DATA SCIENCE CAPSTONE PROJECT ON SPACE X LAUNCHERS

Edwin AKOUSSAH

11/01/2023

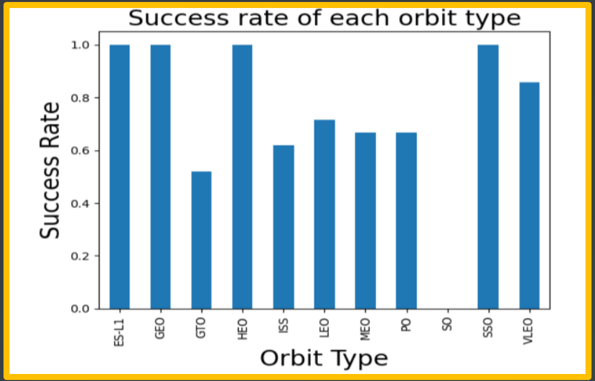
https://github.com/EdAkh/Applied-Data-Science-Capstone

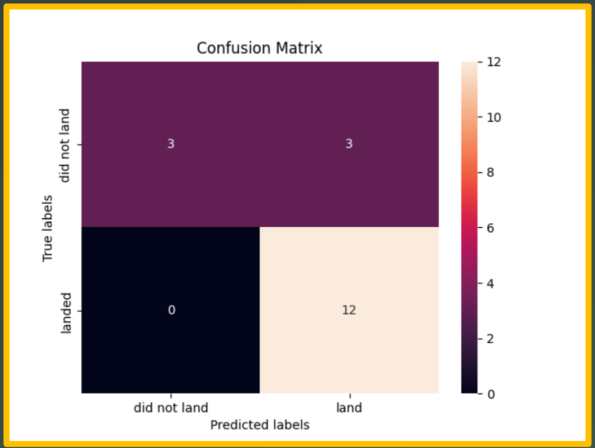
OUTLINE

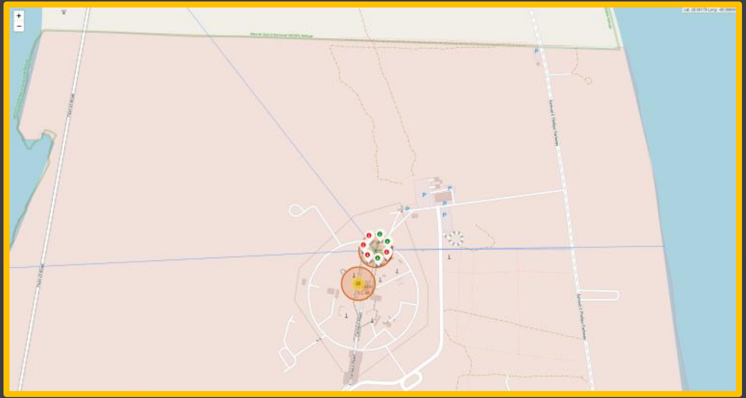
- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

EXECUTIVE SUMMARY

- Summary of methodologies
  - Data collection with SpaceX API and Web scrapping
  - Data Wrangling
  - Exploratory Data Analysis
  - Data visualization with Folium
  - Interactive analysis with a dashboard
  - Prediction with machine learning
- Summary of results
  - Exploratory Data Analysis results
  - Prediction results







INTRODUCTION

- Context

SpaceX is an American company specialized in spacecraft. In this project we will concentrate on one of the launchers of SpaceX, Falcon 9. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch.

METHODOLOGY

METHODOLOGY

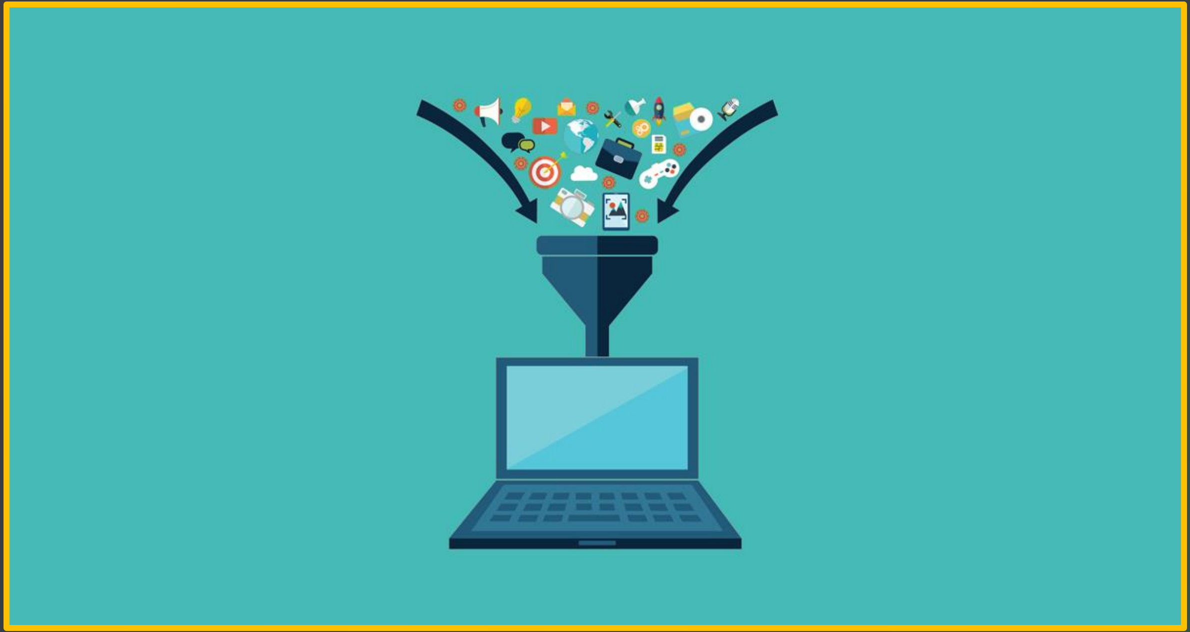
- Data collection methodology
- Data wrangling methodology
- Exploratory data analysis methodology
- Visual analytics methodology
- Predictive analysis methodology

METHODOLOGY

Data Collection

Data collection refers to the process of gathering and measuring information on targeted variables in an established systematic fashion, which then enables one to answer relevant questions and evaluate outcomes.

For data collection we used SpaceX API and Wikipedia for web scraping.



