IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Marcus Kim
6/11/2024

Outline

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

Executive Summary

- Methodology summary: This project covers the entire data analysis process. In order, this followed data collection, API webscraping, data wrangling, data exploration, data visualization, and statistical analysis. When applied to SpaceX rocket launch data, I was able to clean SpaceX's existing data, resulting in my ability to visualize existing, relevant components of that data. Additionally, we could also predict the factors that were most important to successful launches. These include orbit type, launch site, payload mass, and flight number.
- Results summary: Certain orbit types had a higher success rate, while certain launch sites had a higher success rate. Additionally, decision tree classification analysis accurately predicted future launches based on these variables.

Introduction

This project utilizes SpaceX's data and API. To do so, I will be collecting their data online using Python. Additionally, I will be exploring the data utilizing Python and SQL to better understand the variables and pre-existing facts regarding SpaceX's launch history. After visualizing this data to better understand it, I will apply statistical, classification analysis to determine which variables may accurately predict successful future launches. So, I will be exploring the variables that SpaceX displays in their launch data, and analyze their impact on successful launches so less resources are wasted.

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

- Steps for completion
    1. Request data from SpaceXAPI
    2. Assign .json data to a pandas dataframe variable
    3. Filter the data for Falcon 9 launches only
    4. Handle any null values
    5. Send subset to .csv
- Link to notebook in Github → [Here]

Relevant Collection Code (Step 1 and 2)

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

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

Check the content of the response

```
print(response.content)
```

```
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())
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

Data Collection - Scraping

- Steps for completion
    1. Request data from API to access Wikipedia table
    2. Create BeautifulSoup object
    3. Extract column names from Wikipedia table
    4. Assign values to a pandas dataframe for further use
- Link to notebook in Github → [Here]()

Relevant Webscraping flowchart (code):

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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html.parser')
```

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = []

tables = soup.find_all('table')
for table in tables:
        html_tables.append(table)
```

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

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

Data Wrangling

- How data were processed?
    1. First, data was extracted from a csv using .read_csv()
    2. Understand data using .isnull() and .dtypes
        1. Exposed # of null values per column and datatype per column
    3. Understand values of data
        1. Launches per site
        2. Launches per orbit
        3. Mission outcomes per orbit
        4. Outcome of each instance
- Link to notebook in Github → Here

EDA with Data Visualization

- Charts used and why?
  1. Scatterplot
     1. Visualize relationship between numerical independent and dependent variables
     2. Visualize relationship between categorical independent and numerical dependent variables
  2. Bar chart
     1. Visualize and compare quantitative values of numerical variables against 9 categorical variables
  3. Line chart
     1. Visualize success over time
- Link to notebook in Github → [Here]

EDA with SQL

- SQL queries performed
  - Find distinct launch sites
  - Find first successful landing outcome
  - Find successful boosters
  - Find total number of successful and failure missions
  - More @ “Insights drawn from EDA”
- Link to notebook in Github → Here

Build an Interactive Map with Folium

- Map objects added to Folium Map
  1. Marked NASA Johnson Space Center to initialize map
  2. Marked SpaceX launch sites for further visualization
  3. Marked successful and failed launches at each launch site
  4. Marked marker cluster to better visualize range of launch sites
  5. Created mouse position to allow visualization of latitude and longitude for further analysis
  6. Marked distance to nearest coastline to see where launches go to.
- Link to notebook in Github → Here

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- You need present your model development process using key phrases and flowchart
- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

Results

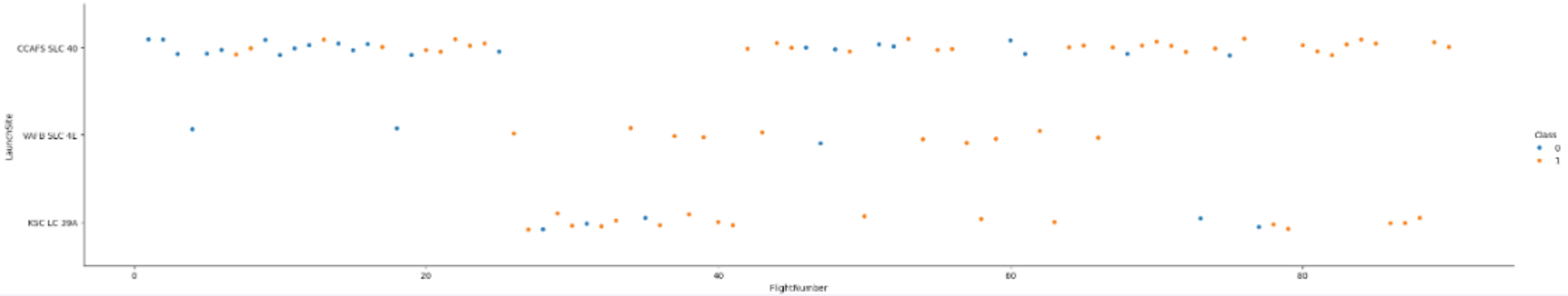
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

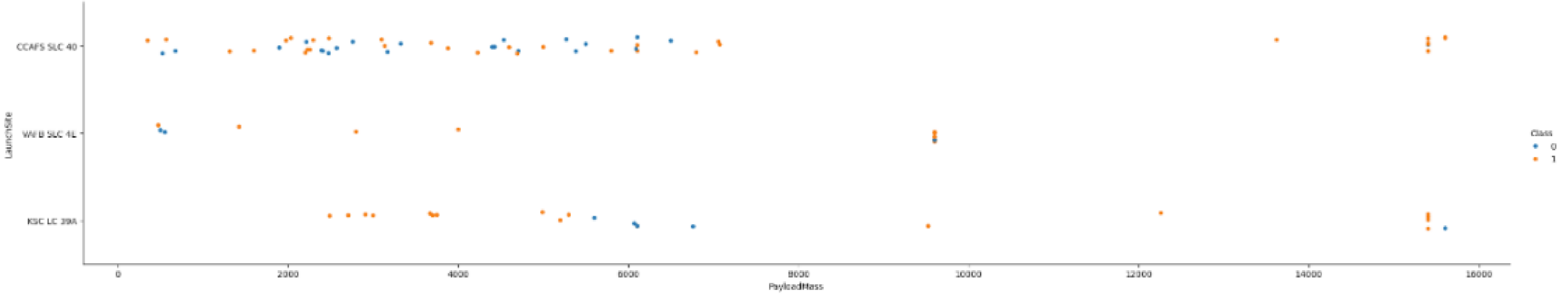
```python
### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.show()
```



Explanation: This code depicts Launch site along the y-axis, and the Flight number on the x-axis. As flight number increases, the success increases (shown by orange, 1)

Payload vs. Launch Site

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.show()
```



