IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Philippe
August 2022

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context.
- The biggest company in the rocket launching industry is Space X. This company provides low cost rocket launches thanks to reusable first stages. To challenge Space X, our goal is to predict successful launches of Space X rockets. This way, we'll be able to bid against the company for a rocket launch.
- Problems we want to find answers to

- What are the factors that influence rocket launches?
- Can we predict successful rocket launches?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - We collected the data using python, SpaceX API and Wikipedia web scraping.
- Perform data wrangling
    - Data was called, we filled the missing values and we created one-hot vectors
- 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

- The data was collected using python, SpaceX API and Wikipedia web scraping.
- We sent a get request to the SpaceX API.
- We transformed the request content into a json then a pandas data frame.
- With Wikipedia and beautifulsoup, we extracted data from launch record tables.

Data Collection – SpaceX API

- We collected data using get request and the SpaceX API, then we convert it to a dataframe and clean it.
- https://github.com/pmassouf/IBM_Data_Science_Project_SpaceX/blob/b40198ed46aaa6ae15b6e6872d8fd8aa4974cae0/jupyter-labs-spacex-data-collection-api%20(2).ipynb

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

In [12]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

In [13]:
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

In [30]:
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

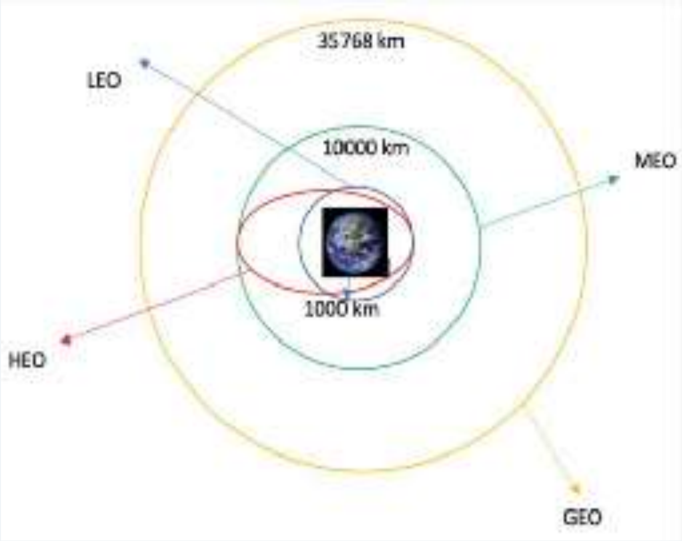
data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

- We performed web scrapping on launch records with the help of BeautifulSoup
- https://github.com/pmassouf/IBM_Data_Science_Project_SpaceX/blob/b40198ed46aaa6ae15b6e6872d8fd8aa4974cae0/jupyter-labs-webscraping.ipynb

