



IBM Data Science Capstone Project: SpaceX Launch Analysis

Abhijeet Singh

22/08/2022

OUTLINE



- Executive Summary
- Introduction
- Methodology
- Results
 - Visualization – Charts
 - Dashboard
- Discussion
 - Findings & Implications
- 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



- Project background:
 - SpaceX advertises Falcon9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollar each, much of the saving is because SpaceX can reuse the first stage.
- Problem Statement:
 - The project task is to predicting if the first stage of the SpaceX Falcon9 rocket will land successfully.

Methodology

METHODOLOGY

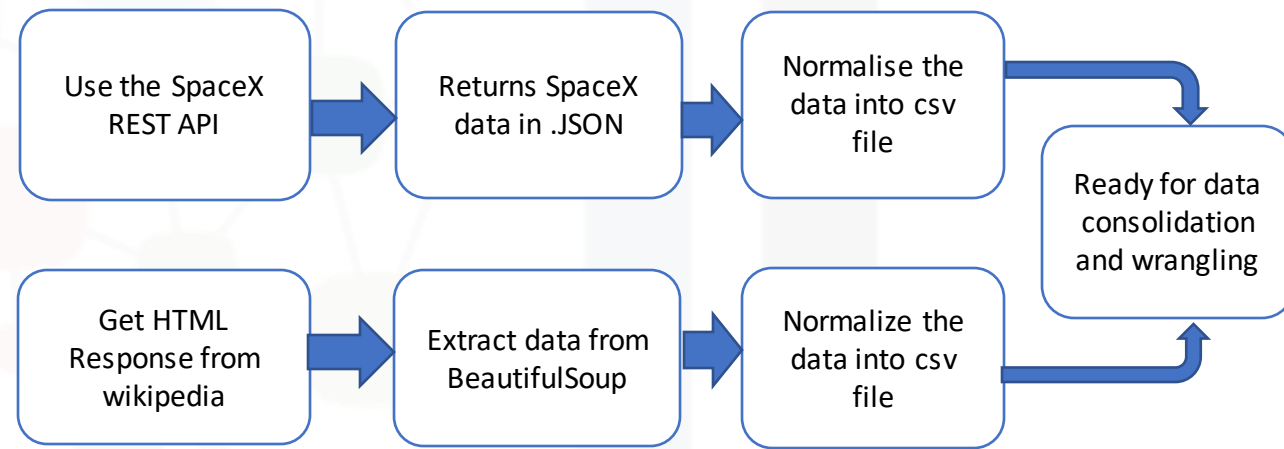


- Executive Summary
 - Data collection methodology:
 - SpaceX Rest API
 - Web Scrapping from Wikipedia
 - Perform data wrangling:
 - One Hot Encoding data fields for machine learning and data cleaning of null values and irrelevant columns
 - Perform exploratory data analysis(EDA) using visualization and SQL
 - Perform interactive visual analytics using Folium and Plotly Dash
 - Perform predictive analysis using classification models
 - LR, KNN, SVM, DT models have been built and evaluated for the best classifier

Data Collection

- The following data set was collected:
 - SpaceX launch data that is gathered from the SpaceX REST API
 - This API will give us data about launches, including the information about the rocket used, payload delivered, launch specifications, landing specifications and landing outcome.
 - The SpaceX REST API endpoints, or URL, starts with <https://api.spacexdata.com/v4/rockets/>
 - Another popular data source for obtaining Falcon9 launch data is web scrapping wikipedia using BeautifulSoup

• SpaceX API



Data Collection And Scrapping SpaceX API

- Data collection with SpaceX REST calls and Web Scrapping from Wikipedia

- https://github.com/Abhijeet-Sih/Capstone-Project/blob/3823feec9a3d3263646de3fc7205b2332b08ac12/Data_Collection&Wrangling.ipynb

Import Libraries and Define Auxiliary Functions

We will import the following libraries into the lab

```
# Requests allows us to make HTTP requests which we will use to get data from an API
import requests

# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd

# Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays
import numpy as np

# Scikit is a library that allows us to represent data
import sklearn

# Setting this option will print all columns of a dataframe
pd.set_option('display_max_columns', None)

# Setting this option will print all of the data in a feature
pd.set_option('display_max_colwidth', None)
```

Below we will define a series of helper functions that will help us use the API to extract information using identification numbers in the launch data.

From the `rocket` column we would like to learn the booster name

```
# Takes the dataset and uses the rocket column to call the API and append the data to t
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the longitude, and the latitude

```
# Takes the dataset and uses the Launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['Launchpad']:
        response = requests.get('https://api.spacexdata.com/v4/launchpads/'+str(x)).json()
        Longitude.append(response['longitude'])
        Latitude.append(response['latitude'])
        LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Go to [Google](#) to find the information you need.

```
# Get the head of the dataframe
data = pd.json_normalize(response.json())
data.head(1)
```

static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads
0	2008-03-17T00:00:00.000Z	1.142554e+09	False	0.0	54940594949595707061e-01	False	{ "time": 33, "engine": "Engine failure #3", "reason": "33 seconds and 10 seconds lost of vehicle" }	0	0	0	{ "seaWebSite": "http://www.nasa.gov/...", "seaWebSite": "http://www.nasa.gov/..." }

You will notice that a lot of the data are IDs. For example the rocket column has no information about the rocket just an identification number

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
```

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

```
# We will remove rows with multiple cores
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]
```

```
# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
```

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

```
# We also want to convert the date utc to a datetime datatype and then extract the date (ignoring the time)
```

```
data['date'] = pd.to_datetime(data['date_utc']).dt.date
```

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

```
# Take the dataset and use the cores column to call the API and append the data to the lists
def get_core_data(s):
    for core in data['cores']:
        if core['core'] != None:
            response = requests.get('https://api.spacetelescope.org/v4/cores/' + core['core'] + '.json')
            Stock.append(response['stock'])
            Serial.append(response['route_count'])
            Serial.append(response['serial'])
        else:
            Stock.append(None)
            Serial.append(None)
            Serial.append(response['route_count'])
            Route.append(str(core['landing_success']) + ' ' + str(core['landing_type']))
            Flight.append(core['flight'])
            Grid.append(response['grid'])
            Waved.append(response['waved'])
            Legn.append(core['legn'])
            Landingpad.append(core['landingpad'])
```

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

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

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

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 successful with the 200 status response code

