IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Elisabeth Kind
27.01.2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- 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 analytics

Introduction

- Project background and context
  - SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage
  - Space Y would like to compete with SpaceX founded by Billionaire industrialist Elon Musk
- Problems you want to find answers
  - Determine the price of each launch
  - Determine likelihood of successful landing of the first stage of the SpaceX Falcon 9 (then it may be reused)
  - Find factors that contribute to successful landing

Section 1

Methodology

Methodology

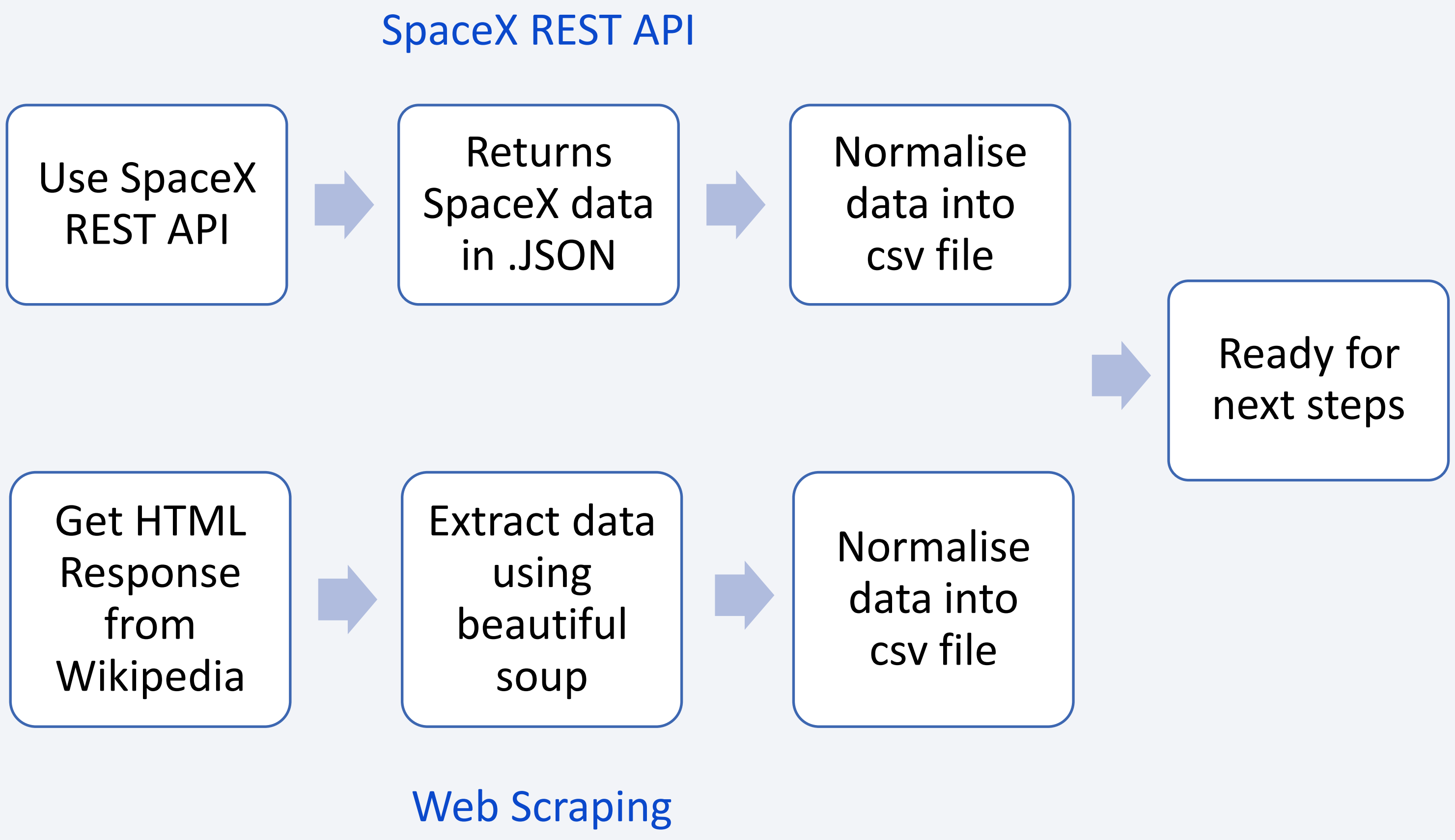
Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scraping (from a Wikipedia page)
- Perform data wrangling
  - One Hot Encoding data fields for Machine Learning and data cleaning of null values and irrelevant rows and 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 were built and evaluated to determine the best classifier

Data Collection

Two datasets were collected:

- Requested rocket launch data from the **SpaceX REST API**
- Requested Falcon 9 launch data from a Wikipedia page via **Web Scraping**



Data Collection – SpaceX API

- Requested rocket launch data from the API
- Decoded response as JSON and turned it into a Pandas dataframe
- Applied functions to extract booster version, launch site, payload data, and core data
- Constructed dataset
- Replaced missing payload mass values with mean & exported dataframe as CSV

https://github.com/elisabethkind/AppliedDataScienceCapstone/blob/master/Data%20Collection%20API.ipynb

1. Request data from SpaceX API

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

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

2. Convert response to a .JSON file

```
response.json()
data=pd.json_normalize(response.json())
```

3. Apply functions to extract data

```
getBoosterVersion(data)
```

```
getPayloadData(data)
```

```
getLaunchSite(data)
```

```
getCoreData(data)
```

4. Construct dataset

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

```
# Create a data from launch_dict
launch_df=pd.DataFrame.from_dict(launch_dict, orient='columns', dtype=None, columns=None)
```

5. Filter dataframe and export to .CSV file

```
# 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'] = data_falcon9['PayloadMass'].replace(np.nan, Mean_PayloadMass)
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

- Requested page from its URL
- Extracted all column/variable names from the HTML table header
- Created a dataframe by parsing the launch HTML tables

https://github.com/elisabethkind/AppliedDataScienceCapstone/blob/master/Data%20Collection%20Web%20Scraping.ipynb

1. Request page from URL

```
html_data = requests.get(static_url)
```

2. Create BeautifulSoup Object

```
soup = BeautifulSoup(html_data.text, 'html.parser')
```

3. Find tables in HTML

```
html_tables = soup.find_all('table')
```

4. Extract all column names from HTML table header <th>

```
column_names = []
```

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

5. Create dictionary with column names as keys

```
launch_dict= dict.fromkeys(column_names)
```

6. Append data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('t
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as n
```

7. Convert dictionary to dataframe

```
df=pd.DataFrame(launch_dict)
```

8. Export to CSV

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

Data Wrangling

- Only included Falcon 9 launches
- Landing outcomes were converted to classes (0 = failed landing, 1 = successful landing)
- Calculated number of launches on each site, per orbit type, and mean success rate

https://github.com/elisabethkind/AppliedDataScienceCapstone/blob/master/Data%20Wrangling.ipynb

1. Load dataset from CSV

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.c
```

2. Inspect for null data & data types

```
df.isnull().sum()/df.count()*100
```

```
df.dtypes
```

3. Calculate nr of launches at each site

```
df.LaunchSite.value_counts()
```

4. Calculate nr and occurrence of each orbit

```
df.Orbit.value_counts()
```

5. Calculate nr and occurrence of mission outcome per orbit type

```
landing_outcomes = df.Outcome.value_counts()
```

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
```