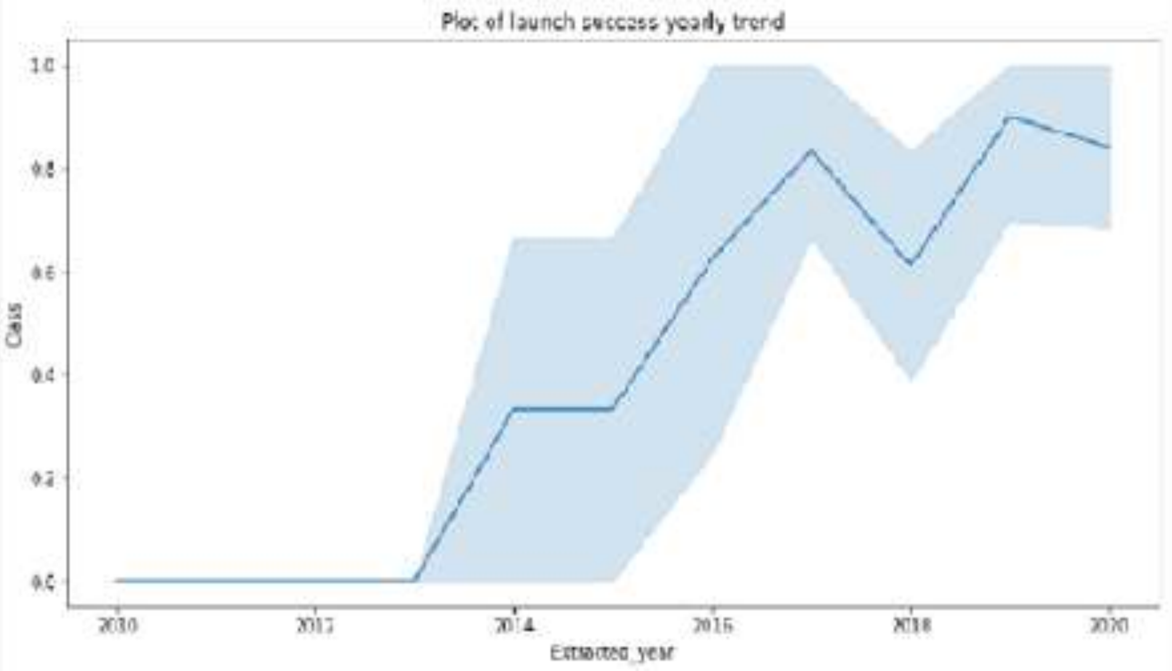
Data Wrangling

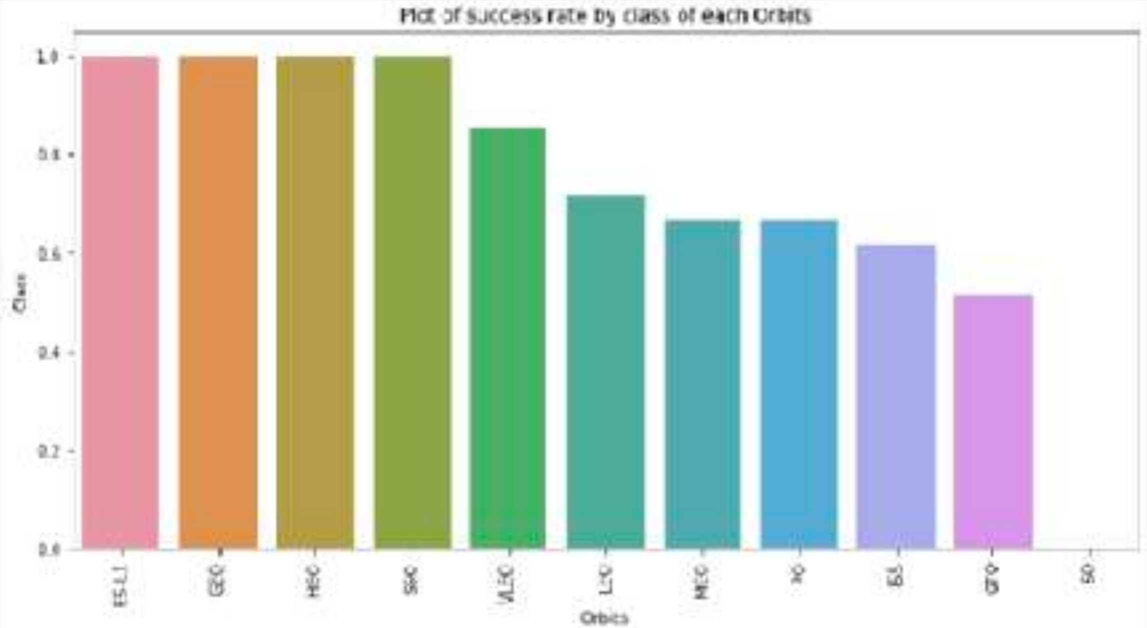
- We performed an exploratory data analysis
- We extracted the number of times each orbit occurred and the number of launches at each site.
- https://github.com/pmassouf/IBM_Data_Science_Project_SpaceX/blob/b40198ed46aaa6ae15b6e6872d8fd8aa4974cae0/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- We have visualised the relationship between successful landings an orbits, year and launcher site
- https://github.com/pmassouf/IBM_Data_Science_Project_SpaceX/blob/b40198ed46aaa6ae15b6e6872d8fd8aa4974cae0/jupyter-labs-eda-





EDA with SQL

- We loaded the dataset in the notebook and perform SQL queries, in order to : find the total number of successful and failed mission, the payload mass or the names of unique launch sites for example.

- https://github.com/pmassouf/IBM_Data_Science_Project_SpaceX/blob/b40198ed46aaa6ae15b6e6872d8fd8aa4974cae0/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We created an interactive Folium map where we marked all launch sites and identified the ones where there is a high success rate
- https://github.com/pmassouf/IBM_Data_Science_Project_SpaceX/blob/b40198ed46aaa6ae15b6e6872d8fd8aa4974cae0/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We created an interactive plotly dashboard
- With a pie chart we can see the total launches by certain sites
- https://github.com/pmassouf/IBM_Data_Science_Project_SpaceX/blob/5fcdfb6803a20c09f8247b3d87b007b5d75158ee/dash_interactivity.py

Predictive Analysis (Classification)

- Using Numpy and Pandas, we loaded the data, transformed it, and divided it into training and testing sets.
- Using GridSearchCV, we constructed various machine learning models and tuned various hyperparameters.
- Our model was measured by accuracy, and it was enhanced through feature engineering and algorithm tweaking.
- The most effective classification model was discovered.
- https://github.com/pmassouf/IBM_Data_Science_Project_SpaceX/blob/5fcdfb6803a20c09f8247b3d87b007b5d75158ee/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

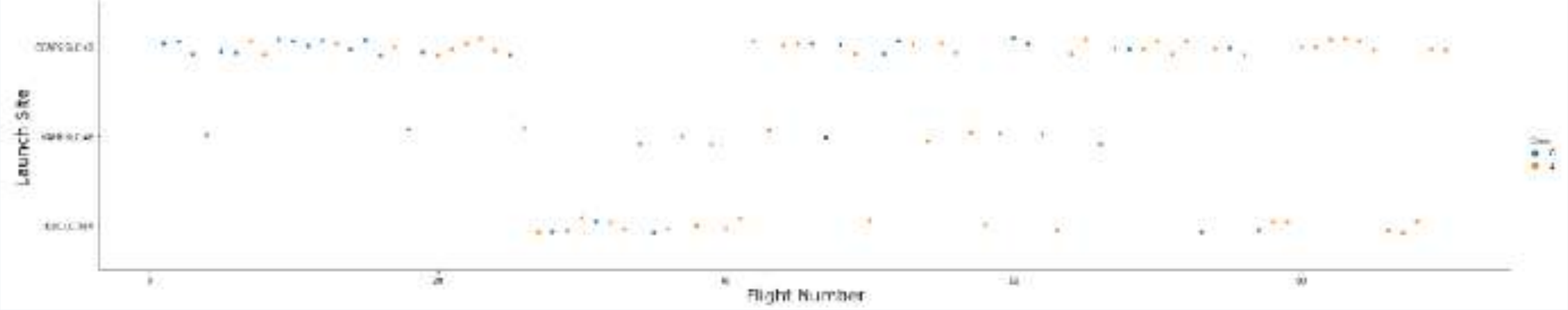
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