```
response.status_code
```

200

```
response.json()
```

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_falcons`.

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

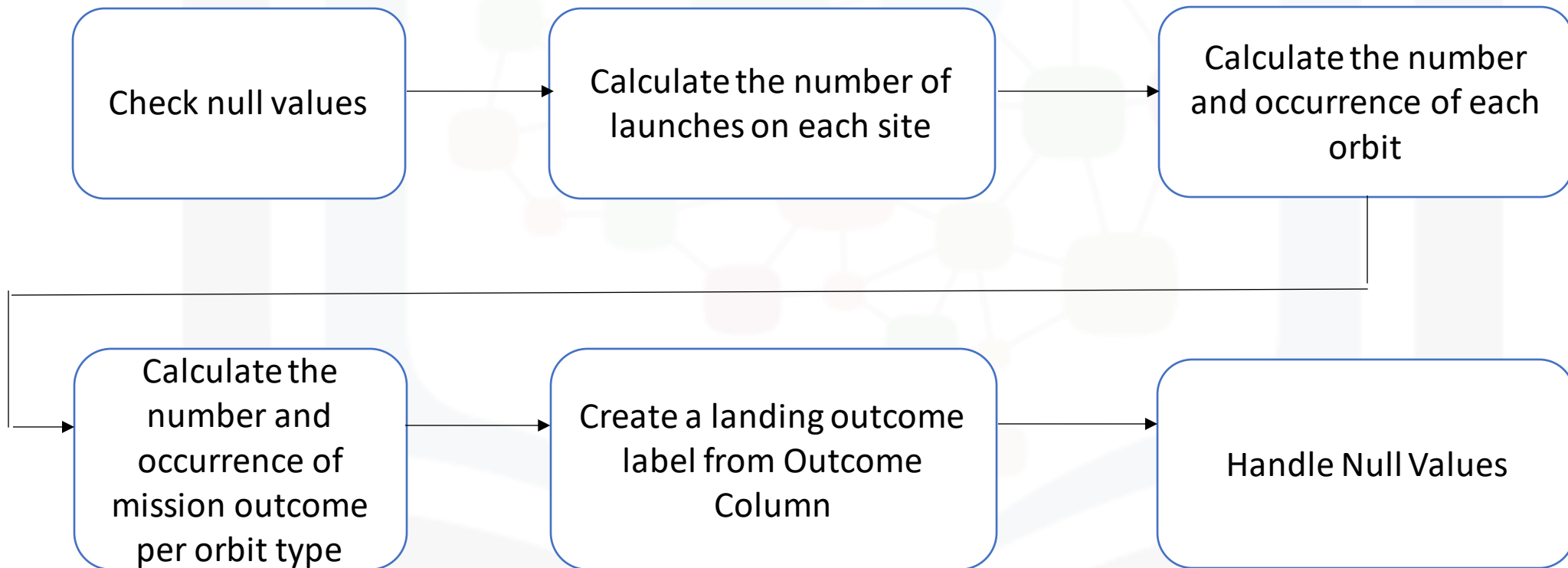
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	Gridfins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Lat
4	6	2010-03-04	Falcon 9	N/A	LEO	CCAFS SLC 40	None	None	1	False	False	False	None	1.0	0	B00031-08
	4	2010-03-22	Falcon 9	\$250,000	LEO	CCAFS SLC 40	None	None	1	False	False	False	None	1.0	0	B00031-08
5	10	2010-03-07	Falcon 9	\$770,000	ISS	CCAFS SLC 40	None	None	1	False	False	False	None	1.0	0	B00037-18
	10	2010-03-29	Falcon 9	\$500,000	PO	VAFB SLC 4E	Orion Crew	1	False	False	False	None	1.0	0	B10012-13	
8	12	2010-12-03	Falcon 9	\$1170,000	GTO	CCAFS SLC 40	None	None	1	False	False	False	None	1.0	0	B1004-08
	12	2010-12-03														
89	102	2006-09-03	Falcon 9	158000	VLEO	KSC LC 39A	True AIGOS	2	True	True	True	SeaWif0310380406236aTcc	5.0	11	B1069-08	
90	103	2006-09-06	Falcon 9	158000	VLEO	KSC LC 39A	True AIGOS	3	True	True	True	SeaWif0310380406236aTcc	5.0	11	B1058-08	
91	104	2006-10-18	Falcon 9	158000	VLEO	KSC LC 39A	True AIGOS	6	True	True	True	SeaWif0310380406236aTcc	5.0	11	B1051-08	
92	105	2006-10-24	Falcon 9	158000	VLEO	CCAFS SLC 40	True AIGOS	3	True	True	True	SeaWif0310380406236aTcc	5.0	11	B1060-08	
93	106	2006-11-03	Falcon 9	368100	MEO	CCAFS SLC 40	True AIGOS	1	True	True	True	SeaWif0310380406236aTcc	5.0	5	B1062-08	

90 rows x 17 columns

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

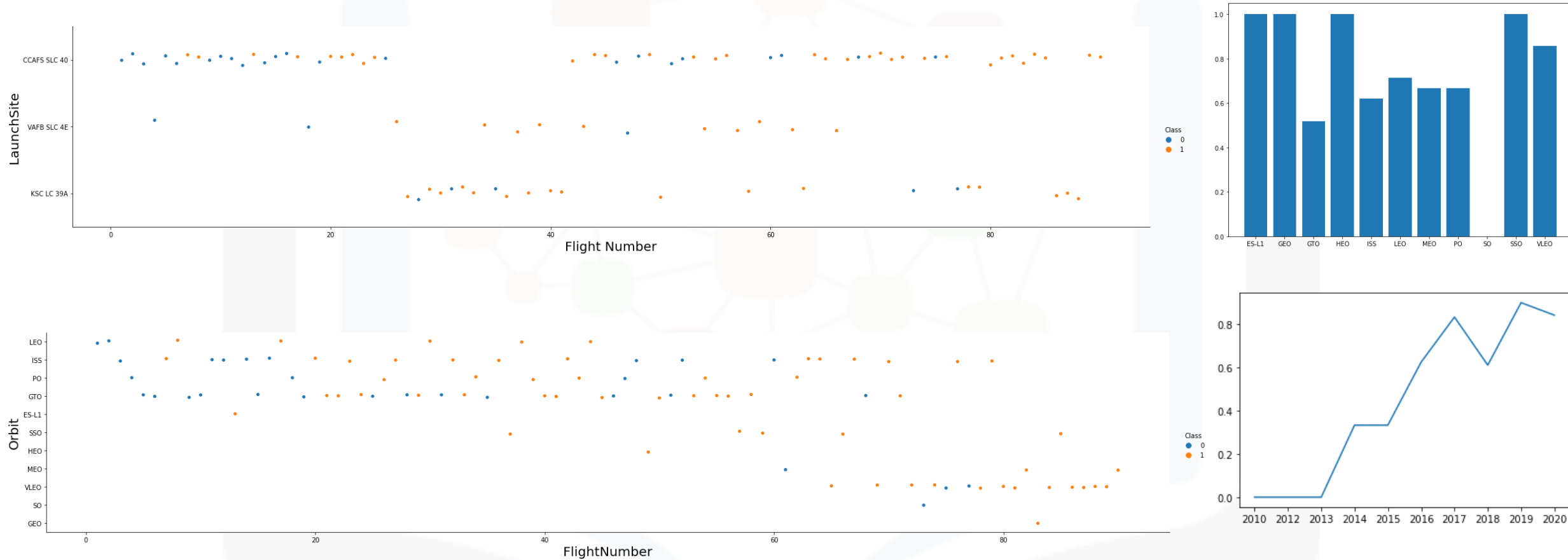
Data Wrangling

EDA Analysis

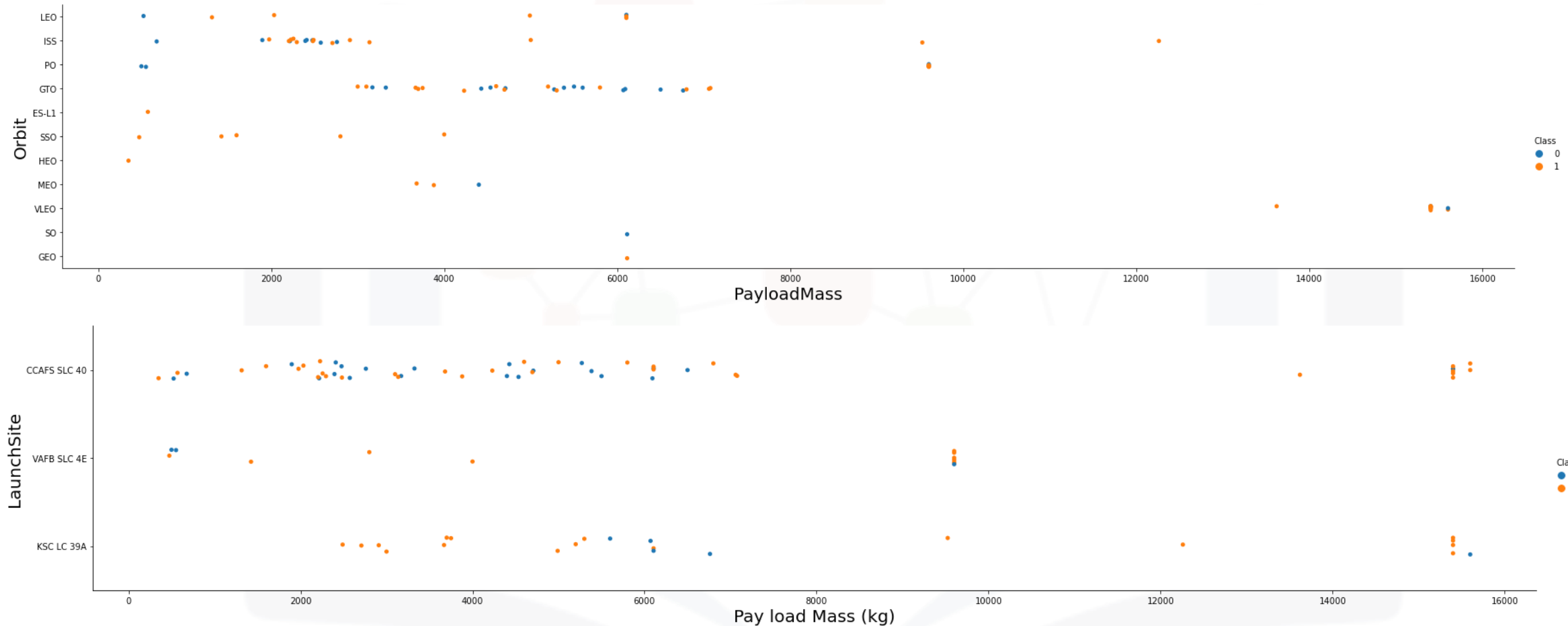


<https://github.com/Abhijeet-Sih/Capstone-Project/blob/a8a0052333ba3732c8596652b90352d70bc977f9/Data%20Wrangling.ipynb>

EDA with Data Visualization



EDA with Data Visualization



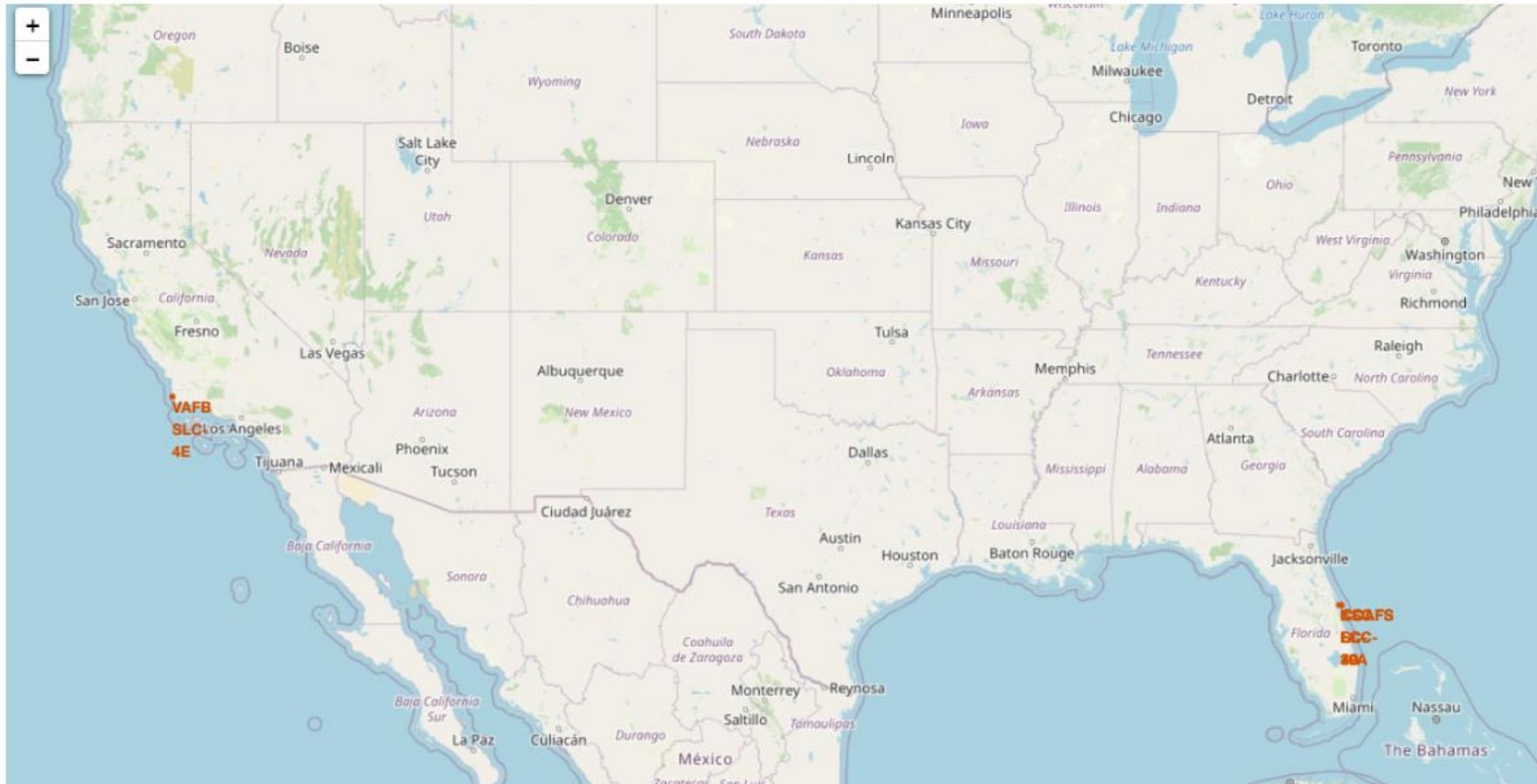
