Winning Space Race with Data Science

Ralf Strasser
5th June 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - EDA with SQL
  - EDA with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - EDA Results
  - Interactive Analytics Results
  - Predictive Analytics Results

Introduction

- Project background and context
  - Space X 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 Space X can reuse the first stage and therefore driving down the cost. An alternate company wants to bid against space X for a rocket launch. The goal of this project is to analyze and predict the outcome of successful first stage landings in the future.
- Problems you want to find answers
  - Identify interdependencies, factors and relationships between parameters and variables that are effecting the outcome of a successful / unsuccessful 1st stage landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Wikipedia regarding SpaceX was utilized for collecting data through REST API and Web Scrapping
- Perform data wrangling
  - Various data cleaning was performed before analysis took place
- 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

- data collection is the process of gathering, measuring, and analyzing accurate data from a variety of relevant sources to find answers to research problems, answer questions, evaluate outcomes, and forecast trends and probabilities.
- Two data collection Methods were applied:
  - REST API: rchitectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used to GET, PUT, POST and DELETE data types, which refers to the reading, updating, creating and deleting of operations concerning resources. The process started with a GET request, transformed into JSON and pandas DF. Data was cleaned, Null values substituted
  - process of extracting content and data from Wikipedia. BeautifulSoup was used to extract data from HTML into a dataframe

Data Collection – SpaceX API

- Get Request
- Json_normalize to convert to js dataframe
- Execute Date Cleaning and substituting Null values
- https://github.com/rstrasser2022/Coursera-Final-Capstone-Project---SpaceX/blob/main/spaceX%20-%20Week1%20API%20Lab.ipynb

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

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

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

Place your flowchart of SpaceX API calls

```
# 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 because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a sing
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 extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

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

Data Collection - Scraping

- Get Request
- Json_normalize to convert to js dataframe
- Execute Date Cleaning and substituting Null values

- https://github.com/rstrasser2022/Coursera-Final-Capstone-Project---SpaceX/blob/main/spaceX%20-%20Week1%20Web%20Scraping.ipynb

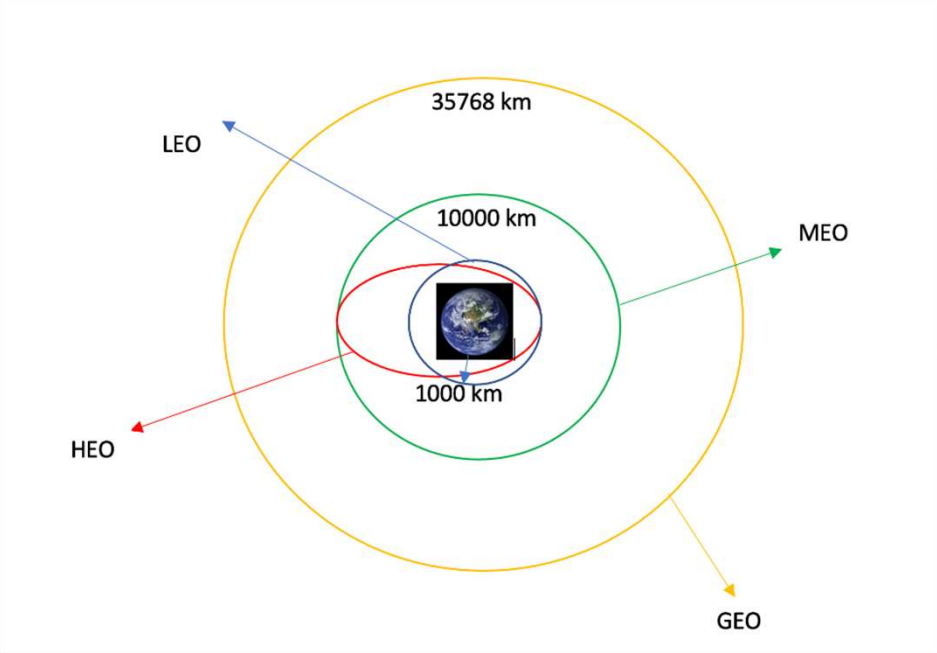
```
# use requests.get() method with the provided static_url
# assign the response to a object
data = requests.get(static_url).text
```

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