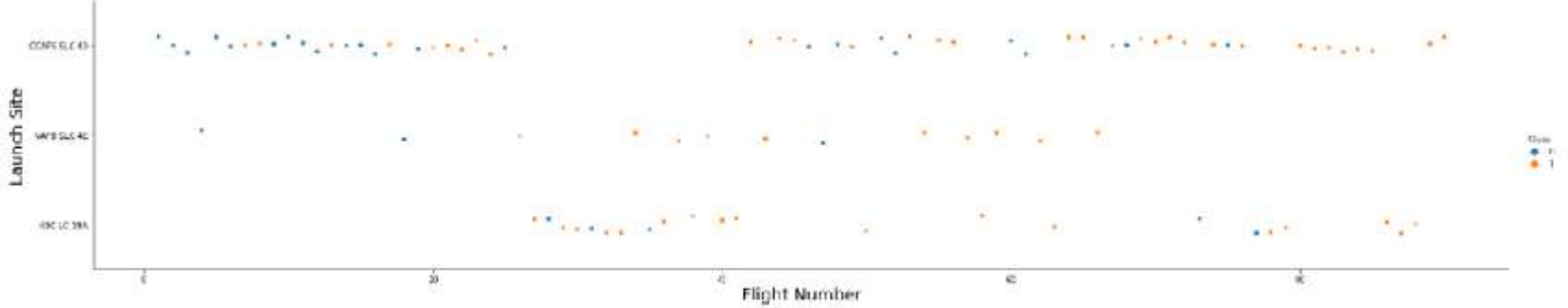
Insights drawn from EDA

Flight Number vs. Launch Site

- The plot led us to the conclusion that a launch site's success rate increases with the size of the flight quantity. Number vs. Launch Site.