Explanation: This scatterplot depicts how certain launch sites (like VAFB SLC) does not launch rockets beyond the 10000 kg payload mass. Additionally, it depicts a high success rate at higher payload masses

Success Rate vs. Orbit Type

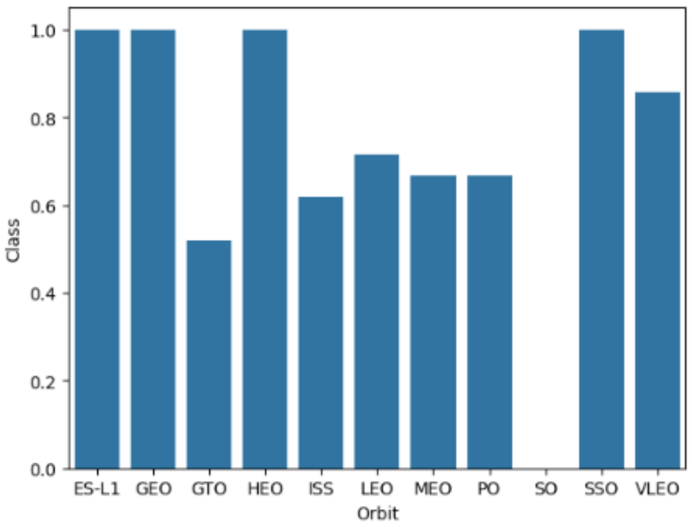
- This chart depicts the code and the resulting bar chart
- This visualizes the success rates of 12 different orbits, regardless of launch site

```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_set = df.groupby('Orbit')['Class'].mean()
orbit_set
```

```
Orbit
ES-L1     1.000000
GEO       1.000000
GTO       0.518519
HEO       1.000000
ISS       0.619048
LEO       0.714286
MEO       0.666667
PO        0.666667
SO        0.000000
SSO       1.000000
VLEO      0.857143
Name: Class, dtype: float64
```

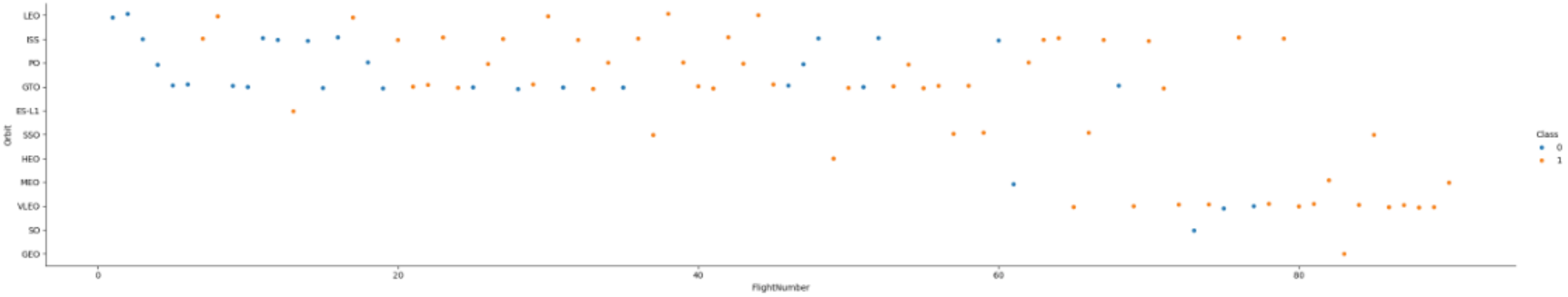
Analyze the ploted bar chart try to find which orbits have high sucess rate.

```
### TASK  4: Visualize the relationship between FlightNumber and Orbit type
sns.barplot(data=orbit_set)
plt.ylabel=("Success Rate")
plt.show()
```