6. Create landing outcome label from Outcome column

```
# 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)
```

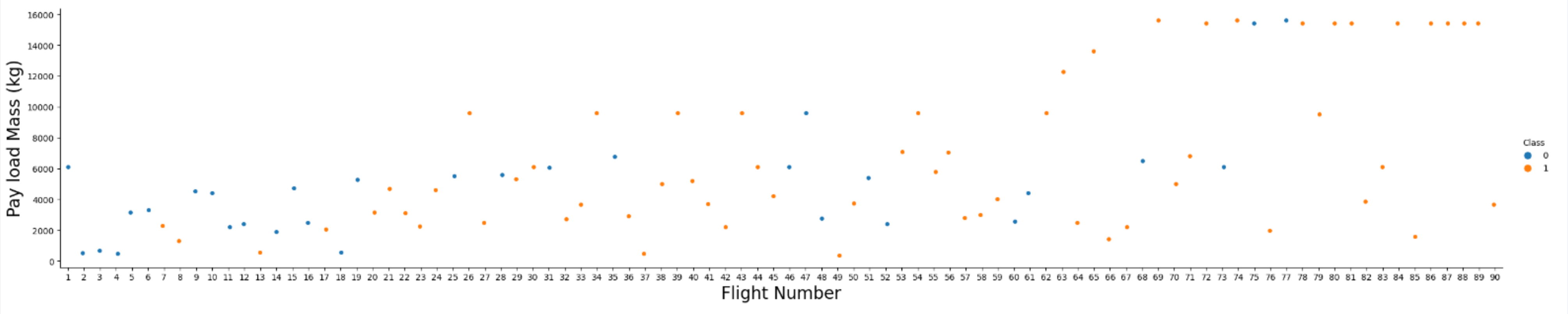
```
df['Class']=landing_class
```

7. Determine success rate

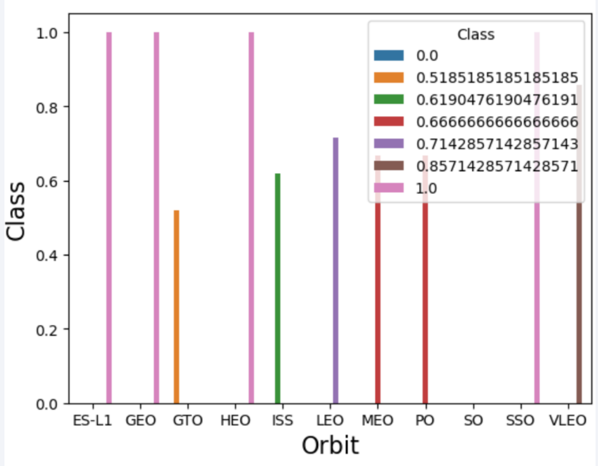
```
df["Class"].mean()
```

EDA with Data Visualization

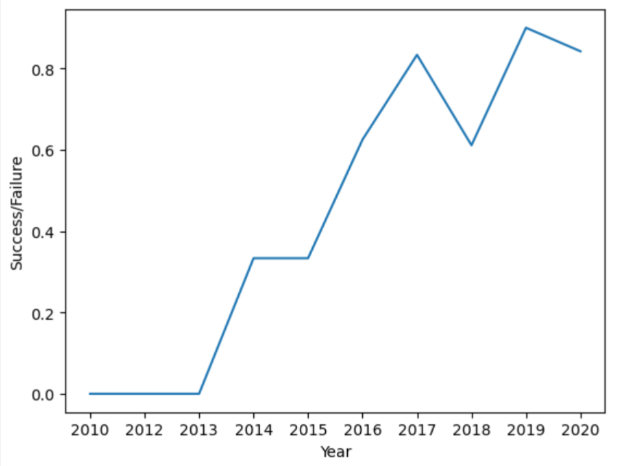
Scatter Plots: see how interactions between FlightNumber, PayloadMass, Launch Site, and Orbit type affect launch outcome



Bar chart: compare success rates of orbit types



Line chart: inspect yearly trends in success rate



https://github.com/elisabethkind/AppliedDataScienceCapstone/blob/master/EDA%20with%20Data%20Visualisation.ipynb

EDA with SQL

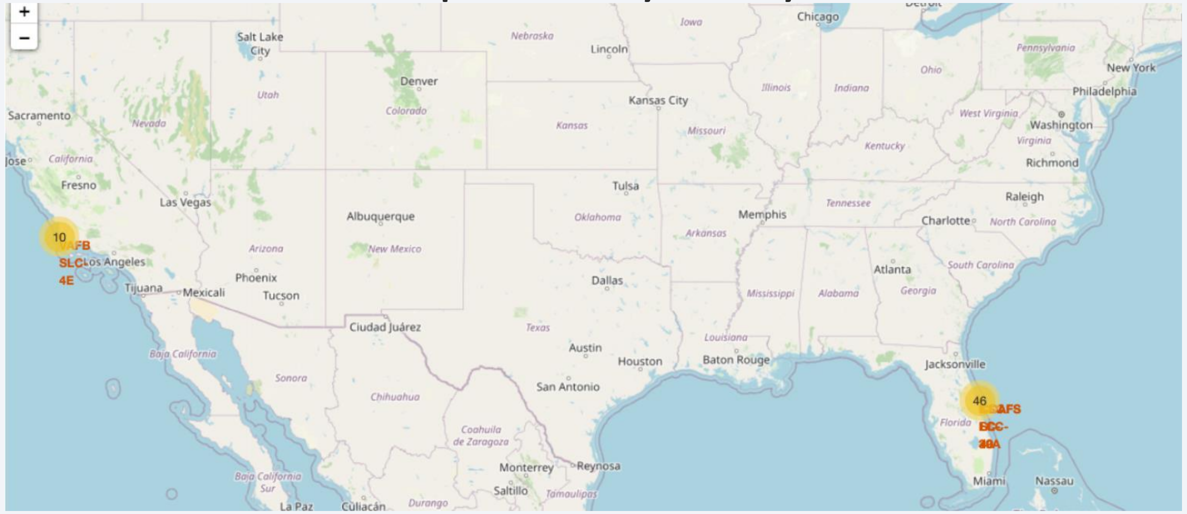
SQL Queries performed:

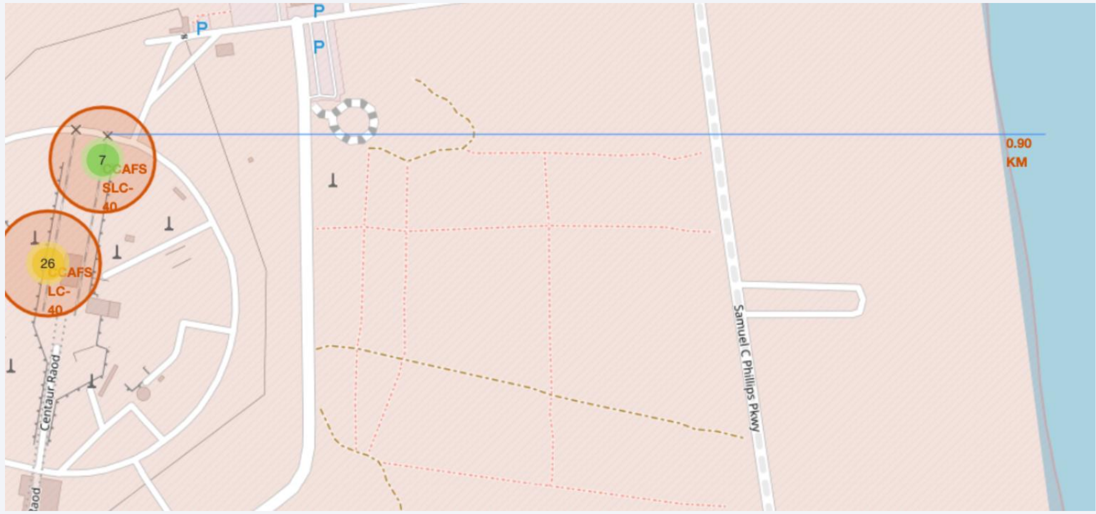
- Display the names of the **unique launch sites** in the space mission
- Display 5 records where launch sites begin with the string **'CCA'**
- Display the total **payload mass** carried by **boosters launched by NASA** (CRS)
- Display average **payload mass** carried by **booster version F9 v1.1**
- List the **date** when the **first successful landing** outcome in ground pad was achieved
- List the names of the **boosters** which have **success in drone ship** and have **payload mass greater than 4000** but **less than 6000**
- List the **total number** of successful and failure **mission outcomes**
- List the names of the **booster_versions** which have **carried the maximum payload mass**
- List the **failed landing_outcomes in drone ship**, their booster versions, and launch site names for in year **2015**
- **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

https://github.com/elisabethkind/AppliedDataScienceCapstone/blob/master/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- All launch sites were marked on map with circles and markers
- Marked the success/failed launches for each site on the map (success = green, failed = red)
- Marked distance between a launch site and nearest coastline with a line for proximity analysis