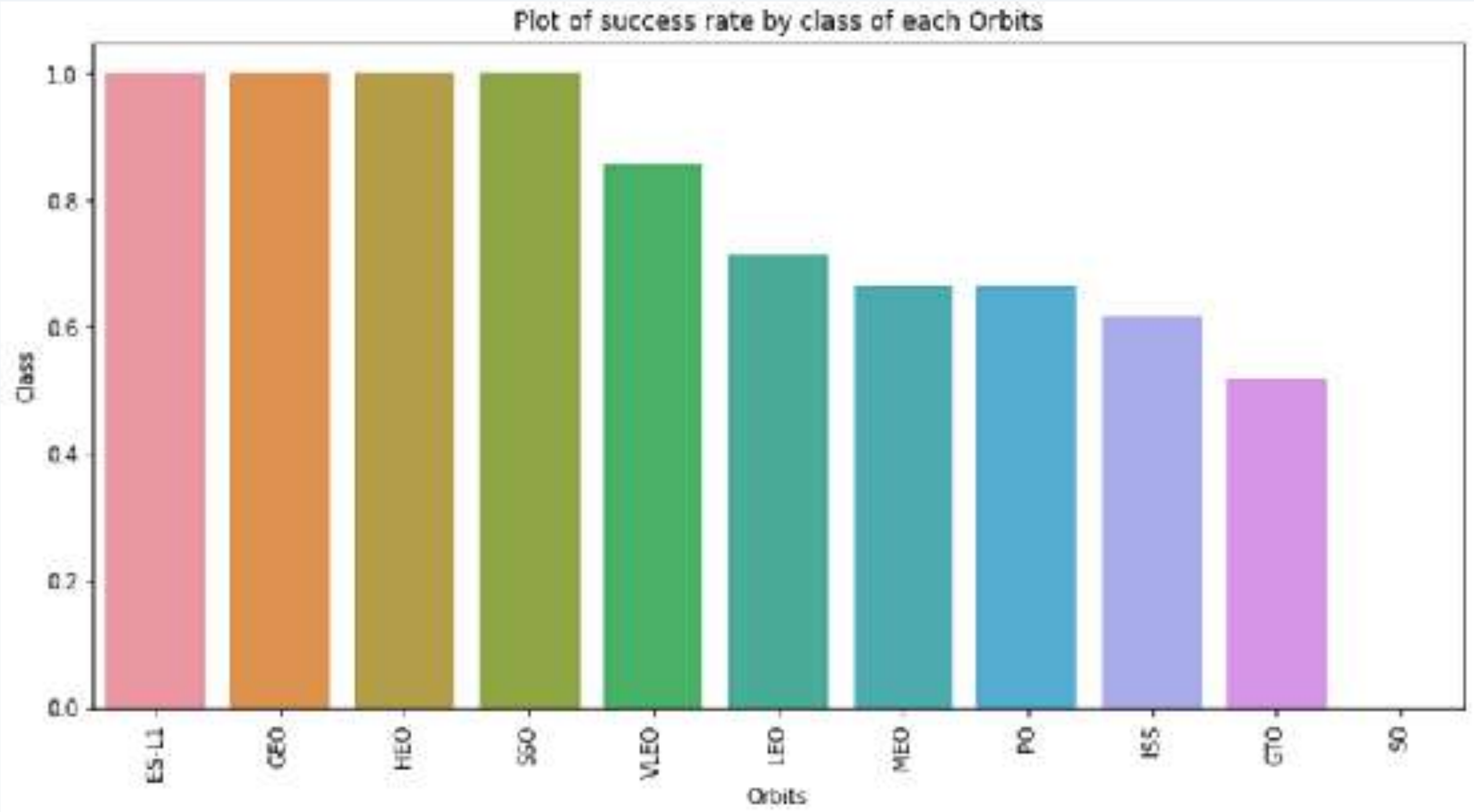
Payload vs. Launch Site



- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket

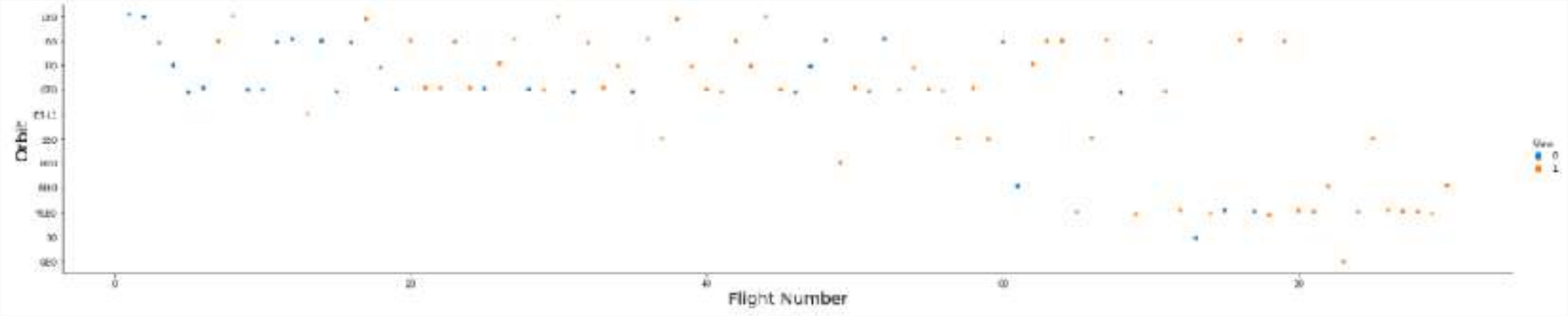
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



Flight Number vs. Orbit Type

- The plot of the Flight Number versus Orbit type is shown below. We note that success in the LEO orbit is correlated with the number of flights, however there is no correlation between the number of flights and the GTO orbit.

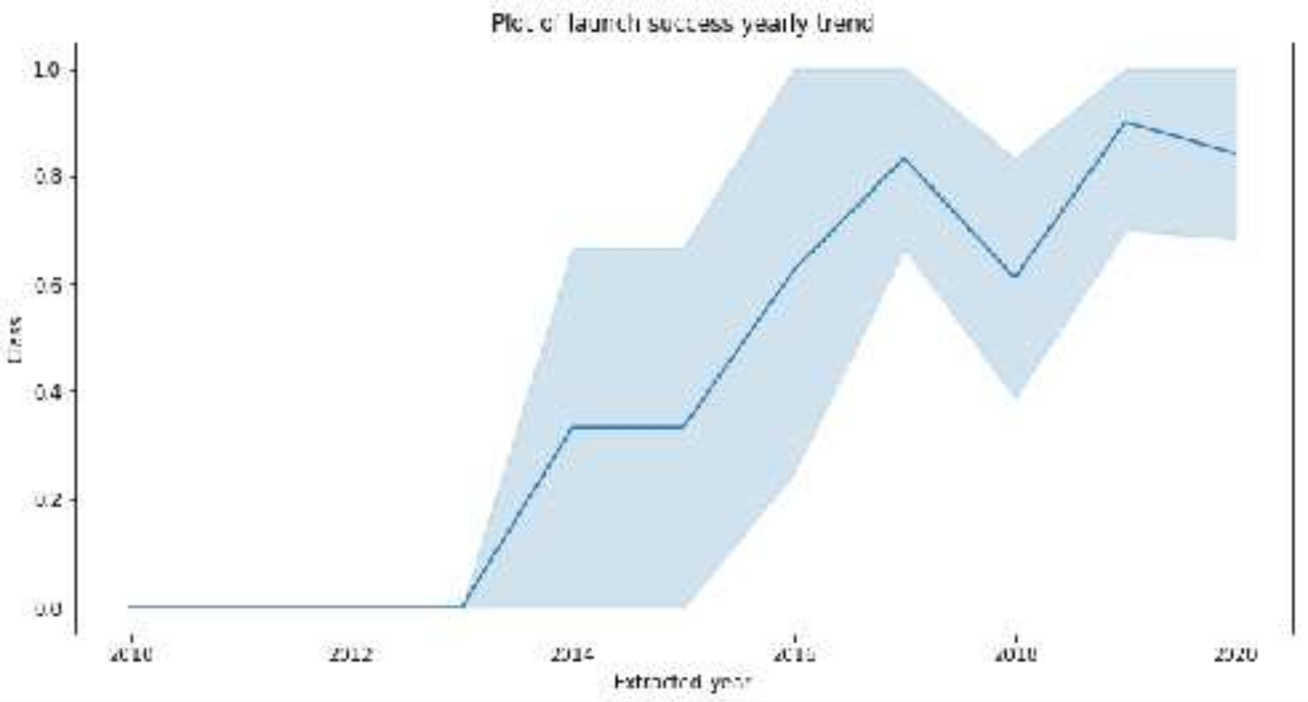


Payload vs. Orbit Type

- We can see that successful landings with heavier payloads tend to occur more frequently in PO, LEO, and ISS orbits.

Launch Success Yearly Trend

- The plot reveals that the success rate has been rising since 2013 and will continue to do so until 2020.