Flight Number vs. Orbit Type

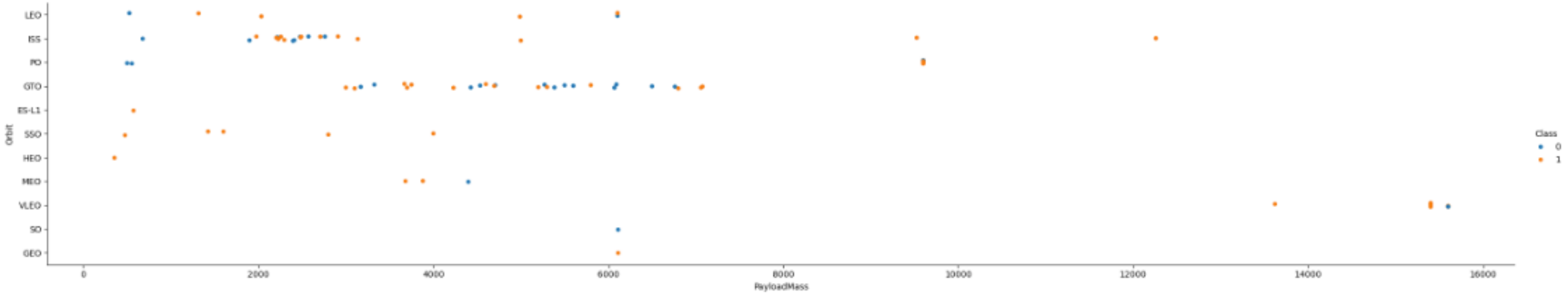
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x='FlightNumber',y='Orbit',hue='Class',data=df, aspect = 5)
plt.show()
```



This plot shows flight number and success/fail against each orbit type. Some insights are that the LEO orbit experiences success with increase flight number, while there is no correlation for GTO.

Payload vs. Orbit Type

```python
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(x='PayloadMass',y='Orbit',hue='Class',data=df, aspect = 5)
plt.show()
```

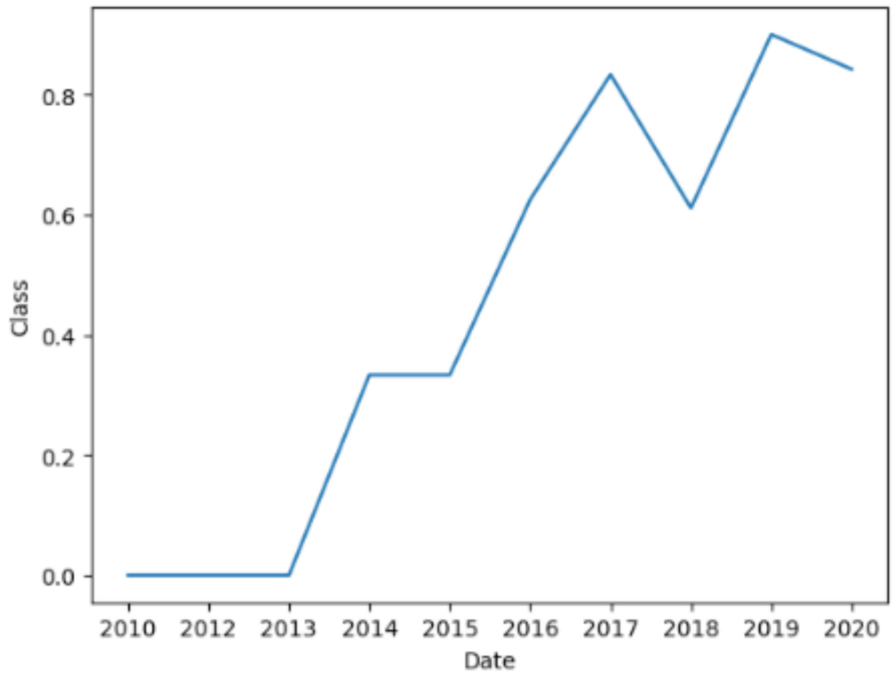


This plot shows payload mass vs. orbit type. It reveals that with heavy payloads, the successful landing rate is more for LEO, Polar, and ISS. For GTO, we can find no correlation. Meanwhile, there are a few number of launches for SO and GEO.

Launch Success Yearly Trend

- This chart simply depicts success rate over time. Although there were no successes before 2013, the success rate has increased since 2013. However, there was a noticeable drop in 2018 in success rates.

```python
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
avg_succ = df.groupby('Date')['Class'].mean()
sns.lineplot(data=avg_succ)
plt.show()
```



All Launch Site Names

- The SQL code below depicts the unique launch sites from SpaceX

```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;

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

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

Launch Site Names Begin with 'CCA'

- The below code depicts Launch sites that begin with, or are similar to, ”CCA”

```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE '%CCA%' LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Task 3

Total Payload Mass

- The below code displays the total payload mass carried by boosters launched by NASA (CRS)

Task 3

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

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)';
```

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

SUM("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- The below code depicts the average payload mass for a booster version “FG v1.1”

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Booster_Version" = "F9 v1.1";
```

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

AVG("PAYLOAD_MASS__KG_")
2928.4

First Successful Ground Landing Date

- This code shows that the first successful ground landing data was 06/04/2010.