https://github.com/Abhijeet-Sih/Capstone-Project/blob/fe12ded58389b2faf89ef4bb1b32d95594580cb8/EDA_with_data_visualization.ipynb

EDA with SQL

- SQL queries performed include:
 - Displaying the names of the unique launch sites in the space mission.
 - Displaying 5 records where launch site begin with string 'KSC'
 - Displaying the total payload mass carried by boosters launched by NASA (CRS)
 - Displaying the average payload mass carried by booster version F9 v1.1
 - Listing the date where the successful landing outcome in drone ship was achieved
 - Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
 - Listing the total number of successful and failure mission outcomes
 - Listing the names of the booster versions which have carried the maximum payload mass
 - Listing the records which will display the month names, successful landing outcomes in ground pad, booster versions, launch site for the months in year 2017
 - Ranking the count of successful landing outcomes between the date 2010-06-04 and 2017-03-30 in descending order

<https://github.com/Abhijeet-Sih/Capstone-Project/blob/a92bcd2fd68e2567bdb2a96a83b726fab373ded2/EDA%20with%20SQL%20.ipynb>

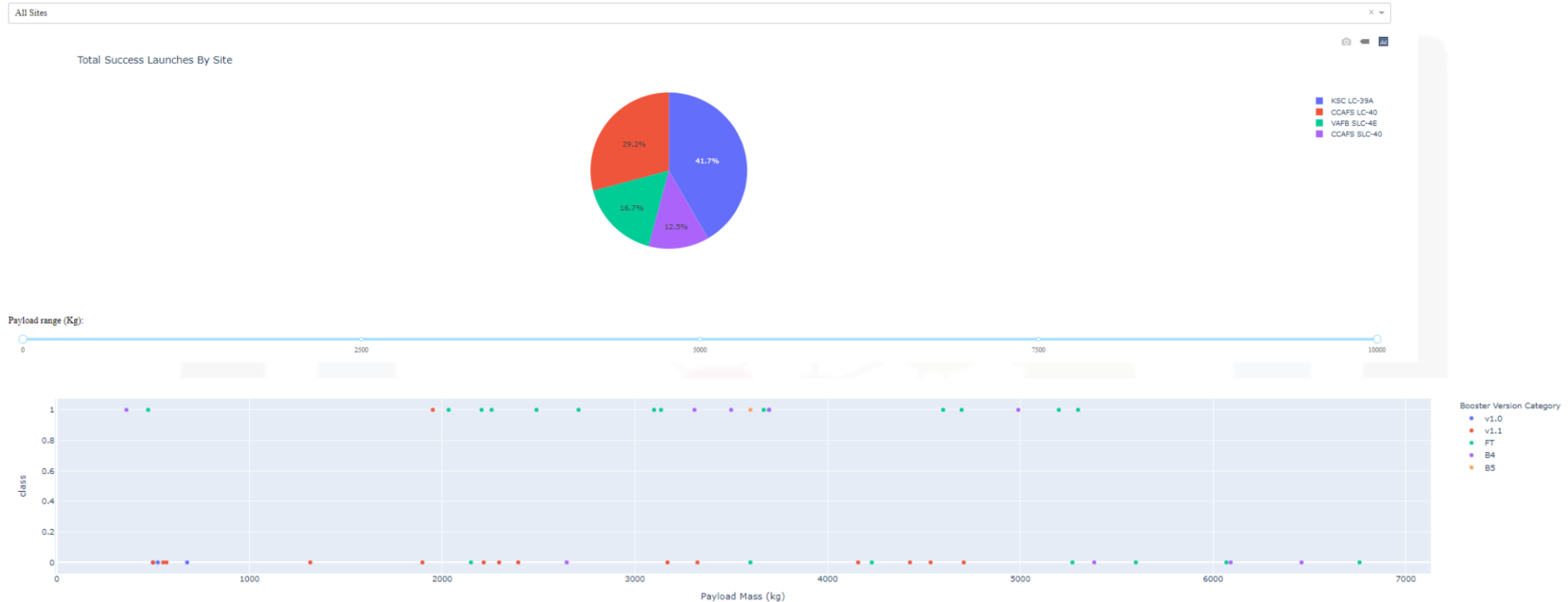
Build an Interactive Map with Folium



https://github.com/Abhijeet-Sih/Capstone-Project/blob/cf77b026ce8c28ab9a0df03ec0969ad9058e92a9/Interactive_map_with_Folium.ipynb

Build a dashboard with Plotly Dash

SpaceX Launch Records Dashboard

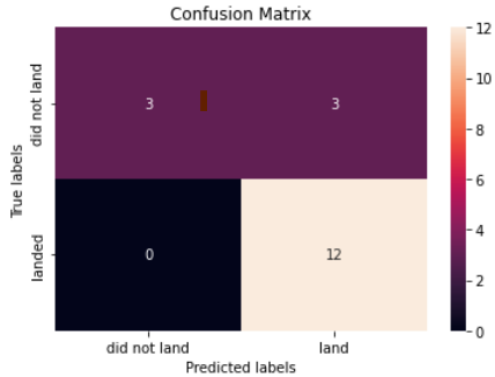


https://github.com/Abhijeet-Sih/Capstone-Project/blob/f93695b96205823667675c05df9b06ec3b2f3752/spacex_dash_app.py

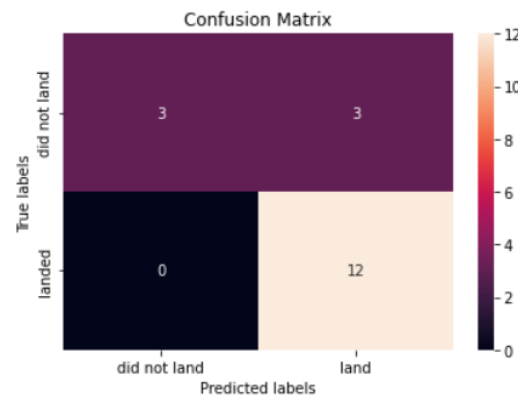
Predictive Analysis

- The SVM, KNN and Logistic Regression model achieved the highest accuracy at 83.33%.

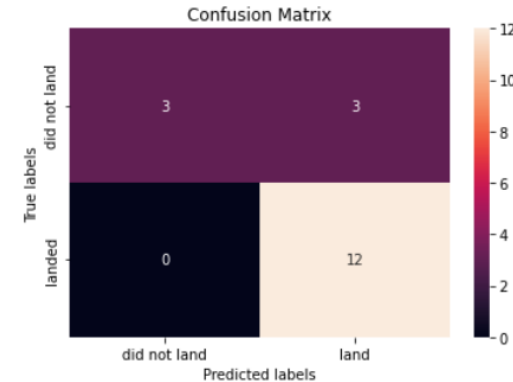
```
yhat=logreg_cv.predict(X_test)  
plot_confusion_matrix(Y_test,yhat)
```



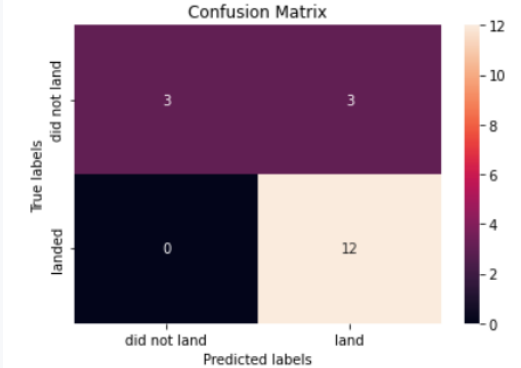
```
yhat=svm_cv.predict(X_test)  
plot_confusion_matrix(Y_test,yhat)
```



