Making effective trips to space with SpaceY

N.Dombrowski

21-10-2022

Outline

- Executive summary
- Introduction
- Methodology
- Results
- Discussion
- Conclusion
- Appendix

Executive summary

- Summary of methodology
  - ✓ Data collection via API and Webscraping
  - ✓ Data cleaning
  - ✓ Exploratory Data Analysis (EDA) by visualization
  - ✓ Exploratory Data Analysis (EDA) by SQL
  - ✓ Generating an interactive map with folium
  - ✓ Creating a dashbord with dash/plotly
  - ✓ Predicting successful launches with Machine Learning

- Summary of all results
  - ✓ Results for the EDA
  - ✓ Results for interactive map and dashbord
  - ✓ Results for machine learning approach

Introduction

- **Project background**

SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars, while other providers cost upward of 165 million dollars. SpaceX is able to provide this lower cost because they can reuse the first stage. The goal of this project is to investigate data on these launches with the main goal to determine if the first stage will land, whether we can determine the cost of a launch.

- Questions to be answered

What factors influence if we have a successul landing?
Can we build a model to predict whether the next landing will be successful or not?

Methodology

Data collection via API

1. **Create request to download data via API**

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

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

2. **Convert the results into a data frame**

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

3. **Clean the data:**
   - Only include Falcon 9 launches
   - Replace missing data with mean values

```
#Filter the data to only include Falcon 9 launches
data_falcon9 = data[data['BoosterVersion']=='Falcon 9']
data_falcon9.head()
```

```
# Calculate the mean value of PayloadMass column
mean_payloadMass = data_falcon9['PayloadMass'].mean()

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

#control
data_falcon9.isnull().sum()
```

Code can be found at: https://github.com/ndombrowski/IBM_DS_capstone/blob/main/Data_collection_via_API.ipynb

Data collection via webscraping

1. **Create request to download data via webscraping**

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

```python
r = requests.get(static_url)
```

2. **Convert the results into a Beautiful Soup object**

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

3. **Extracting relevant columns**

```python
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
```

Code can be found at: https://github.com/ndombrowski/IBM_DS_capstone/blob/main/Data_collection_via_webscraping.ipynb

Data cleaning

1. **Calculate the number of launches per site**

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

2. **Calculate the number of orbits**

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

3. **Evaluate the number of landing outcomes**

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

4. **Since there are different labels for landing outcomes:**

   we define a new class label for successful (1) or unsuccessful landings (0)

```
landing_class = []

for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

Code can be found at: https://github.com/ndombrowski/IBM_DS_capstone/blob/main/Data_wrangling.ipynb

EDA via data visualization

1. **Visualize relationships between different categories, such as flight numbers via launch site, using scatterplots**

```
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

2. **Visualize the relationship between success rate of each orbit type using a bargraph**

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

#plot
mean_orbit_success.plot(kind='bar')

plt.ylabel("Mean success rate",fontsize=20)
plt.show()
```

3. **Visualize the yearly trends for launch successes using a linegraph**

```
sns.lineplot(data=df, x="Year", y="Class")
plt.xlabel("Year",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```

Code can be found at: https://github.com/ndombrowski/IBM_DS_capstone/blob/main/EDA_and_data_vis.ipynb

EDA via SQL

1. Connect to IMB DB to which we have uploaded the SpaceX data

```
%sql ibm_db_sa://my-username:my-password@my-hostname:my-port/my-db-name?security=SSL
```

2. Use different constraint to explore the data:
   - Use 'unique()' to find the landing sites

```
%sql select Unique(LAUNCH_SITE) from SPACEX;
```

   - Use 'min' to identify the first successful landing dates for different conditions

```
%sql select min(Date) from SPACEX where landing__outcome like '%success%'
```

   - Use 'count()' to identify successful mission outcomes

```
%sql select Count(*) as Successful_Missions from SPACEX WHERE mission_outcome like 'Success%'
```

   - Use subqueries to identify booster_versions carrying the max payload mass

```
%sql select booster_version from SPACEX where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEX)
```

Code can be found at: https://github.com/ndombrowski/IBM_DS_capstone/blob/main/EDA_with_SQL.ipynb

Generating an interactive map with folium

- Identify coordinates for each site

```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

- Add coordinates to a foilum map with 'foilum.circle'

```
# Initialize the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=4)

for i, row in launch_sites_df.iterrows():
    coordinate = [row['Lat'], row['Long']]
    folium.Circle(coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup(row['Launch Site'])).add_to(site_map)
    folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12;
color:#d35400;"><b>%s</b></div>' % row['Launch Site'], )).add_to(site_map)

site_map
```

- Added the success/failed launches for each site on the map

```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

for index, record in spacex_df.iterrows():
    coordinate = [record['Lat'], record['Long']]
    folium.map.Marker(coordinate, icon=folium.Icon(color='white',icon_color=record['marker_color'])).add_to(marker_cluster)

site_map
```

- Add distances markers to important landmarks, such as the coast

```
# closest highway marker
distance_marker = folium.Marker(
   closest_highway,
   icon=DivIcon(
       icon_size=(20,20),
       icon_anchor=(0,0),
       html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_highway),
       )
   )
site_map.add_child(distance_marker)
```

Code can be found at: https://github.com/ndombrowski/IBM_DS_capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Creating a dashbord with dash/plotly

- Create a dashbord with plotly and dash
- Create a dropdown menu to view all or specific sites

```
dcc.Dropdown(id='site-dropdown', options = [
        {'label': 'All Sites','value': 'ALL'},
        {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
        {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
        {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
        {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'}
    ],
```

- Add a piechart

```
@app.callback(Output(component_id = 'success-pie-chart', component_property = 'figure'),
              Input(component_id = 'site-dropdown', component_property = 'value'))

def get_pie_chart(entered_site):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        fig = px.pie(filtered_df,
        values = 'class',
        names = 'Launch Site',
```

- Add a scatterplot with a slider for the payload