```
%sql SELECT MIN(DATE) FROM SPACEXTABLE WHERE "Mission_Outcome" = "Success";
```

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

MIN(DATE)
2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

- The code below depicts the booster versions that had successful drone ship landings and had a payload between 4000 and 6000

```
%sql SELECT Booster_Version from SPACEXTBL WHERE Mission_Outcome = "Success" AND Landing_Outcome LIKE "%drone%" AND PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

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

Booster_Version
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- The code below depicts the count of mission outcomes for successful and failed missions

```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

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

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

Boosters Carried Maximum Payload

- The below code shows the boosters that carried the maximum payload

```
%sql select Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

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

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

2015 Launch Records

- The code below shows failed drone ship landing outcomes, booster versions, and launch site during the months in year 2015

```
%sql SELECT substr(Date,6,2) as "Month Number", substr(Date,0,5) as "Year", "Booster_Version","Launch_Site" FROM SPACEXTABLE WHERE substr(Date,0,5) = '2015' and "Landing_Outcome"
```

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

Month Number	Year	Booster_Version	Launch_Site
01	2015	F9 v1.1 B1012	CCAFS LC-40
04	2015	F9 v1.1 B1015	CCAFS LC-40

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

```
%%sql SELECT "LANDING_OUTCOME", COUNT(*) as 'COUNT' FROM SPACEXTBL

WHERE substr(Date,1,4) || substr(Date,6,2) || substr(Date,9,2)

between '20100604' and '20170320' GROUP BY "Landing_Outcome" ORDER BY "COUNT" DESC;
```

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

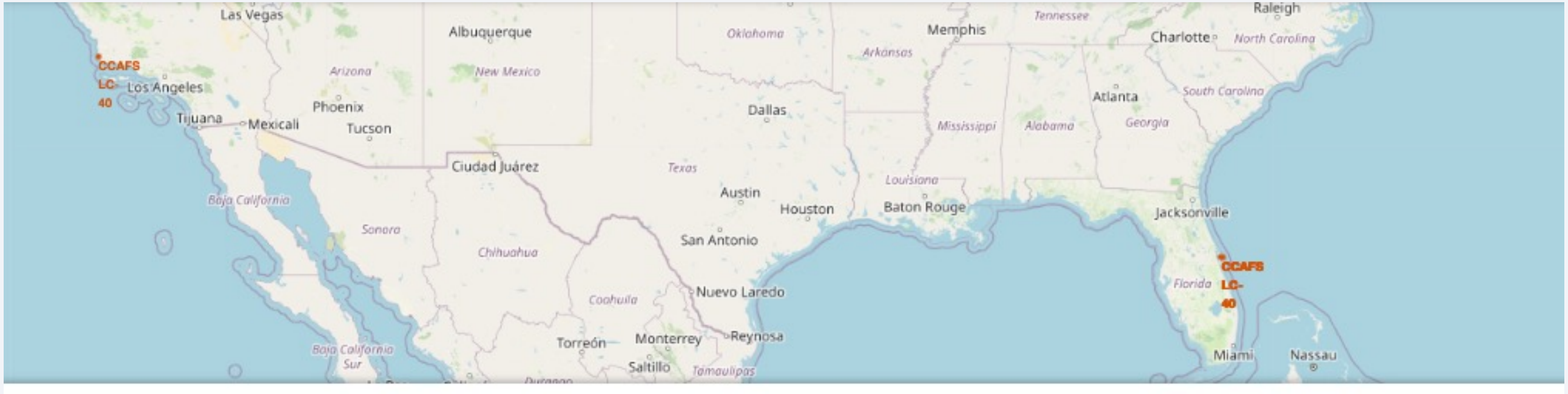
Landing_Outcome	COUNT
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

- This code depicts the count of landing outcomes between 2010-06-04 and 2017-03-20 in DESCENDING order.