METHODOLOGY

Data Collection – SpaceX API

- Using GET request to the SpaceX API to collect launch data

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

- Convert Response into a .json file then a dataframe

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

```
In [12]: # Use json_normalize meethod to convert the json result into a dataframe
static_json_df = response2.json()
data = pd.json_normalize(static_json_df)
```

- We need to proceed the data cleaning

```
In [38]: # Calculate the mean value of PayloadMass column
PayloadMass = data_falcon9.PayloadMass.mean()
print(PayloadMass)

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PayloadMass)

6123.547647058824
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004

METHODOLOGY

Data Collection – Web Scraping

- Using HTTP GET to retrieve the Falcon 9 Wikipedia page html

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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
        html = requests.get(static_url)
        # assign the response to a object
        html.status_code

Out[5]: 200
```

- Then we create a BeautifulSoup object

```
In [7]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(html.text, 'html.parser')
```

- Finding attribute 'table' in the object and extract columns

```
In [10]: # 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 [11]: # Let's print the third table and check its content
         first_launch_table = html_tables[2]
         print(first_launch_table)
```

```
In [13]: column_names = []

         # Apply find_all() function with `th` element on first_launch_table
         # Iterate each th element and apply the provided extract_column_from_header() to get a column name
         # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

         elements = soup.find_all('th')
         for row in range(len(elements)):
             try:
                 name = extract_column_from_header(elements[row])
                 if (name is not None and len(name) > 0):
                     column_names.append(name)
             except:
                 pass
```

- Create a dictionary and convert it into a dataframe

```
In [15]: launch_dict= dict.fromkeys(column_names)

         # Remove an irrelvant 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['Payload mass'] = []
         launch_dict['Orbit'] = []
         launch_dict['Customer'] = []
         launch_dict['Launch outcome'] = []
         # Added some new columns
         launch_dict['Version Booster']=[]
         launch_dict['Booster landing']=[]
         launch_dict['Date']=[]
         launch_dict['Time']=[]
```

```
In [26]: df = pd.DataFrame(launch_dict)
         df.head()
```

Out[26]:

Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time

We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

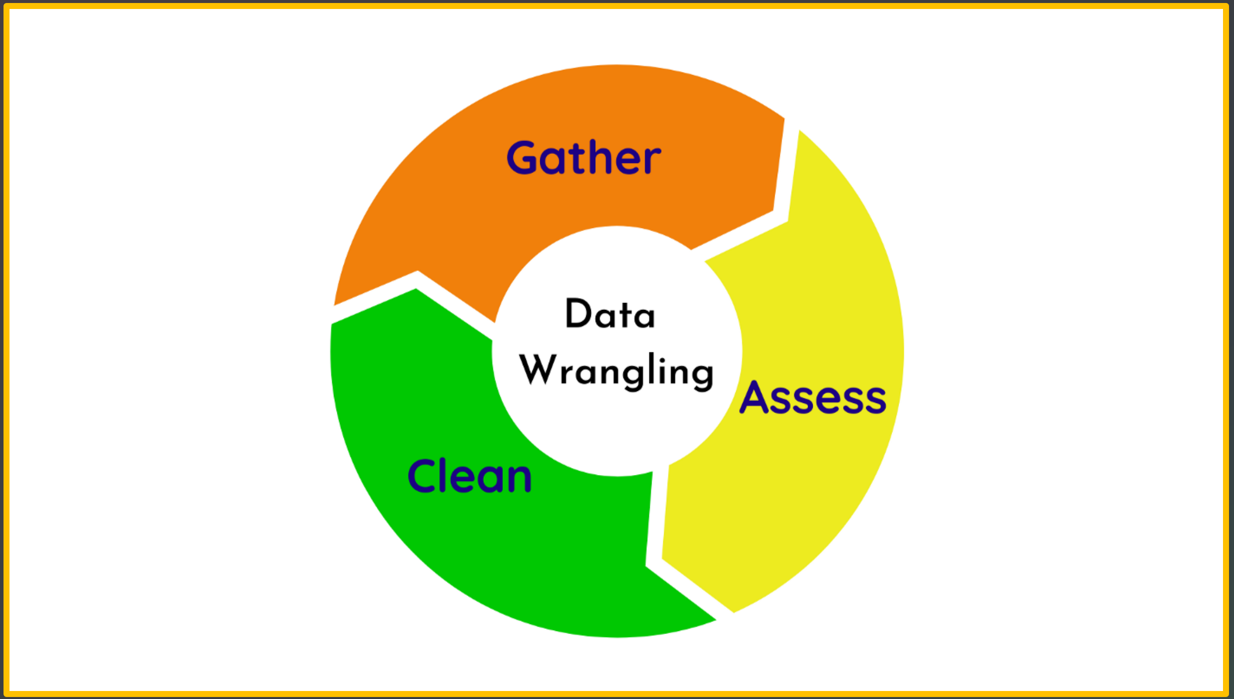
Following labs will be using a provided dataset to make each lab independent.

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

METHODOLOGY

Data Wrangling

Data wrangling, also known as data munging, is the process of cleaning, transforming, and manipulating data in order to make it more suitable for analysis.



METHODOLOGY

Data Wrangling

- Firstly we calculate the number of launches, the number and occurrence of each orbits and the number and occurrence of mission outcome per orbit type.

```
In [7]:  # Apply value_counts() on column LaunchSite
         df.value_counts('LaunchSite')

Out[7]:  LaunchSite
         CCAFS SLC 40    55
         KSC LC 39A      22
         VAFB SLC 4E     13
         dtype: int64
```

```
In [8]:  # Apply value_counts on Orbit column
         df.value_counts('Orbit')

Out[8]:  Orbit
         GTO      27
         ISS      21
         VLEO     14
         PO        9
         LEO       7
         SSO       5
         MEO       3
         ES-L1     1
         GEO       1
         HEO       1
         SO        1
         dtype: int64
```

```
In [11]:  # landing_outcomes = values on Outcome column
          landing_outcomes = df.value_counts('Outcome')
          landing_outcomes

Out[11]:  Outcome
          True ASDS      41
          None None      19
          True RTLS      14
          False ASDS      6
          True Ocean      5
          False Ocean     2
          None ASDS       2
          False RTLS      1
          dtype: int64
```

- Creating 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`:

```
In [14]:  # 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 [15]:  df['Class']=landing_class
          df[['Class']].head(8)
```

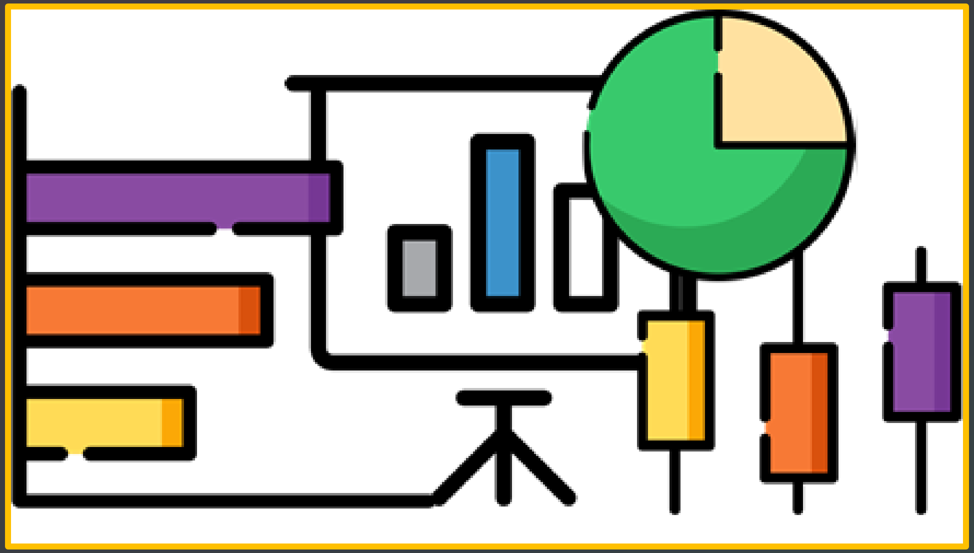
Out[15]:

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

METHODOLOGY

Exploratory Data Analysis (EDA)

The goal of EDA is to identify patterns, trends, and relationships within the data that can provide insight and inform further analysis or modeling.



