IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jelger Oud
20/12/2024

Outline

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

Executive Summary

- In today's highly competitive aerospace industry:
- **SpaceX** has transformed satellite launches with its innovative approach to reusable rockets, specifically the **Falcon 9** and **Falcon Heavy**.
- **Key benefit**: This approach dramatically lowers the cost per kilogram of payload.
- **Challenges**: Despite its cost advantages, reliability concerns persist when compared to traditional launch systems like **Soyuz** and **Ariane 5**.
- **Competitive edge**: To sustain its low-cost advantage over conventional launch vehicles, the success of a **Falcon 9** mission is primarily defined by the ability to recover or land the booster safely.
- **Booster recovery success**: Several factors influence the likelihood of successful booster recovery, including:
    - **Orbital parameters**
    - **Payload weight**
    - **Booster configurations**
    - **Launch site locations**
- Leveraging these parameters, the **supervised machine learning classification model** developed in this study achieved an **accuracy rate of nearly 94%** in predicting booster recovery outcomes.

Introduction

- **Project Background and Context**
- SpaceX promotes its **Falcon 9** rocket launches at a cost of **$62 million** per launch, while competing providers charge upwards of **$165 million**. The significant cost reduction is largely attributed to SpaceX's ability to **reuse the first-stage booster** of its rockets.
- If we can accurately **predict whether the first stage of the Falcon 9 rocket will successfully land**, we can estimate launch costs more effectively. Such insights would be valuable for **competitors** looking to **bid against SpaceX** in the commercial launch market.
- **Project Objective**
- In this capstone project, we aim to develop a **predictive model** using **machine learning techniques** to forecast the **successful landing of the Falcon 9 first stage** based on historical launch data provided on SpaceX’s website.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

- **Data Source**
  The data used in this project was collected through the **SpaceX REST API** by making **GET requests** to retrieve historical launch records.
- **Steps Taken:**

1. A **GET request** was sent to the SpaceX API endpoint to access launch data.
2. The **response content** was received in **JSON format**.
3. The JSON data was **parsed and decoded** into a structured format.
4. The data was then **converted into a Pandas DataFrame** for further analysis and preprocessing.

- **Code Repository**
  The complete code for the **SpaceX API calls notebook** can be accessed on **GitHub**:

👉 *[https://github.com/JellaGit/C10_Git/blob/main/jupyter-labs-spacex-data-collection-api-v2.ipynb]*

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:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
response.status_code
```

```
200
```

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

```
# Use json_normalize meethod to convert the json result into a dataframe

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe

data.head() # 1st 5 rows

```

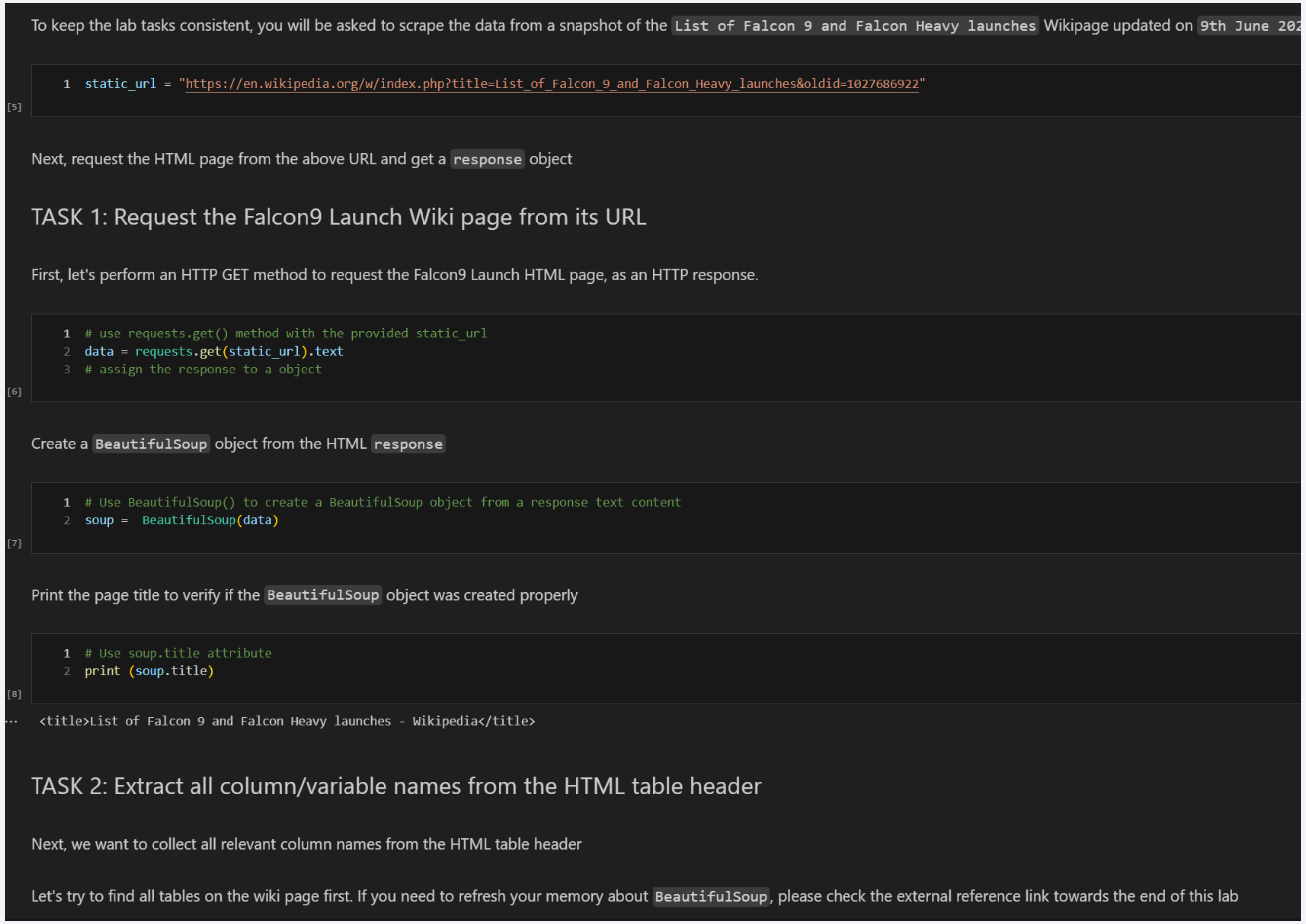
	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads	launchpad	flight_numbe
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5090995de566f86	
								Successful first stage burn and						

Data Collection - Scraping

- **Data Source**
  To complement the data obtained from the SpaceX API, additional **Falcon 9 historical launch records** were gathered through **web scraping** from a **Wikipedia page**.
- **Steps Taken:**

1. Utilized the **Beautiful Soup** library along with the **requests** module to fetch the webpage content.
2. Extracted the **HTML table** containing Falcon 9 launch data.
3. **Parsed the HTML table** into a structured format.
4. Created a **Pandas DataFrame** to store and process the extracted data.

- **Code Repository**
  The complete code for the **web scraping notebook** is available on **GitHub**:
  👉[https://github.com/JellaGit/C10_Git/blob/main/Web%20scraping%20Falcon%209%20and%20Falcon%20Heavy%20Launches%20Records.ipynb]



To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 202`

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

Next, request the HTML page from the above URL and get a `response` object

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.

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

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

```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup =  BeautifulSoup(data)
```

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

```python
# Use soup.title attribute
print (soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

