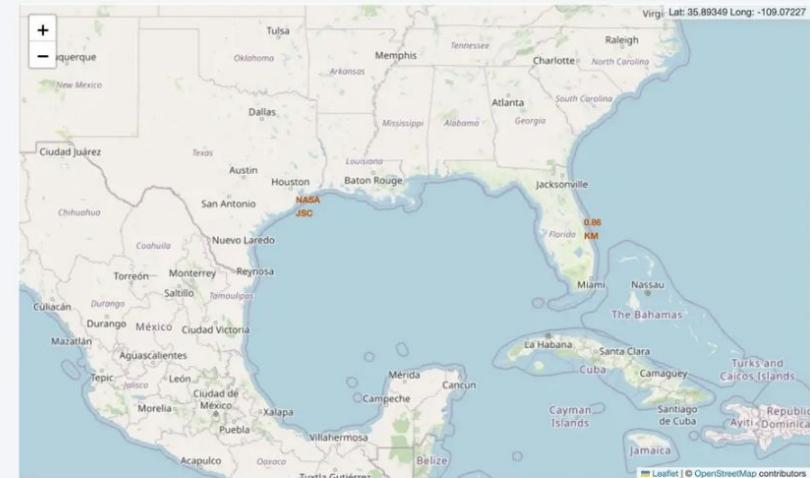
1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when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

# Build an Interactive Map with Folium

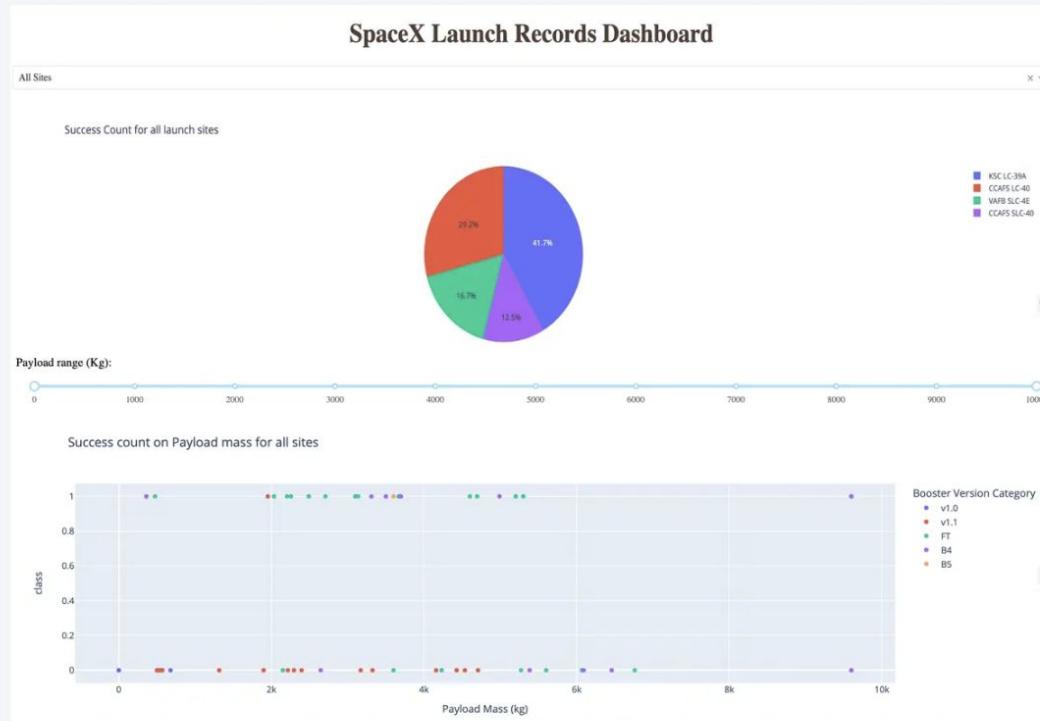
A circle marker was created to show NASA Johnson Space Center's coordinate



Distance marker was created to show distances between a launch site to its proximities



# Build a Dashboard with Plotly Dash



Dropdown option of pie chart created to show the success launches of all / each site

Scatter plot with payload range slider to show the success launches of all / each site by payload mass

My GitHub reference: [7. Hands-on Lab Build an Interactive Dashboard with Plotly Dash\\_spacex\\_dash\\_app.py](#)

# Predictive Analysis (Classification)

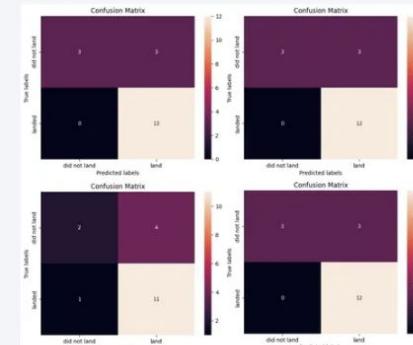
- LR, SVM, Decision Tree and KNN objects are created and fit with GridSearchCV object to find the best parameters, then the models are trained on the training set.
- The accuracy of test data are calculated for each machine learning model. It is found that the methods performed best are LR, SVM, KNN where all 3 achieved the highest accuracy of 83.33%.