METHODOLOGY

Exploratory Data Analysis (EDA)

For exploratory data analysis we used data visualization and SQL.

Data Visulization to analyse relation between:

- Payload and Flight Number
- Flight Number and Launch Site
- Payload and Launch Site
- Flight Number and Orbit Type
- Payload and Orbit Type
- Success Rate vs Orbit Type
- Space X Success Rate

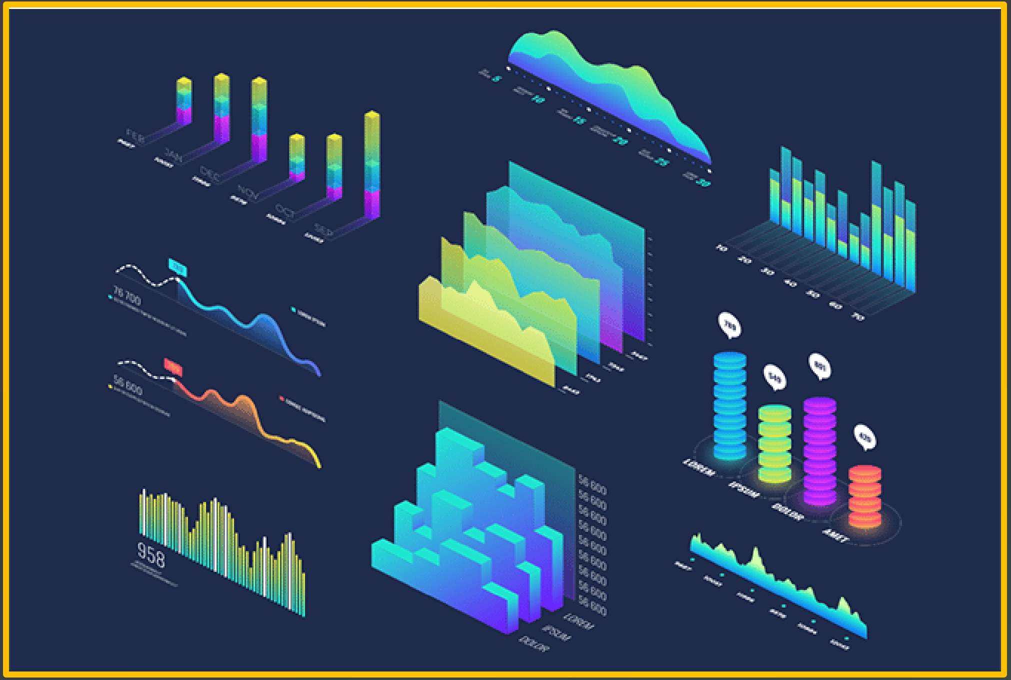
SQL to display information about:

- All Launch Site Names
- Payload Mass
- Booster Versions
- Mission Outcomes
- Booster Landings

METHODOLOGY

Data Visualization

Data visualization is the process of creating graphical representations of data in order to better understand and analyse it. The goal of data visualization is to take complex data and present it in a way that is easy to understand and interpret, making it easier to identify patterns, trends, and insights.

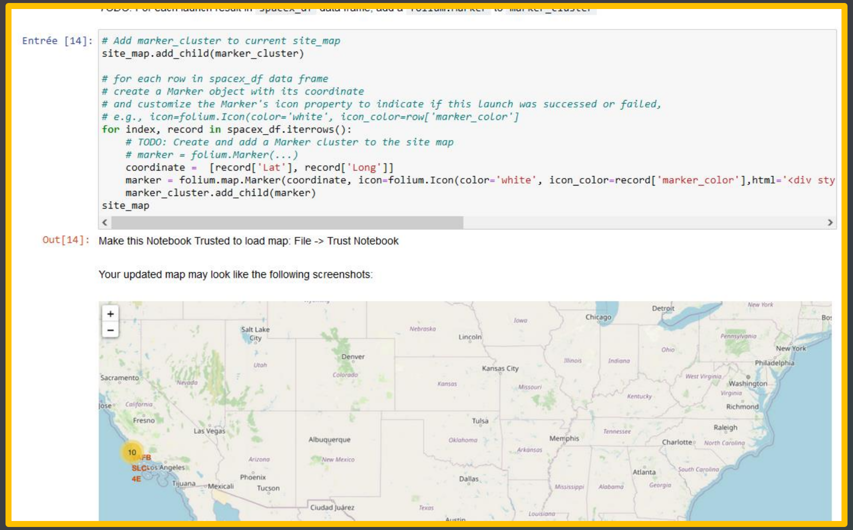


METHODOLOGY

Data Visualization

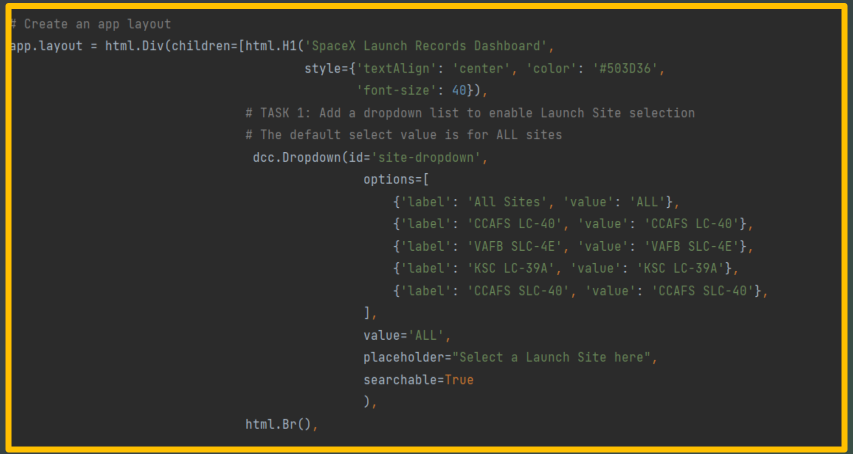
<u>Building an interactive map with Folium containing:</u>

- All the launches sites
- Successful and failed launches
- Distances between launch sites and highway, railway, coastline and cities



<u>Building an interactive dashboard showing:</u>

- A pie chart of the total success for all sites or by launch site
- A scatter chart of link between payload and success for all sites or by launch site



METHODOLOGY

Predictive Analysis

Predictive models are built using a variety of techniques, such as regression analysis, decision trees, and neural networks, and are used in a wide range of industries, including finance, healthcare, and retail.