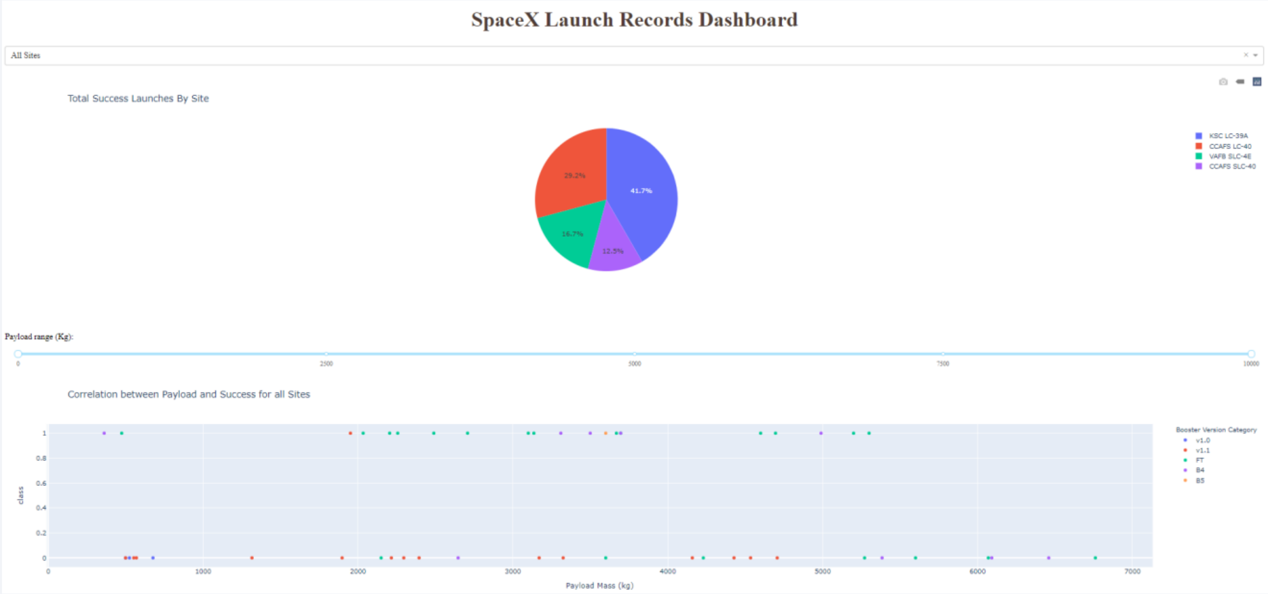




https://github.com/elisabethkind/AppliedDataScienceCapstone/blob/master/Interactive%20Visual%20Analytics%20and%20Dashboard.ipynb

Build a Dashboard with Plotly Dash

- Launch site drop-down: display plots for specific site or all sites combined
- Pie chart visualizing success rate in percentage
- Payload range slider: select specific payload range to see if variable payload is correlated to mission outcome
- Scatter plot of payload vs. launch outcome: visually observe how payload may be correlated with mission outcomes for selected site(s)
- Color-label of Booster version on each scatter point so that we may observe mission outcomes with different boosters



https://github.com/elisabethkind/AppliedDataScienceCapstone/blob/master/spacex_dash_app.py

Predictive Analysis (Classification)

Created **Logistic regression, SVM, Decision tree**, and **KNN** models as follows:

- Split the data into training and testing data (using train_test_split)
- Create model
- Train model and select hyperparameters (using GridSearchCV)
- Calculate the model's accuracy

https://github.com/elisabethkind/AppliedDataScienceCapstone/blob/master/Machine%20Learning%20Prediction.ipynb

1. Split data into training and testing data

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

2. Create logistic regression object & fit it to find the best parameters

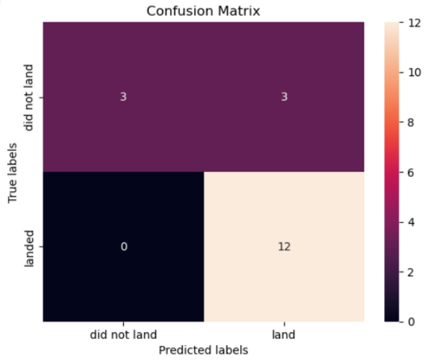
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}#
lr=LogisticRegression()

logreg_cv=GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

3. Evaluate the model's accuracy

```
logreg_accuracy = logreg_cv.score(X_test, Y_test)
logreg_accuracy
```

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



→ Repeat for SVM, decision tree, and KNN

4. Find method that performs best by comparing accuracy scores

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.800000	0.800000	0.800000	0.800000
F1_Score	0.888889	0.888889	0.888889	0.888889
Accuracy	0.833333	0.833333	0.666667	0.833333

Results

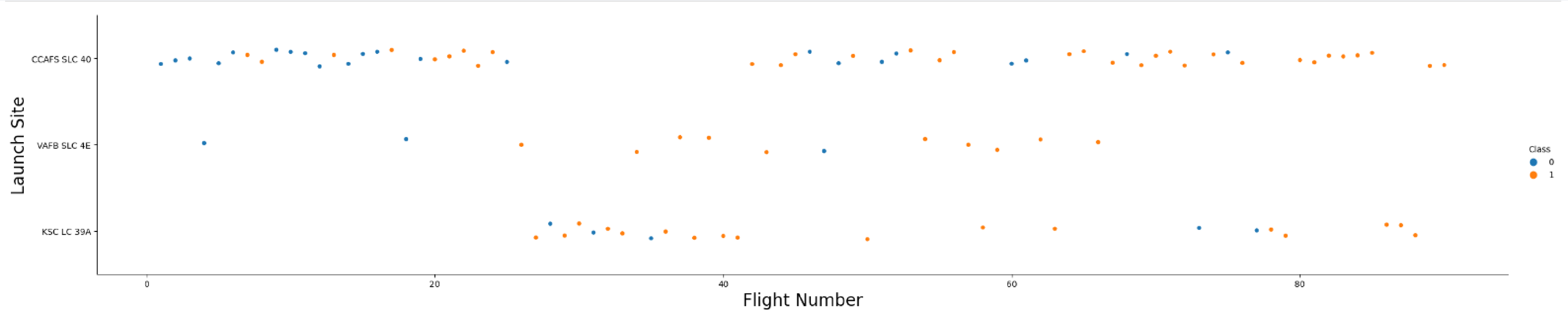
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