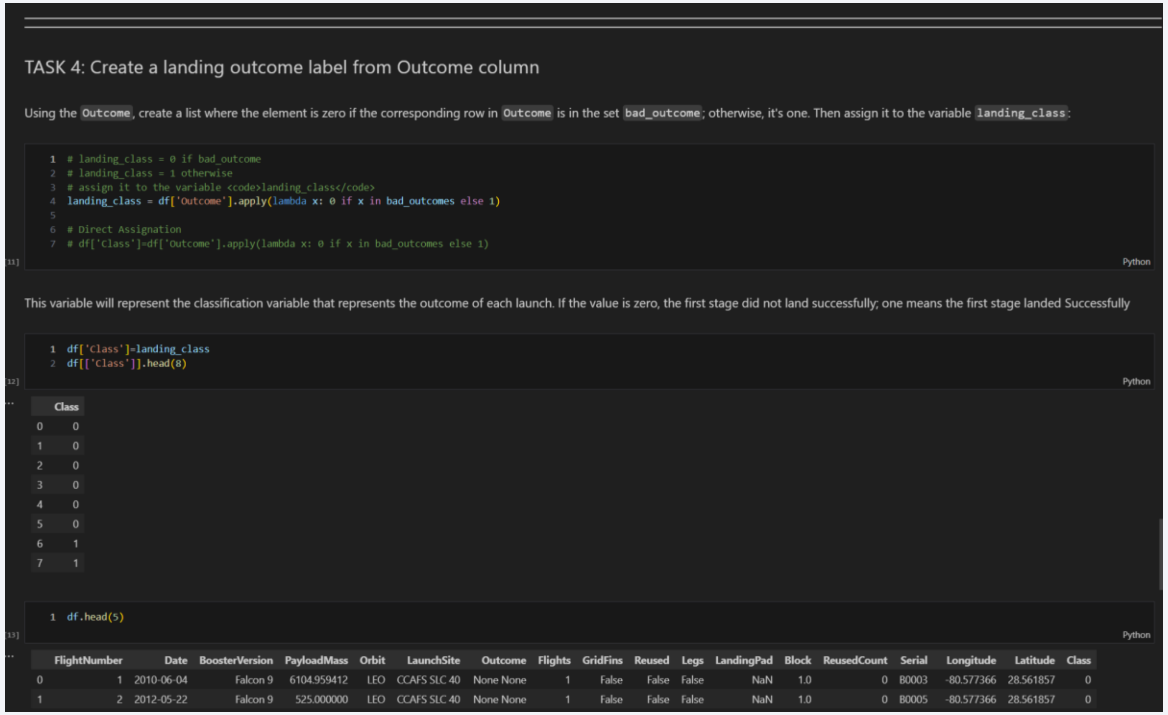
TASK 2: Extract all column/variable names from the HTML table header

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

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

Data Wrangling

- **1. Filtering Falcon 9 Launches**
- The **Booster Version** column was used to filter the data, ensuring only **Falcon 9 launches** were retained in the dataset.
- This step helped remove irrelevant records and focus solely on Falcon 9 missions.
- **2. Handling Missing Values**
- **Payload Mass** column had **missing values**.
- These were **replaced** by the **mean value** of the column to maintain consistency and prevent data loss.
- **GitHub Repository**
  The complete **data wrangling notebooks** can be found here:
  👉 [https://github.com/JellaGit/C10_Git/blob/main/labs-jupyter-spacex-Data%20wrangling-v2.ipynb]



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
# assign it to the variable <code>landing_class</code>
landing_class = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)

# Direct Assignation
# df['Class']=df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 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