```
yhat = svm_cv.predict(X_test)  
plot_confusion_matrix(Y_test,yhat)
```



```
yhat = knn_cv.predict(X_test)  
plot_confusion_matrix(Y_test,yhat)
```



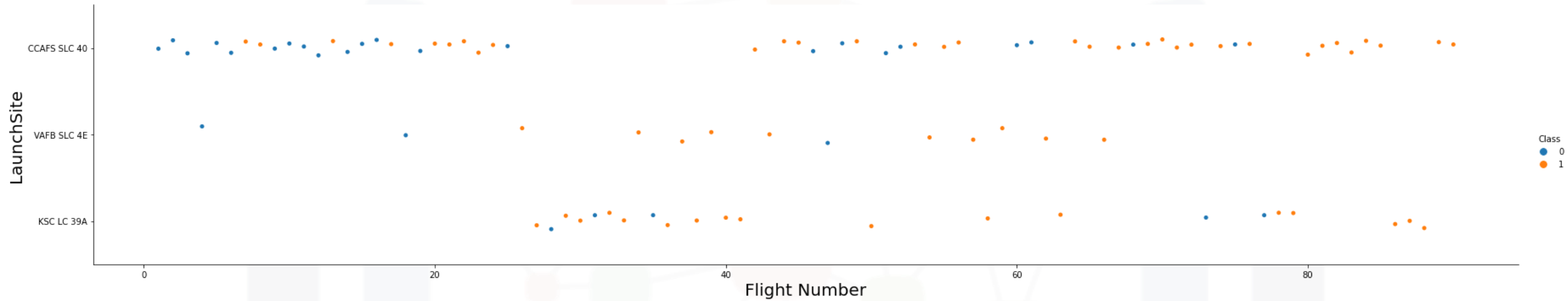
<https://github.com/Abhijeet-Sih/Capstone-Project/blob/0bf4e589b04fe1b175e05ecb31ae87fd827239e0/Predictive%20Analysis.ipynb>

Results

- The SVM, KNN, Logistic Regression and Decision Tree models are the equivalent in terms of accuracy for this dataset.
- Low weighted payloads perform better than the heavier payloads.
- The success rates for the SpaceX launches is directly proportional to time in years.
- KSC LC 39A had the most successful launches from all the sites.
- Orbits GEO, HEO, SSO, ES-L1 has the best Success Rates.

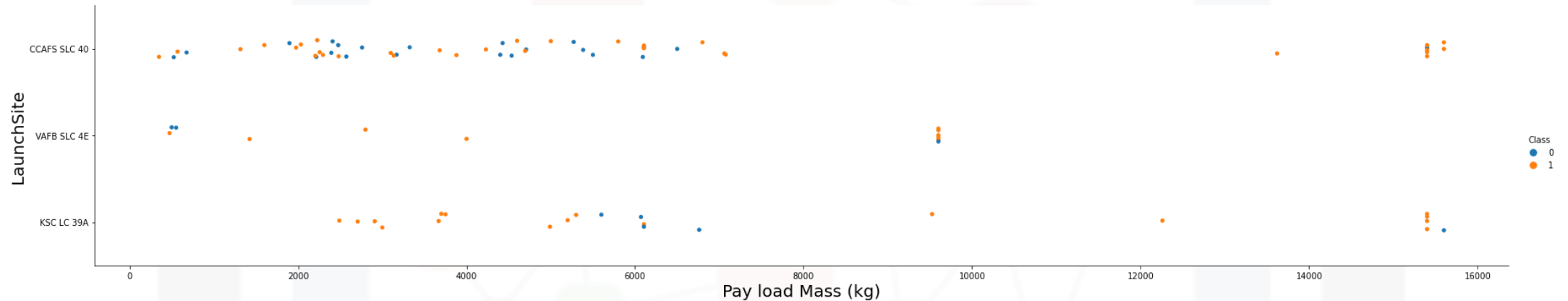
Insights Drawn From EDA

Flight Number vs. Launch Site



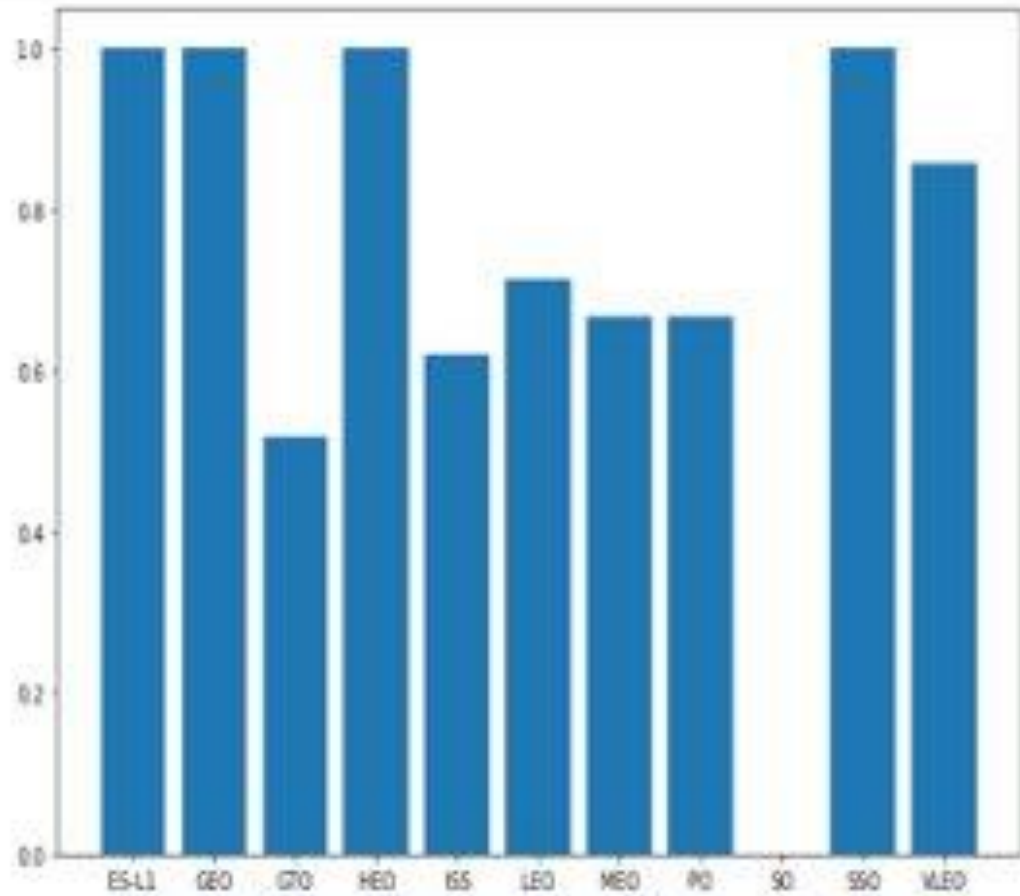
- Launches from the site of CCAFS SLC 40 are significantly higher than the launches from the other sites.

Payload vs. Launch Site



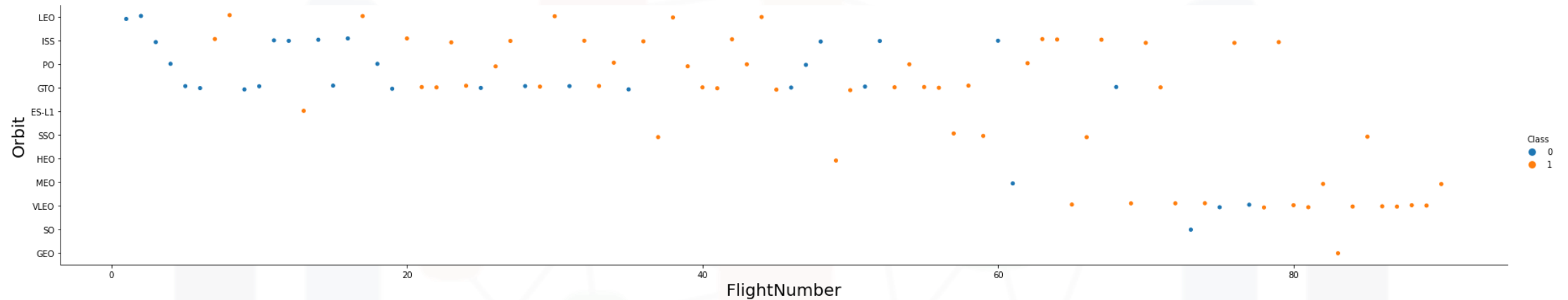
- Majority of payloads with lower mass have been launched from CCAFS SLC 40.

Success Rate vs. Orbit Type



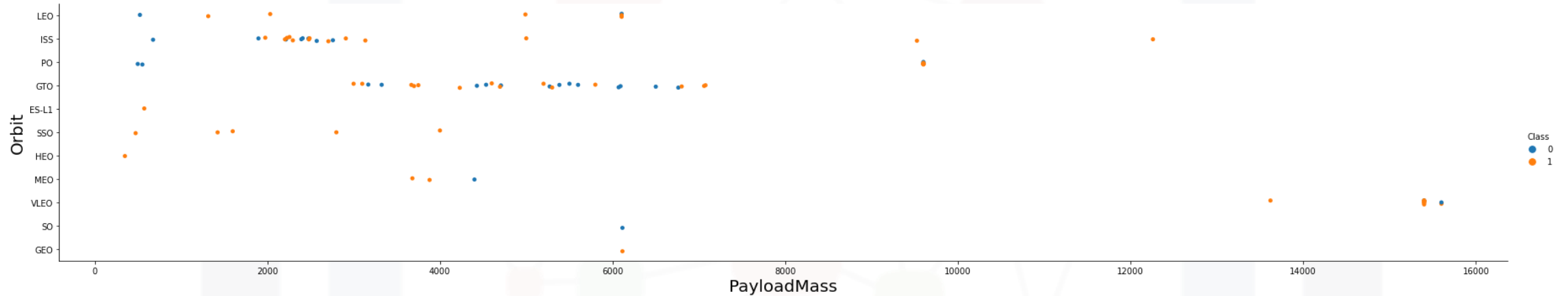