## TASK 12

Find the method performs best:

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

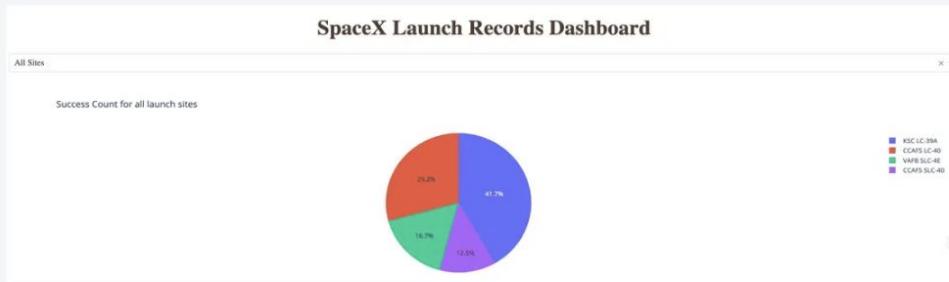
```
LR Accuracy: 83.33%  
SVM Accuracy: 83.33%  
Decision Tree Accuracy: 72.22%  
KNN Accuracy: 83.33%
```



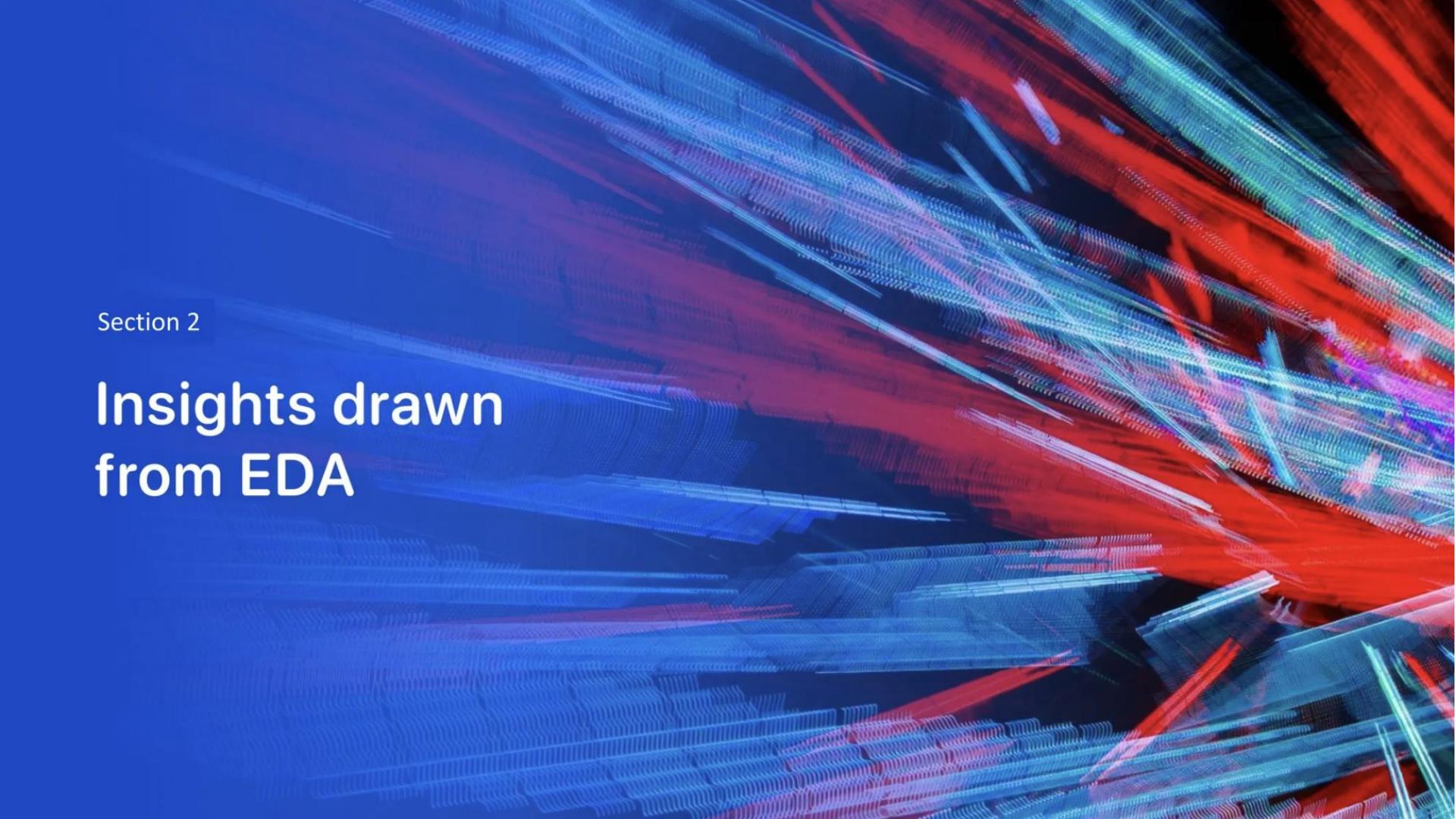
# Results

---

- LR, SVM, KNN are top-performing models for forecasting outcomes in this data.
- Lighter payloads have a higher performance compared to heavier ones.
- The likelihood of a SpaceX launch succeeding increases with the number of years of experience, suggesting a trend towards flawless launches over time.
- Launch Complex 39A at Kennedy Space Center has the highest number of successful launches compared to other launch sites.
- GEO, HEO, SSO, ES L1 orbit types exhibit the highest rates of successful launches.



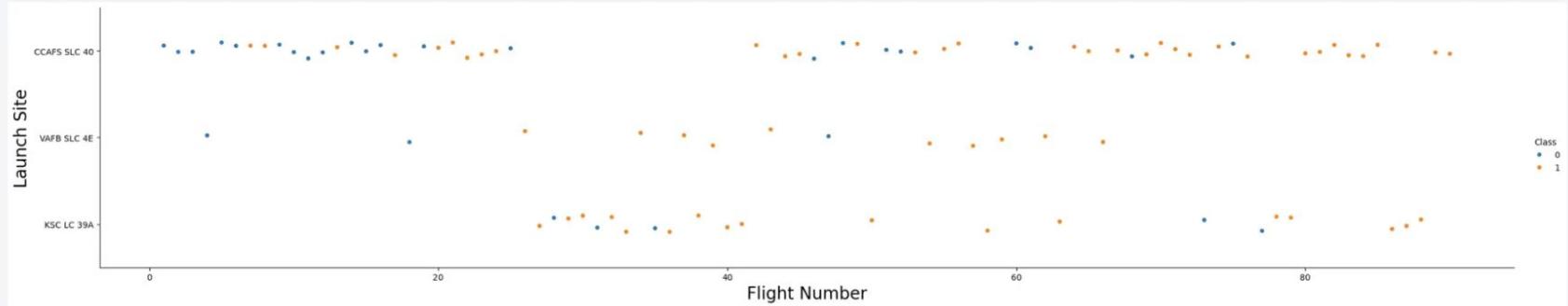
KSC LC-39A has the most successful launches overall