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

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

```
df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0

EDA with Data Visualization

1. Exploratory Data Analysis (EDA)

- **Pandas** and **Matplotlib** were used to explore and analyze relationships between variables.
- Statistical summaries and visual patterns were extracted to understand data distributions.

2. Feature Engineering

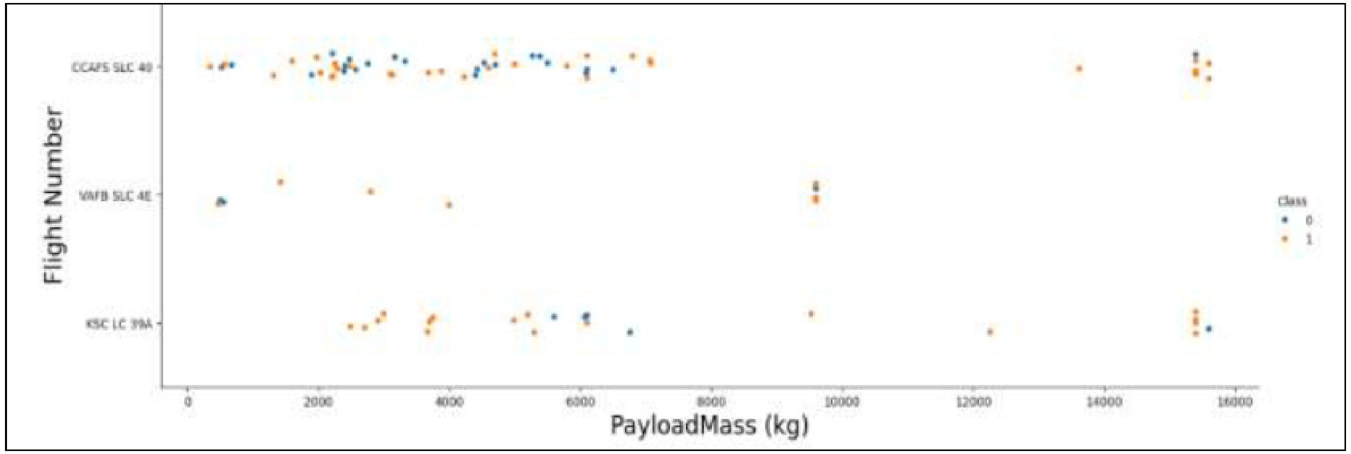
- Prepared and transformed data to create meaningful **features** for modeling.
- Enhanced dataset usability by identifying key variables influencing launch success.

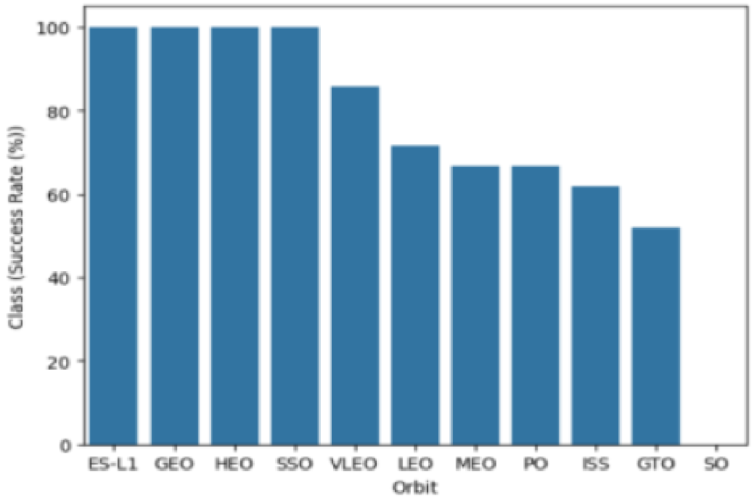
3. Data Visualizations

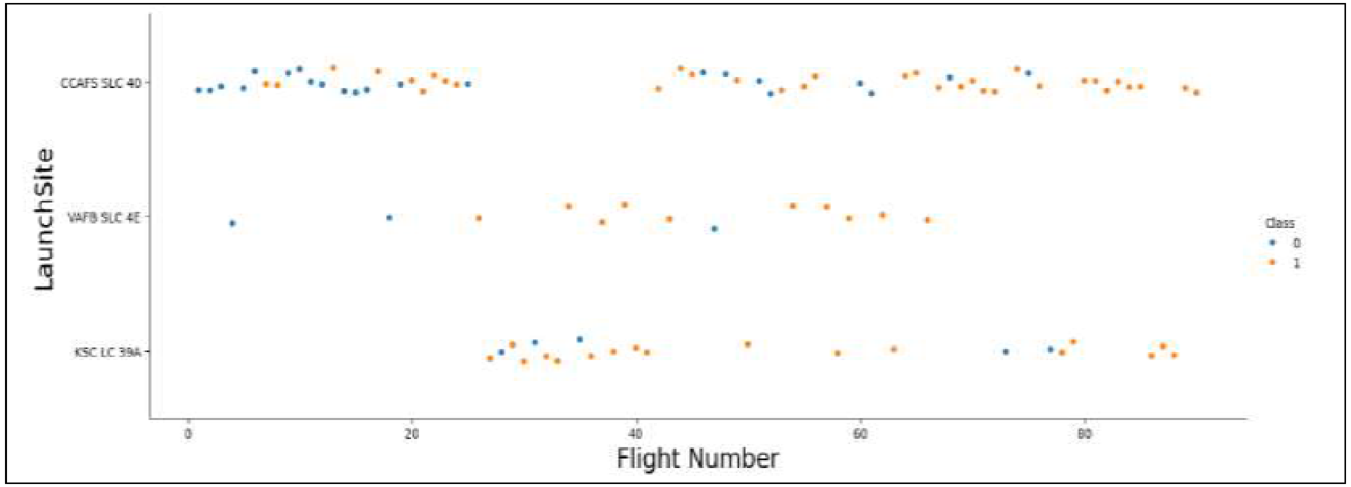
- **Scatter Plots:**
  - Visualized relationships between:
    - **Flight Number** and **Launch Site**
    - **Payload Mass** and **Launch Site**
    - **Flight Number** and **Orbit Type**
    - **Payload Mass** and **Orbit Type**
- **Bar Charts:**
  - Examined **success rates** across **different orbit types**.
- **Line Plots:**
  - Visualized **yearly trends** in **launch success** rates.

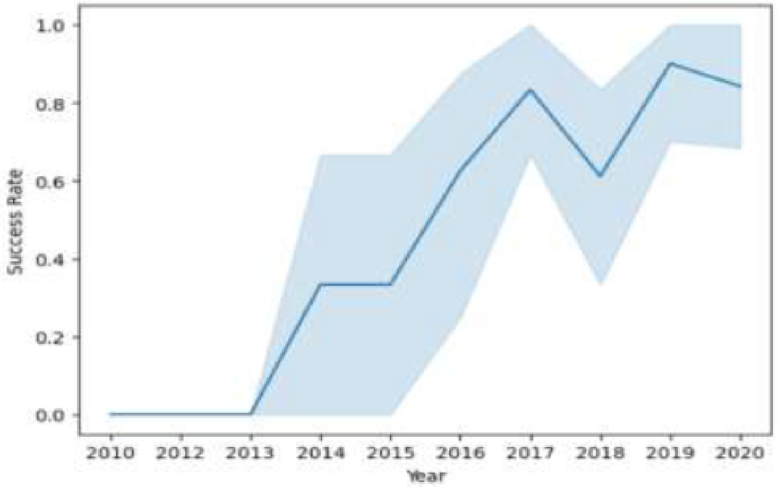
GitHub Repository
The completed **EDA and data visualization notebooks** can be accessed here:
👉[https://github.com/JellaGit/C10_Git/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb]

EDA Visualization Charts









EDA with SQL

1.Display the names of the unique launch sites in the space mission:
%%sql
SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;

2.Display 5 records where launch sites begin with the string 'CCA':
%%sql
SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;

3.Display the total payload mass carried by boosters launched by NASA (CRS):
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';

4.Display the average payload mass carried by booster version F9 v1.1:
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';

5.List the date when the first successful landing outcome in the ground pad was achieved:
%%sql
SELECT MIN(DATE) AS First_Successful_Landing FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';

Notes:
•Queries extract insights into **launch site details**, **payload mass**, and **landing success timelines**.

SQL functions such as SUM(), AVG(), DISTINCT, and MIN() are used for aggregations and filtering.

•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 SPACEXTBL where LANDING_OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;

•List the total number of successful and failure mission outcomes
%%sql
select count(MISSION_OUTCOME) as mission outcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;

•List the names of the booster versions which have carried the maximum payload mass. Use a subquery
%%sql
select BOOSTER_VERSION as booster version from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);

GitHub URL of Completed EDA with SQL Notebook
👉
[https://github.com/JellaGit/C10_Git/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb]

Build an Interactive Map with Folium

👉 Interactive Map with Folium - SpaceX Launch Sites

Key Features in the Notebook

- **Map Creation:** Used **Folium** to generate an interactive map displaying **launch sites**.
- **Markers:** Placed **markers** to indicate **launch outcomes** (success = 1, failure = 0).
- **Circles and Lines:** Visualized **distances** and **coverage areas** around launch sites.
- **Marker Clusters:** Grouped multiple markers for **better visualization**.
- **Mouse Hover and Popups:** Displayed **site details** and **launch outcomes** on interaction.