```
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')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            print(flight_number)
            datatimelist=date_time(row[0])
```

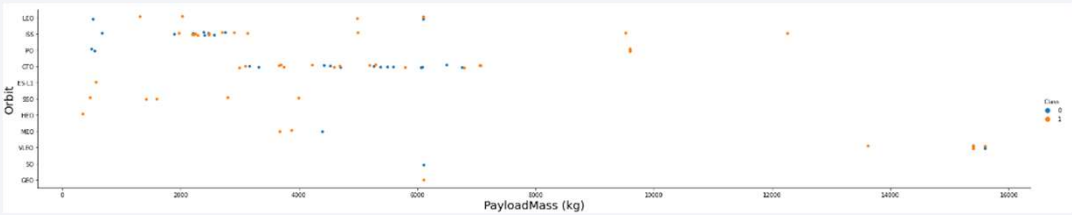
Data Wrangling

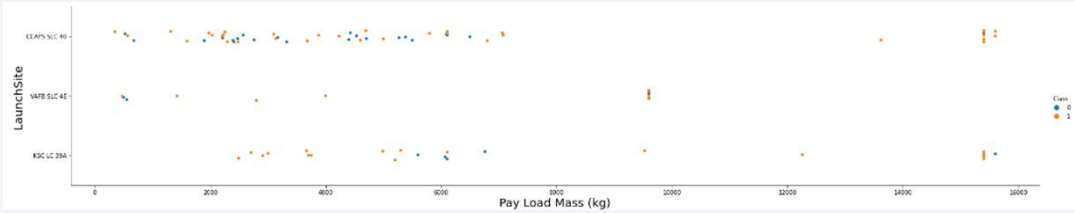
- Data wrangling—also called data cleaning, data remediation, or data munging—refers to a variety of processes designed to transform raw data into more readily used formats.
- After Data Wrangling an Exploratory Data Analysis (EDA) was performed
- First the number of launches per site were calculated and then categorized in the different dedicated orbits (see graph on right)
- https://github.com/rstrasser2022/Coursera-Final-Capstone-Project---SpaceX/blob/main/spaceX%20-%20Week1%20Data%20Wrangling.ipynb

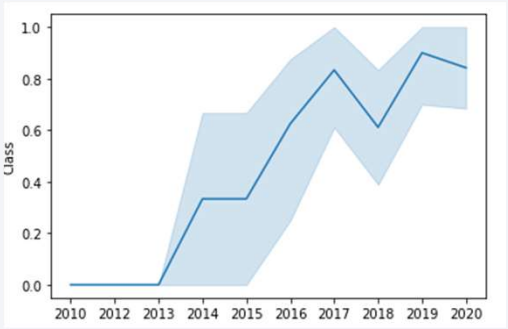


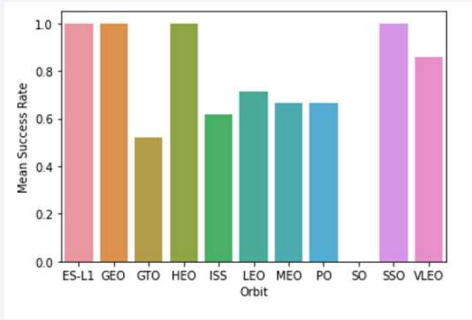
EDA with Data Visualization

- First Scatterplots were created to visualize relationships between:
  - Flight Number and Launch Site
  - Payload and Launch Site
  - Flight Number and Orbit Type
  - Payload and Orbit Type
- Once initial relationships were established bar graph and line plots were generated to analyze success rates
  - Success rate and orbit type
  - Launch Success versus yearly trend
- https://github.com/rstrasser2022/Coursera-Final-Capstone-Project---SpaceX/blob/main/spaceX%20-%20Week2%20EDA%20with%20Data%20Visualization.ipynb









EDA with SQL

- Summary of SQL Queries:
  - 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 acheived.
  - 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. Use a subquery
  - 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/rstrasser2022/Coursera-Final-Capstone-Project---SpaceX/blob/main/spaceX%20-%20Week2%20EDA%20SQL.ipynbSpaceX/blob/main/spaceX%20-%20Week2%20EDA%20SQL.ipynb

```
%sql select distinct launch_site from SPACEXTBL
```

```
%sql select * from SPACEXTBL where launch_site like 'CCA%'
```

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where customer = 'NASA (CRS)'
```

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where booster_version like 'F9 v1.1%'
```

```
%sql select min(date) from SPACEXTBL where MISSION_OUTCOME = 'Success'
```

```
%sql select booster_version from SPACEXTBL where MISSION_OUTCOME = 'Success' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
```

```
%sql select mission_outcome, count(*) from SPACEXTBL group by mission_outcome
```

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

```
%sql select landing__outcome, booster_version, launch_site from SPACEXTBL where date like '2015%'
```