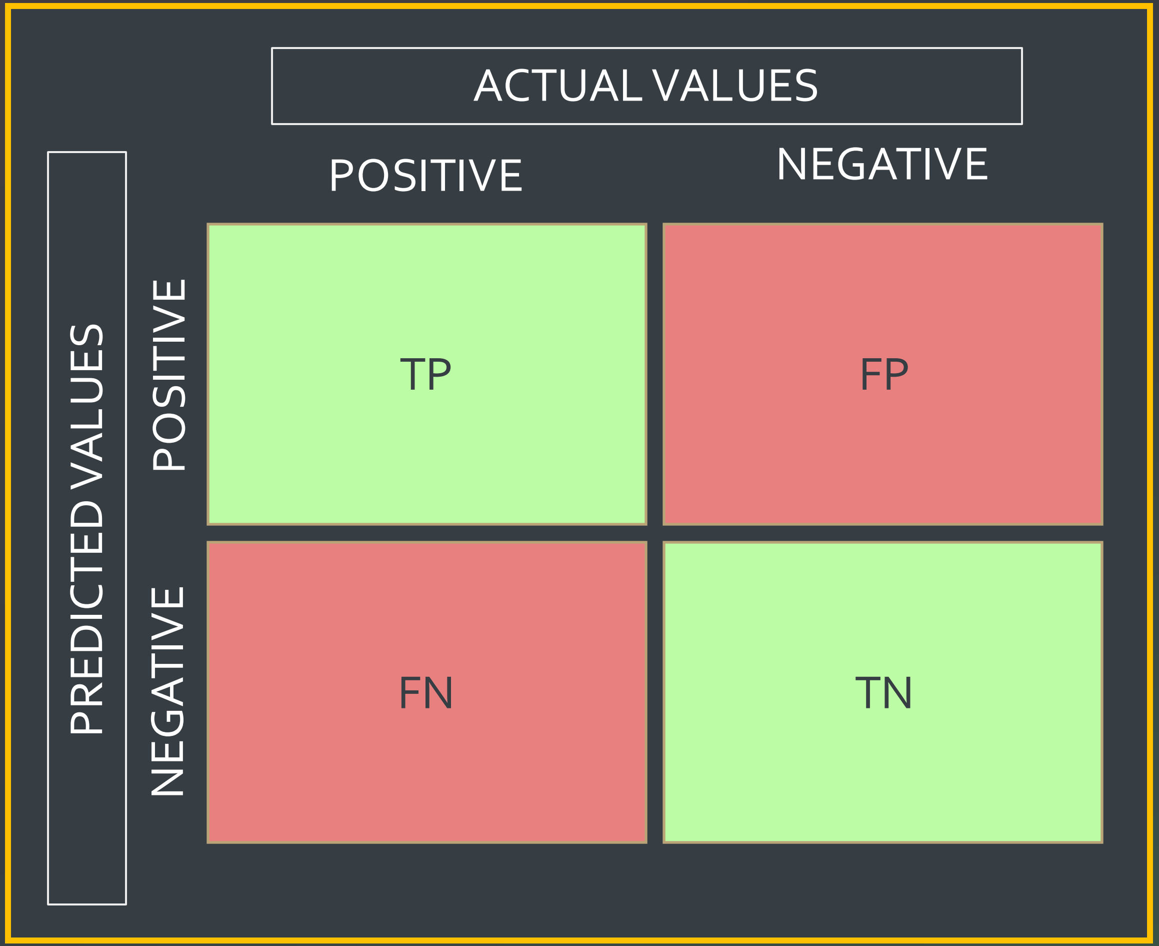
METHODOLOGY

Predictive Analysis

Loading our dataframe
Create columns for 'Class' using Numpy
Standardized the data
Split the into a training and test data
Using four machine learning classification models for training and calculate accuracy:

- Logistic regression
- Support Vector Machine (SVM)
- Decision tree
- K-Nearest neighbors

Evaluate which model has the best accuracy by using GridsearchCV



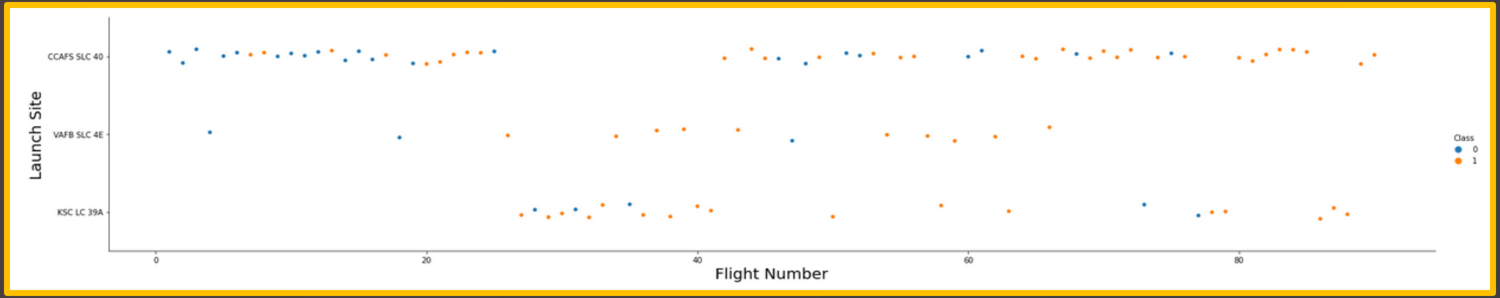
RESULTS

RESULTS

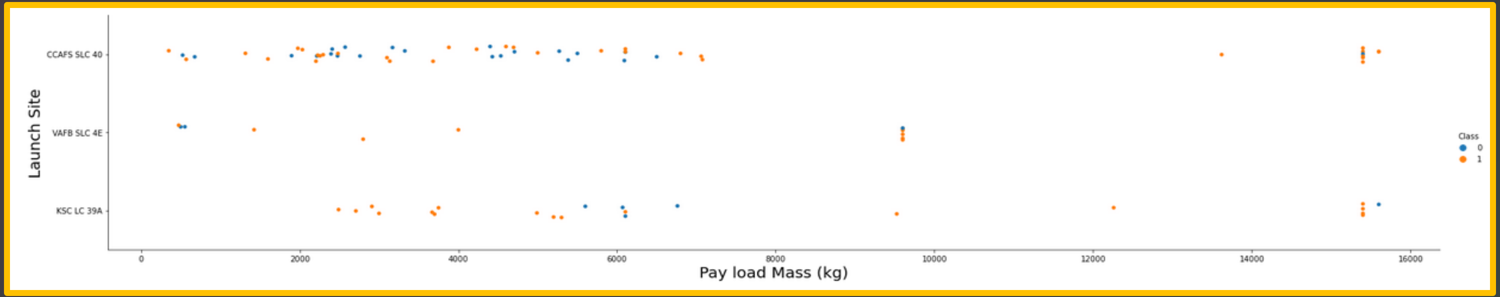
- Exploratory Data Analysis results
- Interactive Analytics results
- Predictive Analysis results

RESULTS

Exploratory data analysis with visualization results



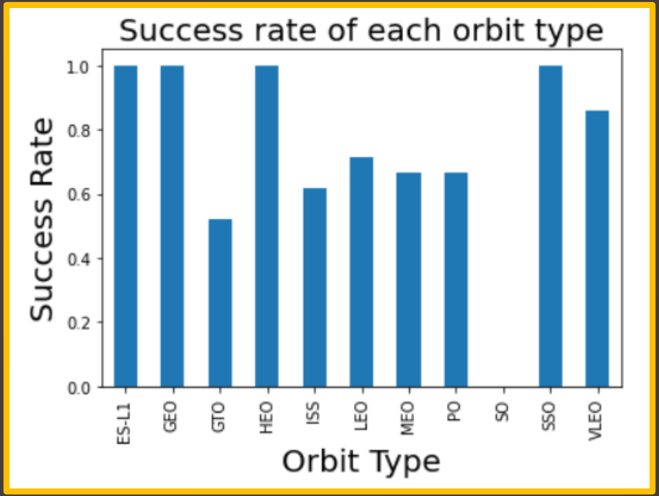
- The more there is flights, the more the success rate seems to increase a lot for a launch site.