The background of the slide features a dynamic, abstract pattern of glowing particles. These particles are arranged in various lines and clusters, creating a sense of depth and motion. The colors used are primarily shades of red, blue, and green, which stand out against a dark, almost black, background. The overall effect is reminiscent of a microscopic view of a complex system or a futuristic digital landscape.

Section 2

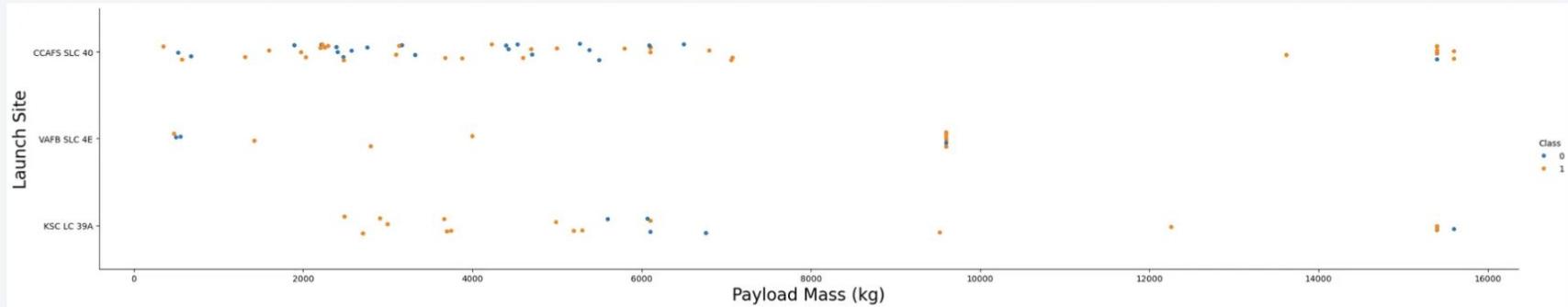
## Insights drawn from EDA

# Flight Number vs. Launch Site



- Total number of launches from launch site CCAFS SLC 40 are significantly higher than the other launch sites.

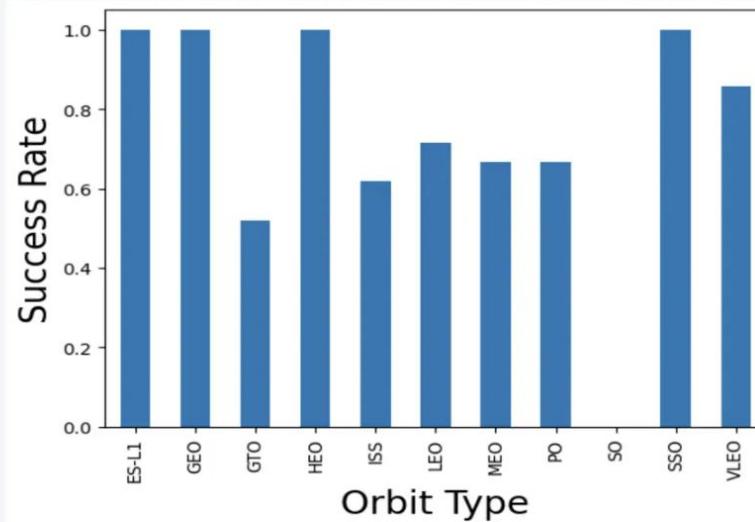
# Payload vs. Launch Site



- Payloads with lower mass are have more launches compared to those with higher mass across all three launch sites.