- The orbit types ES-L1, GEO, HEO, SSO are among the highest success rates.

Flight Number vs. Orbit Type



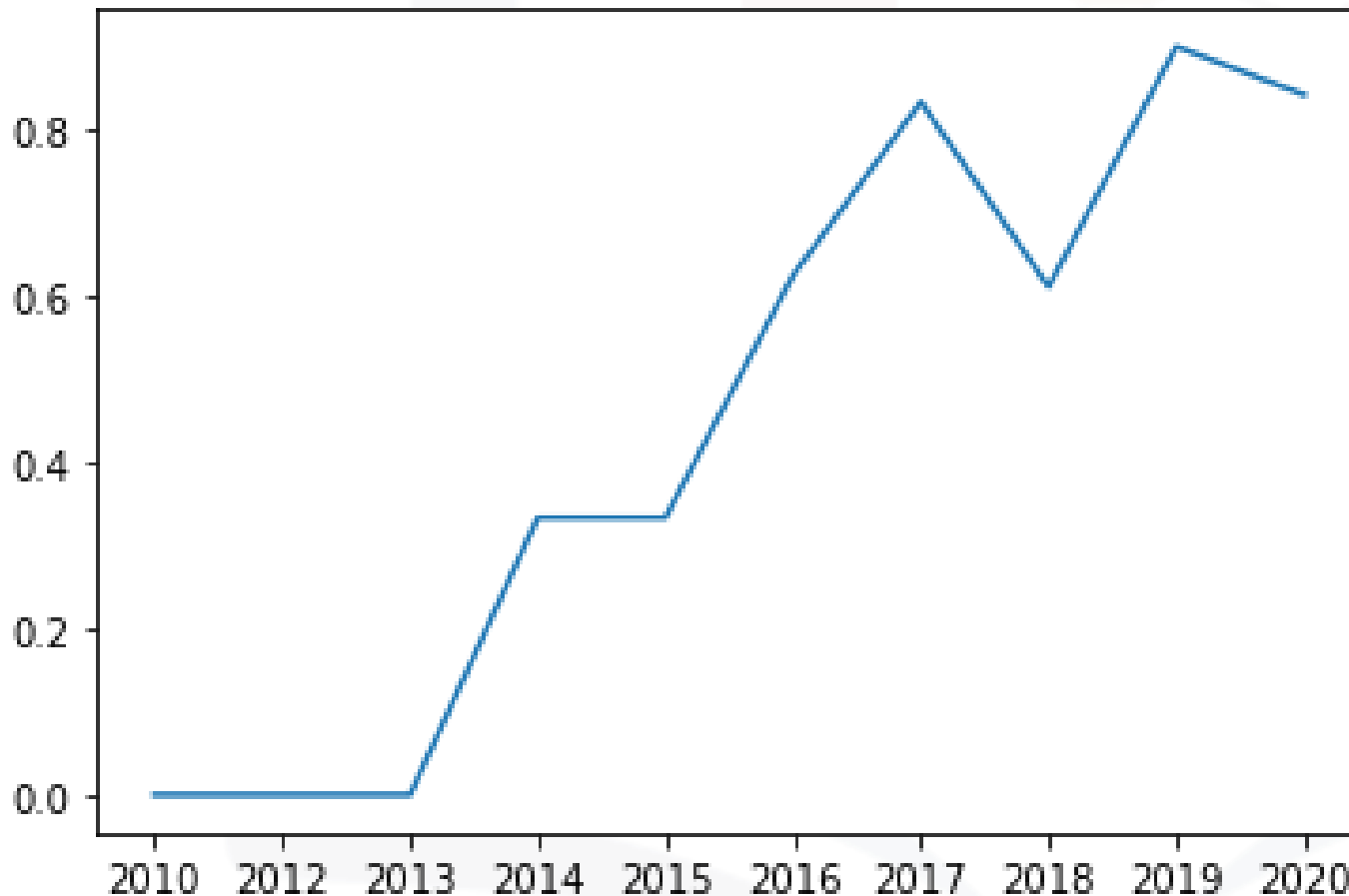
- A trend can be observed of shifting to VLEO launches in recent years

Payload vs. Orbit Type



- There are strong correlation between ISS and payload at the range around 2000, as well as between GTO and the range of 4000-8000.

Launch Success Yearly Trend



- Launch success rate has increased significantly since 2013 and has stabilized since 2019, potentially due to advance in technologies and lessons learned.

All Launch Site Names

launch_site

CCAFS LC-40

CCAFS SLC-40

KSC LC-39A

VAFB SLC-4E

- `%sql SELECT DISTINCT(Launch_Site) from SPACEXTBL;`

Launch Site Names Begin with 'CCA'

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- `%sql SELECT * FROM SPACEXTBL WHERE launch_site LIKE 'CCA%' LIMIT 5;`

Total Payload Mass

`total_payload_mass`

`45596`

- `%sql SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass FROM SPACEXTBL WHERE Customer = 'NASA (CRS)' GROUP BY Customer;`

Average Payload Mass By F9 v1.1

payload_F9 v1.1

2928