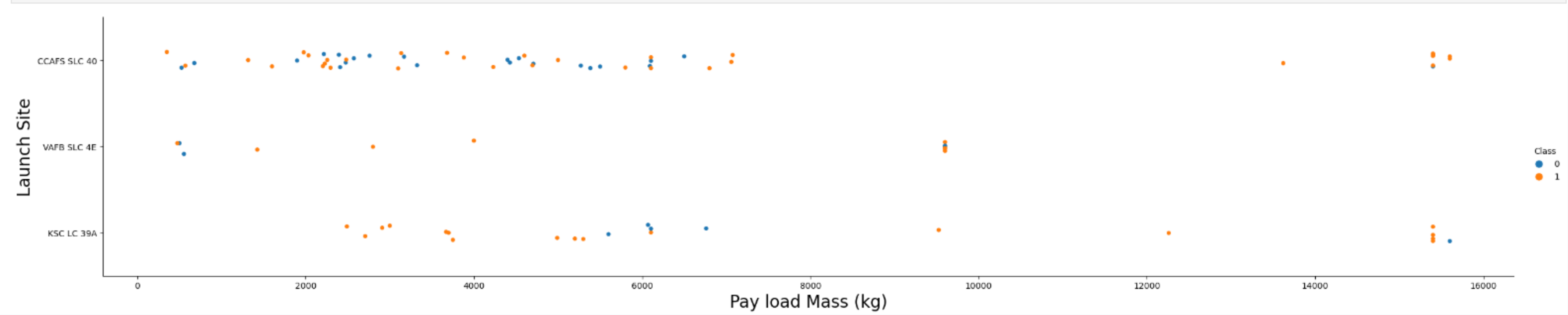
Flight Number vs. Launch Site

- Flight numbers vary by launch site such that there appear to be more launches from CCAFS SLC 40 than from the other sites
- Successful launches seem to increase with flight numbers



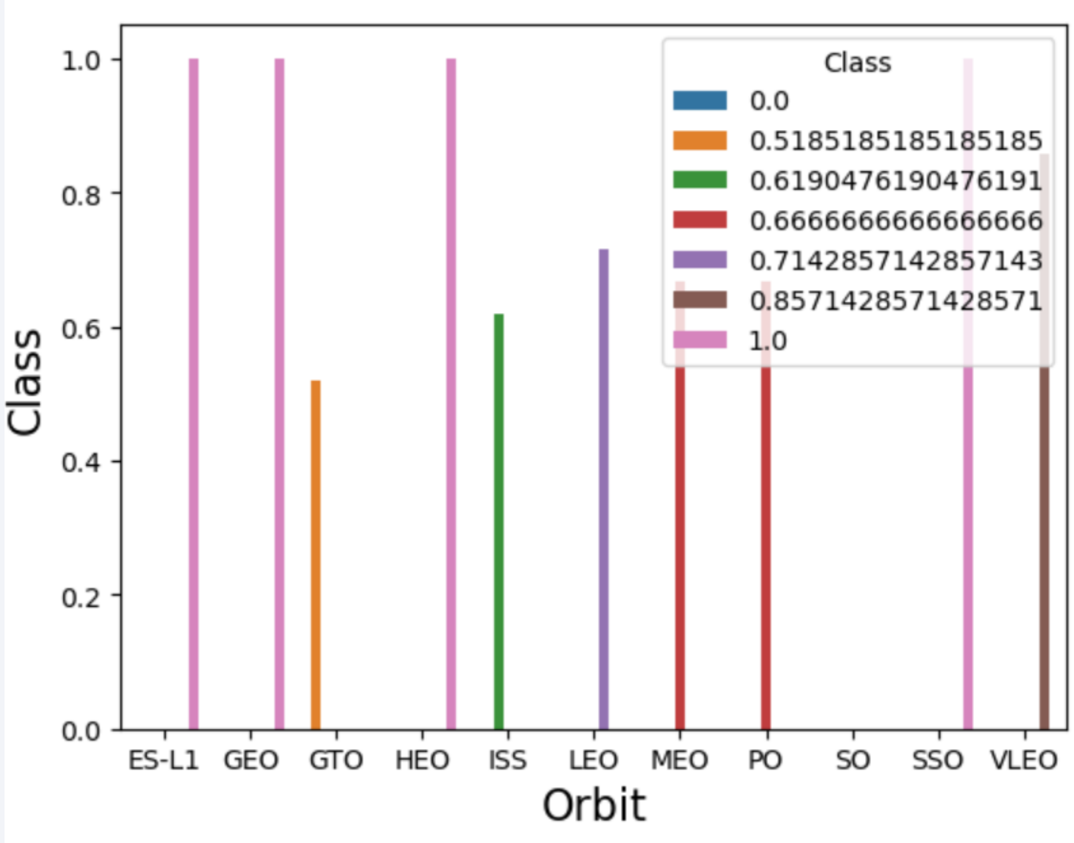
Payload vs. Launch Site

- Most launches have payloads between 0 and 7000 kg
- KSC LC 39A seems to be particularly successful for payload masses between 2000 and 5000 kg
- WAFB SLC 4E may be particularly successful for payload masses between 1000 and 4000 kg
- CCAFS SLC 40 seems to have a lower success rate than the other sites
- Success rate seems to increase with payload mass



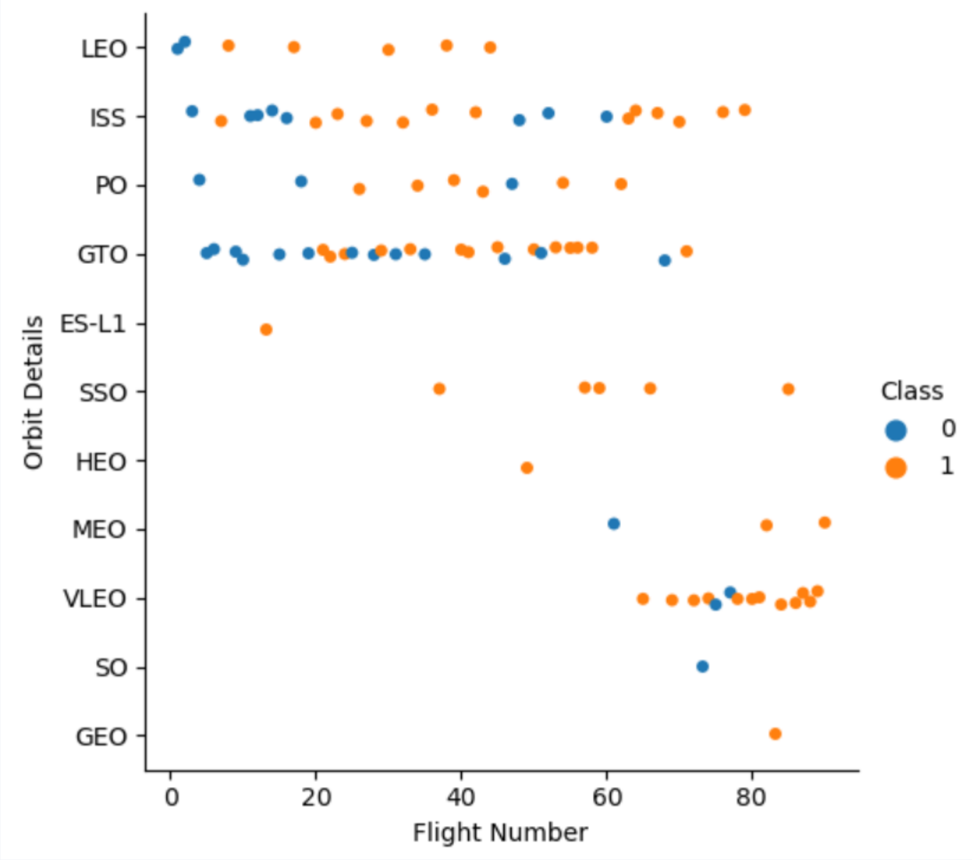
Success Rate vs. Orbit Type

- Success rate varies by orbit type
- ES-L1, GEO, HEO, and SSO have the highest success rate (100%)
- SO has the lowest success rate (0%)