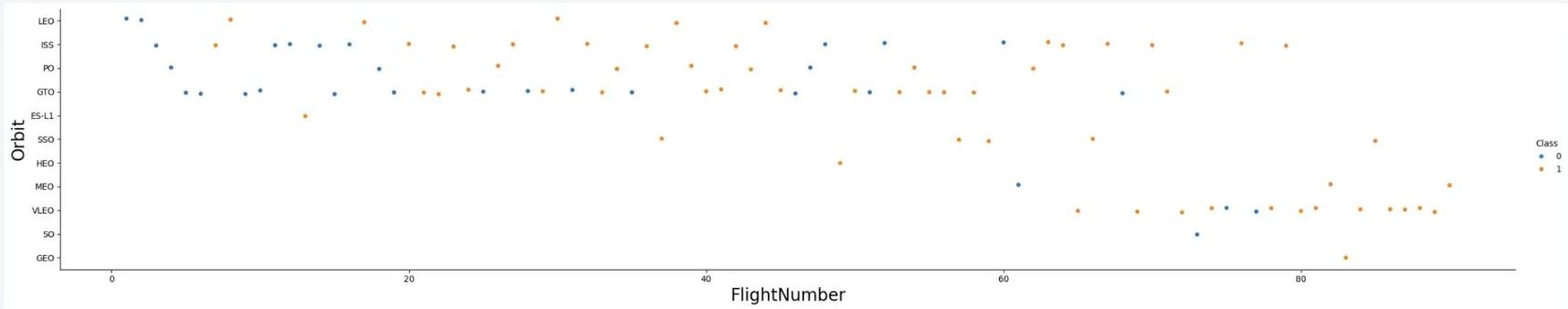
# Success Rate vs. Orbit Type

---



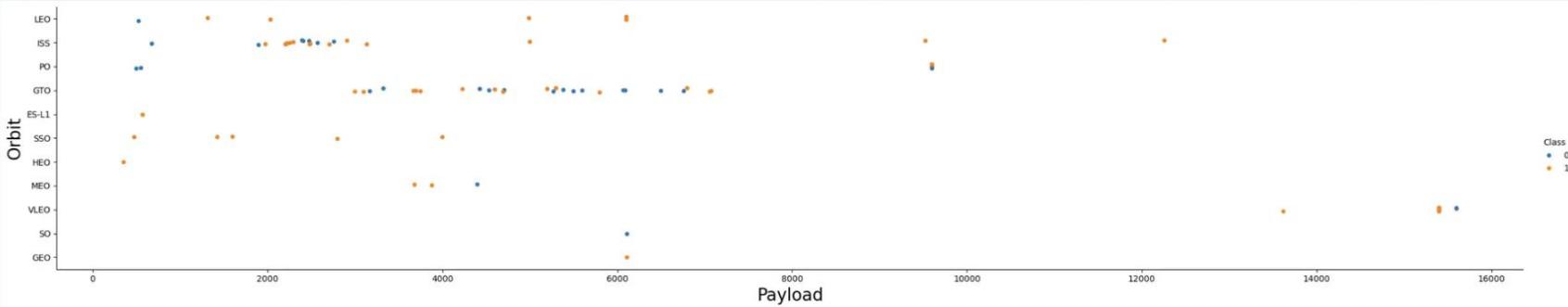
- Orbit types ES-L1, GEO, HEO, SSO have the highest success rate among all.

# Flight Number vs. Orbit Type



- LEO, ISS, PO, GTO orbits have the most launches in the earlier years, but it slowly shifted to VLEO orbit in the later years.

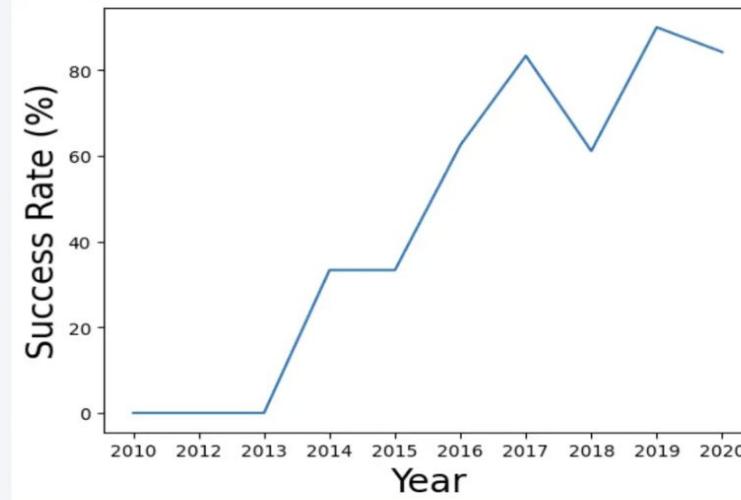
# Payload vs. Orbit Type



- Heavy payloads tend to have higher successful landing rates for PO, LEO, and ISS orbits, but for GTO orbit, success is less predictable with an almost equal mix of successes and failures.

## Launch Success Yearly Trend

---



- The success rate of launches have been increasing since 2013 till 2020, possibly due to technology advancement and experience.

# All Launch Site Names

---

## Task 1

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

```
In [15]: %sql select distinct Launch_Site FROM SPACEXTABLE  
* sqlite:///my_data1.db  
Done.
```

```
Out[15]: Launch_Site  
CCAFS LC-40  
VAFB SLC-4E  
KSC LC-39A  
CCAFS SLC-40
```

- Performed an SQL query to obtain all launch site names

# Launch Site Names Begin with 'CCA'

Task 2

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

```
In [16]: %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_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (0)
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 (0)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

- Performed an SQL query to obtain 5 launch site names that begin with 'CCA'

# Total Payload Mass

---

## Task 3

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

```
In [17]: %sql select sum(PAYLOAD_MASS__KG_) from SPACETABLE where customer = 'NASA (CRS)'  
* sqlite:///my_data1.db  
Done.  
Out[17]: sum(PAYLOAD_MASS__KG_)  
45596
```

- Performed an SQL query to obtain the total payload mass carried by boosters launched by NASA (CRS)

# Average Payload Mass by F9 v1.1

---

## Task 4

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

```
In [18]: %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = 'F9 v1.1'  
* sqlite:///my_data1.db  
Done.  
Out[18]: avg(PAYLOAD_MASS__KG_)  
2928.4
```

- Performed an SQL query to calculate the average payload mass carried by booster version F9 v1.1

# First Successful Ground Landing Date

---

## Task 5

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

*Hint: Use min function*

In [21]:

```
%%sql
SELECT min(Date)
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (ground pad)';
```

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

Out[21]:

min(Date)

2015-12-22

- Performed an SQL query to find the dates of the first successful landing outcome on ground pad

# Successful Drone Ship Landing with Payload between 4000 and 6000

---

```
%sql select distinct Booster_Version from SPACEXTABLE where Landing_Outcome = 'Success  
(drone ship)' and PAYLOAD_MASS_KG_ > 4000 and PAYLOAD_MASS_KG_ < 6000
```

## Task 6

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

```
In [22]: %sql select distinct Booster_Version from SPACEXTABLE where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_  
* sqlite:///my_data1.db  
Done.  
Out[22]: Booster_Version  
F9 FT B1022  
F9 FT B1026  
F9 FT B1021.2  
F9 FT B1031.2
```

- Performed an SQL query to list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

# Total Number of Successful and Failure Mission Outcomes

---

## Task 7

List the total number of successful and failure mission outcomes