```
@app.callback(Output(component_id = 'success-payload-scatter-chart', component_property = 'figure'),
              Input(component_id = 'site-dropdown', component_property = 'value'),
              Input(component_id = 'payload-slider', component_property = 'value'))


def get_payload_fig(entered_site, payload):
    filtered_df = spacex_df[spacex_df['Payload Mass (kg)'].between(payload[0], payload[1])]
    if entered_site == 'ALL':
        fig3 = px.scatter(filtered_df, y = 'class', x= 'Payload Mass (kg)',
        color = 'Booster Version Category',
```

Code can be found at: https://github.com/ndombrowski/IBM_DS_capstone/blob/main/code/spacex_dash_app.py

Predict successful launches with Machine Learning

1. **Standardize the data**

```python
transform = preprocessing.StandardScaler()
X = transform.fit(X).transform(X)
```

2. **Divide data into training and test set**

```python
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2)
```

3. **Identify the best parameters for several modeling approaches (logistic regression, svm, decision tree and knn) and train a model**

```python
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
```

```python
logreg_cv = GridSearchCV(lr, parameters, cv=10)
```

```python
logreg_cv.fit(X_train, Y_train)
```

4. **Validate the model by getting the accuracy score and generating a confusion matrix**

```python
knn_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```

We can plot the confusion matrix