Purpose:

- **External reference** for **peer reviews** and **collaboration**.
- Provides **reproducible steps** to create **interactive visualizations** using **Folium**.
- Helps explore **launch data geographically** to analyze **patterns** and **success rates**.

Build a Dashboard with Plotly Dash

Launch Site Dropdown Input Component:
•Allows users to **select a launch site** from a dropdown menu.
•Filters data dynamically based on the **selected site**.

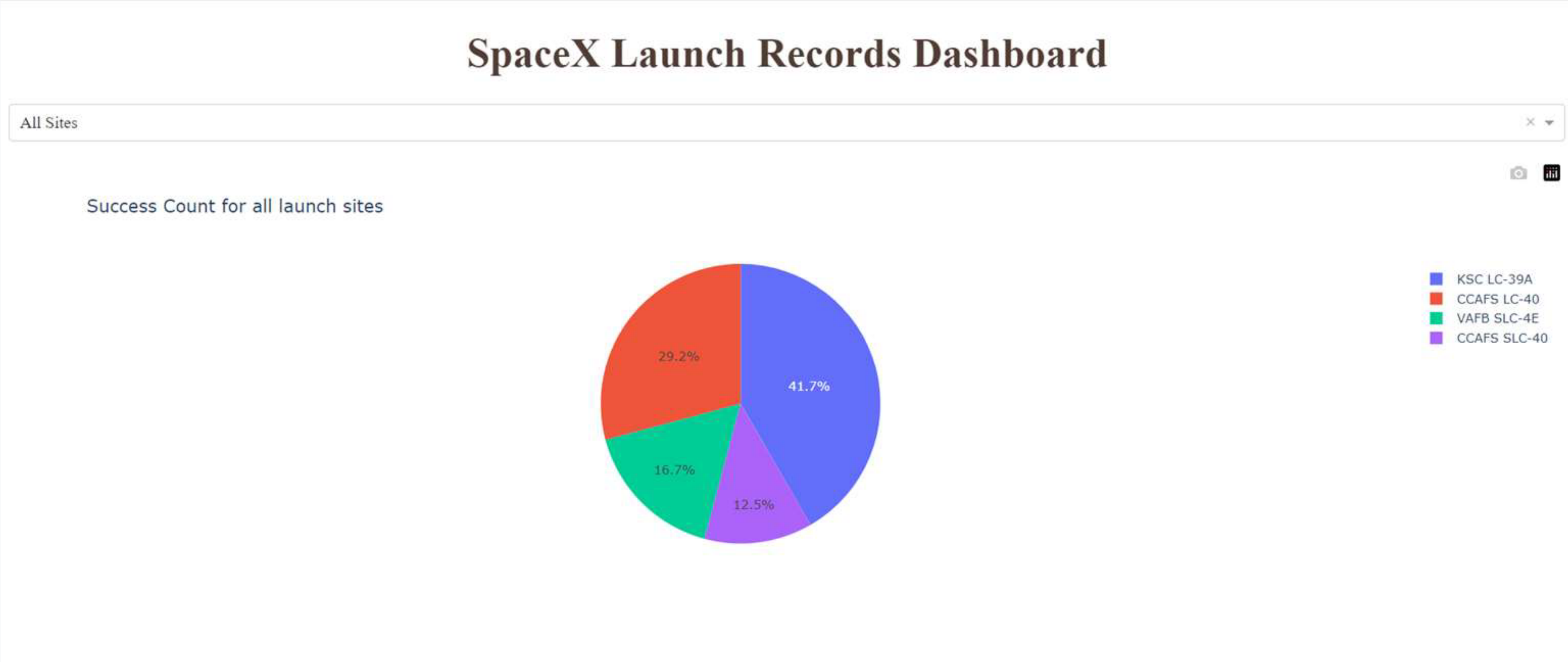
Success Pie Chart (Callback Function):
•Displays the **success rate** of launches based on the selected site.
•Updates dynamically when the user changes the **dropdown input**.

Payload Range Slider:
•Interactive **slider** to filter launches based on **payload mass**.
•Allows for exploration of success patterns based on payload size.

Scatter Plot for Success vs Payload:
•Plots **payload mass** against **launch outcomes**.
•Visualizes trends and success rates for different payload sizes.
•Dynamically updates using **callback functions** based on slider values.

GitHub URL of Completed Plotly Dash Application
👉[https://github.com/JellaGit/C10_Git/blob/main/Dashboard.ipynb]

SpaceX Launch Records Dashboard



Predictive Analysis (Classification)

Overview of Finding the Best Predictive Analysis Technique

1. Data Preparation:
•Loaded the dataset into a **Pandas DataFrame** for preprocessing and analysis.
•Performed **Exploratory Data Analysis (EDA)** to understand data distribution and relationships.

2. Defining Labels (Outcome Variable):
•Created a **NumPy array** from the **Class** column, representing whether the Falcon 9 booster landed successfully (**1**) or failed (**0**).
•Converted the column into a NumPy array using: Y = data['Class'].to_numpy()

3. Feature Standardization:
•Used **StandardScaler()** from **sklearn.preprocessing** to **normalize features (X)**, ensuring all features are on the same scale, improving model performance. from sklearn.preprocessing import StandardScaler scaler = StandardScaler() X = scaler.fit_transform(X)

4. Data Splitting for Training and Testing:
•Split the data into **training** and **testing** sets using **train_test_split()** from **sklearn.model_selection**.
•Allocated **80%** of the data for training and **20%** for testing, with a **random state of 2** for reproducibility. from sklearn.model_selection import train_test_split X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)

5. Model Selection:
•Evaluated different supervised machine learning models like **Logistic Regression**, **K-Nearest Neighbors (KNN)**, **Support Vector Machines (SVM)**, and **Decision Trees**.
•Compared models based on **accuracy scores**, **precision**, and **recall** to identify the **best predictive model**.

Outcome
•Achieved a **94% accuracy** using the **best classification model** based on the analysis.
•The results demonstrate high reliability in predicting **Falcon 9 booster landing success** based on key features like **payload mass**, **launch site**, and **orbit type**.
Let me know if you'd like more details about model comparisons or evaluation metrics! 🚀

Predictive Analysis (Classification)

Finding the Best Machine Learning Model
1. Model Initialization and Hyperparameter Tuning:
•Created objects for each ML algorithm (**SVM**, **Decision Trees**, **K-Nearest Neighbors (KNN)**, and **Logistic Regression**).
•Used **GridSearchCV** to perform **hyperparameter tuning** by testing a range of parameter values for each model.