Section 3

Launch Sites Proximities Analysis

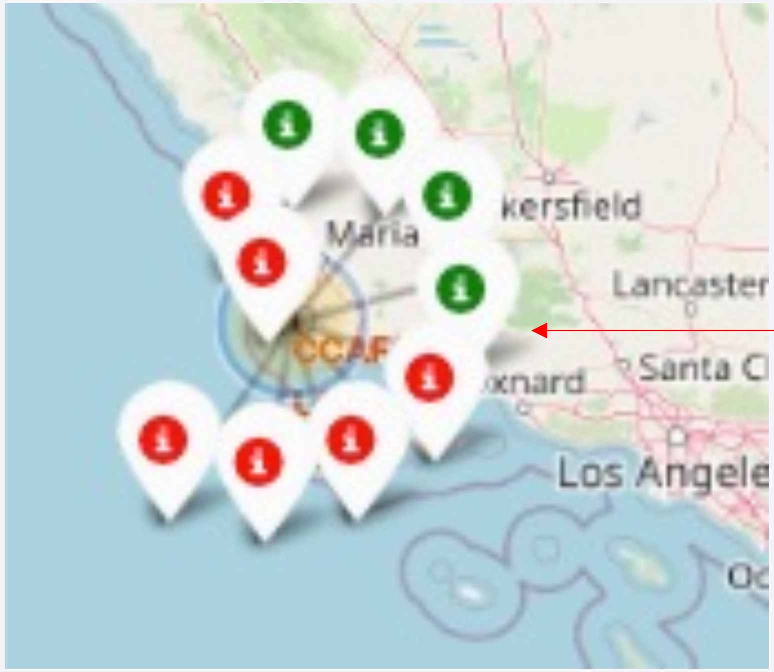
Map 1: Launch Site Locations

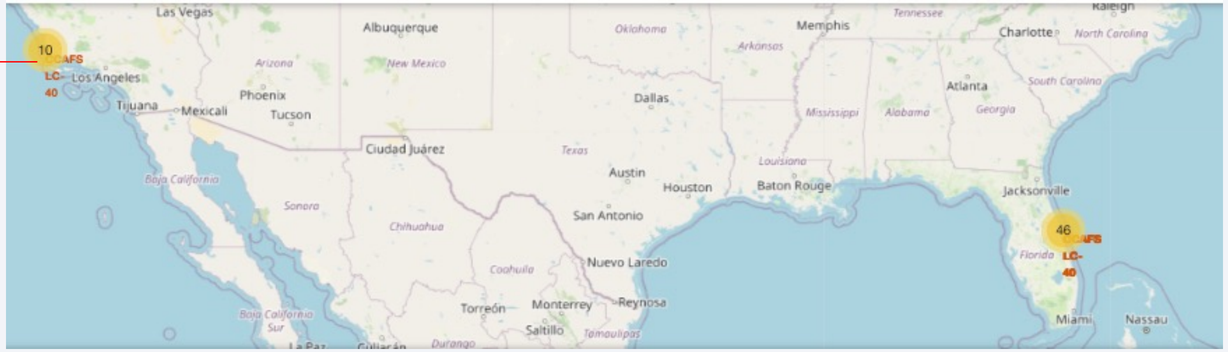


Explanation: This depicts SpaceX's 4 launch sites

1. CCAFS LC-40 @ coordinates (28.562302, -80.577356)
2. CCAFS SLC-40 @ coordinates (28.563197, -80.576820)
3. KSC LC-39A @ coordinates (28.573255, -80.646895)
4. VAFB SLC-4E @ coordinates (34.632834, -120.610745)

Map 2: Marker Cluster at each Launch Site





- This map provides depth to the previous graph. It shows the number of launches in a colored area for better visibility.
- See the red arrow above