- %sql SELECT AVG(PAYLOAD_MASS__KG_) AS "payload_F9 v1.1" FROM SPACEXTBL WHERE booster_version = 'F9 v1.1' GROUP BY booster_version;

First Successful Ground Landing Date

Date_first_success_launch

2015-12-22

- %sql SELECT MIN(DATE) as "Date_first_success_launch" FROM SPACEXTBL WHERE landing__outcome LIKE '%ground pad%';

Successful Drone Ship Landing with Payload Between 4000 and 6000

booster_version

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

- %sql SELECT booster_version FROM SPACEXTBL WHERE landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 and 6000;

Total Number of Successful and Failure Mission Outcome

success	failure
100	1

- %sql SELECT SUM(CASE WHEN mission_outcome LIKE 'Success%' THEN 1 ELSE 0 END) AS Success, SUM(CASE WHEN mission_outcome LIKE 'Failure%' THEN 1 ELSE 0 END) AS Failure FROM SPACEXTBL;

Boosters Carried Maximum Payload

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

- %sql SELECT booster_version FROM SPACEXTBL WHERE payload_mass__kg_ IN (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL);

Failed Landing Outcomes in year 2015

Month	landing__outcome	booster_version	launch_site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- %sql SELECT TO_CHAR(date, 'Month') AS "Month", landing__outcome, booster_version, launch_site FROM SPACEXTBL WHERE EXTRACT(YEAR FROM date) = 2015 AND landing__outcome = 'Failure (drone ship)';

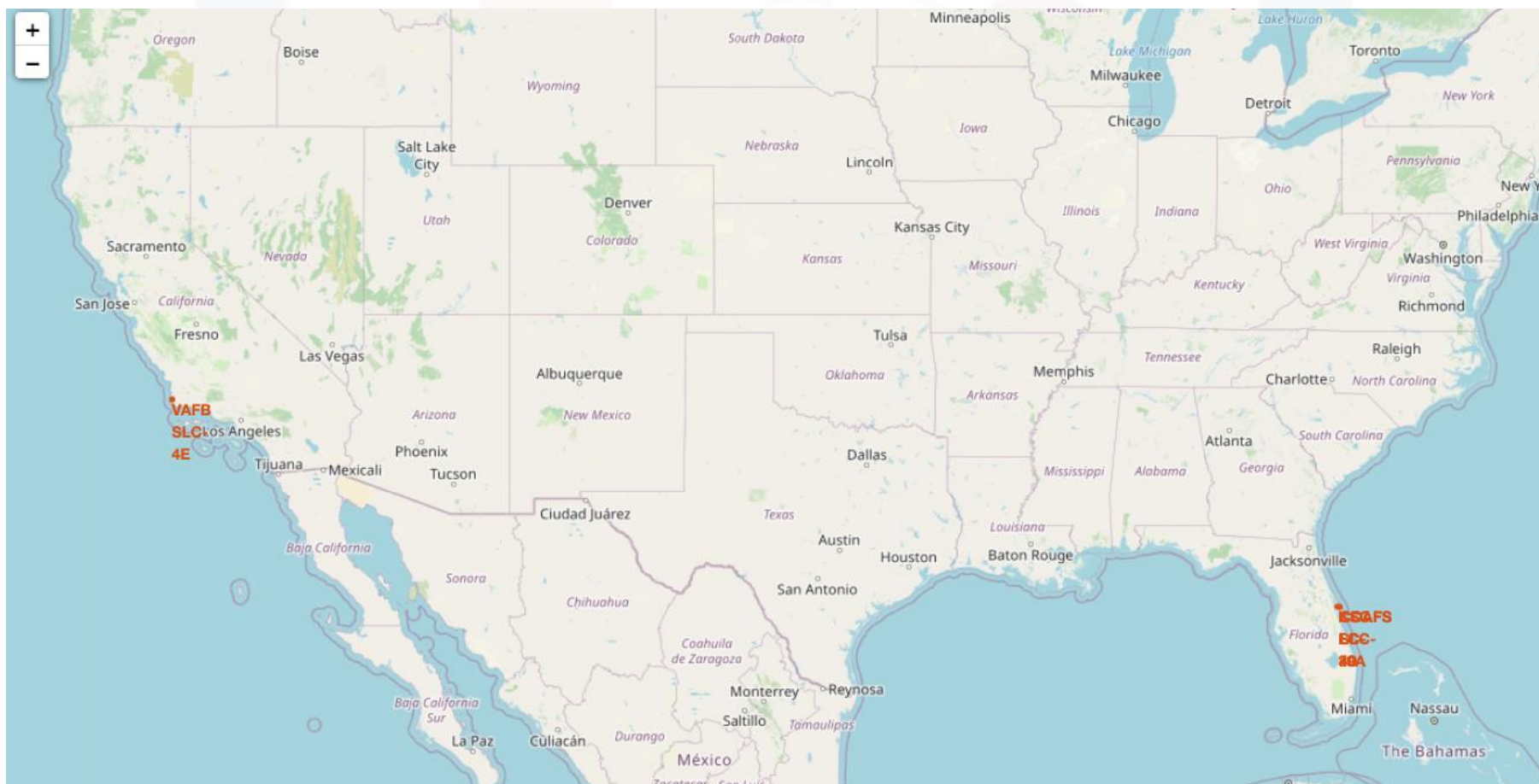
Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

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

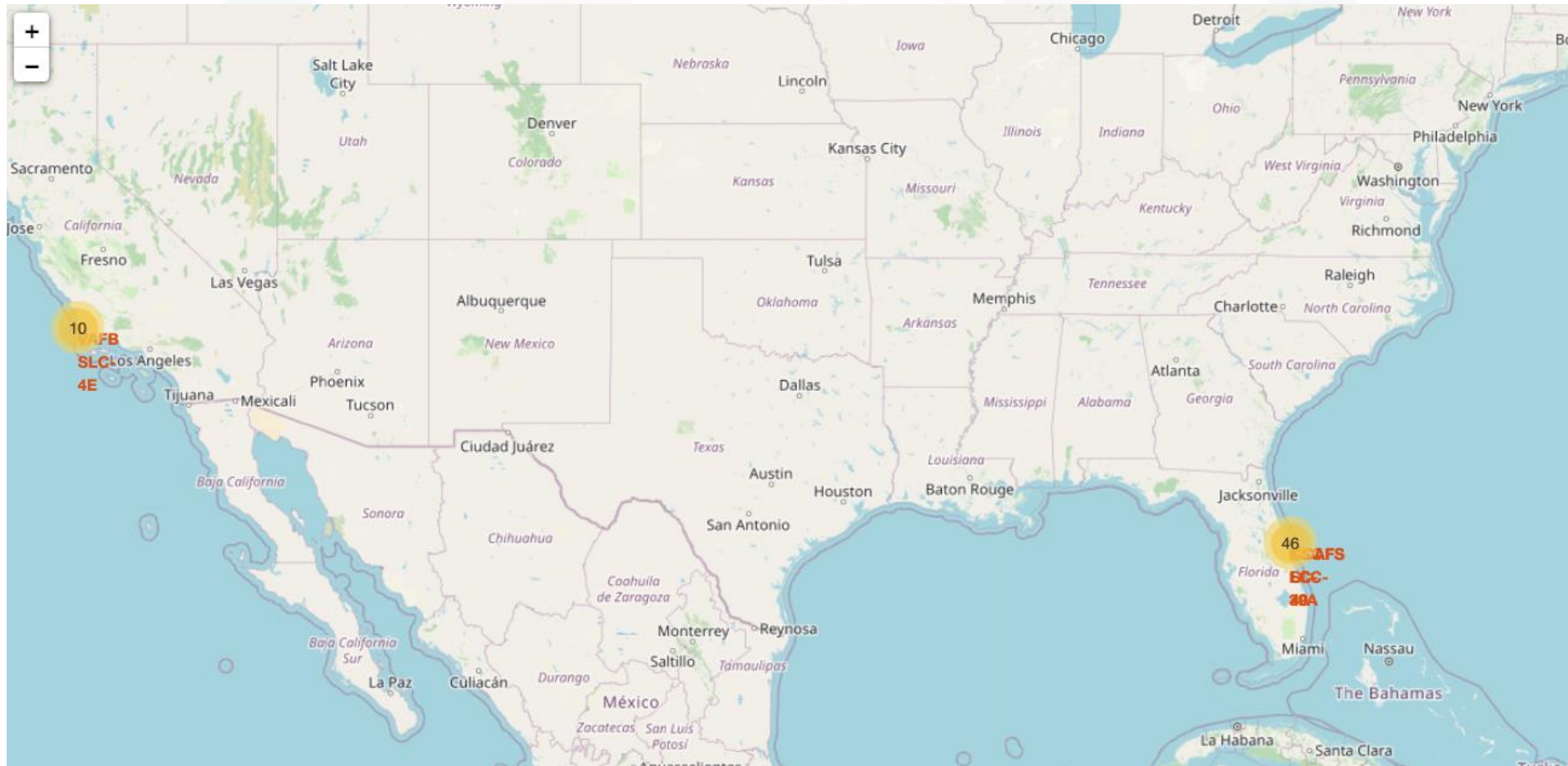
- %sql SELECT landing__outcome, COUNT(*) AS count_of_lo FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing__outcome ORDER BY count_of_lo DESC;

Launch Site Proximities Analysis

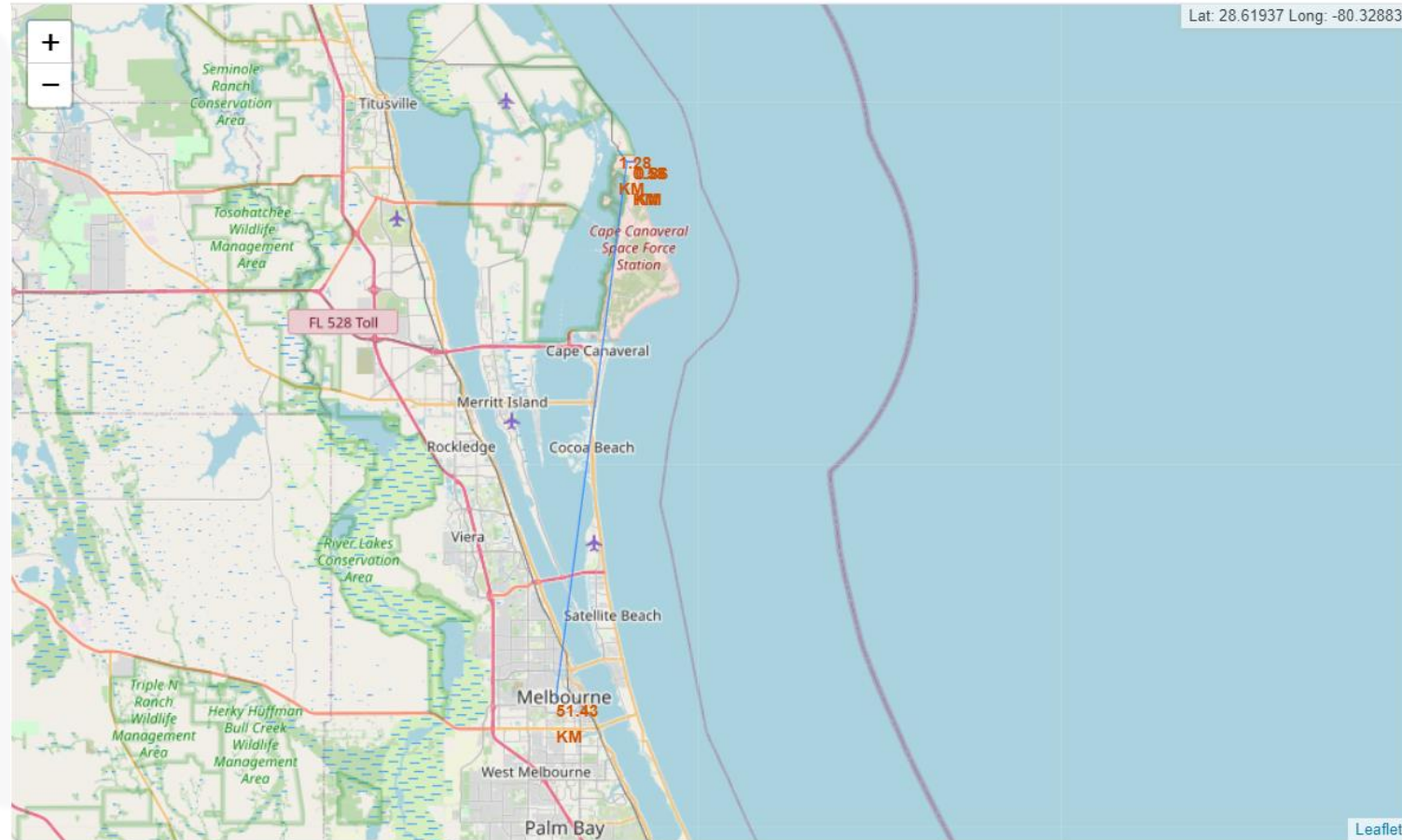
All Launch Site on The Map



Marked the success/failed launches for each site on the map



Distances Between a Launch Site and its Proximities



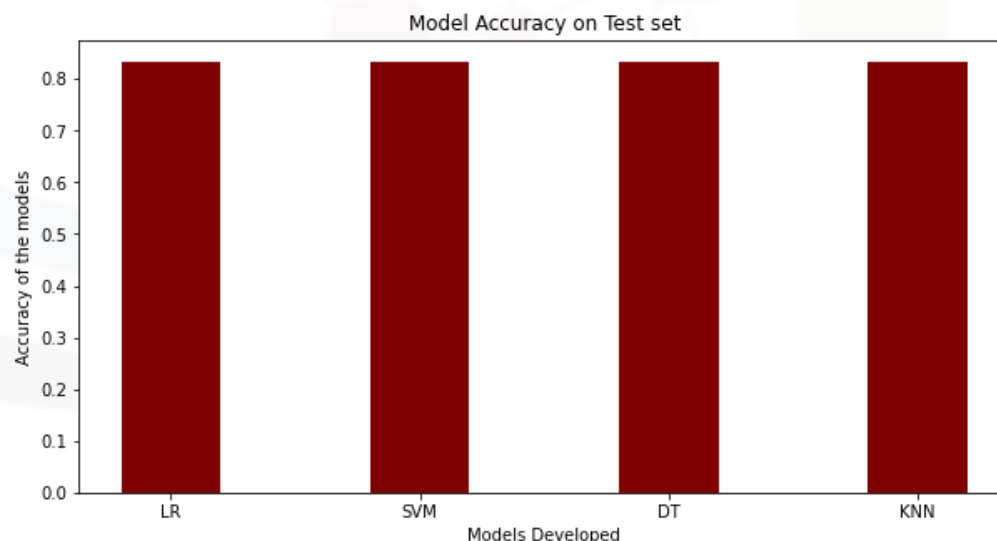


Predictive Analysis

Classification Accuracy

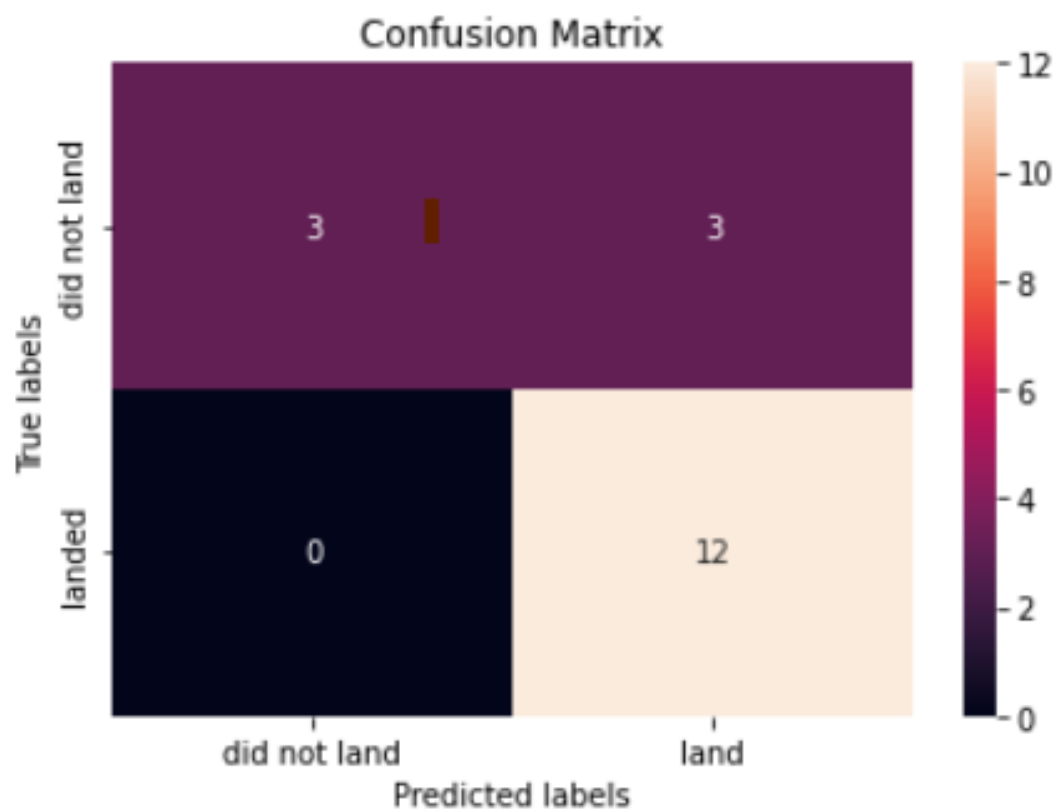
```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))  
print('Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))  
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))  
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.8333333333333334
Accuracy for K nearsdt neighbors method: 0.8333333333333334