•Example for **SVM**: from sklearn.svm import SVC from sklearn.model_selection import GridSearchCV svm = SVC() param_grid = {'C': [0.1, 1, 10], 'gamma': [0.01, 0.1, 1], 'kernel': ['rbf', 'linear']} grid_search = GridSearchCV(svm, param_grid, cv=10) grid_search.fit(X_train, Y_train)

2. Cross-Validation and Model Fitting:
•Used **10-fold cross-validation** (cv=10) to split the training data into multiple subsets for validation.
•Fit the training data into each **GridSearchCV** object to find the **best hyperparameters**.

3. Extracting Best Parameters and Validation Scores:
•After training, retrieved the **best parameters** and **best validation accuracy** for each model: print("Best Parameters:", grid_search.best_params_) print("Best Validation Score:", grid_search.best_score_)
•This process was repeated for all models (**SVM**, **KNN**, **Decision Trees**, and **Logistic Regression**).

4. Final Model Evaluation on Test Data:
•Evaluated the **test accuracy** for each model using the score() method: test_accuracy = grid_search.score(X_test, Y_test) print("Test Accuracy:", test_accuracy)
•Plotted **confusion matrices** for each model to visualize prediction performance: from sklearn.metrics import plot_confusion_matrix plot_confusion_matrix(grid_search.best_estimator_, X_test, Y_test, cmap='Blues') plt.show()

Key Results:
•**SVM** achieved the **highest accuracy (94%)** with the best hyperparameters (C=1.0, gamma=0.03, kernel='sigmoid').
•The **confusion matrices** highlighted areas where each model misclassified the data, helping to understand performance weaknesses.
•**Decision Trees** and **KNN** showed slightly lower accuracy, making **SVM** the preferred model for this dataset.

Results

- The table below shows accuracy score of test data for each of the methods comparing them to show which performed best.
- Techniques include SVM, Classification Trees, K Nearest Neighbors and Logistic Regression.

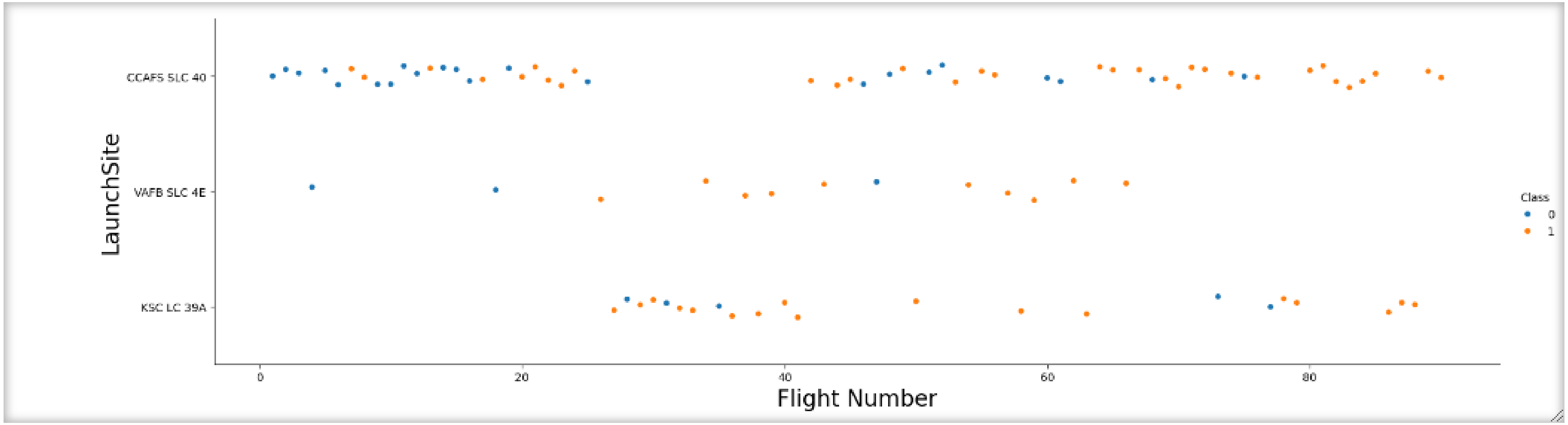
	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.888889
KNN	0.833333

- **GitHub Repository for Predictive Analysis Lab**
  👉https://github.com/JellaGit/C10_Git/blob/main/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

Section 2

Insights drawn from EDA

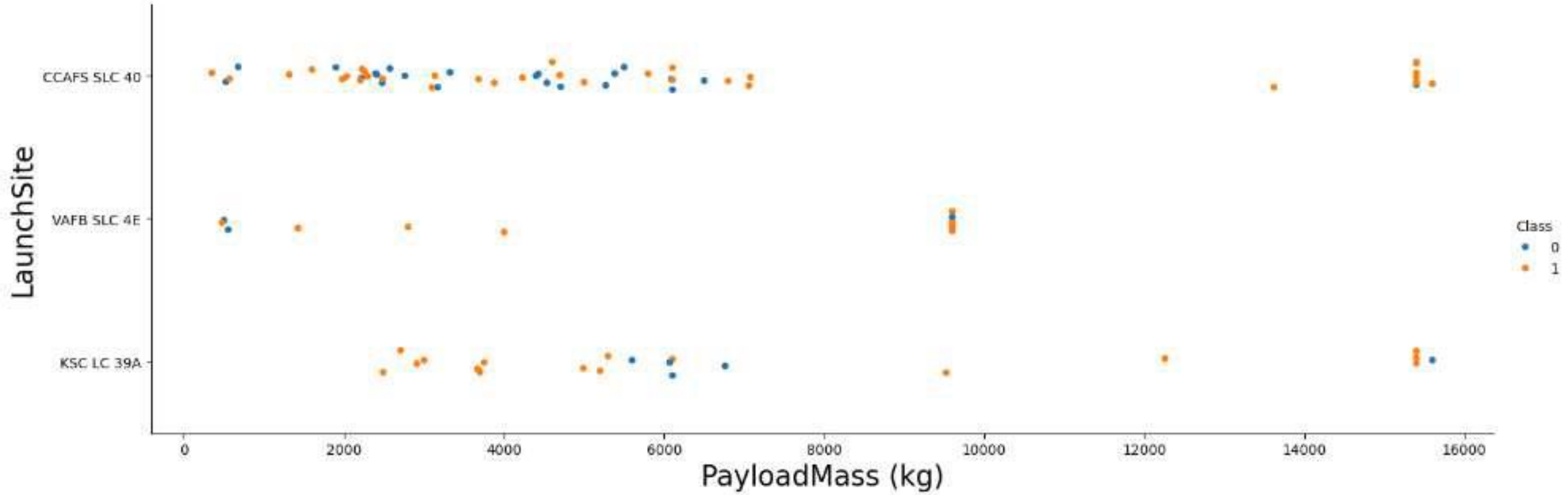
Flight Number vs. Launch Site



Now try to explain the patterns you found in Flight Number and Launch Site.

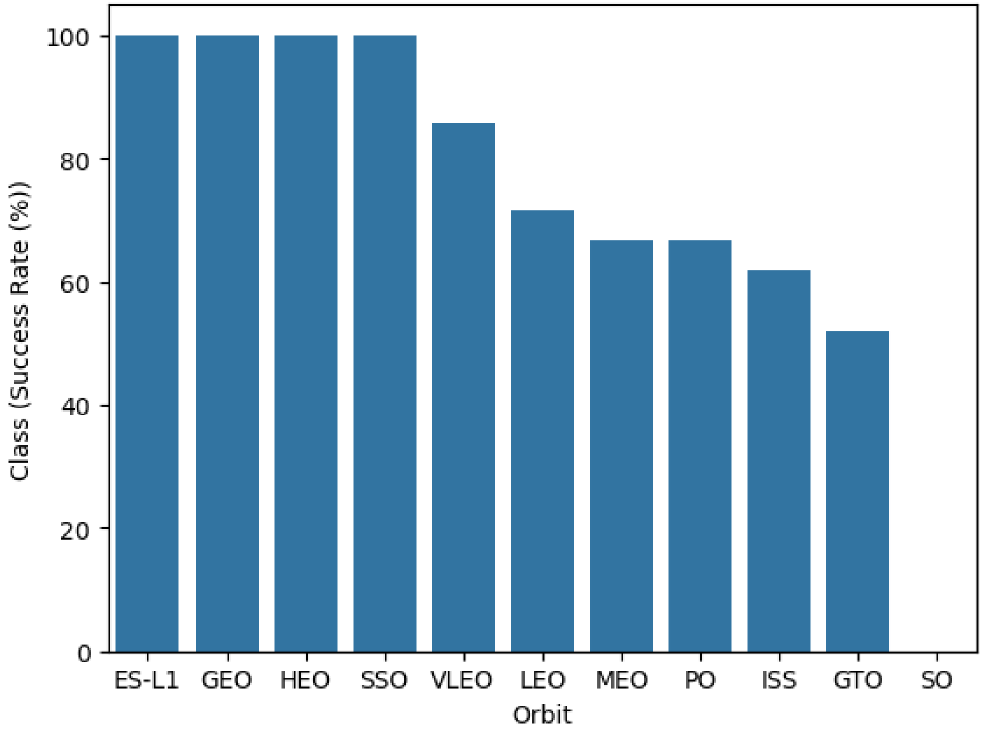
As the flight number increases it is clear to be seen that the success ration increases in all Launch sites. Focusing on VAFB SLC 4E almost all lauches after 20 Fight Number is successful except 1, while all the launches of other two are succesful after 80 Flights.

Payload vs. Launch Site



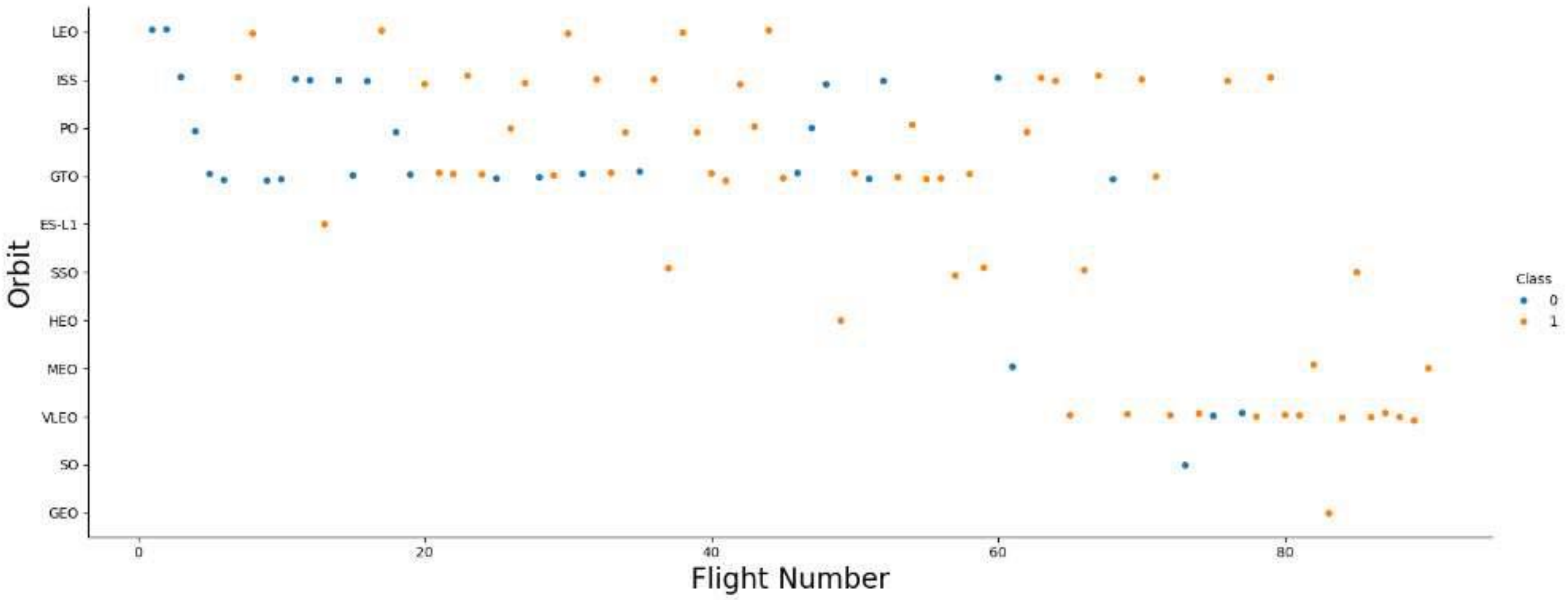
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type



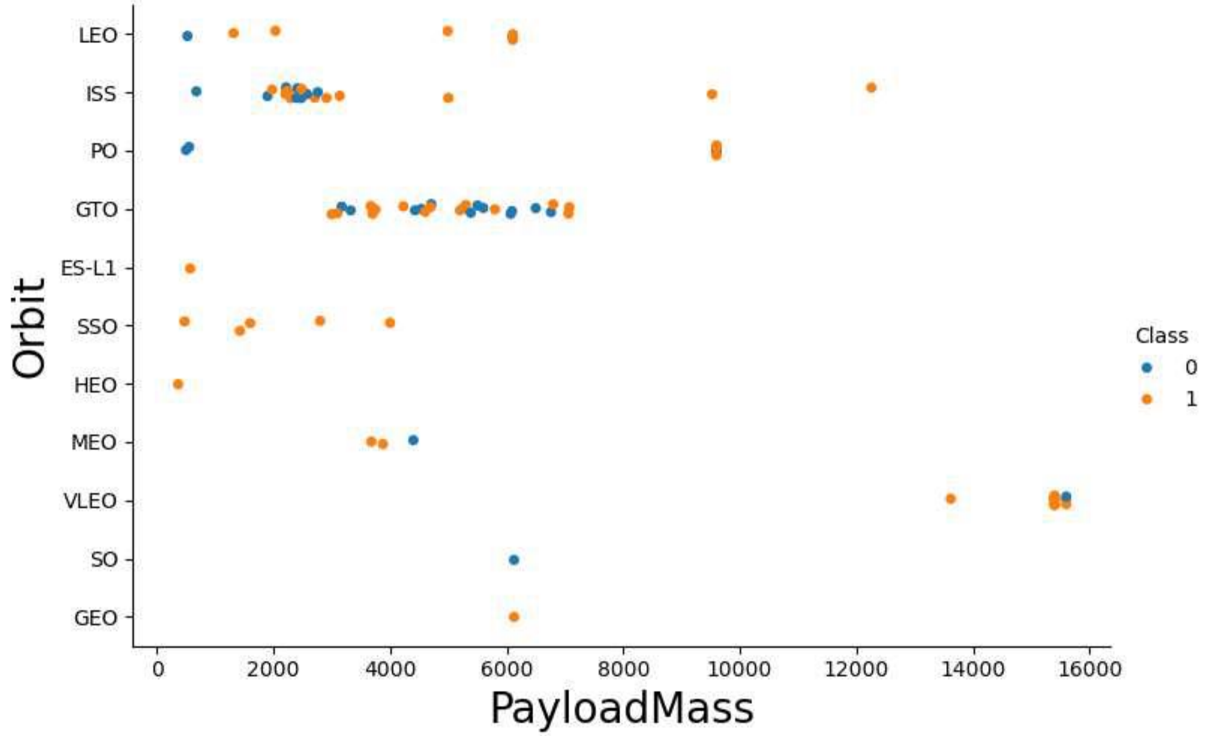
Analyze the ploted bar chart try to find which orbits have high sucess rate. Orbit ES-L1, GEO, HEO, SSO are the successful orbits having 100% success rate while orbit SO having success rate 0%. From orbits VLEO till GTO success rate is continuously decreasing and stay between 85% to 50% respectivley.

Flight Number vs. Orbit Type



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

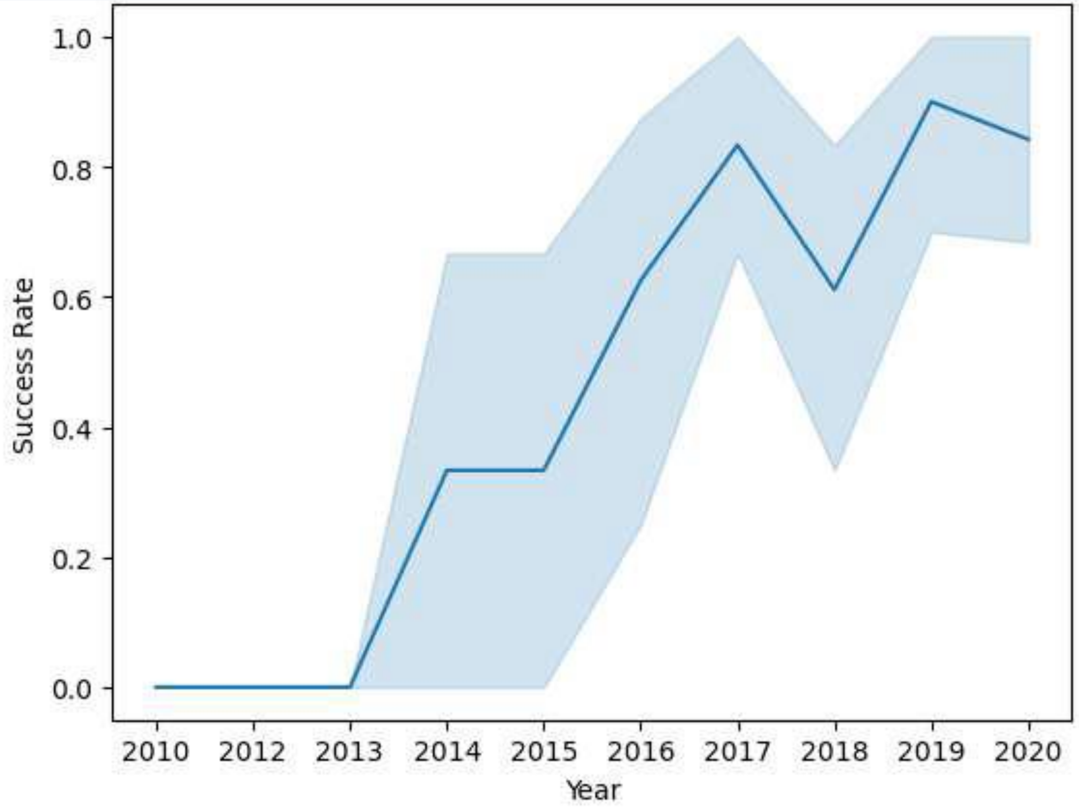
Payload vs. Orbit Type



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

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

Launch Success Yearly Trend



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

All Launch Site Names

- Find the names of the unique launch sites
- All launch sites are taking as a list called launch site. Using distinct we get all the unique different sites used in launching.