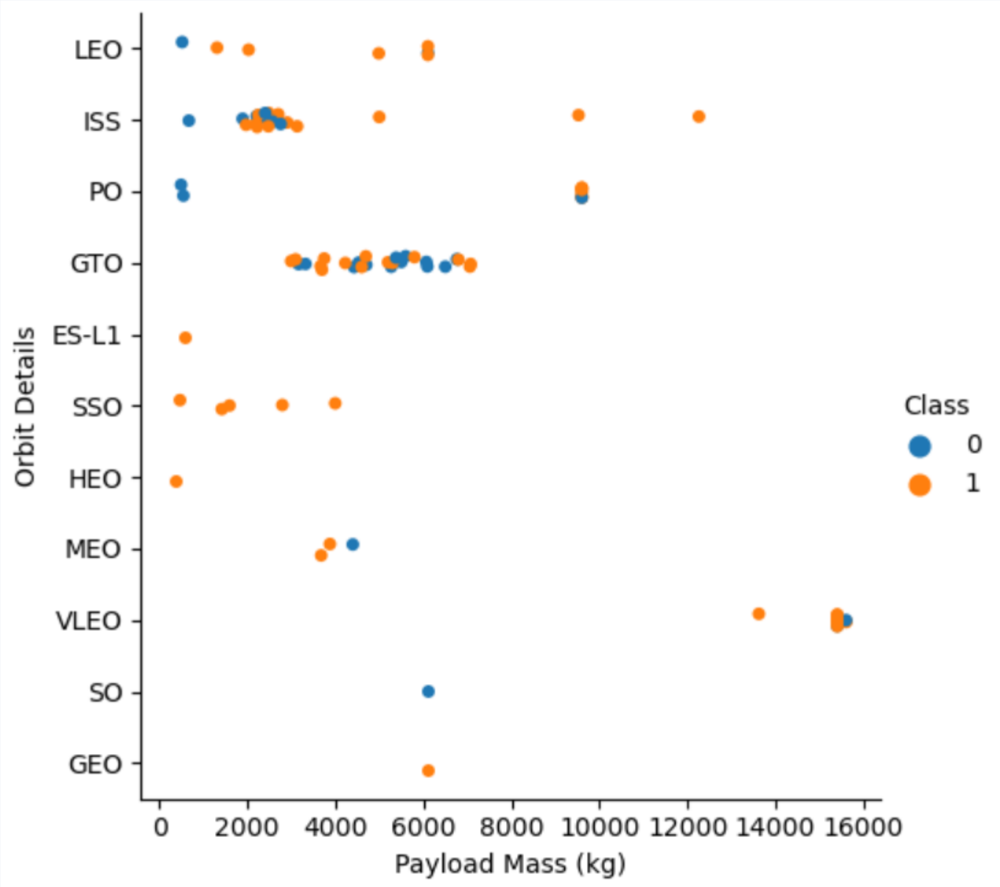
Flight Number vs. Orbit Type

- Flight numbers vary by orbit type
- Majority of first flights were in LEO, ISS, PO, and GTO but have decreased (except for ISS)
- Flights in VLEO have notably increased



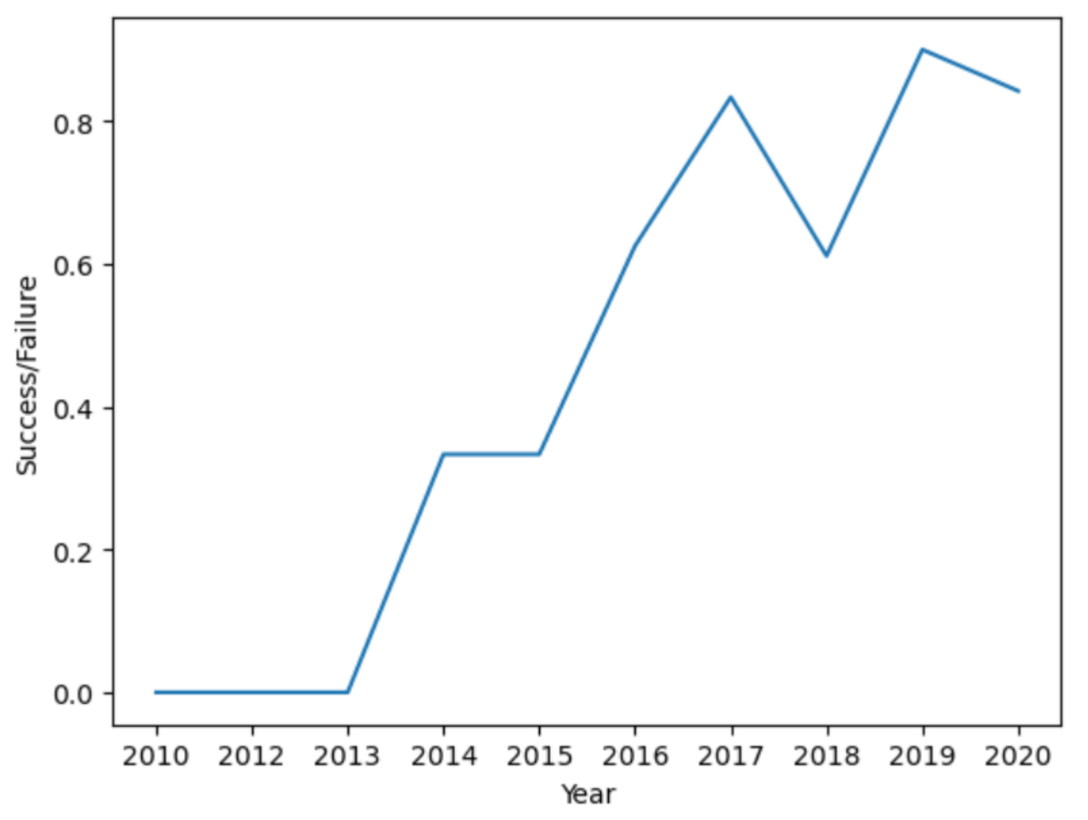
Payload vs. Orbit Type

- Payload varies by orbit type
- GTO is most used with payload between 4000 and 8000 kg (but mixed outcomes)
- ISS is most used with payload between 2000 and 4000 kg
- SSO seems particularly successful for low payload (< 4000 kg)
- LEO and ISS may be most successful for payload between 4000 and 13 000 kg
- VLEO is only used with high payload (> 13 000 kg)



Launch Success Yearly Trend

- Success rate increased massively with time
- Drop in 2018
- Seems to plateau around 0.8



All Launch Site Names

Extracted all unique launch site names used in the dataset:

```
%sql select distinct Launch_Site from SPACEXTBL
```

Result:

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

Launch Site Names Begin with 'CCA'

Extracted 5 records of launch sites begin with `CCA`:

```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

Result:

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

Total Payload Mass

Calculated the total payload carried by boosters from NASA:

```
%sql select sum(payload_mass__kg_) from SPACEXTBL WHERE customer = 'NASA (CRS)'
```

Result:

```
45596
```

Average Payload Mass by F9 v1.1

Calculated the average payload mass carried by booster version F9 v1.1:

```
%sql select avg(payload_mass__kg_) from SPACEXTBL WHERE booster_version = 'F9 v1.1'
```

Result:

```
2928
```

First Successful Ground Landing Date

Extracted the dates of the first successful landing outcome on ground pad:

```
%sql select min(DATE) from SPACEXTBL WHERE landing__outcome = 'Success (ground pad)'
```

Result:

2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Extracted the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000:

```
%sql select booster_version from SPACEXTBL where landing__outcome = 'Success (drone ship)'\
and payload_mass__kg_ between 4000 and 6000
```

Result:

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

Total Number of Successful and Failure Mission Outcomes

Calculated the total number of successful and failure mission outcomes:

```
%sql select mission_outcome, count(mission_outcome) from SPACEXTBL GROUP BY mission_outcome
```

Result:

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

Extracted the names of the booster which have carried the maximum payload mass:

```
%sql select booster_version, payload_mass__kg_ from SPACEXTBL\
where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL)
```

Result:

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

Max. payload mass:
15 600 kg

2015 Launch Records

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

```
%sql select booster_version, launch_site from SPACEXTBL where landing__outcome = 'Failure (drone ship)' and year(DATE) = 2015
```

Result:

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

Ranked 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 count(landing__outcome), landing__outcome from SPACEXTBL \
where DATE between '2010-06-04' and '2017-03-20' group by landing__outcome\
order by count(landing__outcome) desc
```

Result:

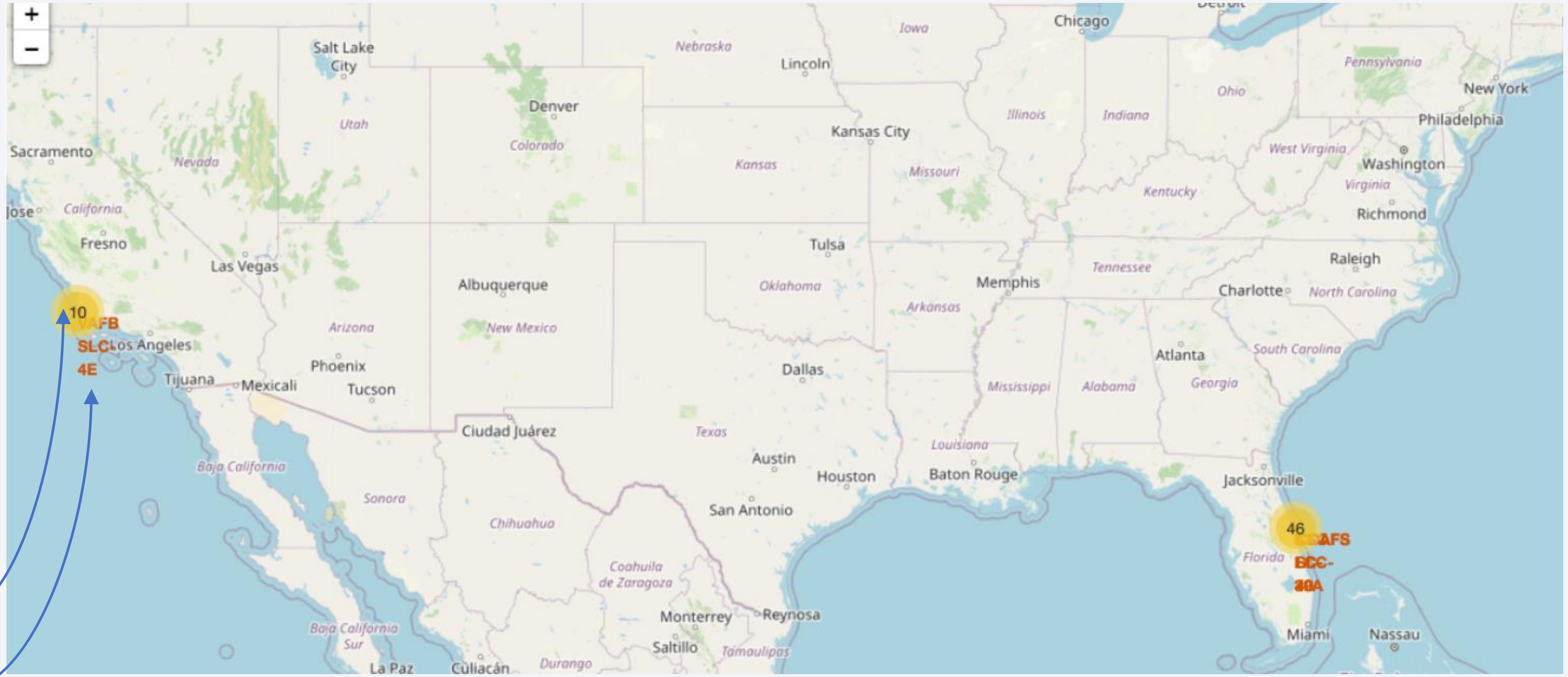
1	landing__outcome
10	No attempt
5	Failure (drone ship)
5	Success (drone ship)
3	Controlled (ocean)
3	Success (ground pad)
2	Failure (parachute)
2	Uncontrolled (ocean)
1	Precluded (drone ship)

Section 3

Launch Sites Proximities Analysis

Folium Map with all Launch Sites

- Marked and labeled all launch sites on a map
- Launch sites seem to be located on the eastern and western coast