- We can see that there is no payload mass over 10000 kg in launch site VAFB SLC 4E.

RESULTS

Exploratory data analysis with visualization results

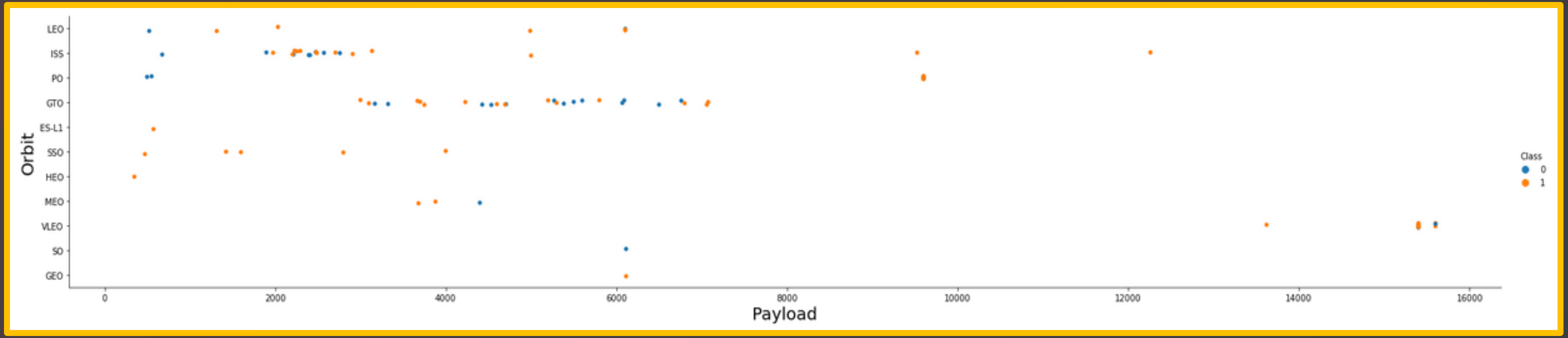


- The orbits with the highest success rates are ES-L1, GEO, HEO, SSO and VLEO.

- We can see that in the LEO orbit the success rate appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

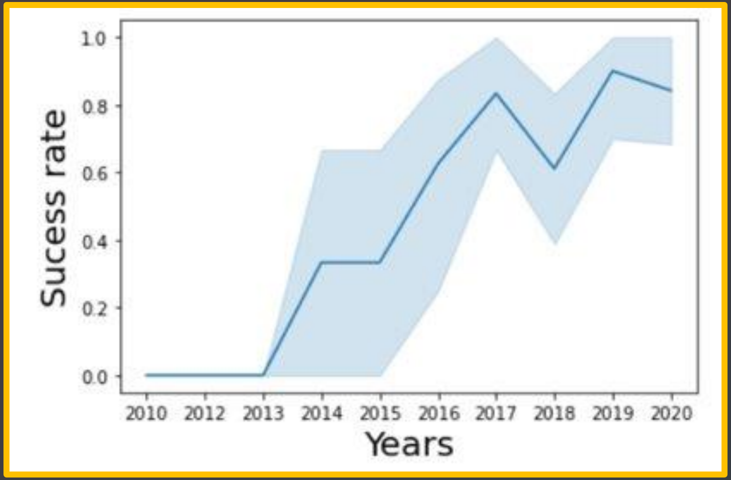
RESULTS

Exploratory data analysis with visualization results



- 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.



- Since 2013 the launches success rate never stop to increase.

RESULTS

Exploratory data analysis with SQL results

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

Entrée [7]:
```
%sql select distinct(LAUNCH_SITE) from SPACEXTBL
```

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

Out[7]:

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

- We used 'distinct' to show only unique value of launch sites.

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

Entrée [55]:
```
%sql select * from SPACEXTBL where (LAUNCH_SITE) like 'CCA%' limit 5
```

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

Out[55]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- 'limit 5' allow to display the five records where launch sites begin with 'CCA'.

RESULTS

Exploratory data analysis with SQL results

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

```
Entrée [9]: %sql select sum(PAYLOAD_MASS__KG_) as TOTAL_PAYLOAD_MASS from SPACEXTBL where (CUSTOMER) = 'NASA (CRS)'

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

Out[9]:

TOTAL_PAYLOAD_MASS
45596

- With the function 'sum()', we calculated the total payload mass carried by boosters launched by NASA and found 45 596 kg as a result.

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

```
Entrée [10]: %sql select avg(PAYLOAD_MASS__KG_) as AVG_PAYLOAD_MASS from SPACEXTBL where (BOOSTER_VERSION) = 'F9 v1.1'

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

Out[10]:

AVG_PAYLOAD_MASS
2928.4

- By using thefunction 'avg()' we found that the average payload mass carried bybooster version F9 v1.1 was 2928.4 kg.

RESULTS

Exploratory data analysis with SQL results

```
[36]: %sql select * from SPACEXTBL where "Landing _Outcome" = "Success (ground pad)";
 * sqlite:///my_data1.db
Done.
```

[36]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
22-12-2015	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

- We found that the first successful landing staged was the 22/12/2015.

```
Entrée [25]: %sql select min(DATE) as FIRST_SUCCESSFUL_LANDING from SPACEXTBL where "LANDING _OUTCOME" = 'Success (ground pad)'
 * sqlite:///my_data1.db
Done.
```

Out[25]:

FIRST_SUCCESSFUL_LANDING
01-05-2017

```
Entrée [26]: %sql select BOOSTER_VERSION from SPACEXTBL where "LANDING _OUTCOME" = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 a
 * sqlite:///my_data1.db
Done.
```

Out[26]:

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

- With this query we can see which boosters have a success in drone ship fora payload between 4000 kg and 6000 kg.

RESULTS

Exploratory data analysis with SQL results

List the total number of successful and failure mission outcomes

Entrée [58]:
```
%sql select count(*) as TOTAL_SUCCESSFUL_MISSION_OUTCOME from SPACEXTBL where (MISSION_OUTCOME) like 'Success%'
```
```
 * sqlite:///my_data1.db
Done.
```

Out[58]:

TOTAL_SUCCESSFUL_MISSION_OUTCOME
100

Entrée [59]:
```
%sql select count(*) as TOTAL_FAILURE_MISSION_OUTCOME from SPACEXTBL where (MISSION_OUTCOME) like 'Failure (in flight)'
```
```
 * sqlite:///my_data1.db
Done.
```

Out[59]:

TOTAL_FAILURE_MISSION_OUTCOME
1

- The queries return the total of successful missions and failed ones.

[19]:
```
%sql select BOOSTER_VERSION, PAYLOAD_MASS__KG_ from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL) order by BOOSTER
```
```
 * sqlite:///my_data1.db
Done.
```

[19]:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

- To list the booster versions who carried the maximum payload mass we used the clause 'where' and the function 'max()'.

RESULTS

Exploratory data analysis with SQL results

[17]:
```
%sql select substr(DATE, 4, 2) as "Month", "LANDING _OUTCOME", BOOSTER_VERSION, LAUNCH_SITE from SPACEXTBL where "LANDING _OUTCOME" like 'Failure (dron
```

* sqlite:///my_data1.db
Done.

[17]:

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

- With this query, we can see the failed landing with drone ship in 2015 and by month, booster versions and launch sites.