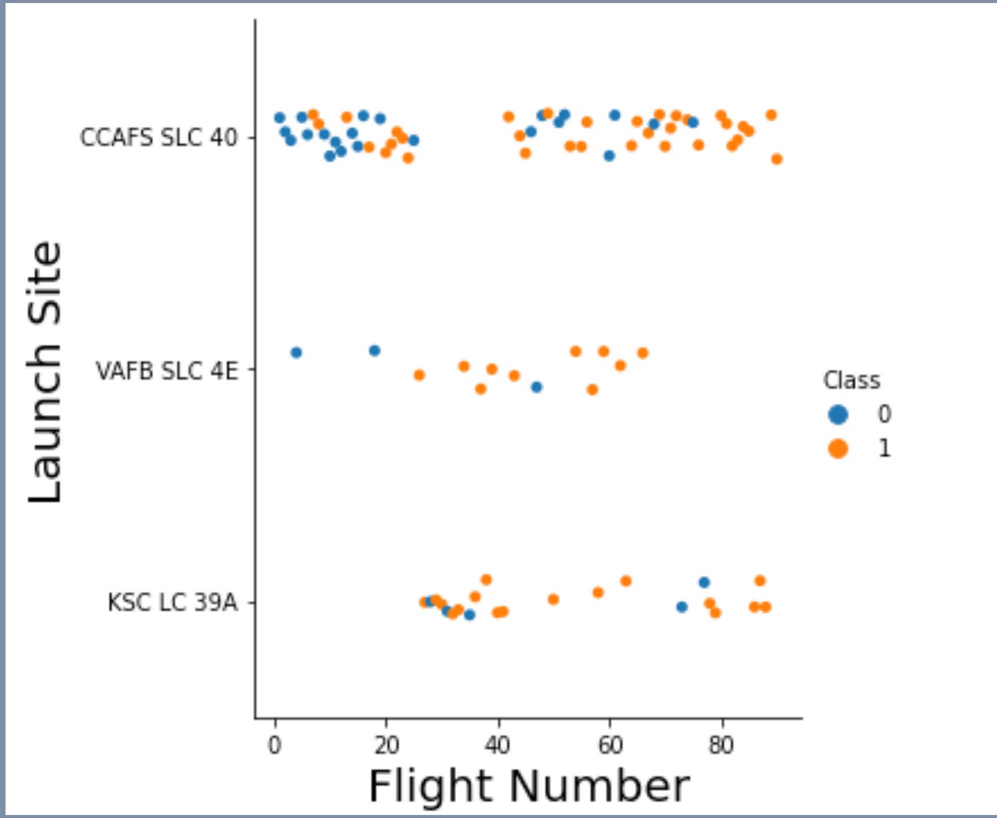
```python
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Code can be found at: https://github.com/ndombrowski/IBM_DS_capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.ipynb

Results:

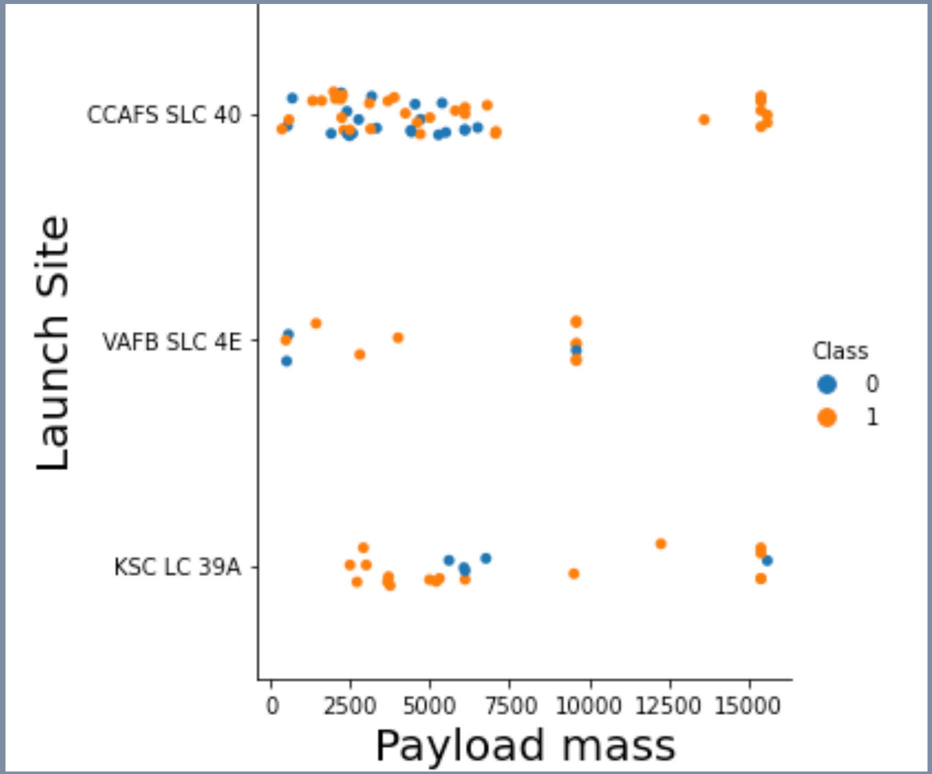
(1) Deriving insights from EDA

Different launch sites have different success rates



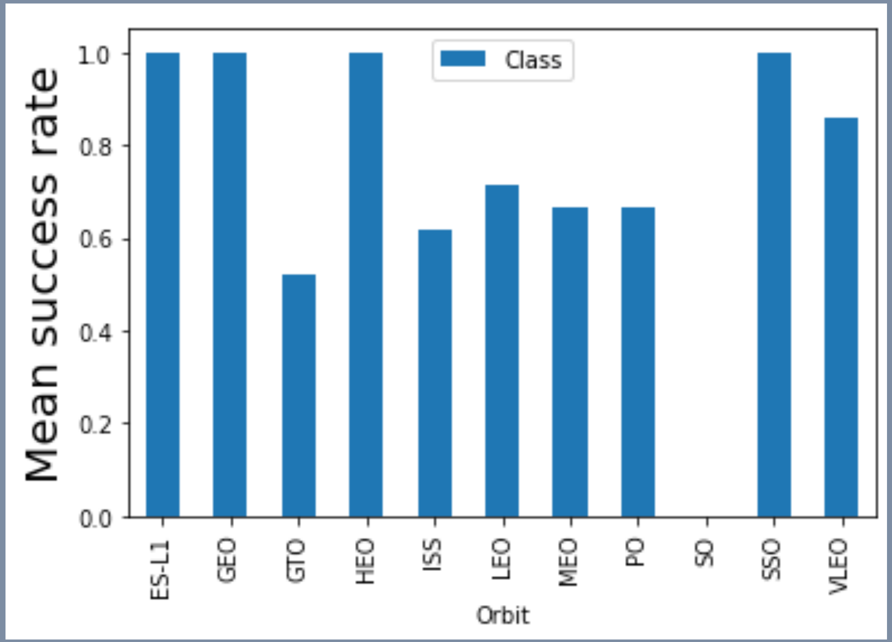
- Success rates for CCAFS SLC 40 increase over time
- The majority of launches were successful at VAFB SLC 4E and KSC LC 39A

Payload mass affects success rate



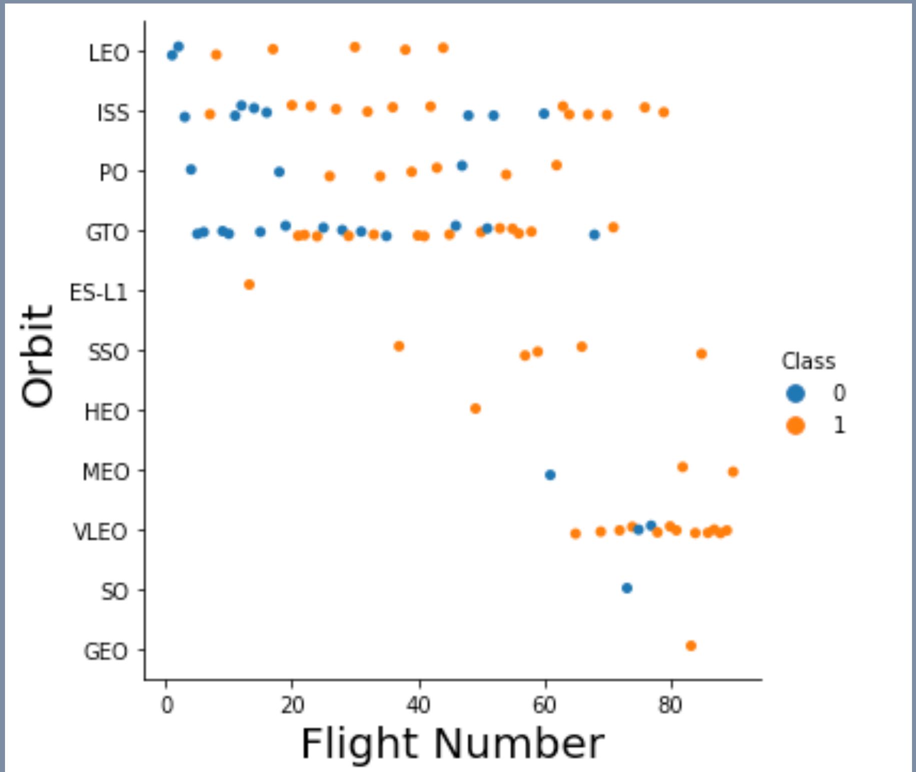
- At CCAFS SLC 40 the most successful launches are launches with a high payload mass
- There were no launches at VAFB with a payload mass > 10 000

Orbits affect success rates



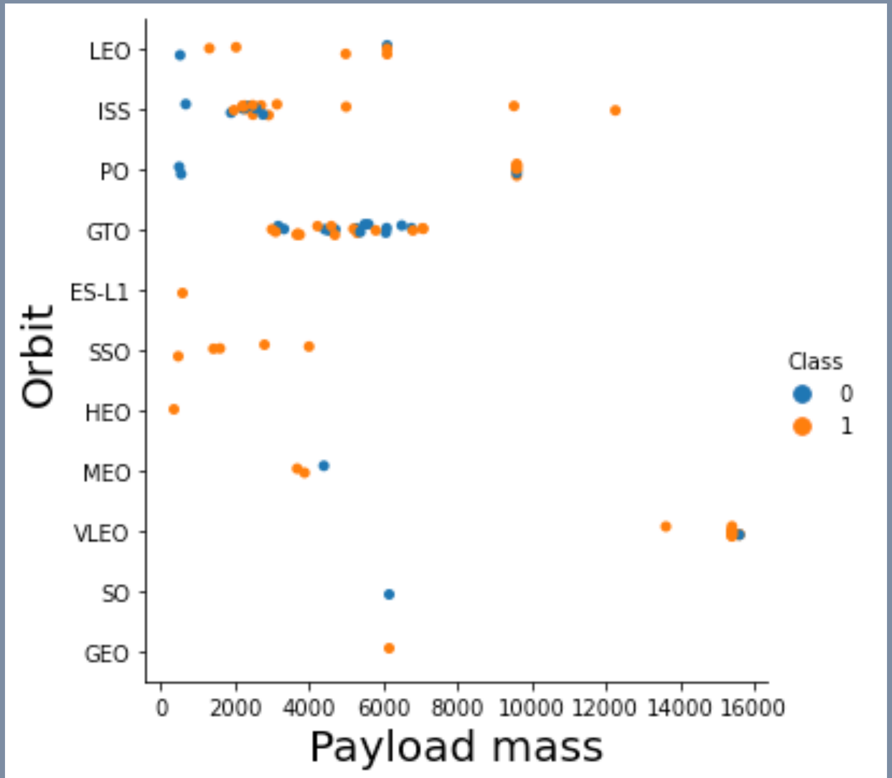
- ES-L1, GEO , HEO and SSO have only successful launches
- SO has no successful launches

Relationship between orbits and flights



- In the LEO orbit the Success appears related to the number of flights
- There seems to be no relationship between flight number when in GTO orbit.

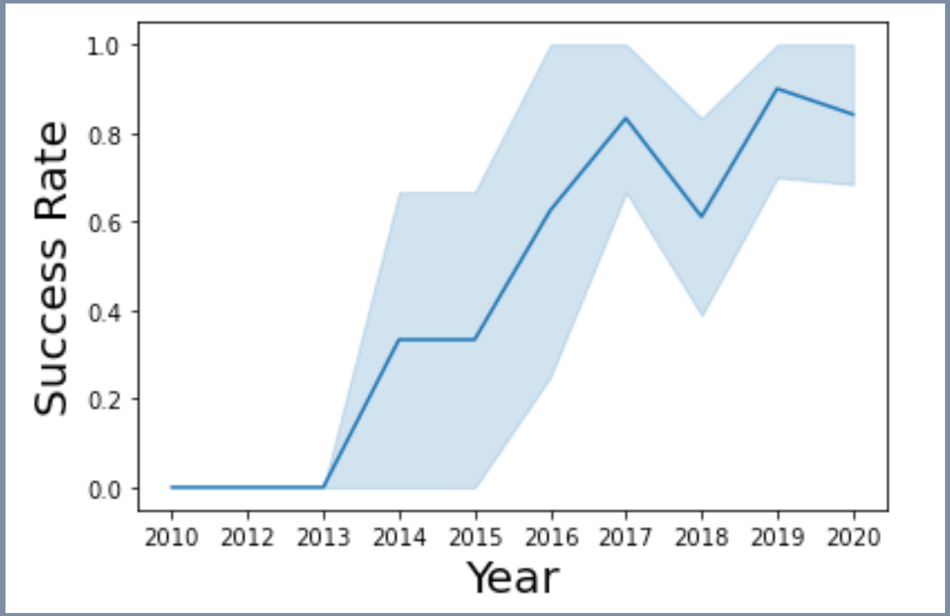
Relationship between payload and orbit



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

- For GTO we cannot landing rates

Success rates increased over the years



- SpaceX started with a success rate of ~30% in 2014 and this rapidly increased to around 80% in 2017

There are 4 unique landing sites:

```
%sql select Unique(LAUNCH_SITE) from SPACEX;
```

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

```
%sql select Unique(LAUNCH_SITE) from SPACEX;
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

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

Lets look at some records:

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

```
%sql select * from SPACEX WHERE LAUNCH_SITE like 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
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	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

Identify the total payload mass

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

```
%sql select SUM(payload_mass__kg_) as payloadmass from SPACEX WHERE customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

payloadmass
45596

Identify the average payload mass

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

```
%sql select AVG(payload_mass__kg_) as avg_payloadmass  from SPACEX WHERE booster_version like 'F9 v1.0%'
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

avg_payloadmass
340

The first successful ground pad landing

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql select min(Date) from SPACEX where landing__outcome like '%success%'
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

1
2015-12-22

Boosters between a mass of 4000 and 6000

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 SPACEX where landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ > 4000 AND payload_mass__kg_ < 6000
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

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

Mission successes and failures

List the total number of successful and failure mission outcomes

```
%sql select Count(*) as Successful_Missions from SPACEX WHERE mission_outcome like 'Success%'
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

successful_missions
100

```
%sql select Count(*) as Failure_Missions from SPACEX WHERE mission_outcome like 'Failure%'
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

failure_missions
1

Which boosters carried the max payload?

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql select booster_version from SPACEX where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEX)
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
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

List of failed landings in 2015

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

```
%sql select booster_version,launch_site,landing__outcome from SPACEX where landing__outcome = 'Failure (drone ship)' AND YEAR(Date) = 2015
```

```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

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

Ranking of landing outcomes

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

```
%sql select landing__outcome, count(landing__outcome) as count from SPACEX WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' group by landing__outcome  order by count desc
```