All Launch Site Names

- To display just distinct launch sites from the SpaceX data, we utilized the keyword DISTINCT.

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

```
pd.read_sql("select * from spacexdata where Launch_Site like 'CCA%' limit 5", conn)
```

	index	Date	Time_(UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	0	2010-06-04 00:00:00	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	1	2010-12-08 00:00:00	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2	2012-05-22 00:00:00	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	3	2012-10-08 00:00:00	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	4	2013-03-01 00:00:00	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- We performed the previous query to show 5 records for launch sites that start with "CCA."

Total Payload Mass

- Using the following query, we determined that NASA's boosters carried a total of 45596 kilograms of payload.

- ```
  pd.read_sql("select sum(PAYLOAD_MASS__KG_) from spacexdata where Customer='NASA (CRS)'", conn)
  ```

	sum(PAYLOAD_MASS__KG_)
0	45596
  ```

Average Payload Mass by F9 v1.1

- The average mass of the payload that booster version F9 v1.1 can carry was calculated to be 2928.4.Present your query result with a short explanation here

```
pd.read_sql("select avg(PAYLOAD_MASS__KG_) from spacexdata where Booster_Version='F9 v1.1'", conn)
```

	avg(PAYLOAD_MASS__KG_)
0	2928.4

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

```
pd.read_sql("select min(Date) from spacexdata where Landing__Outcome='Success (ground pad)'", conn)
```

	min(Date)
0	2015-12-22 00:00:00

Successful Drone Ship Landing with Payload between 4000 and 6000

- In order to find boosters that have successfully landed on drone ships, we employed the WHERE clause. We then used the AND condition to identify successful landings with payload masses larger than 4,000 but less than 6,000.

```
pd.read_sql("select distinct Booster_Version from spacexdata where Landing__Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000", conn)
```

	Booster_Version
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- To filter MissionOutcome for a success or a failure.

```
pd.read_sql("select substr(Mission_Outcome,1,7) as Mission_Outcome, count(*) from spacexdata  group by 1", conn)
```

	Mission_Outcome	count(*)
0	Failure	1
1	Success	100

Boosters Carried Maximum Payload

- Using a subquery in the WHERE clause and the MAX() method, we were able to identify the booster that had carried the most payload.

```
pd.read_sql("select distinct Booster_Version from spacexdata where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacexdata)", conn)
```

	Booster_Version
0	F9 B5 B1048.4
1	F9 B5 B1049.4
2	F9 B5 B1051.3
3	F9 B5 B1056.4
4	F9 B5 B1048.5
5	F9 B5 B1051.4
6	F9 B5 B1049.5
7	F9 B5 B1060.2
8	F9 B5 B1058.3
9	F9 B5 B1051.6
10	F9 B5 B1060.3
11	F9 B5 B1049.7

2015 Launch Records

- In order to filter for failure landing outcomes in drone ship, their booster versions, and launch site names for the year 2015, we employed the WHERE clause

```
pd.read_sql("select distinct Landing__Outcome, Booster_Version, Launch_Site from spacexdata where Landing__Outcome='Failure (drone ship)'", conn)
```

	Landing__Outcome	Booster_Version	Launch_Site
0	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
1	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
2	Failure (drone ship)	F9 v1.1 B1017	VAFB SLC-4E
3	Failure (drone ship)	F9 FT B1020	CCAFS LC-40
4	Failure (drone ship)	F9 FT B1024	CCAFS LC-40

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

- 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 we used the WHERE clause and a groupby