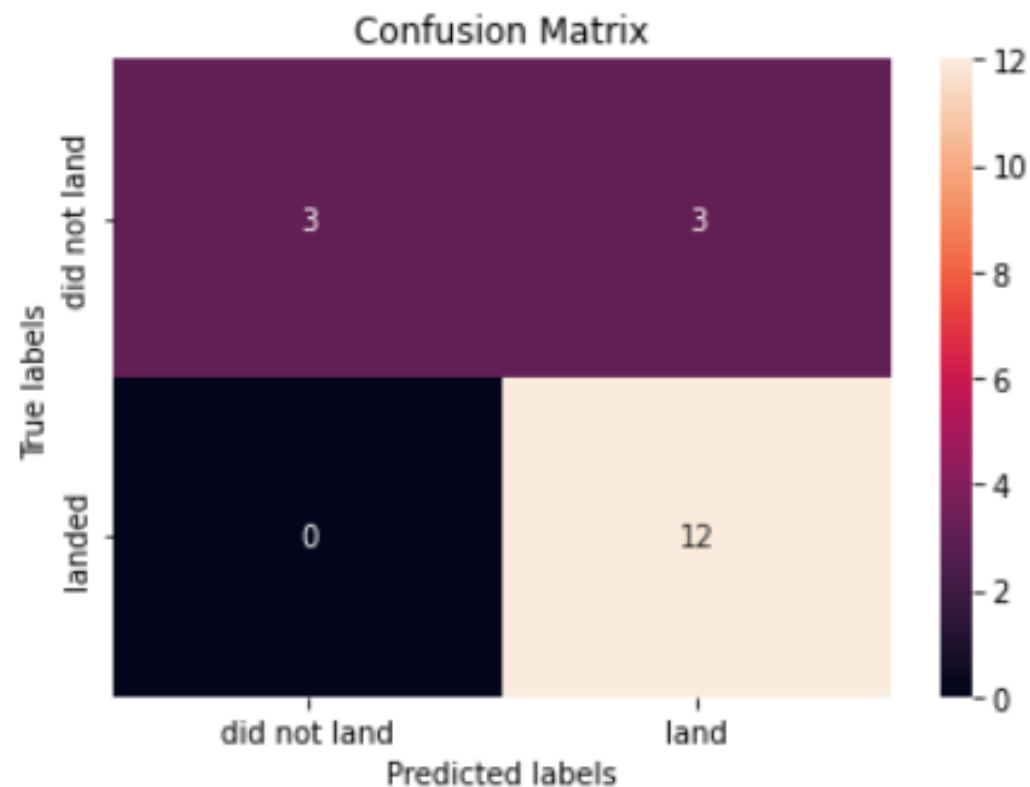


Confusion Matrix

```
yhat=logreg_cv.predict(X_test)  
plot_confusion_matrix(Y_test,yhat)
```

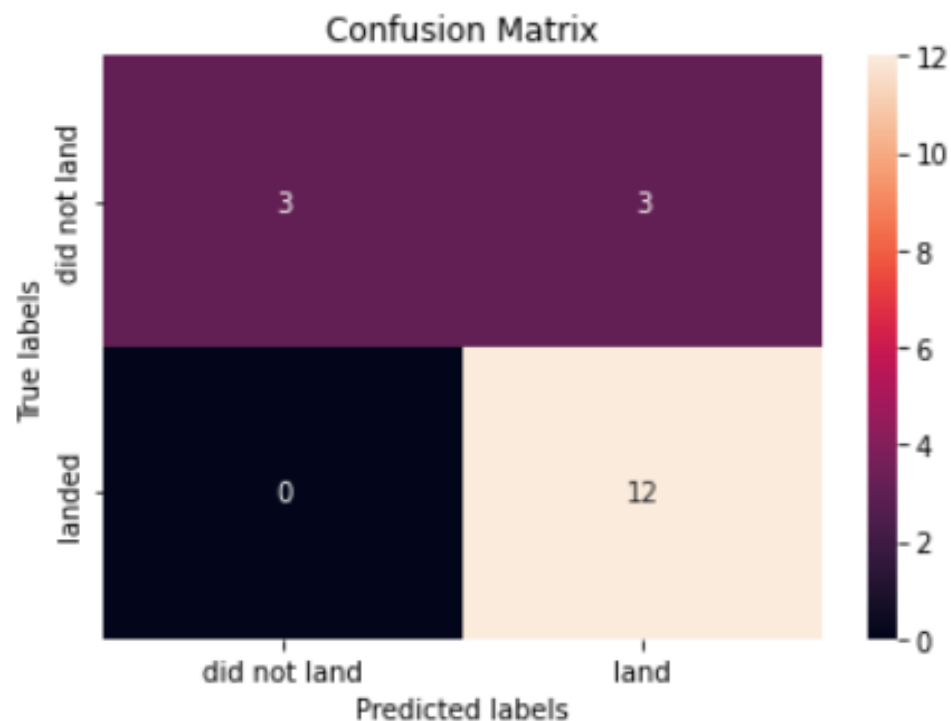


```
yhat = svm_cv.predict(X_test)  
plot_confusion_matrix(Y_test,yhat)
```

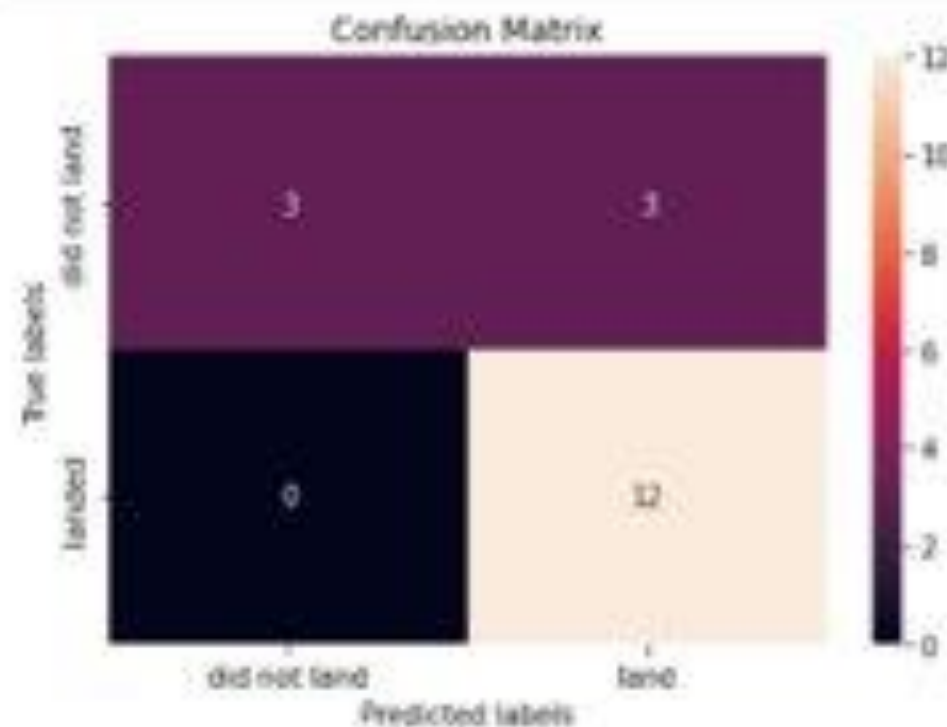


Confusion Matrix

```
yhat=svm_cv.predict(X_test)  
plot_confusion_matrix(Y_test,yhat)
```



```
yhat = knn_cv.predict(X_test)  
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- The SVM, KNN, Logistic Regression and Decision Tree models are equivalent in terms of accuracy for this dataset.
- Low weighted payloads perform better than the heavier payloads.
- The success rates for the SpaceX launches is directly proportional to time in years.
- KSC LC 39A had the most successful launches from all the sites.
- Orbits GEO, HEO, SSO, ES-L1 has the best Success Rates.

Thank You!!!