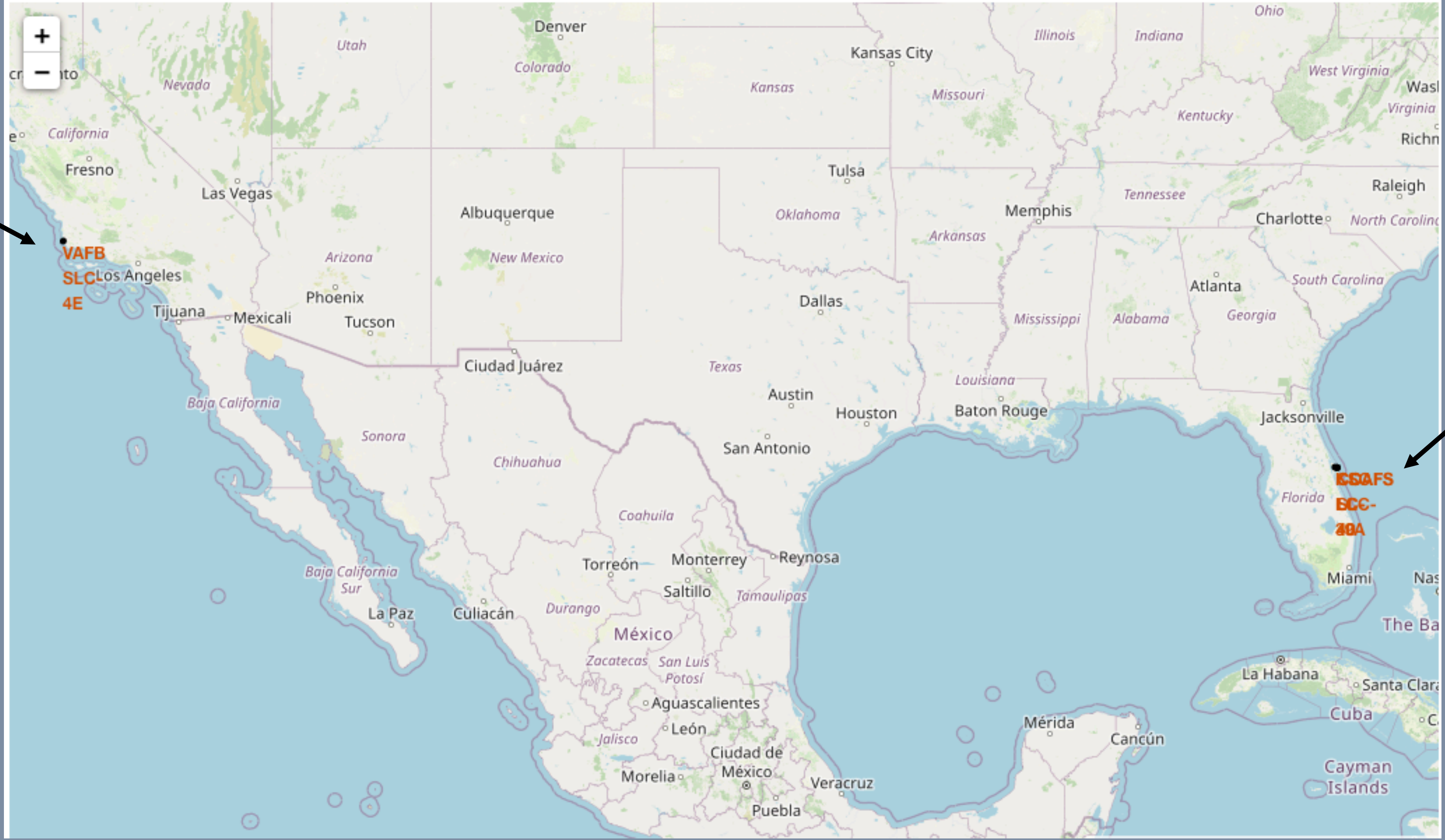
```
 * ibm_db_sa://jty26738:***@6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30376/bludb
Done.
```

landing__outcome	COUNT
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

Results:

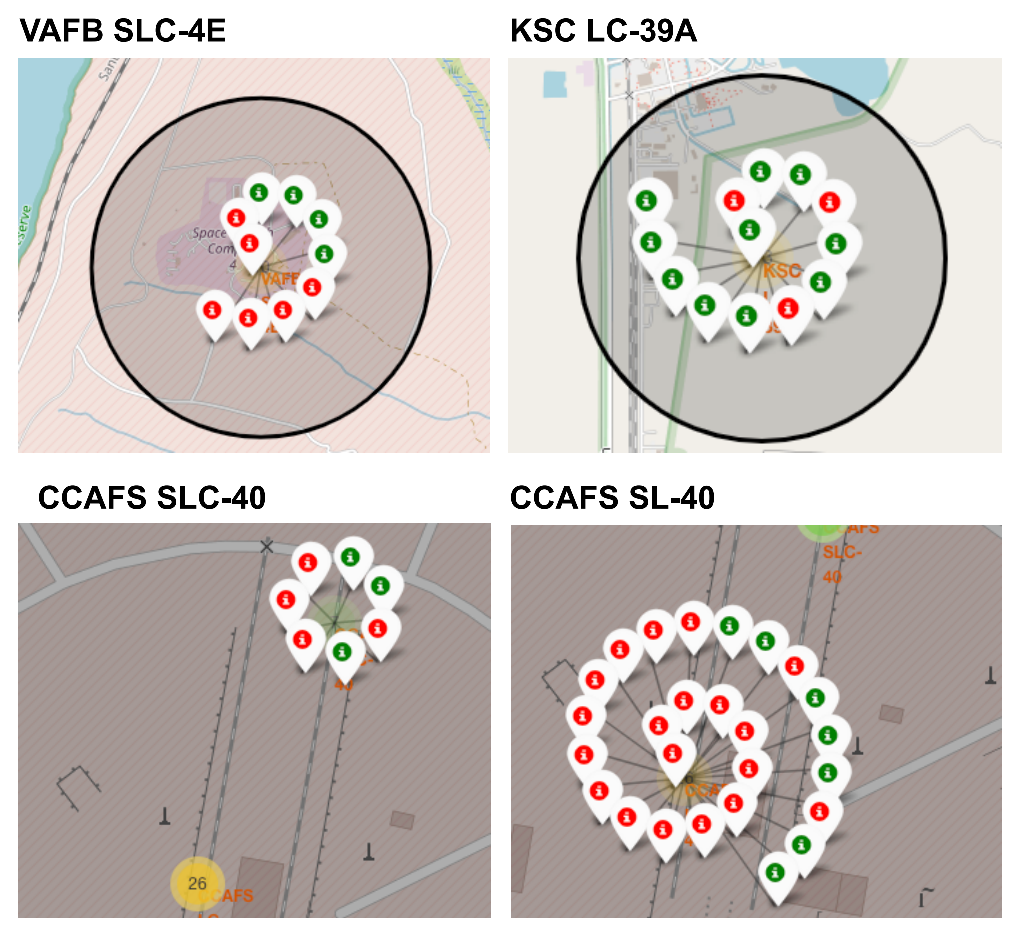
(2) Deriving insights from the interactive map and dashbord