marker

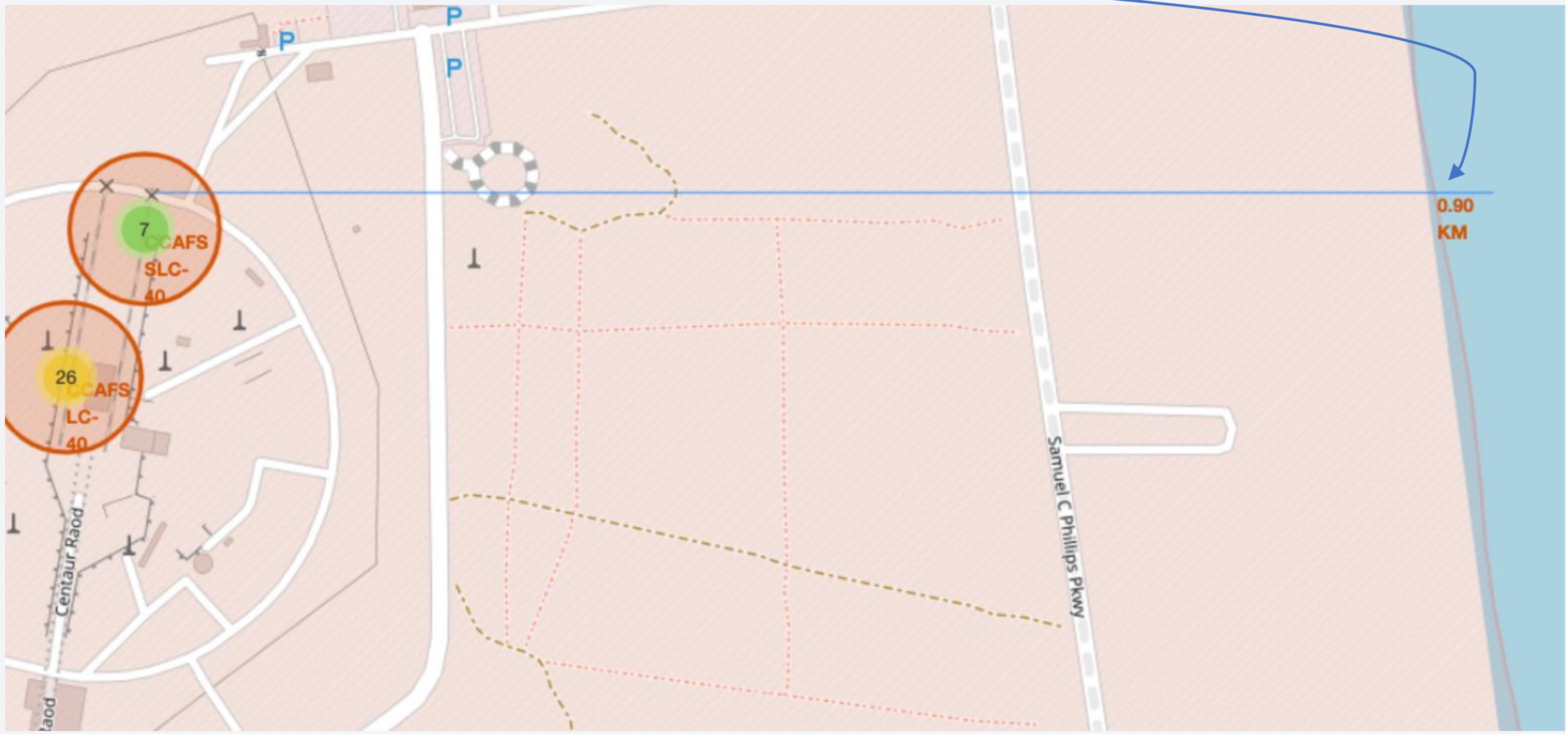
label

Folium Map with Launch Outcomes

- Launch outcomes were color-labeled for each launch site on the map (success = green, failure = red)

Folium Map with Launch Site Proximities

- Added lines marking distance between selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed
- Launch sites seem to be in close proximity to coastline, railway and highway

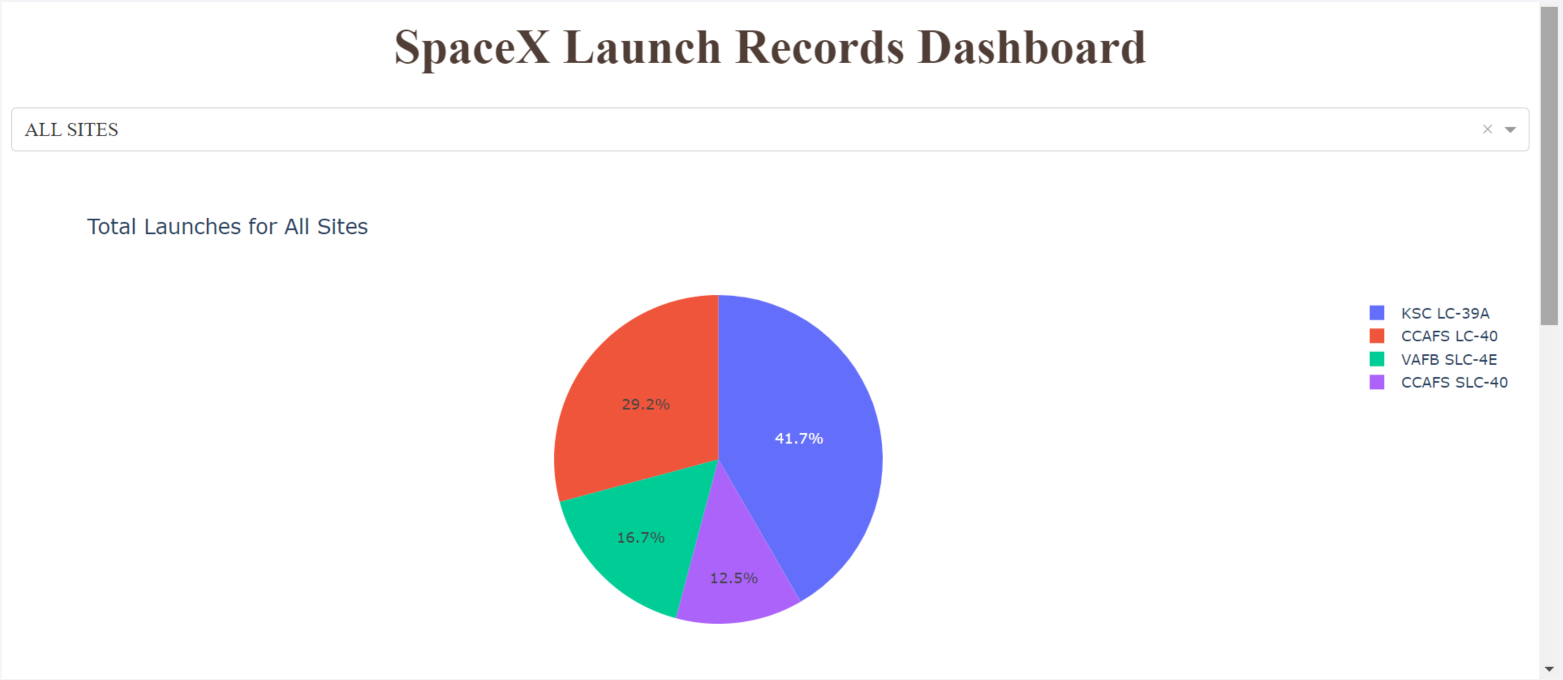


Section 4

Build a Dashboard with Plotly Dash

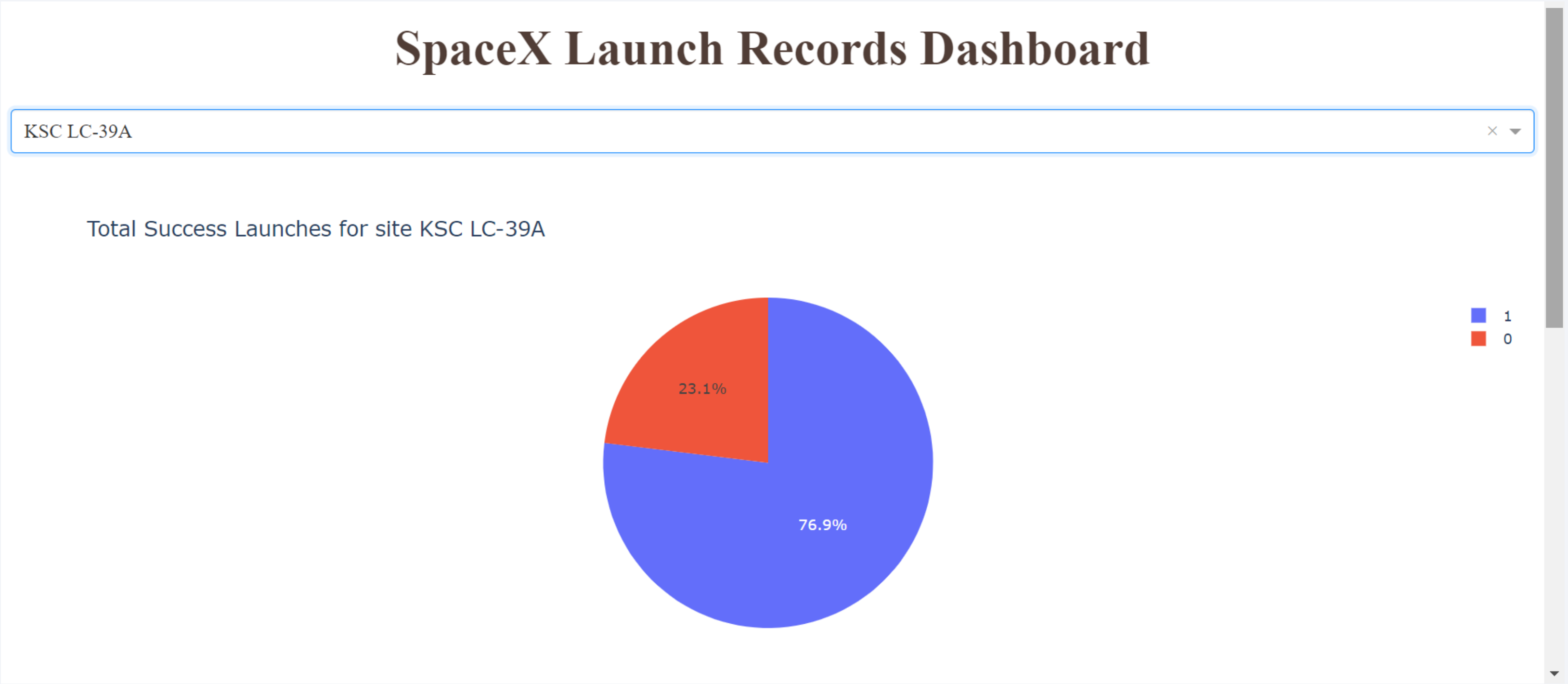
Launch Success Count for all Sites

- KSC LC-39A appears to have the highest amount of successful launches



Highest Launch Success Ratio

- KSC LC-39A has the highest launch success ratio (76.9%)