```
%sql select distinct LAUNCH_SITE as "LauchSite" from SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

- This is the list of launch site specifically starts with CCA with the of LIKE query and using LIMIT query to get only top 5 results.

```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Using SUM query can make the total of all payload mass from Payload_Mass_Kg _

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

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

payloadmass
619967

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- The average payloadmass can be find by using AVG on the column of payload_mass_kg

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

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

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

payloadmass
6138.287128712871

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- This is the .first date of a successful landing outcome by using MIN on the date column

```
%sql select min(DATE) from SPACEXTBL;
```

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

min(DATE)
2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

- This is the list from BOOSTER Version column and applying the two conditions on them:

```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING_OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000
```

* sqlite:///my_data1.db
Done.

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Used the COUNT together with the GROUP BY statement to return total number of missions outcomes.

```
%sql select "MISSION_OUTCOME", count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Using a Subquerry to return and pass the Max payload and used it list all the boosters that have carried the Max payload of 15600kgs.

```
%sql select BOOSTER_VERSION, payload as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

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

Booster_Version	boosterversion
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

- Used the subsrt in the select statement to get the month and year from the date column where substr (Date,7,4)='2015' for year and Landing_outcome was ‘Failure (drone ship) and return the records nmatching the filter

```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Ou
```

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

substr(Date,7,4)	substr(Date, 4, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing _Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone ship)

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

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

- Using DESC statement.