```
In [23]: %sql select distinct Mission_Outcome, count(*) from SPACEXTABLE group by Mission_Outcome  
* sqlite:///my_data1.db  
Done.
```

```
Out[23]:
```

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

- Performed an SQL query to calculate the total number of successful and failure mission outcomes

# Boosters Carried Maximum Payload

```
%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_)from SPACEXTABLE)
```

## Task 8

List the names of the booster\_versions which have carried the maximum payload mass. Use a subquery

```
In [25]: %sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_)from SPACEXTABLE)
* sqlite:///my_data1.db
Done.
```

```
Out[25]: 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

- Performed an SQL query to list the names of the booster which have carried the maximum payload mass

# 2015 Launch Records

---

```
%sql select substr(Date, 6,2) as Month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE where Landing_Outcome = 'Failure (drone ship)' and substr(Date,0,5) = '2015'
```

## 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: SQLITE 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.

```
In [26]: %sql select substr(Date, 6,2) as Month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE where La
          * sqlite:///my_data1.db
Done.
```

```
Out[26]:   Month  Landing_Outcome  Booster_Version  Launch_Site
              01  Failure (drone ship)  F9 v1.1 B1012  CCAFS LC-40
              04  Failure (drone ship)  F9 v1.1 B1015  CCAFS LC-40
```

- Performed an SQL query to list the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015

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

---

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

## 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.

```
In [27]: %sql select Landing_Outcome, count(*) as 'Count' from SPACEXTABLE where Date between '2010-06-04' and '2017-03-20'  
* sqlite:///my_data1.db  
Done.  
Out[27]:
```

Landing_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

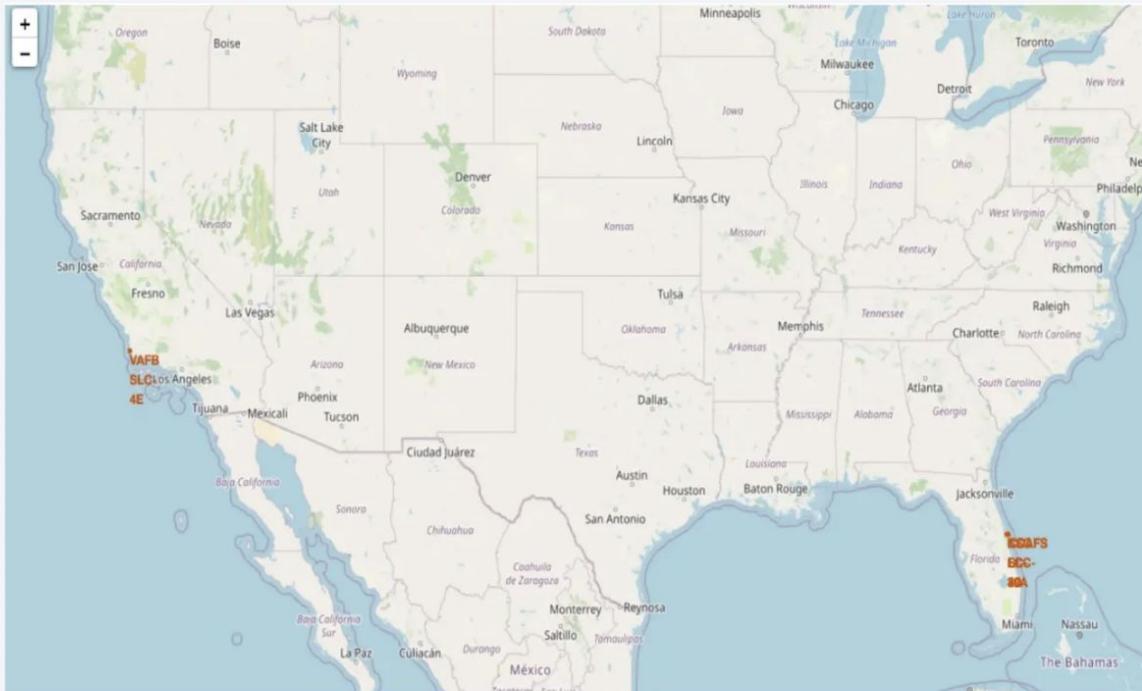
- Performed an SQL query to rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

The background of the slide is a photograph taken from space at night. It shows the curvature of the Earth against the dark void of space. City lights are visible as numerous small white and yellow dots, primarily concentrated in the lower right quadrant where the United States and Mexico would be. There are also larger, more intense clusters of lights in South America and Europe. The atmosphere appears as a thin blue layer above the planet's surface.

Section 3

# Launch Sites Proximities Analysis

# All launch sites on a map



- The launch sites are labelled by a marker with their names on the map.