1 launch is in the west and 3 in the east of the US



CCAFS LC-40
CCAFS SLC-40
KSC LC-39A

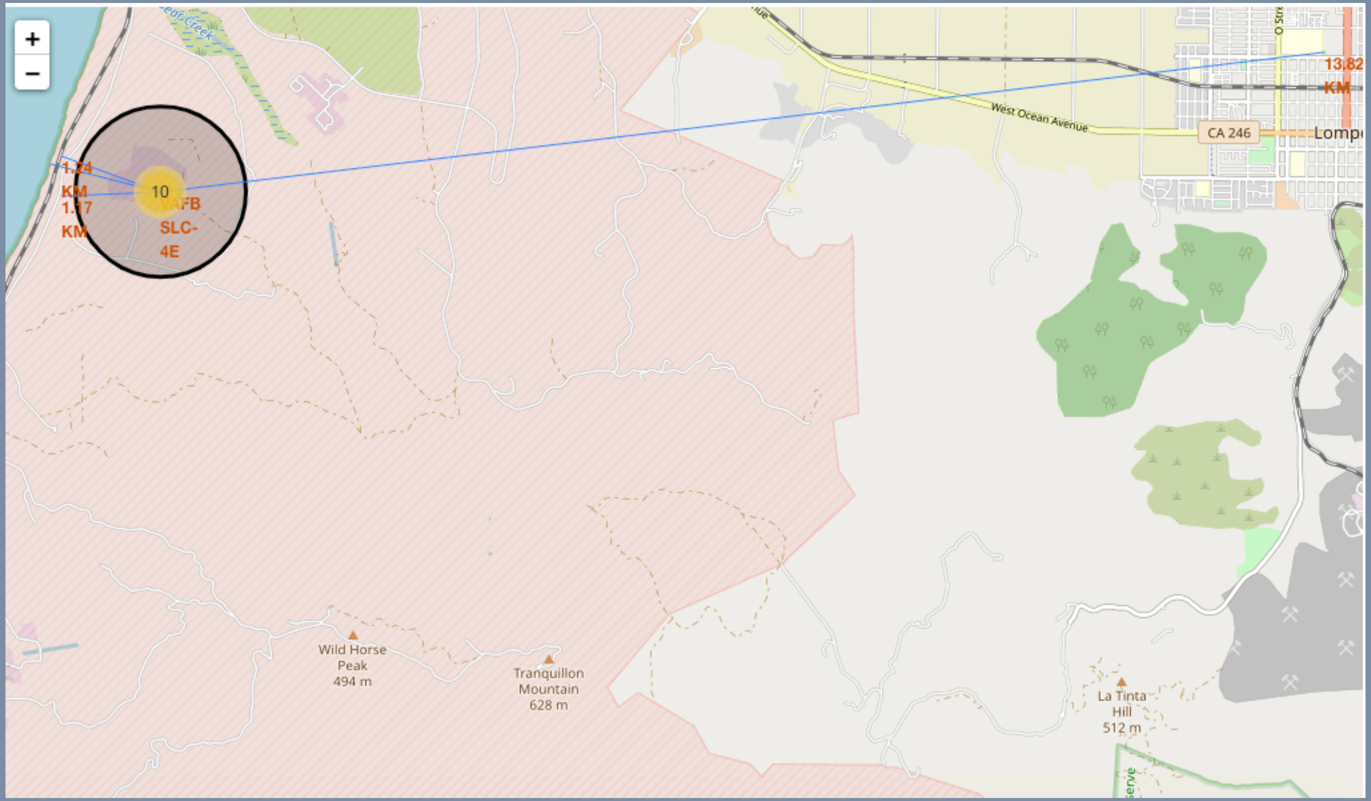
Successful launches per site



Green marker shows successful launches and red marker shows failures

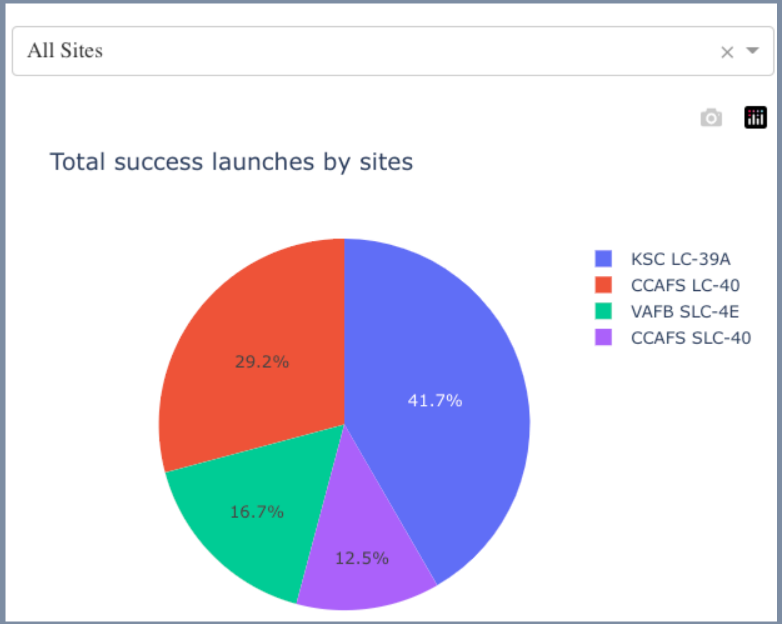
- KSC LC-39A is the site with most successful launches
- CCAFS SL-40 is the site with the most failures

Distance of launch site to key locations



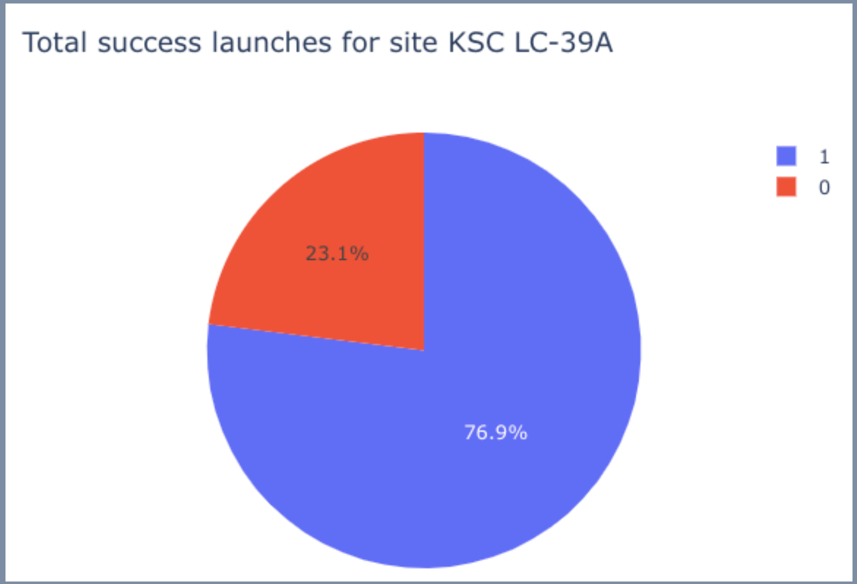
- Launch sites are close to the coast (safety concerns) and highways as well as rail roads (transport of humans and goods)
- Launch sites are not too close to cities to avoid safety issues during mission failures but close enough for workers to transit