```
%sql SELECT * FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORDE
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-10-2020	12:25:57	F9 B5 B1051.6	KSC LC-39A	Starlink 13 v1.0, Starlink 14 v1.0	15600	LEO	SpaceX	Success	Success
18-08-2020	14:31:00	F9 B5 B1049.6	CCAFS SLC-40	Starlink 10 v1.0, SkySat-19, -20, -21, SAOCOM 1B	15440	LEO	SpaceX, Planet Labs, PlanetIQ	Success	Success
18-07-2016	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-04-2018	22:51:00	F9 B4 B1045.1	CCAFS SLC-40	Transiting Exoplanet Survey Satellite (TESS)	362	HEO	NASA (LSP)	Success	Success (drone ship)

Section 3

Launch Sites Proximities Analysis

Folium Map | 1

There are total 4 launch sites of SpaceX :

- VAFB SLC 4E: Vandenberg Space Launch Complex 4 (CA)
- KSC LC29A: Kennedy Space Center Merritt Island (FL)
- CCAFS LC40: Cape Canaveral Launch Complex 40 (FL)
- CCAF SLC40: Cape Canaveral Space Launch Complex 40(FL)

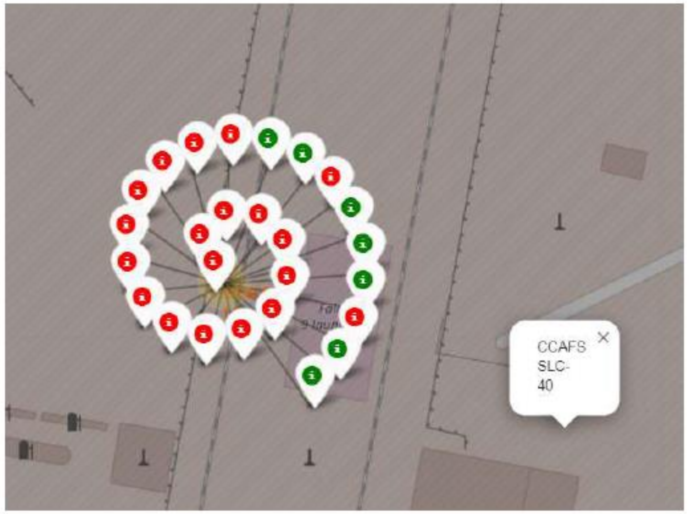
Folium Map | 2

- These are the successful and unsuccessful launch sites.
  Green shows successful launch while red are unsuccessful launches.
- The KSC LC 39A have mostly successful launches which is 10 successful while 3 unsuccessful.

Kennedy Space Center (FL)
KSC LC 39A

Folium Map | 3

- Launch site CCAFS SLC 40 have two launch site close to each other.
- The first site have 26 launches out of which only 7 were successful while the other left representation shows total 7 launches out of which only 3 were successful.
- In general it is clear that the launches from this area are mostly unsuccessful.

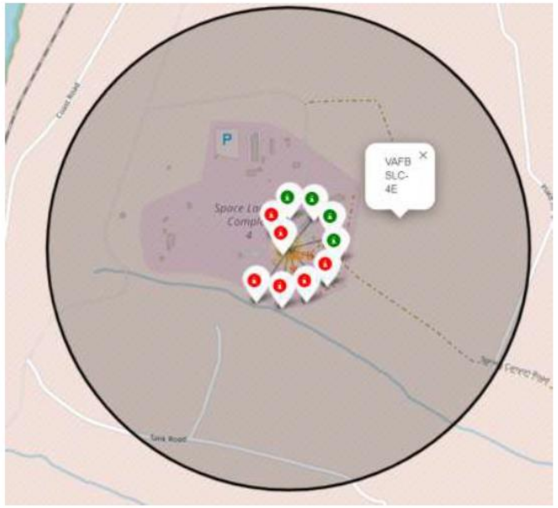


Cape Canaveral (FL)
CCAFS-LC40

Cape Canaveral (FL)
CCAFS-SLC40

Folium Map | 4

- This is the launch site close to L os Angeles which is totally opposite to other three sites.
- The ratio of successful launches was not good enough having 6 unsuccessful launches with 4 successful in total of 10 launches.



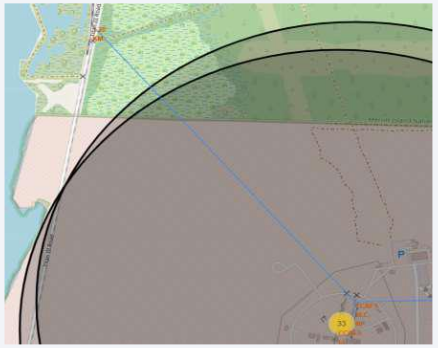
Vandenberg Space Launch Complex 4 (CA)
VAFB SLC-4E

Folium Map | 5

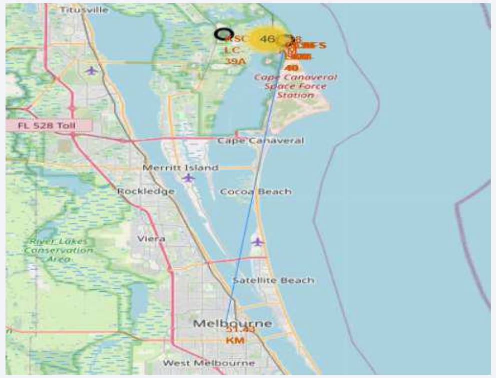
- Launch site we use to check the distance with highways, train ways and city is Cape Canaveral (FL) CCAFS SLC 40
- This map shows the line that defines the closest distance of launch site with the highway.
- Distance with the highway is 0 58 km.

Folium Map | 6

- The distance with railway line is almost 3.28 km.



- Melbourne is the closest big city from the launch site which is around 51.43 km.