```
pd.read_sql("select Landing__Outcome, count(*) from spacexdata where Date between '2011-06-04' and '2017-03-20' group by Landing__Outcome order by 2 desc", conn)
```

	Landing__Outcome	count(*)
0	No attempt	10
1	Success (drone ship)	5
2	Failure (drone ship)	5
3	Success (ground pad)	3
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1

Section 3

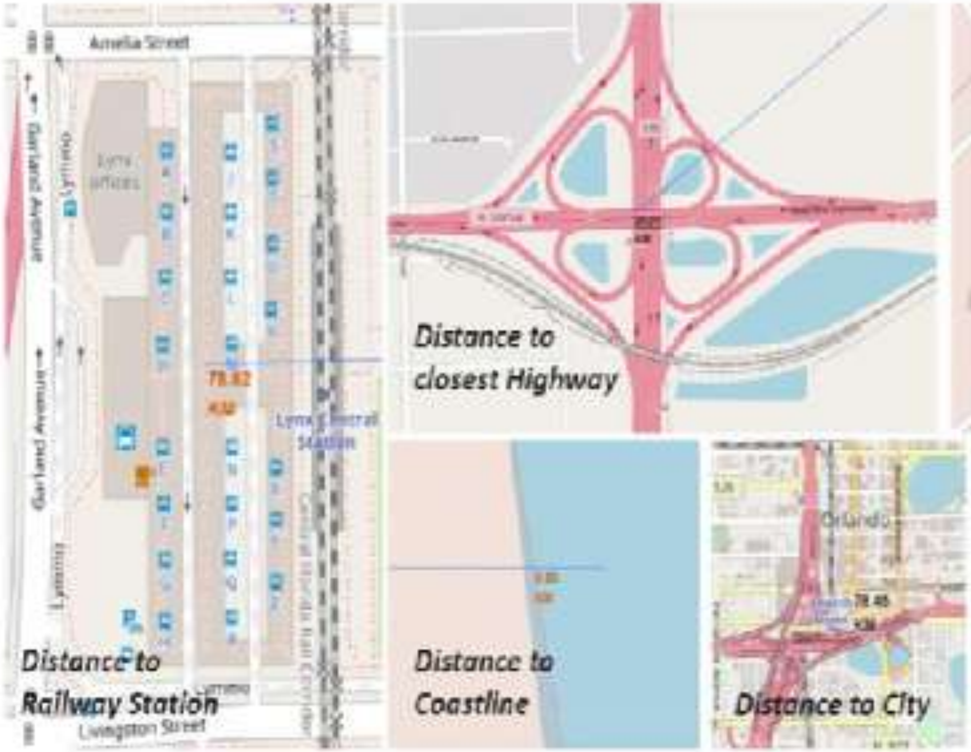
Launch Sites Proximities Analysis

All launch sites global map markers

- The launch sites are in Florida and California

Markers for launch sites

Distances to landmarks



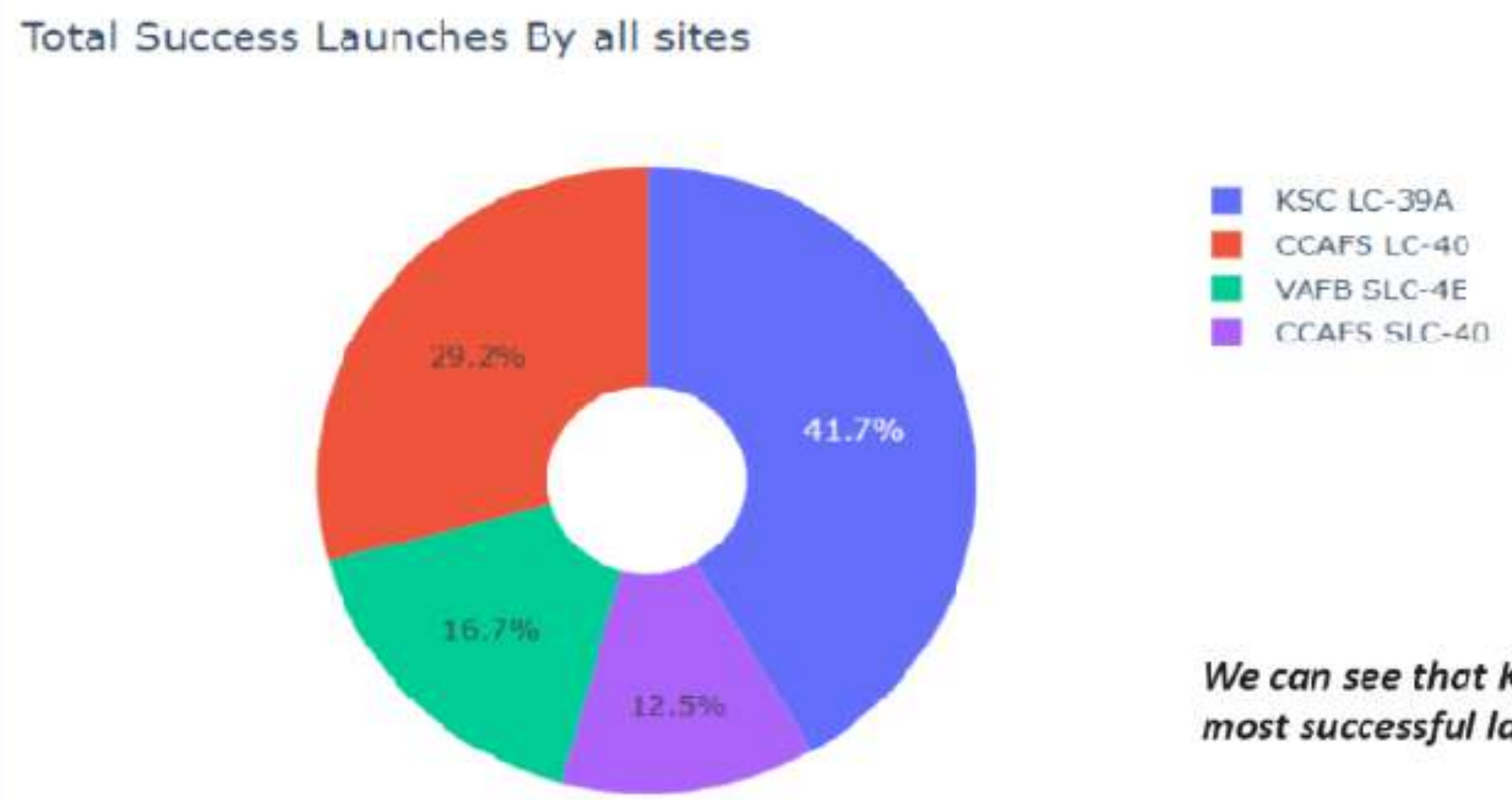
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

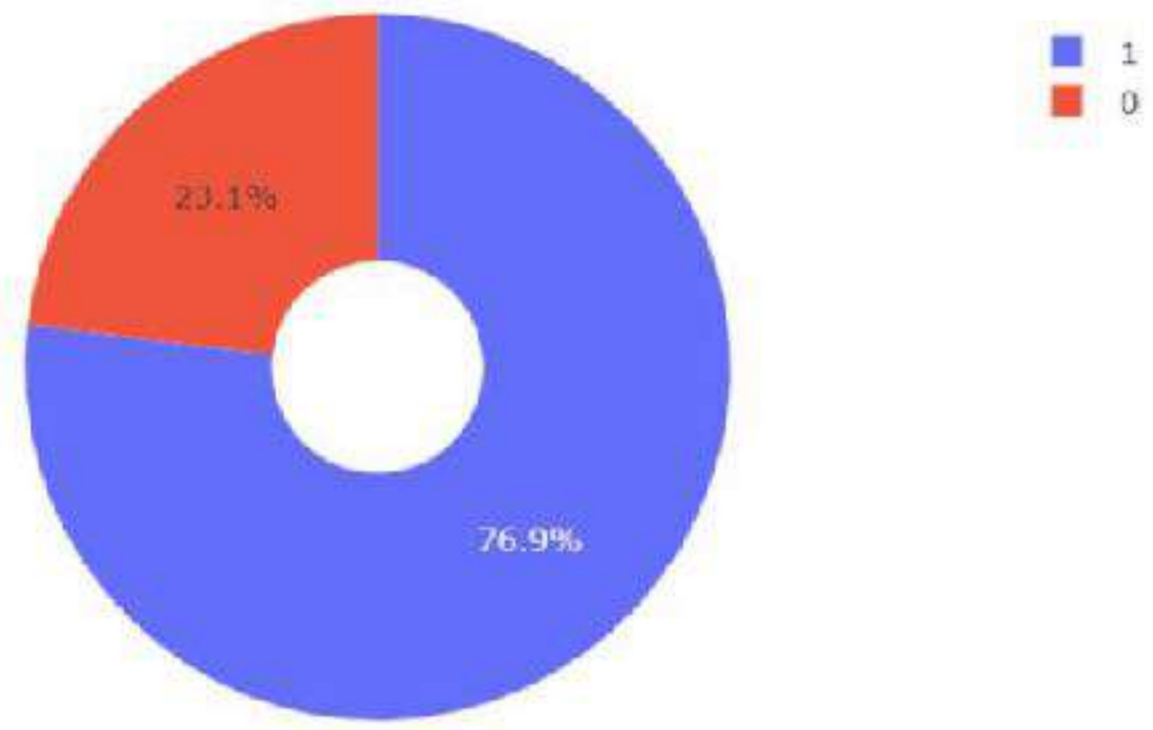