Displaying the success by launch site



- KSC LC-39A had the most successful launches
- VAFB SLC-4E is the site with the least number of successfull launches

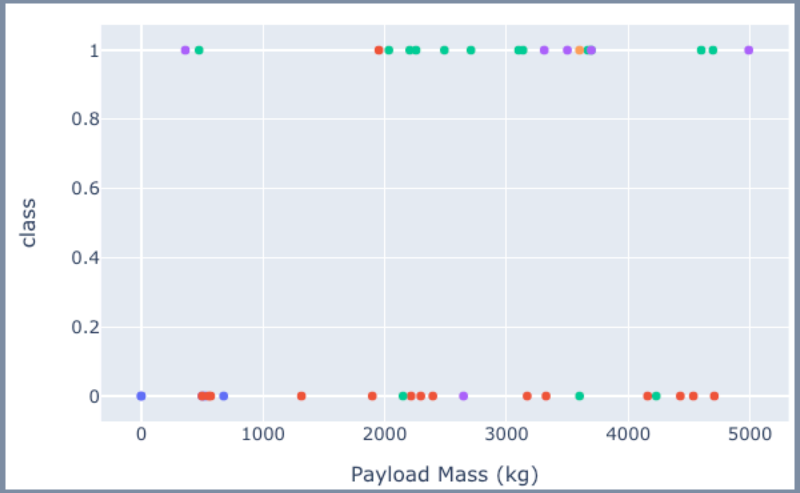
Displaying the success by launch site



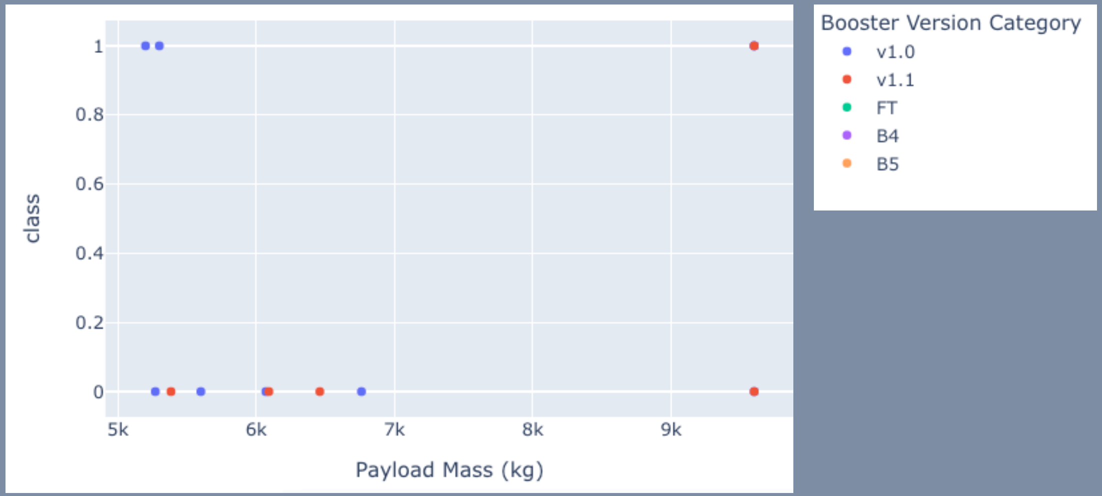
- KSC LC-39A had 76.9% successful launches and 23.1% failures

Displaying the success by launch site

Payload between 0 – 5000 kg



Payload between 5000 – 10 000 kg



- Booster v1.0 and v1.1 are used preferably with higher payload but both have a low success rate
- Booster FT is used with lower payload and has a good success rate

Results:

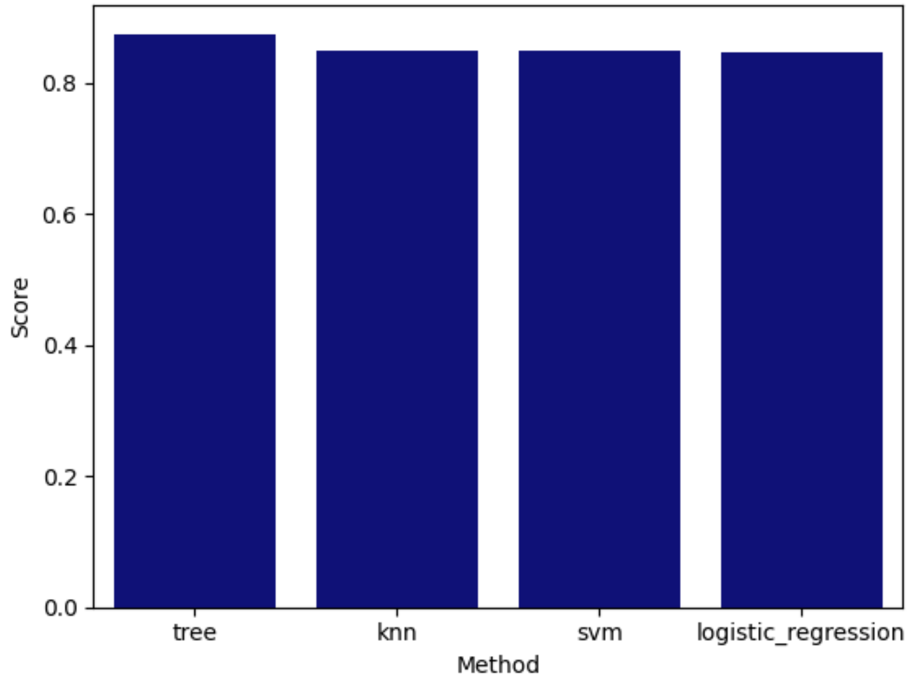
(2) Deriving insights from the machine learning approach