```
%sql select landing__outcome, count(*) from (select * from SPACEXTBL where date between '2010-06-04' and '2017-03-20') group by landing_outcome order
```

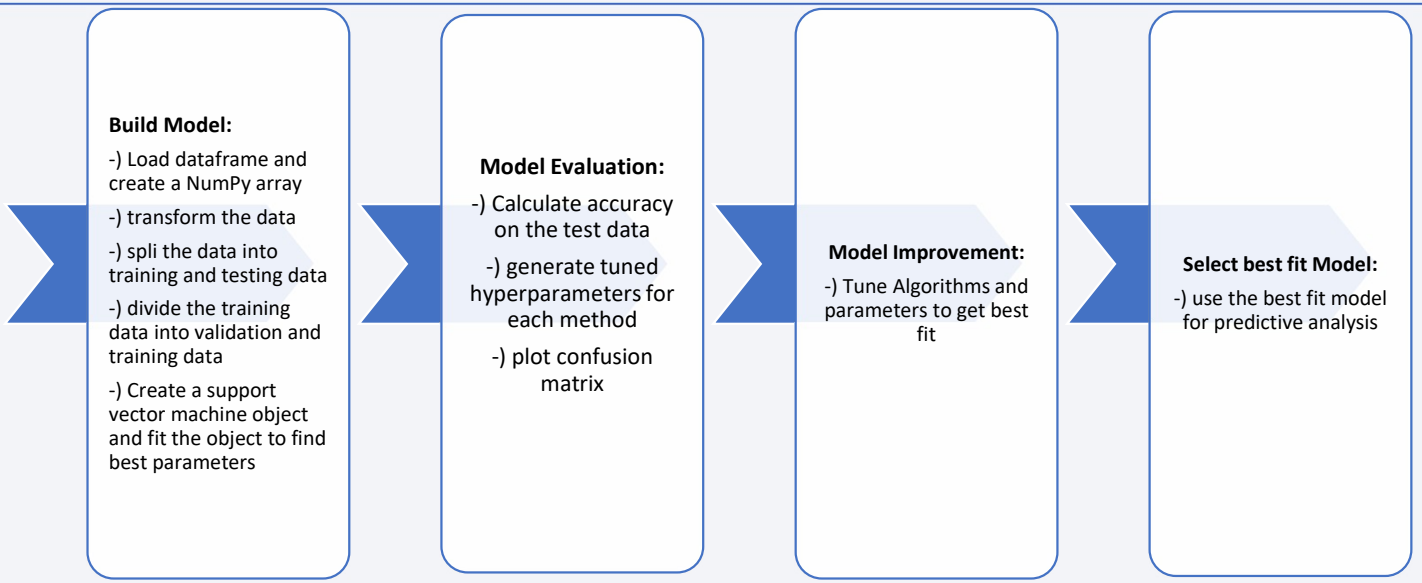
Build an Interactive Map with Folium

- Summarize of map objects created and added to a folium map
  - An interactive map was created and NASA space center initial LAT/LONG added.
  - After for each lunch site a circle was added
  - Added the success/failed launches for each site on the map, with a green circle if success and red if failed launch
  - Distances between launch sites to its proximities were calculated (highways, railways, coastlines and proximity to cities)
- https://github.com/rstrasser2022/Coursera-Final-Capstone-Project---SpaceX/blob/main/spaceX%20-%20Week3%20Interactive%20Visual%20Analytics.ipynb