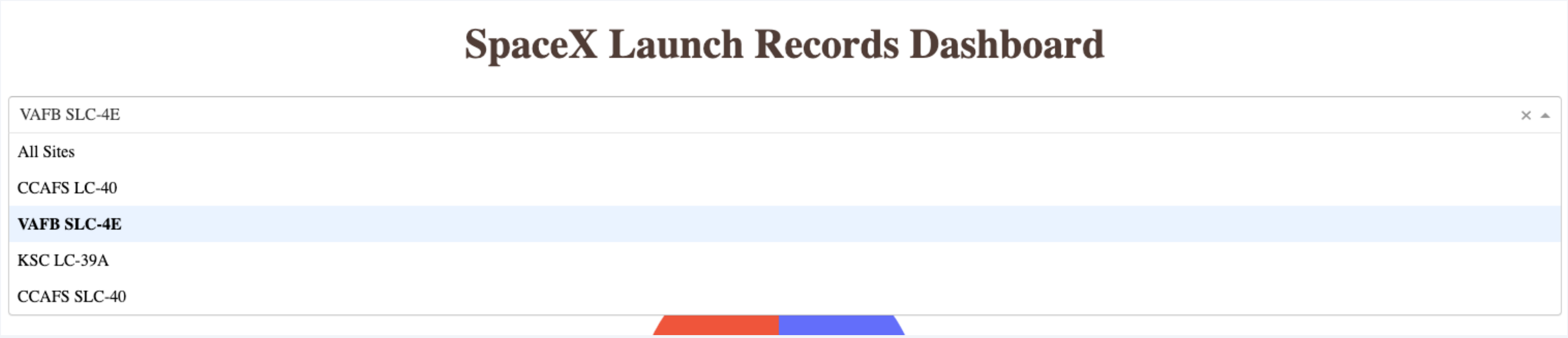
Map 3: Proximity to Coastline

The blue line depicts the distance between the coastline and a launch site

Section 4

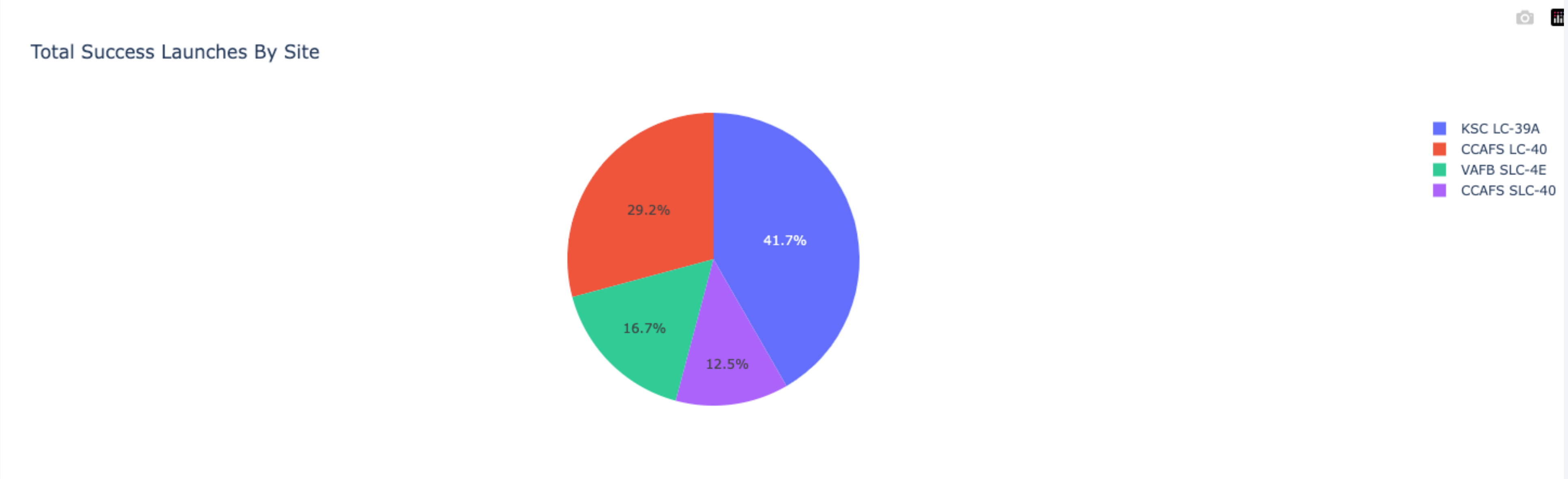
Build a Dashboard with Plotly Dash

Dashboard Dropdown and Title



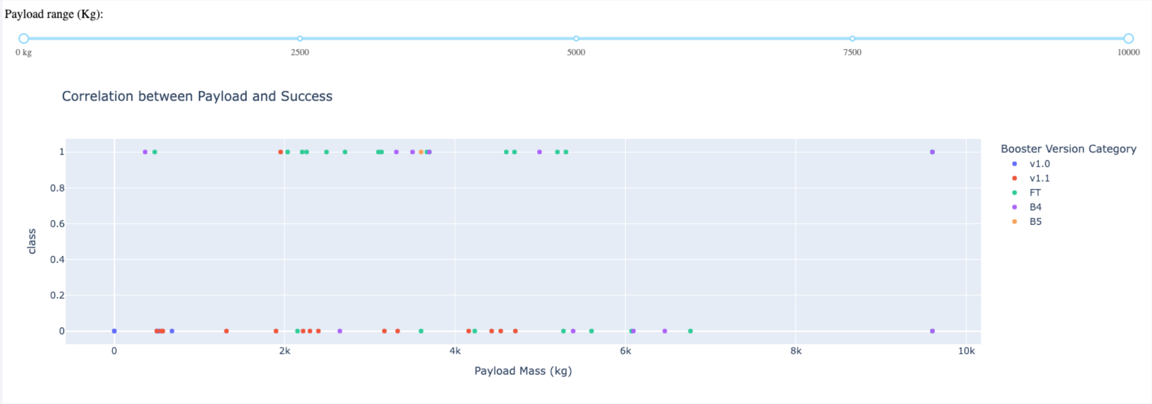
This depicts the dropdown menu

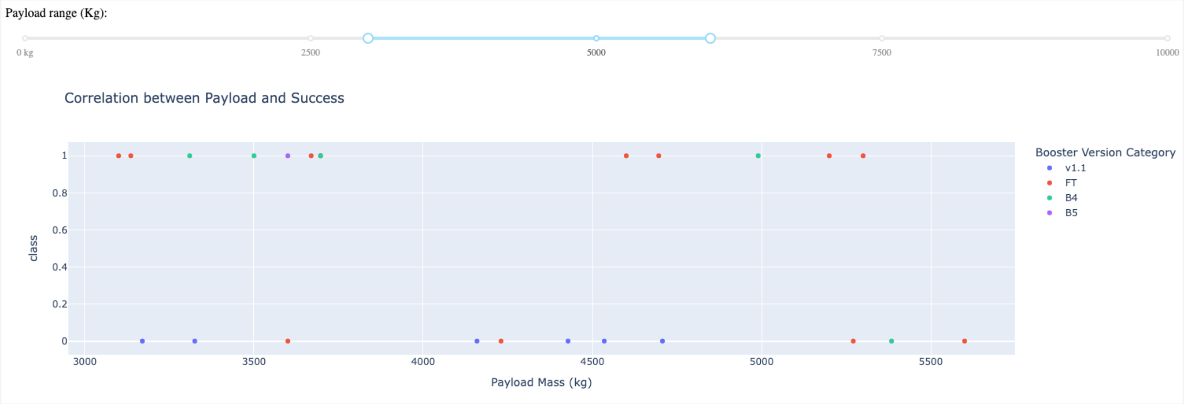
Most successful launch sites



This depicts the percentage of successful launches by site. KSC LC-39A in blue has the highest percentage, while CCAFS SLC-40 in purple has the lowest percentage

Payload vs. Success at different intervals





This depicts the correlation between Payload and Success between the min and max values of 0 and 10000 kg.

This depicts the correlation between Payload and Success between the selected values of 3000 and 6000 kg.

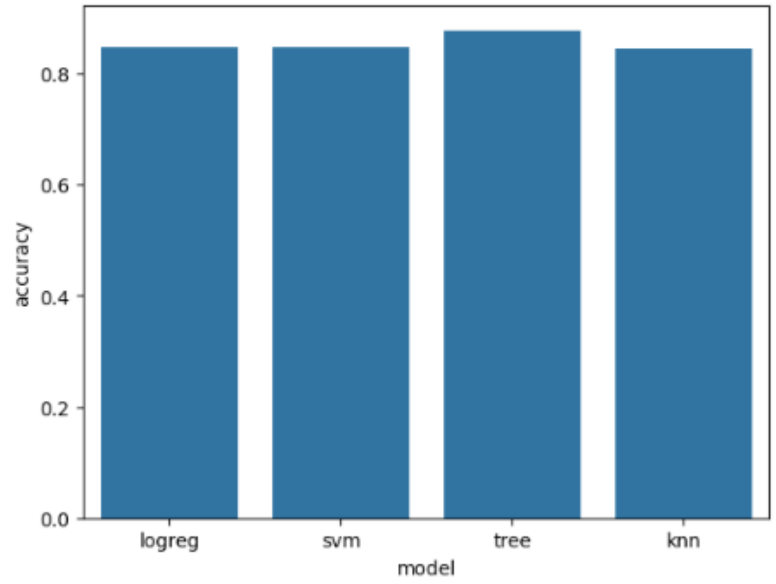
Section 5

Predictive Analysis (Classification)