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

[44]:
```
%sql select ("LANDING _OUTCOME"), count("LANDING _OUTCOME") as "Total Count" from SPACEXTBL where (DATE) between '04-06-2010' and '20-03-2017' group by
```

* sqlite:///my_data1.db
Done.

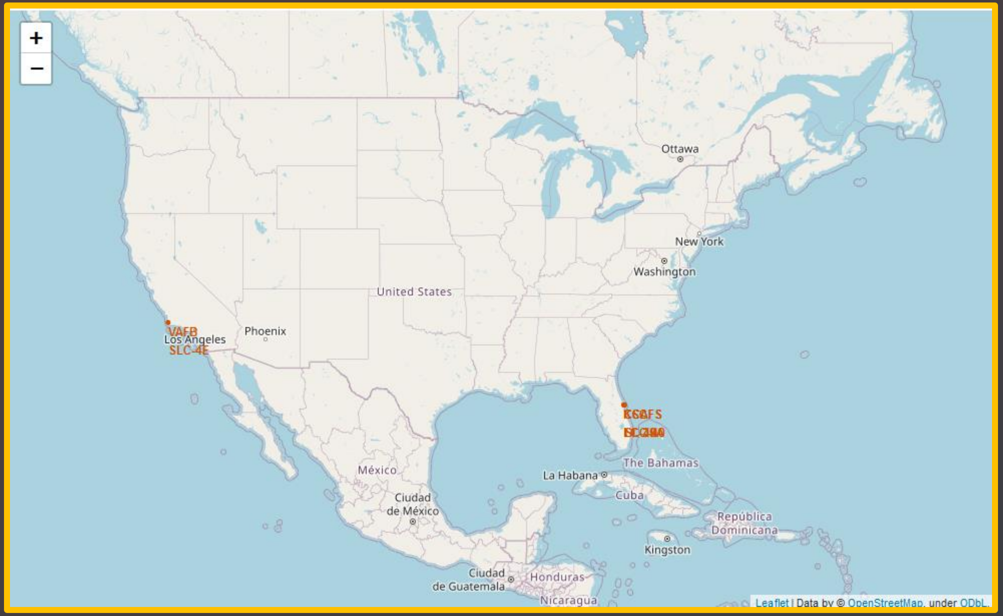
[44]:

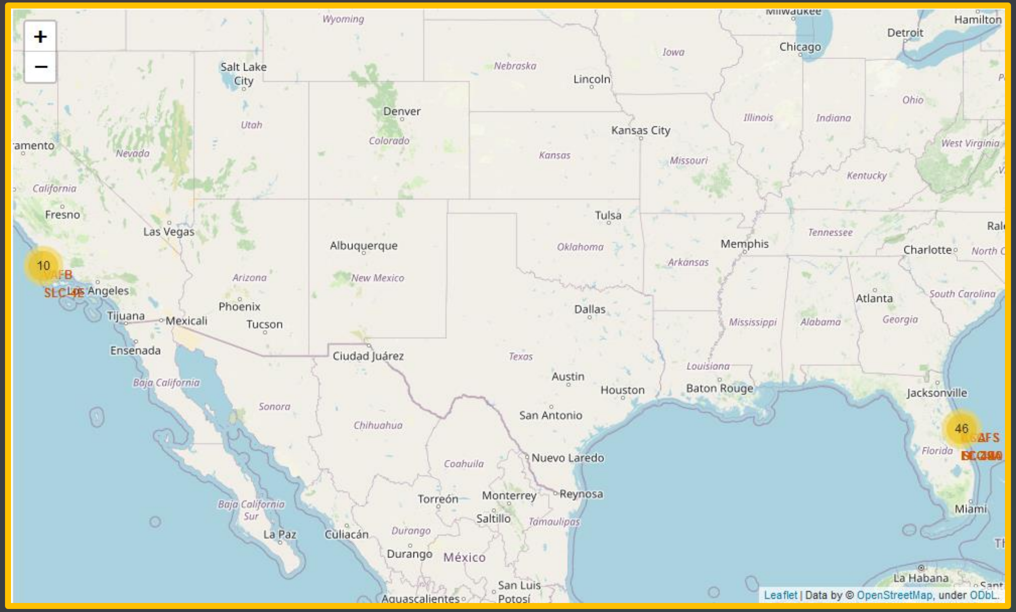
Landing _Outcome	Total Count
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

- A rank of all landing outcomes by descending order between 04/06/2010 and 20/03/2017.

RESULTS

Interactive Analytics with Folium





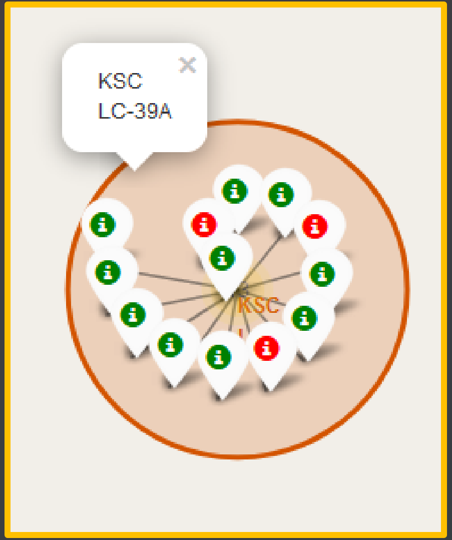
- We generated a map with the launch sites
- Then created clusters containing the success/failed launches on each site.

RESULTS

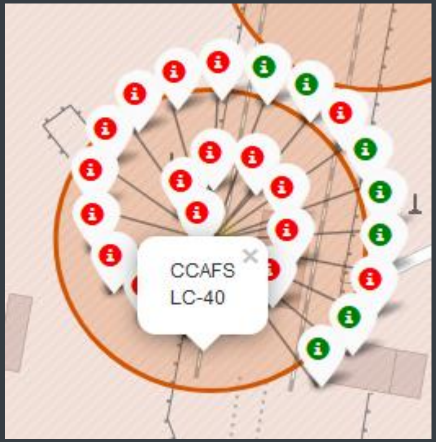
Interactive Analytics with Folium

- The green markers show successful launches and red markers show failures.
- We can see that KSC LC-39A had the most successful launches.