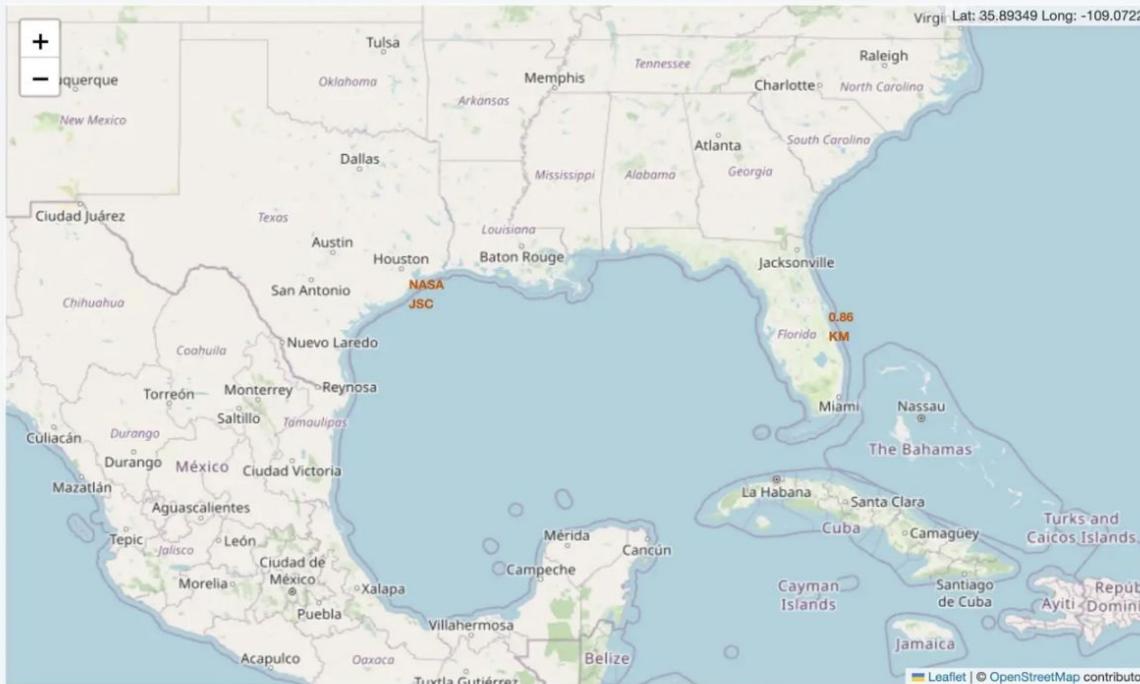
## All success/failed launches for each site on the map

---



- The launch records are grouped in clusters on the map, then labelled by green markers for successful launches, and red markers for unsuccessful ones.

# Distances between a launch site to its proximities

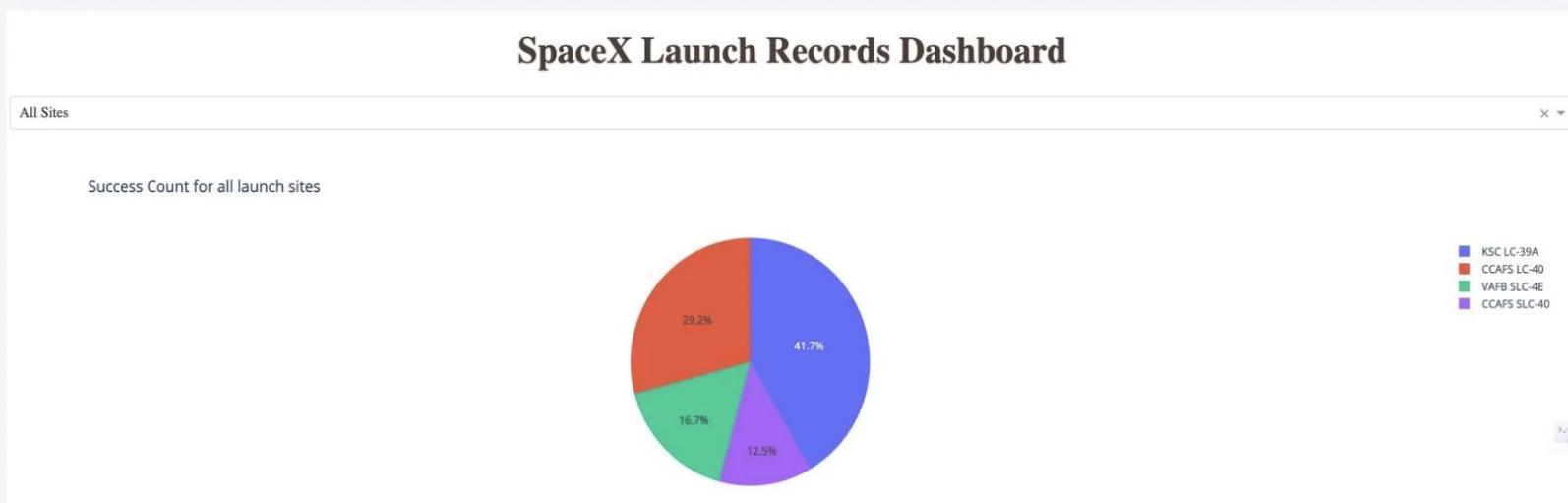


- The closest coastline from NASA JSC is marked as a point using MousePosition and the distance between the coastline point and the launch site, which is approximately 0.86 km.