The decision tree is slighly better than other methods

```
scores = {'logistic_regression':logreg_cv.best_score_, 'svm':svm_cv.best_score_, 'tree':tree_cv.best_score_,
'knn':knn_cv.best_score_}
scores_df = pd.DataFrame({'Method': list(scores.keys()), 'Score': list(scores.values())})
scores_df = scores_df.sort_values(by = "Score", ascending = False)
```

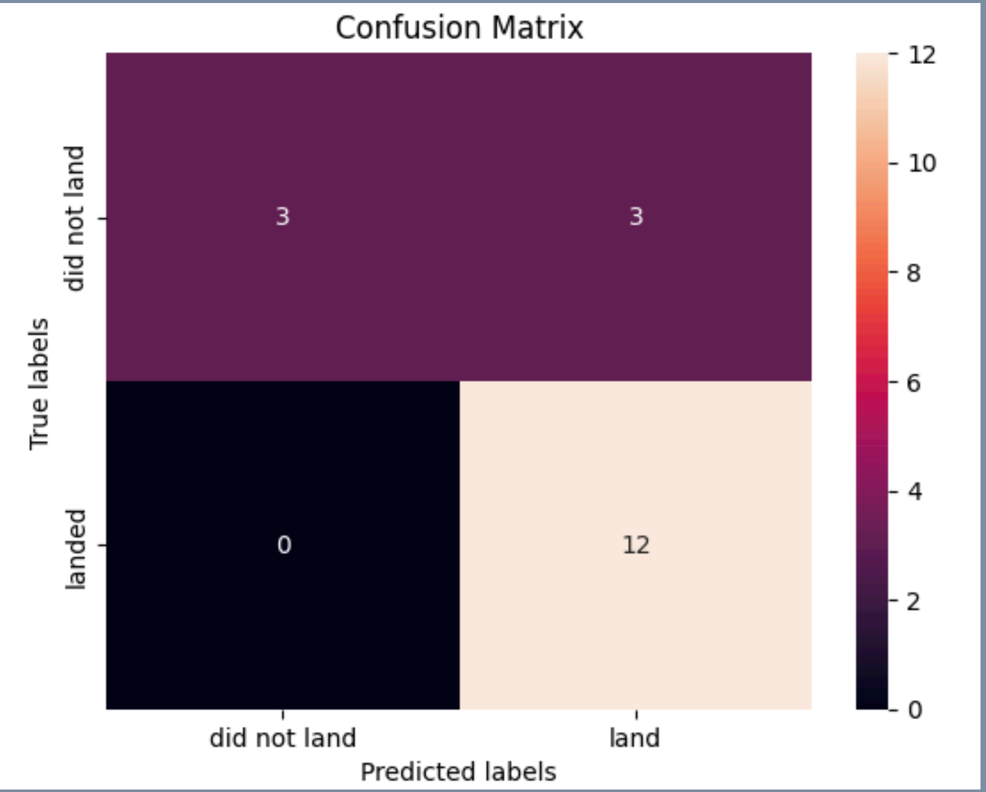
```
import seaborn as sns
%matplotlib inline

sns.barplot(data=scores_df, x="Method", y="Score", color = "darkblue")
```

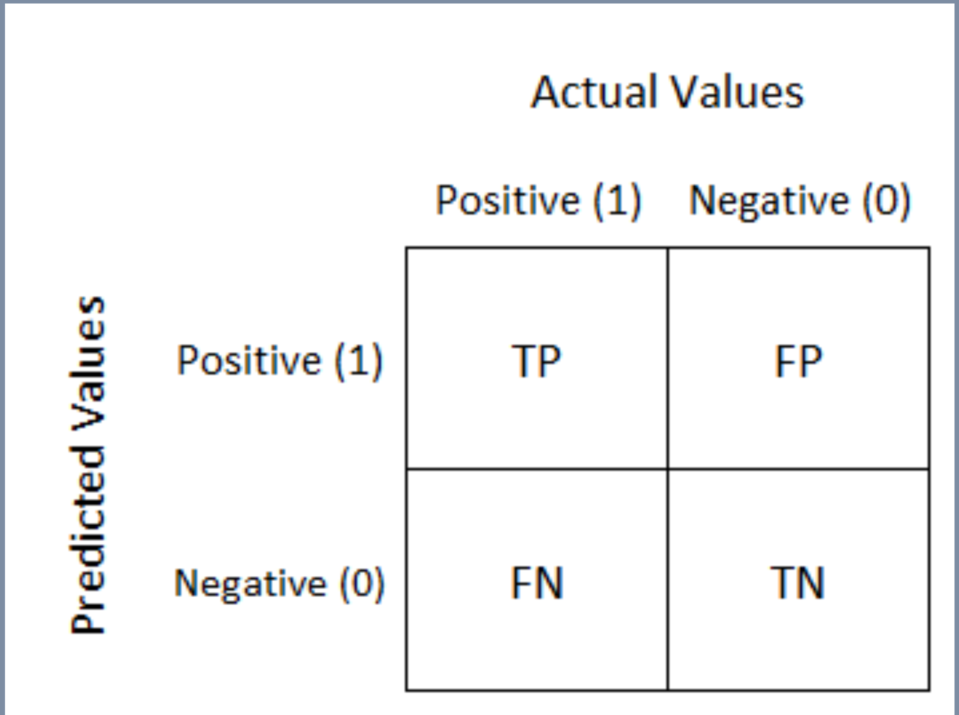
```
<AxesSubplot:xlabel='Method', ylabel='Score'>
```



The decision tree is slighly better than other methods

Confusion matrix for decision tree method
(only one example shown because all methods produce similar plots)





- The classifier can distinguish between different classes
- The main issue is that there are some false positives

Conclusion

Based our analysis, we can conclude that:

- Launches became more successful over the years
- The most successful launches are launches with a high payload mass but this depends on the site
- KSC LC-39A is the site with most successful launches
- We can predict launch success using our model build using the decision tree model

Appendix

- **Link to github repository with all code notebooks:** https://github.com/ndombrowski/IBM_DS_capstone