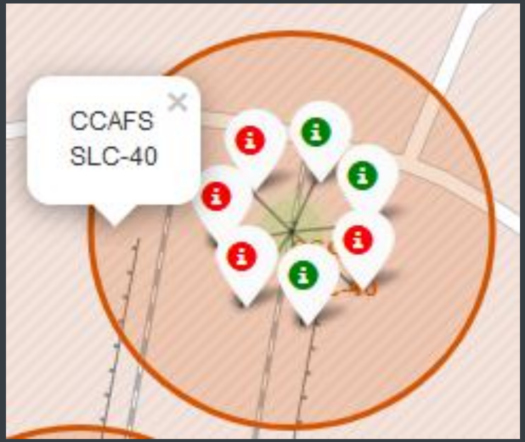
VAFB SLC-4E



KSC LC-39A



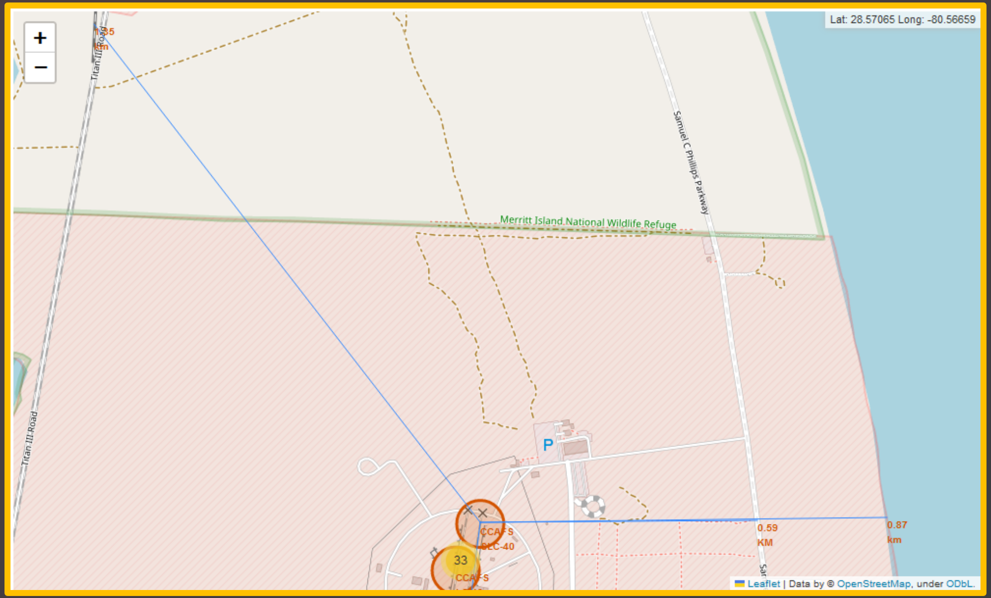
CCAFS LC-40

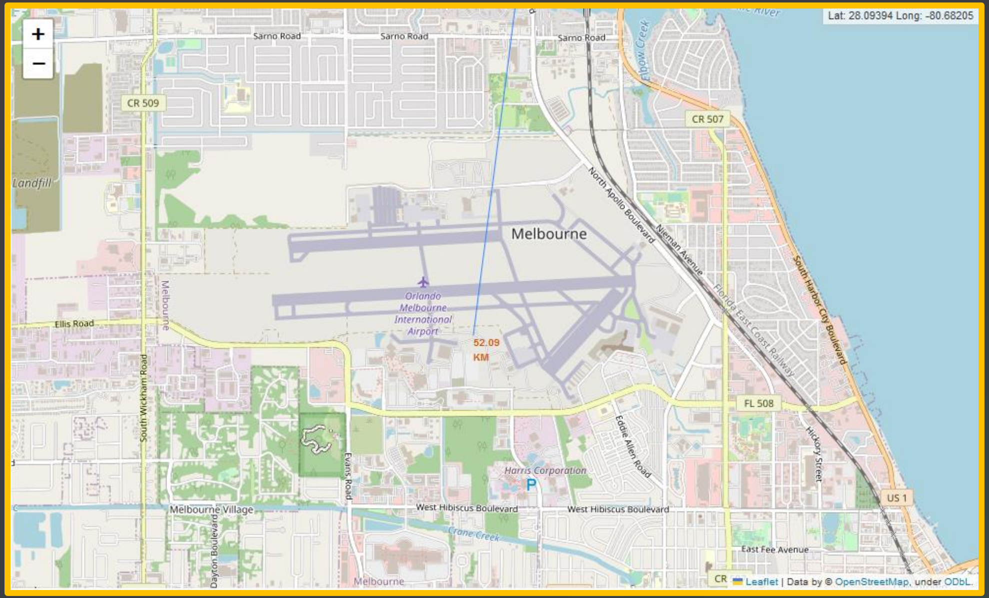


CCAFS SLC-40

RESULTS

Interactive Analytics with Folium

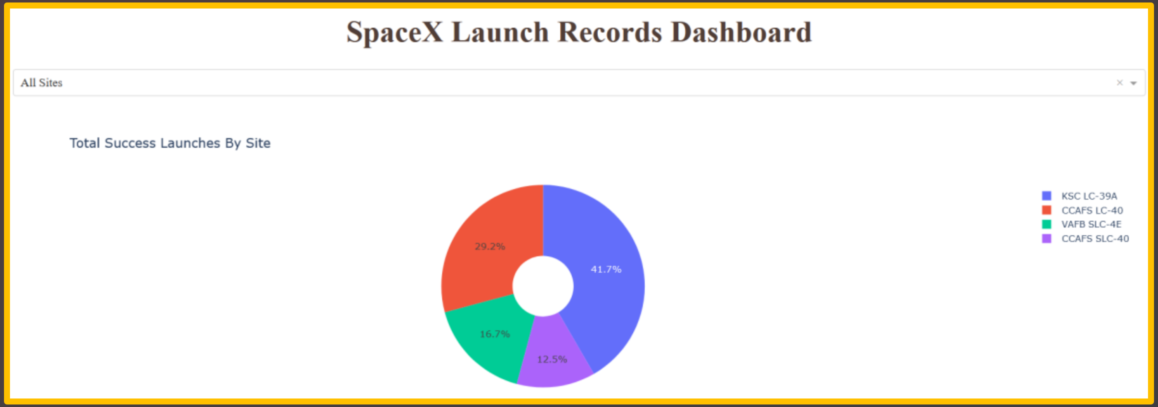


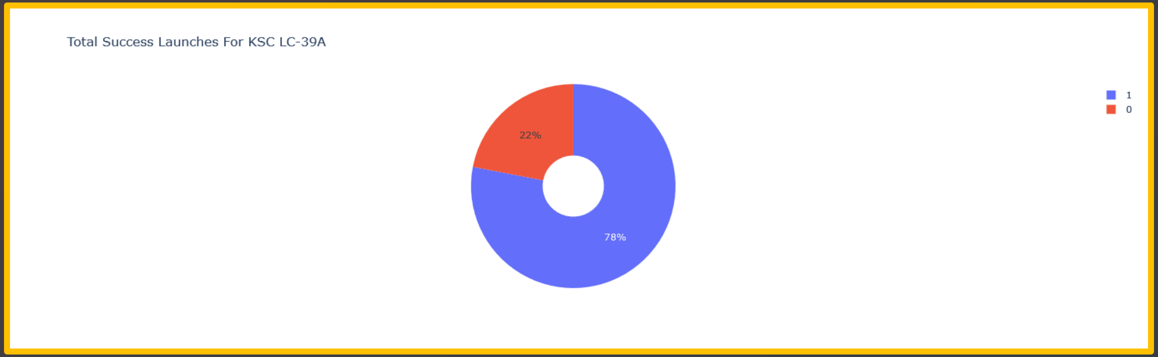


- We calculated the distance between launch site CCAFS SLC-40 and the coastline, the nearest highway, railway and city. To demonstrate that launch sites are located near coastline and transport infrastructure but far from cities to minimize accidents.