Section 4

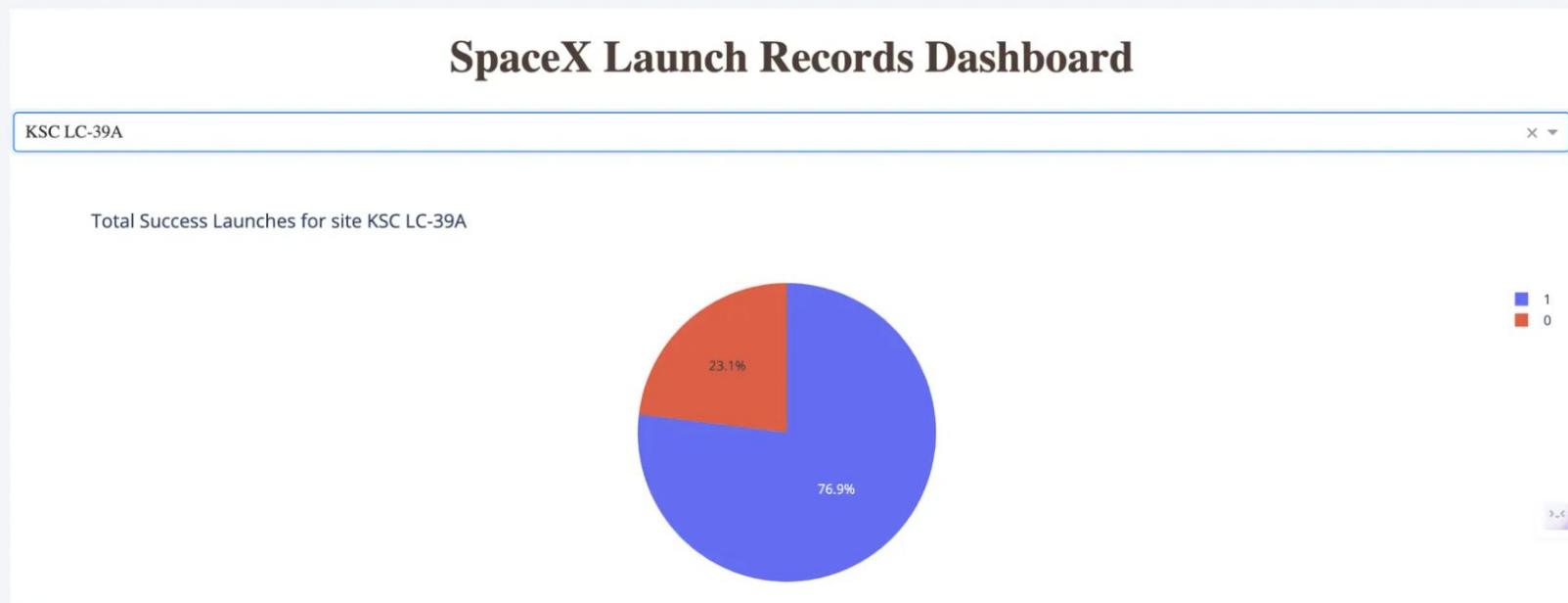
# Build a Dashboard with Plotly Dash

# Total success launches for all sites



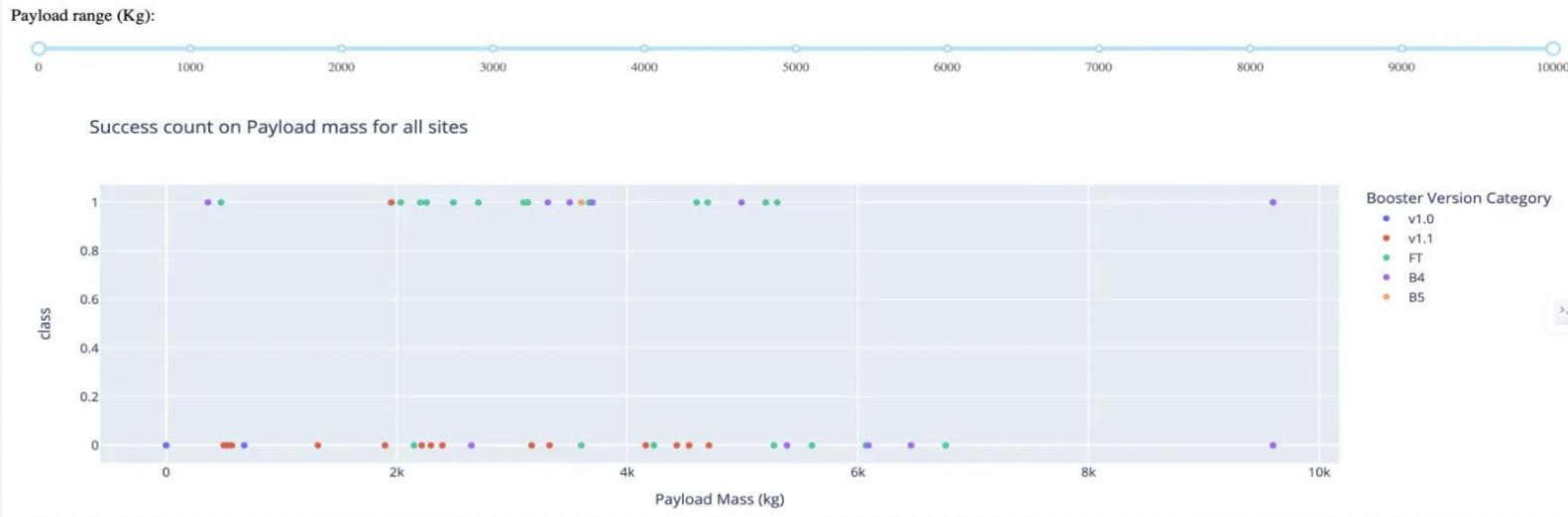
- KSC LC-39A has the highest amount of success launches with 41.7% from the entire record, whereas CCAFS SLC-40 has the lowest amount of success launches with only 12.5%.

## Success ratio of the launch site with the highest success launches



- KSC LC-39A which is the launch site with highest amount of success, has a 76.9% success rate for the launches from its site, and 23.1% failure rate.