Payload vs. Launch Outcome

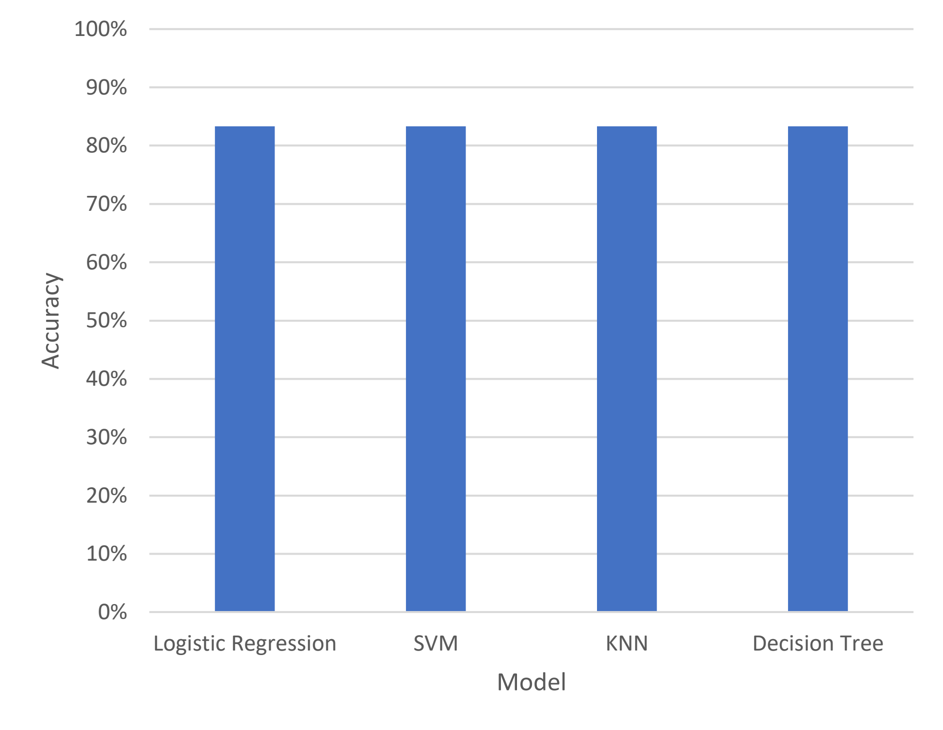
- Lower payload appears to have a larger success rate
- Booster v1.1 seems particularly successful for lower payload masses

Section 5

Predictive Analysis (Classification)

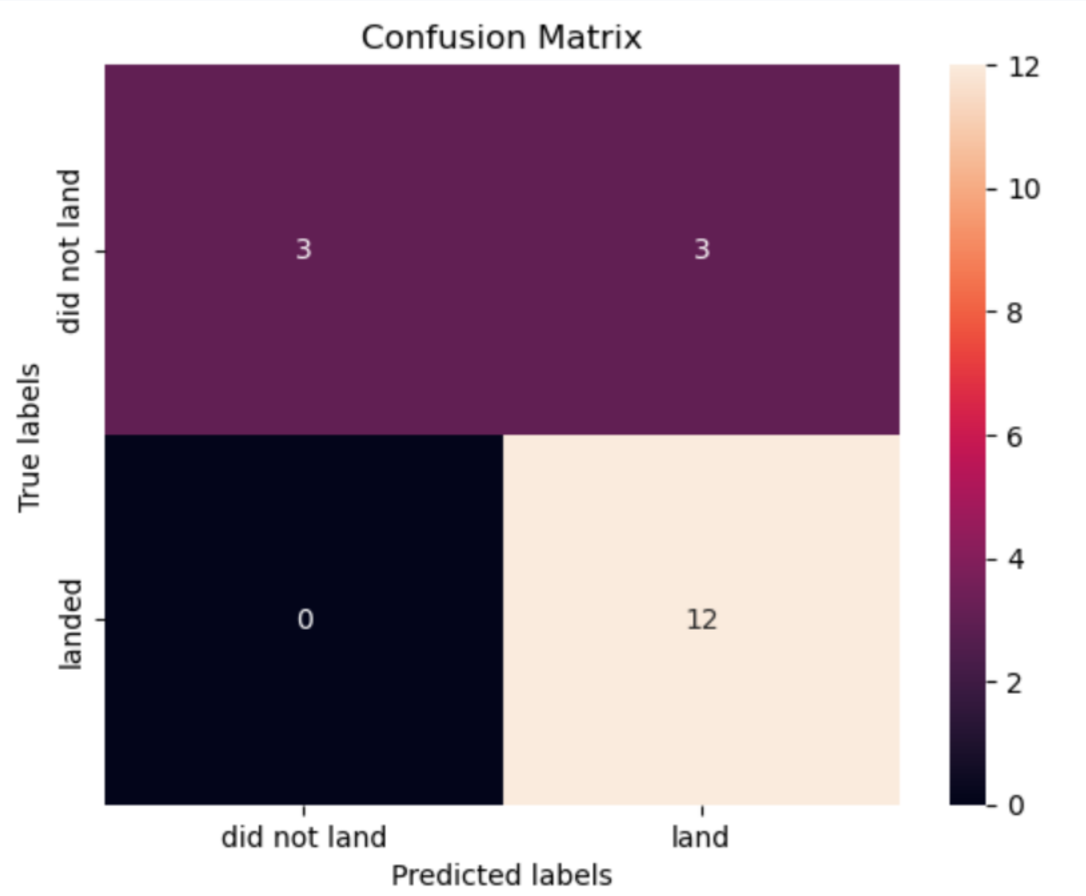
Classification Accuracy

- Built and trained Regression, SVM, KNN and Decision Tree models to predict launch outcome based on the available data
- All classification models performed equally well in accuracy (83.34%)



Confusion Matrix

- The models had perfect accuracy in predicting failed landing
- Predictions of successful landing also had good accuracy



Conclusions

- Success rate increases with flight numbers (likely due to experience)
- Launch site:
  - KSC LC 39A seems to be most successful (especially for payload masses between 2000 and 5000 kg)
- Orbit type:
  - ES-L1, GEO, HEO, and SSO orbits have the highest success rate (100%)
  - SSO seems particularly successful for low payload (< 4000 kg)
  - LEO and ISS may be most successful for payload between 4000 and 13 000 kg
  - VLEO is highly successful with high payload (> 13 000 kg)
- Payload mass:
  - Lower payload mass (< 5000 kg) appears to have a higher success rate

Appendix

All codes are available via my GitHub repository:

https://github.com/elisabethkind/AppliedDataScienceCapstone

Thank you!