RESULTS

Interactive Analytics with Dash

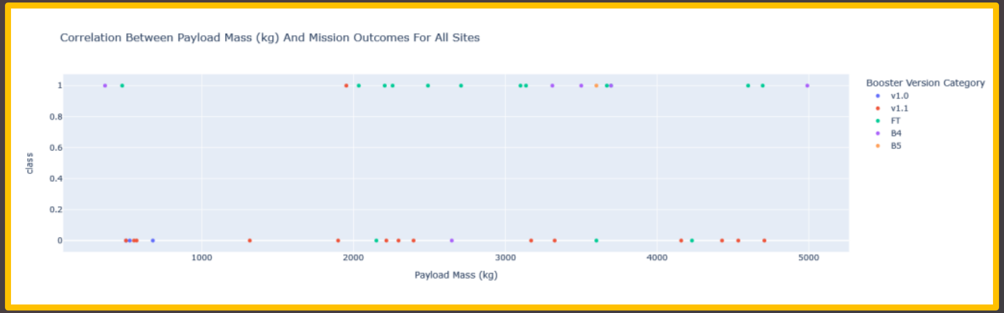




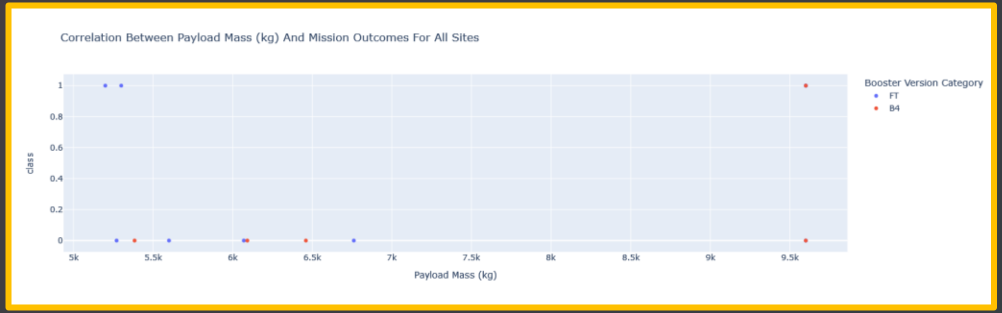
- We can see that the launch site KSC LC-39A has the most successful launches.
- KSC LC-39A has the best success/fail ratio with a success rate of 78% and 22% failure rate.

RESULTS

Interactive Analytics with Dash



- Payload range 0 to 5000 kg has the highest success rate.



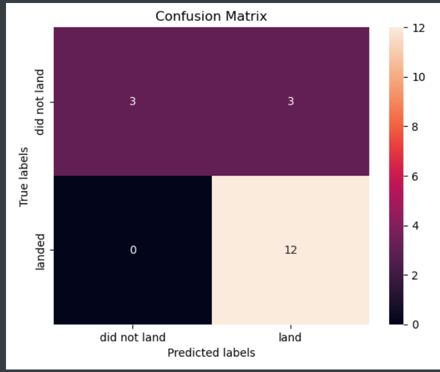
- Payload range 5000 to 10000 kg has the lowest success rate.

- F9 Booster version FT has the highest launch success rate from all versions.

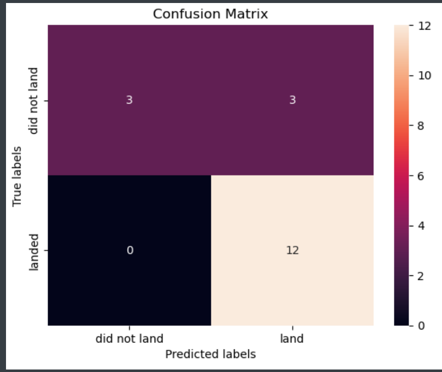
RESULTS

Predictive Analysis – Confusion Matrix

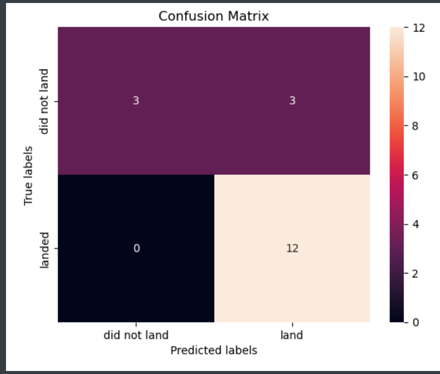
Logistic Regression



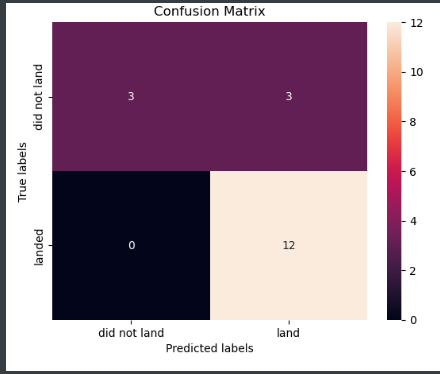
Support Vector Machine



Decision Tree



K-Nearest Neighbors



- After testing the four models, we got the same confusion matrices.

Metrics:

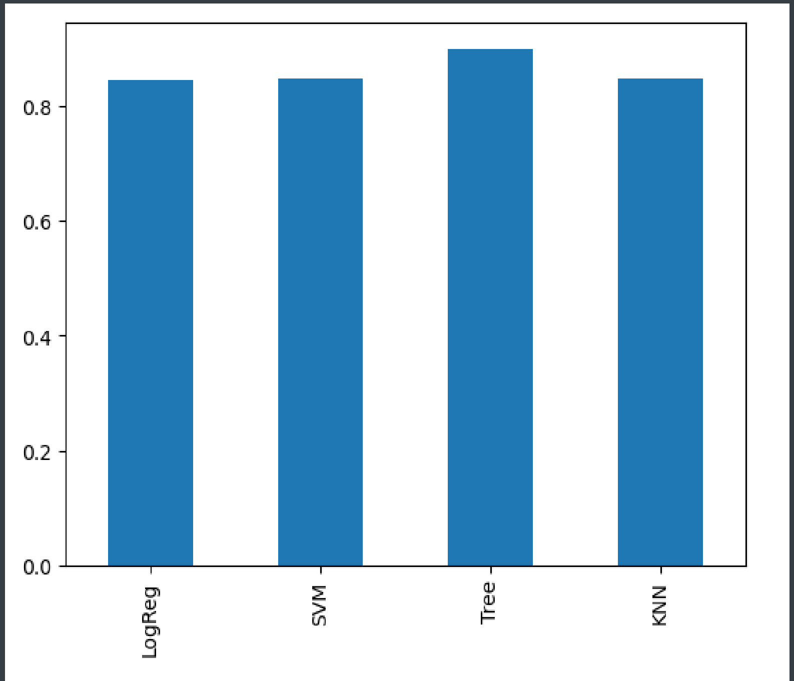
- Accuracy: 83%
- Precision: 50%
- Negative Predictive Value: 80%
- Specificity: 20%
- Sensitivity: 100%

F1-score = 60%

RESULTS

Predictive Analysis – Classification Accuracy

	Accuracy
LogReg	0.846429
SVM	0.848214
Tree	0.889286
KNN	0.848214



- On the training set, the best model is Decision Tree with an accuracy of 89%. But as we can see, it performs slightly better than the others.

CONCLUSION

CONCLUSION

In conclusion:

- Orbits ES-L1, GEO, HEO, SSO and VLEO have the highest success rates.
- From 2013 to 2020 launch success rate never stop to increase.
- The launch site with the most successful launches is KSC LC-39A.
- The Decision Tree model predict that first stage will land with an accuracy of 89%.

Thank You!