# Payload vs. launch outcome



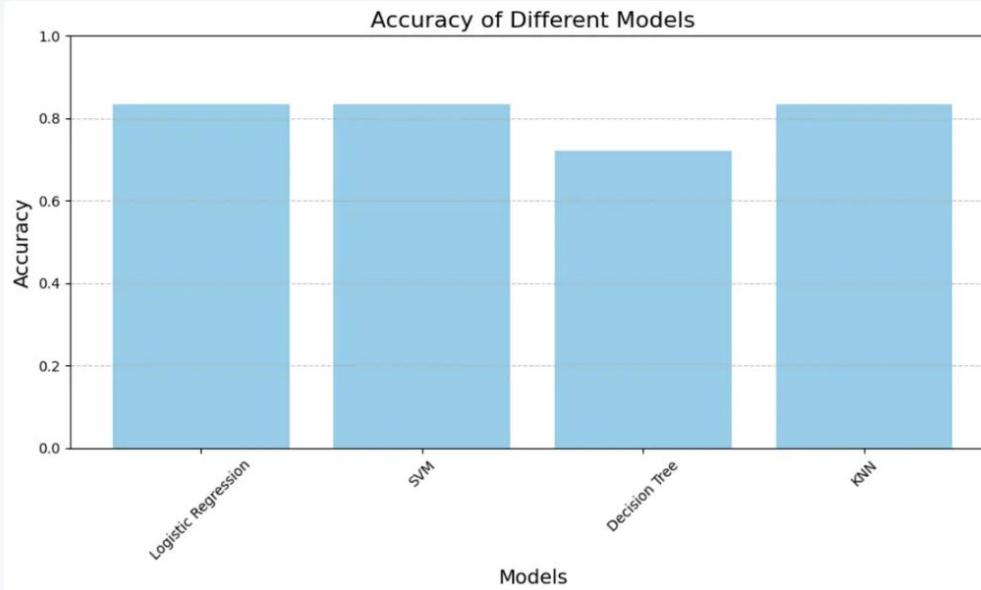
- The payload range that has the highest success launches is between 2,000 to 4,000 kg, which can be seen by the most number of plots in that range, followed by the payload range of 4,000 to 6,000 kg, with the second most number of plots.
- Booster version FT (green spots) has the highest success launches, followed by B4 (purple spots) with the second highest success launches, among all booster versions.

The background of the slide features a dynamic, abstract design. It consists of several thick, curved lines that transition from a bright yellow at the top right to a deep blue at the bottom left. These lines create a sense of motion and depth, resembling a tunnel or a stylized landscape. The overall effect is modern and professional.

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy



## TASK 12

Find the method performs best:

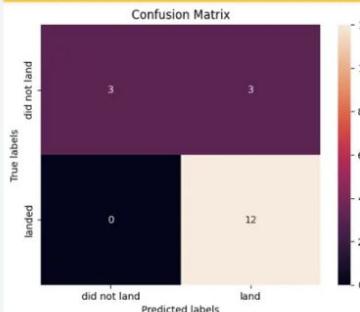
```
print('LR Accuracy:', '{:.2%}'.format(logreg_accuracy))
print('SVM Accuracy:', '{:.2%}'.format(svm_accuracy))
print('Decision Tree Accuracy:', '{:.2%}'.format(tree_accuracy))
print('KNN Accuracy:', '{:.2%}'.format(knn_accuracy))
```

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

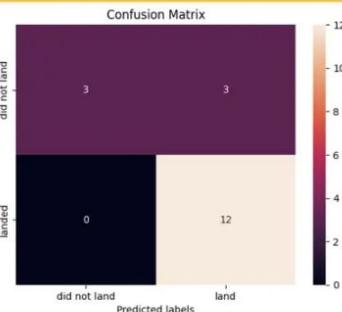
- The model that performed best are LR, SVM, KNN where all 3 achieved the highest accuracy of 83.33%.

# Confusion Matrix

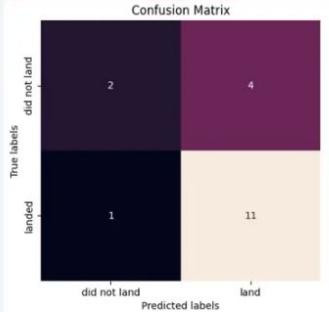
Confusion Matrix of LR



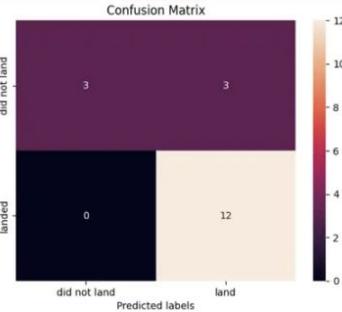
Confusion Matrix of SVM



Confusion Matrix of Decision Tree



Confusion Matrix of KNN

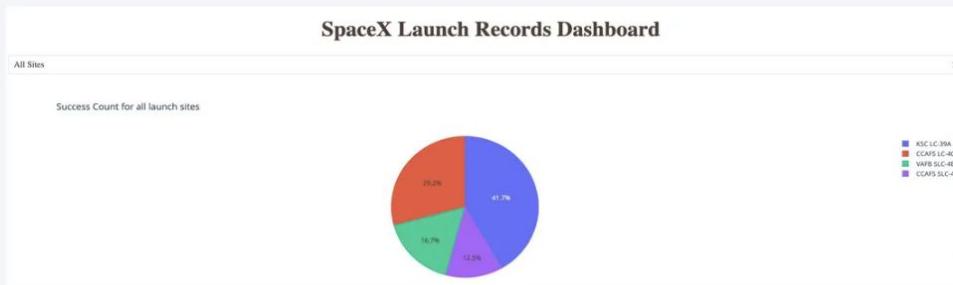


- LR, SVM, KNN models are good as their confusion matrix show that they predicted all 12 successful landing correctly, with 0 error.
- However, the Decision Tree model only predicted 11 successful landing correctly, with one of them wrongly predicted as a failed / did not land.
- LR, SVM, KNN models have the same accuracy of 83.33% as displayed earlier, hence the same confusion matrix.

# Conclusions

---

- LR, SVM, KNN are top-performing models for forecasting outcomes in this data.
- Lighter payloads have a higher performance compared to heavier ones.
- The likelihood of a SpaceX launch succeeding increases with the number of years of experience, suggesting a trend towards flawless launches over time.
- Launch Complex 39A at Kennedy Space Center has the highest number of successful launches compared to other launch sites.
- GEO, HEO, SSO, ES L1 orbit types exhibit the highest rates of successful launches.



KSC LC-39A has the most successful launches overall

Thank you!