Classification Accuracy

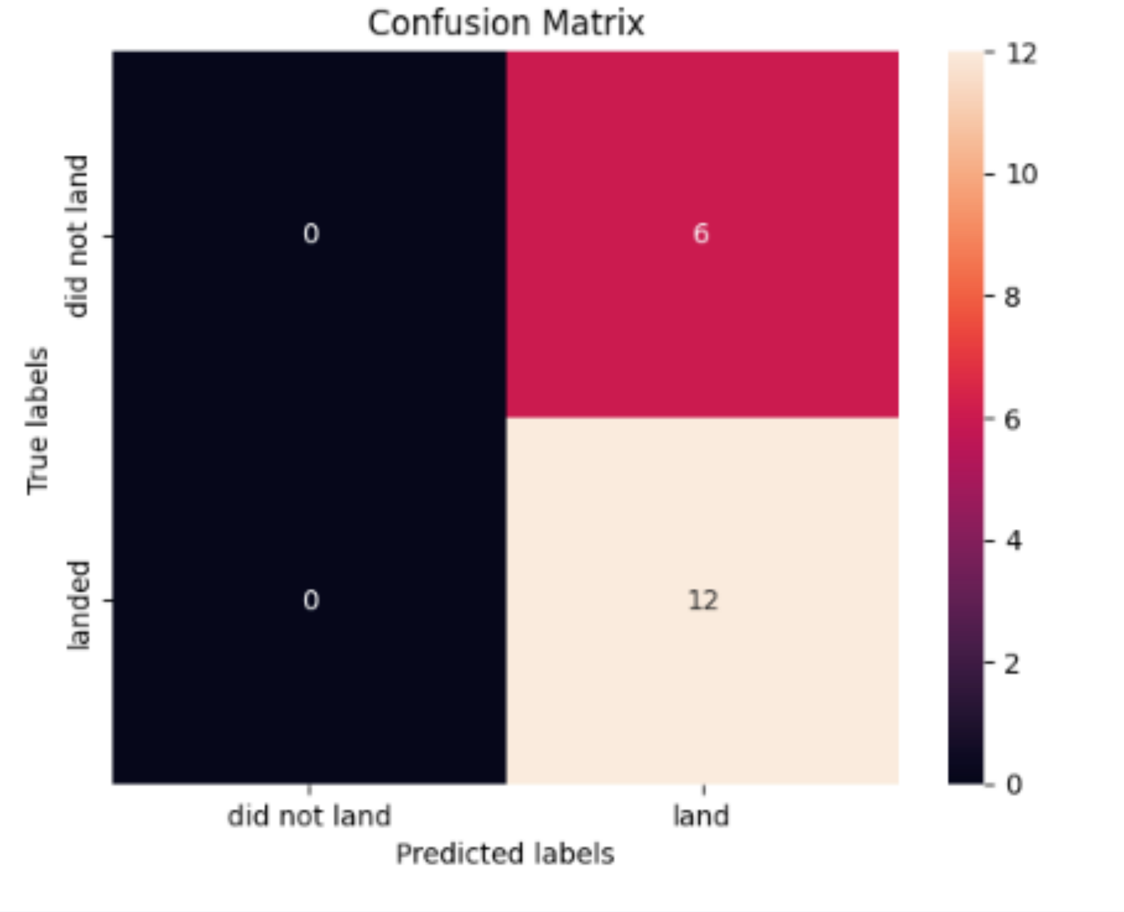
- This model depicts the accuracy for logistic regression, support vector machines, decisions tree, and k-nearest neighbor
- This shows that decision tree classification provided the highest accuracy

```
x_labels = ["logreg","svm","tree","knn"]
y_labels = [logreg_cv.best_score_,svm_cv.best_score_,tree_cv.best_score_,knn_cv.best_score_]

data = {'model':x_labels, 'accuracy':y_labels}
df = pd.DataFrame(data)
df.set_index('model', inplace=True)

sns.barplot(x='model',y='accuracy',data=df)
plt.show()
```



Confusion Matrix

- This confusion matrix depicts the decision tree analysis
- This model accurately predicted true negatives and true positives.
- However, it inaccurately predicted flights that did not actually land as those that landed. However, this is not a worrisome problem
- It is more important to predict false positives, which it did correctly.