Success rate of every launch site

•



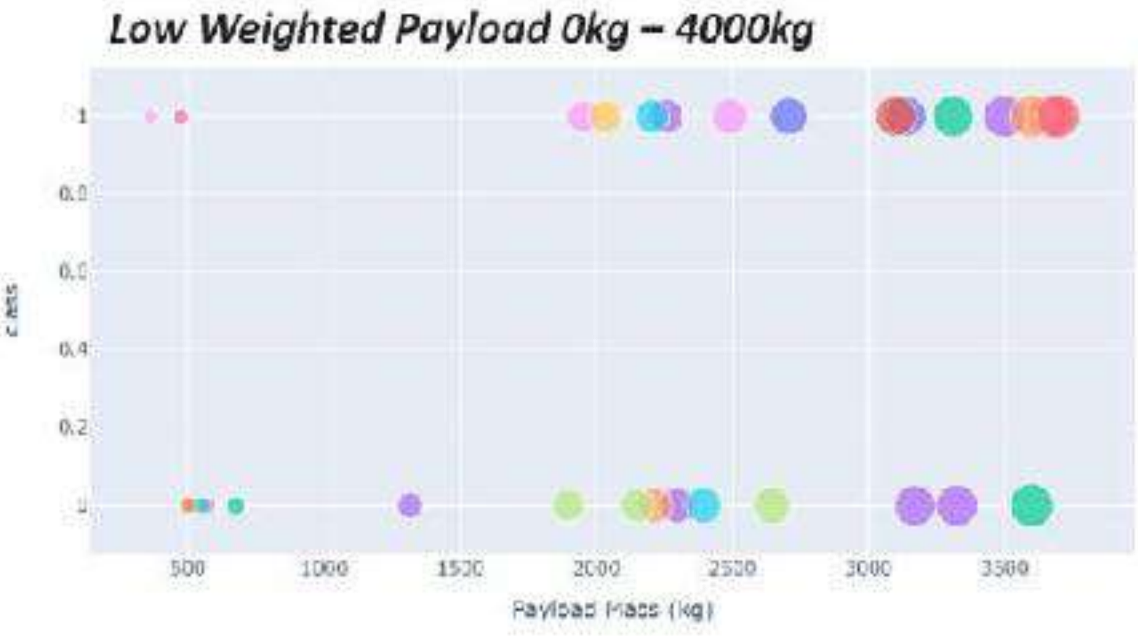
We can see that KSC LC-39A had the most successful launches from all the sites

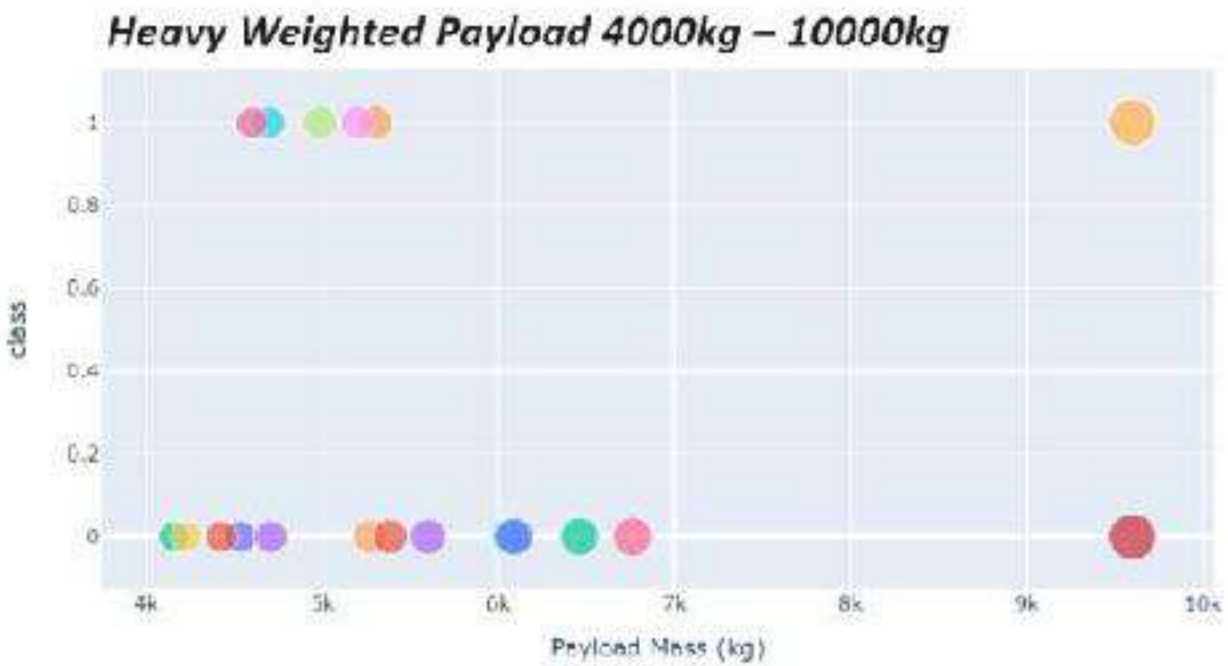
Launch site with the highest success rate



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Comparing low weight payloads





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Scores on test data for each method

- Logistic Regression: 0.944
- SVM: 0.944
- Decision Tree: 0.888
- KNN: 0.888

Conclusion: Logistic Regression and SVM deliver the best performance on test data.

Confusion Matrix

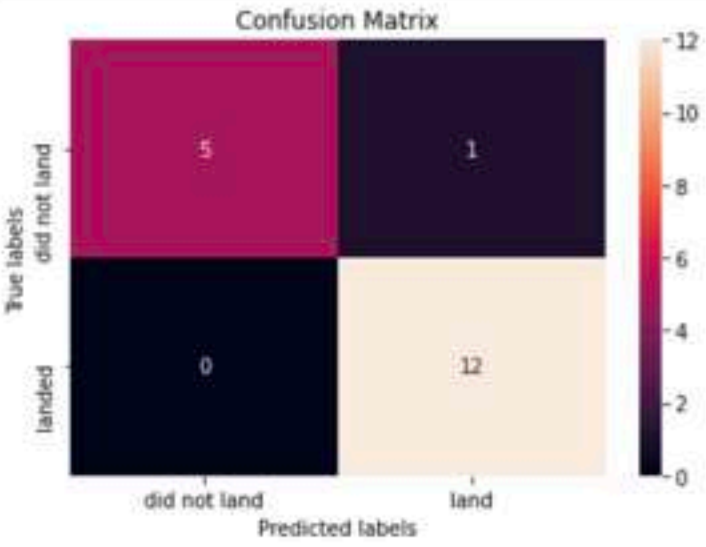
Calculate the accuracy on the test data using the method `score`:

```
print('score on train data: ', lr_cv.score(X_train, Y_train))
print('score on test data : ', lr_cv.score(X_test, Y_test))
```

```
score on train data:  0.875
score on test data :  0.9444444444444444
```

Lets look at the confusion matrix:

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

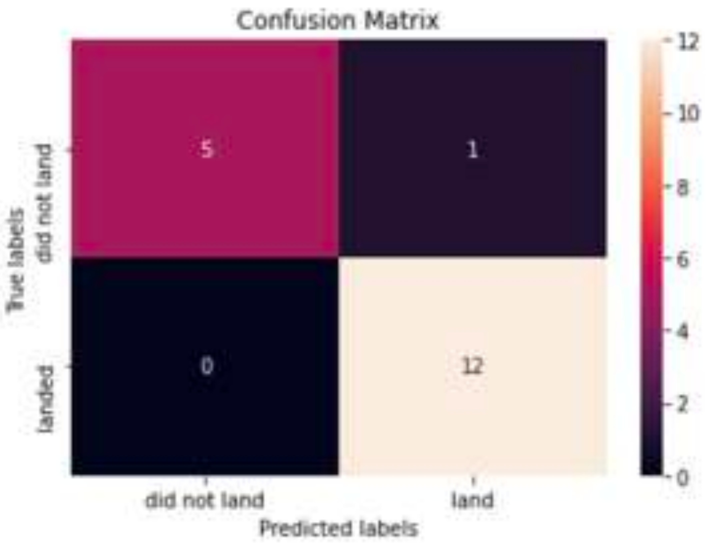



```
print('score on train data: ', svm_cv.score(X_train, Y_train))
print('score on test data : ', svm_cv.score(X_test, Y_test))
```

```
score on train data:  0.8611111111111112
score on test data :  0.9444444444444444
```

We can plot the confusion matrix

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



Conclusions

- The success rate at a launch site increases with the size of the flight quantity.
- Logistic Regression and SVM are the best machine learning method for this task.
- The launch success rate increased from 2013 to 2020.
- The highest success rate was in the ES-L1, GEO, HEO, SSO, and VLEO orbits.
- Of all the sites, KSC LC-39A had the most successful launches.

Thank you!