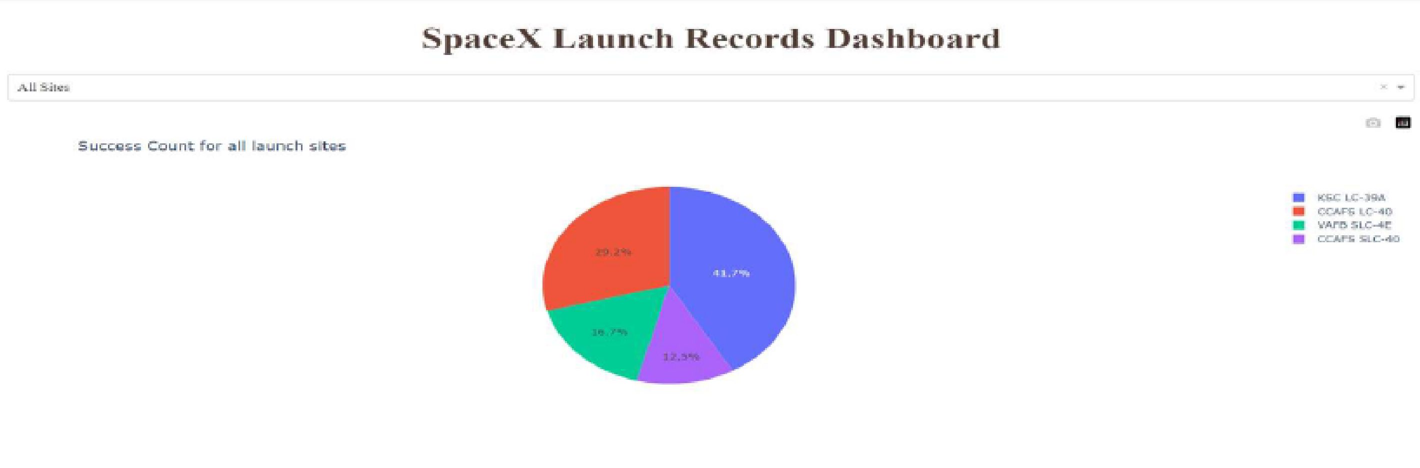
Section 4

Build a Dashboard with Plotly Dash

Plotly Dashboard | 1

We built an interactive dashboard with Plotly including

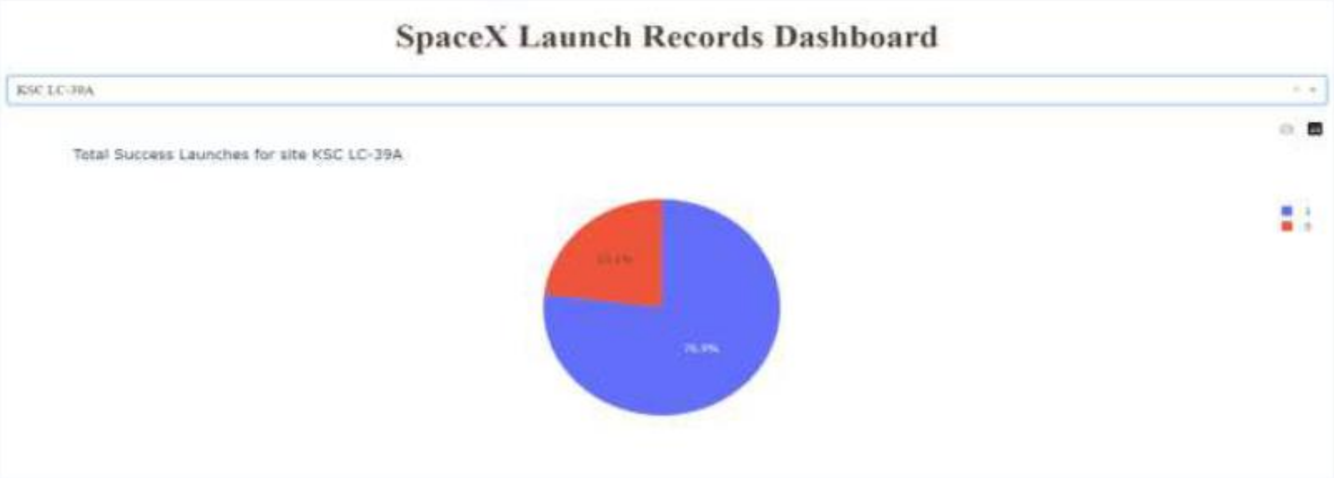
- Dropdown menu for selecting launch sites
- Pie charts displaying success rate
- Scatter chart displaying launch site, payload mass, success/failure
- Range slider for selecting range of payload mass in kg

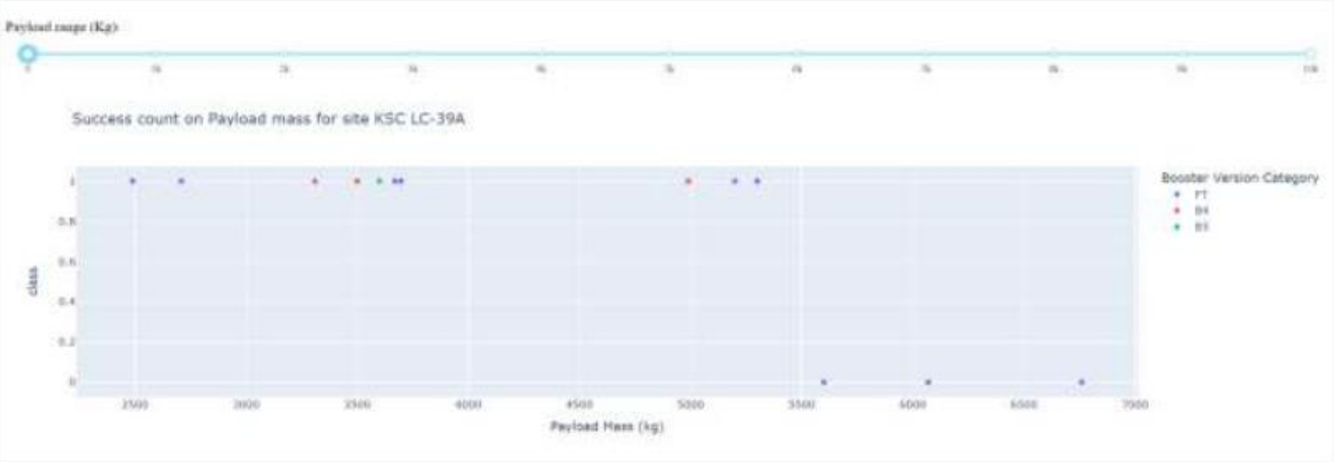


Plotly Dashboard | 2

Getting the following information by analyzing

- Site with largest successful launches
- Site with highest launch success rate
- Payload range(s) with highest launch success rate
- Payload range(s) with lowest launch success rate
- F 9 Booster version (v 1 0 v 1 1 FT, B 4 B 5 etc) with highest launch success rate





Section 5

Predictive Analysis (Classification)

Classification Accuracy

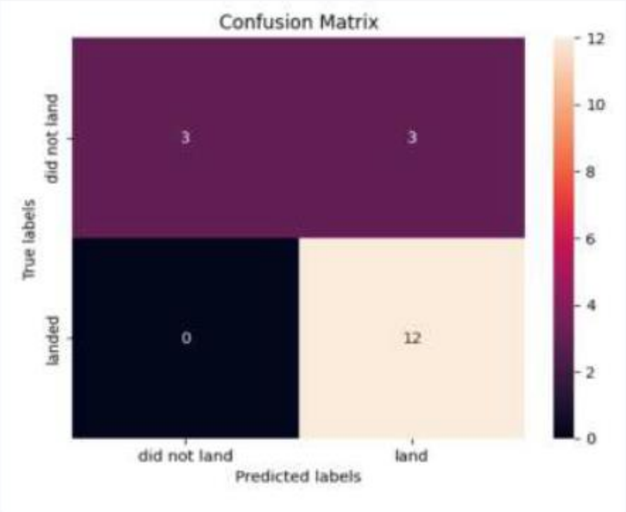
The final results shows in two column one is of predicted method column while other is Test Data Accuracy

- Total 4 predictive methods use which is Logistic Regression, SVM, Decision Tree and KNN
- Decision Tree shows more accuracy which is 0.88 as compare other three which 0.83

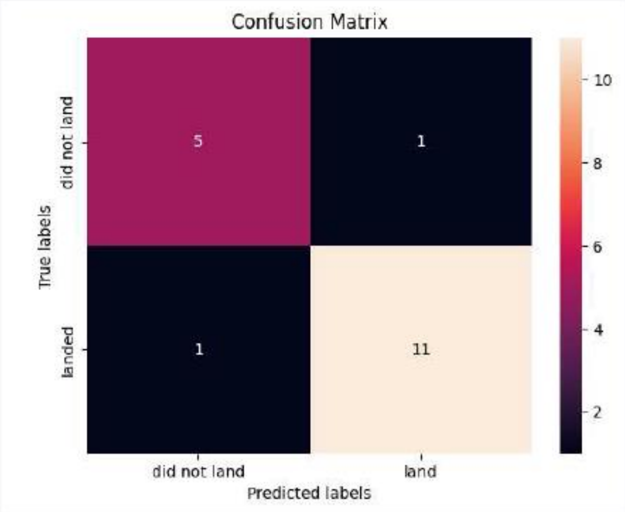
	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.888889
KNN	0.833333

Confusion Matrix

- Decision Tree is the model predicted more accuracy than other three.
- Other predicted about unsuccessful landings were 100% correct while decision tree prediction about successful landing is more accurate than others.



Confusion matrix of KNN, Logistics Regression and SVM



Confusion matrix of decision tree

Conclusions

- • Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
- We can deduce that, as the flight number increases in each of the 3 launch sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80^{th} flight
- If you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).
- Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.
- LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS. However, for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here
- A finally the success rate since 2013 kept increasing till 2020.

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!