Conclusions

- The most notable points to be made regard orbit type and launch site
  - Orbit type:
    - ES-L1, GEO, and SSO resulted in the highest success rates
    - SO had a 0% success rate
  - Launch Site:
    - KSC LC-39A had the highest success rate
    - CCAFS SLC-40 had the lowest success rate

Appendix

dataset_part_1

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad
1	2010-06-04	Falcon 9	6123.547647058820	LEO	CCSFS SLC 40	None None	1	FALSE	FALSE	FALSE	
2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	FALSE	FALSE	FALSE	
3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	FALSE	FALSE	FALSE	
4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	FALSE	FALSE	FALSE	
5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	FALSE	FALSE	FALSE	
6	2014-01-06	Falcon 9	3325.0	GTO	CCSFS SLC 40	None None	1	FALSE	FALSE	FALSE	
7	2014-04-18	Falcon 9	2296.0	ISS	CCSFS SLC 40	True Ocean	1	FALSE	FALSE	TRUE	
8	2014-07-14	Falcon 9	1316.0	LEO	CCSFS SLC 40	True Ocean	1	FALSE	FALSE	TRUE	
9	2014-08-05	Falcon 9	4535.0	GTO	CCSFS SLC 40	None None	1	FALSE	FALSE	FALSE	
10	2014-09-07	Falcon 9	4428.0	GTO	CCSFS SLC 40	None None	1	FALSE	FALSE	FALSE	
11	2014-09-21	Falcon 9	2216.0	ISS	CCSFS SLC 40	False Ocean	1	FALSE	FALSE	FALSE	
12	2015-01-10	Falcon 9	2395.0	ISS	CCSFS SLC 40	False ASDS	1	TRUE	FALSE	TRUE	5e9e3032383ecb761
13	2015-02-11	Falcon 9	570.0	ES-L1	CCSFS SLC 40	True Ocean	1	TRUE	FALSE	TRUE	
14	2015-04-14	Falcon 9	1898.0	ISS	CCSFS SLC 40	False ASDS	1	TRUE	FALSE	TRUE	5e9e3032383ecb761
15	2015-04-27	Falcon 9	4707.0	GTO	CCSFS SLC 40	None None	1	FALSE	FALSE	FALSE	
16	2015-06-28	Falcon 9	2477.0	ISS	CCSFS SLC 40	None ASDS	1	TRUE	FALSE	TRUE	5e9e3032383ecb6bb
17	2015-12-22	Falcon 9	2034.0	LEO	CCSFS SLC 40	True RTLS	1	TRUE	FALSE	TRUE	5e9e3032383ecb267
18	2016-01-17	Falcon 9	553.0	PO	VAFB SLC 4E	False ASDS	1	TRUE	FALSE	TRUE	5e9e3033383ecbb9e
19	2016-03-04	Falcon 9	5271.0	GTO	CCSFS SLC 40	False ASDS	1	TRUE	FALSE	TRUE	5e9e3032383ecb6bb
20	2016-04-08	Falcon 9	3136.0	ISS	CCSFS SLC 40	True ASDS	1	TRUE	FALSE	TRUE	5e9e3032383ecb6bb
21	2016-05-06	Falcon 9	4696.0	GTO	CCSFS SLC 40	True ASDS	1	TRUE	FALSE	TRUE	5e9e3032383ecb6bb
22	2016-05-27	Falcon 9	3100.0	GTO	CCSFS SLC 40	True ASDS	1	TRUE	FALSE	TRUE	5e9e3032383ecb6bb
23	2016-07-18	Falcon 9	2257.0	ISS	CCSFS SLC 40	True RTLS	1	TRUE	FALSE	TRUE	5e9e3032383ecb267
24	2016-08-14	Falcon 9	4600.0	GTO	CCSFS SLC 40	True ASDS	1	TRUE	FALSE	TRUE	5e9e3032383ecb6bb
25	2016-09-01	Falcon 9	5500.0	GTO	CCSFS SLC 40	None ASDS	1	TRUE	FALSE	TRUE	5e9e3032383ecb6bb
26	2017-01-14	Falcon 9	9600.0	PO	VAFB SLC 4E	True ASDS	1	TRUE	FALSE	TRUE	5e9e3033383ecbb9e
27	2017-02-19	Falcon 9	2490.0	ISS	KSC LC 39A	True RTLS	1	TRUE	FALSE	TRUE	5e9e3032383ecb267

← Falcon 9 subset

Normalized data →

FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS
1.0	6104.959411764710	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2.0	525.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3.0	677.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
4.0	500.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
5.0	3170.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
6.0	3325.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
7.0	2296.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
8.0	1316.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
9.0	4535.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
10.0	4428.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
11.0	2216.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
12.0	2395.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
13.0	570.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
14.0	1898.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
15.0	4707.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
16.0	2477.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
17.0	2034.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
18.0	553.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
19.0	5271.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
20.0	3136.0	1.0	1.0	0.0	1.0	2.0	1.0	0.0	0.0	0.0	0.0	1.0
21.0	4696.0	1.0	1.0	0.0	1.0	2.0	0.0	0.0	0.0	1.0	0.0	0.0
22.0	3100.0	1.0	1.0	0.0	1.0	2.0	1.0	0.0	0.0	1.0	0.0	0.0
23.0	2257.0	1.0	1.0	0.0	1.0	2.0	1.0	0.0	0.0	0.0	0.0	1.0
24.0	4600.0	1.0	1.0	0.0	1.0	2.0	0.0	0.0	0.0	1.0	0.0	0.0
25.0	5500.0	1.0	1.0	0.0	1.0	3.0	0.0	0.0	0.0	1.0	0.0	0.0
26.0	9600.0	1.0	1.0	0.0	1.0	3.0	1.0	0.0	0.0	0.0	0.0	0.0
27.0	2490.0	1.0	1.0	0.0	1.0	3.0	1.0	0.0	0.0	0.0	0.0	1.0

Thank you!