Build a Dashboard with Plotly Dash

- An interactive dashboard with plotly dash was build.
  - A dropdown for all available launch sites was created
  - A pie chart to show successful launches count for all sites was added
  - Scatter charts showed the correlation between payload and launch success. Sliders for payload were added.
- https://github.com/rstrasser2022/Coursera-Final-Capstone-Project---SpaceX/blob/main/Spacex_plotly_app.py

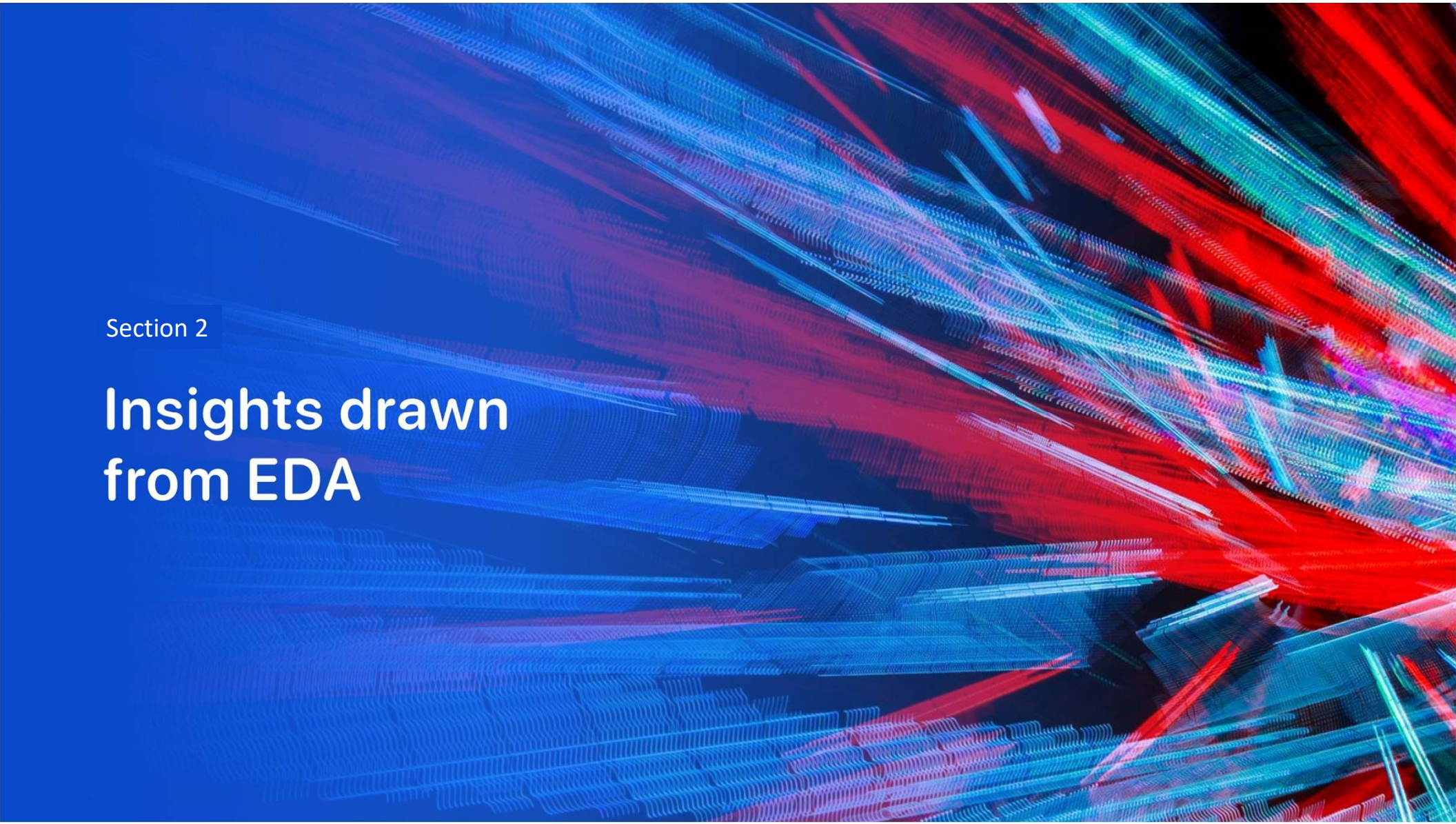
Predictive Analysis (Classification)



- https://github.com/rstrasser2022/Coursera-Final-Capstone-Project---SpaceX/blob/main/SpaceX%20-%20Week%204%20-%20Machine%20Learning.ipynb

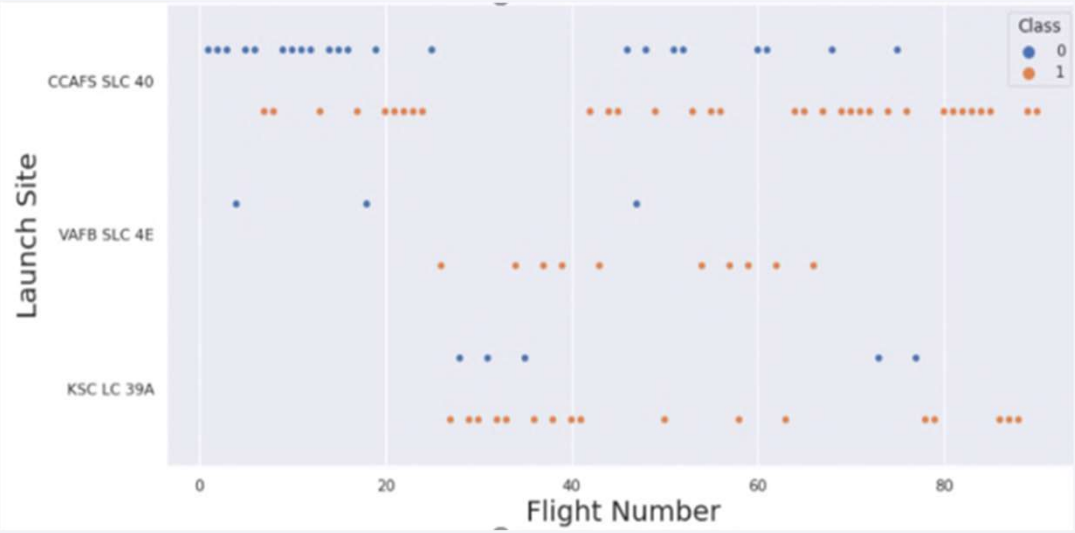
Results

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


Section 2
Insights drawn
from EDA

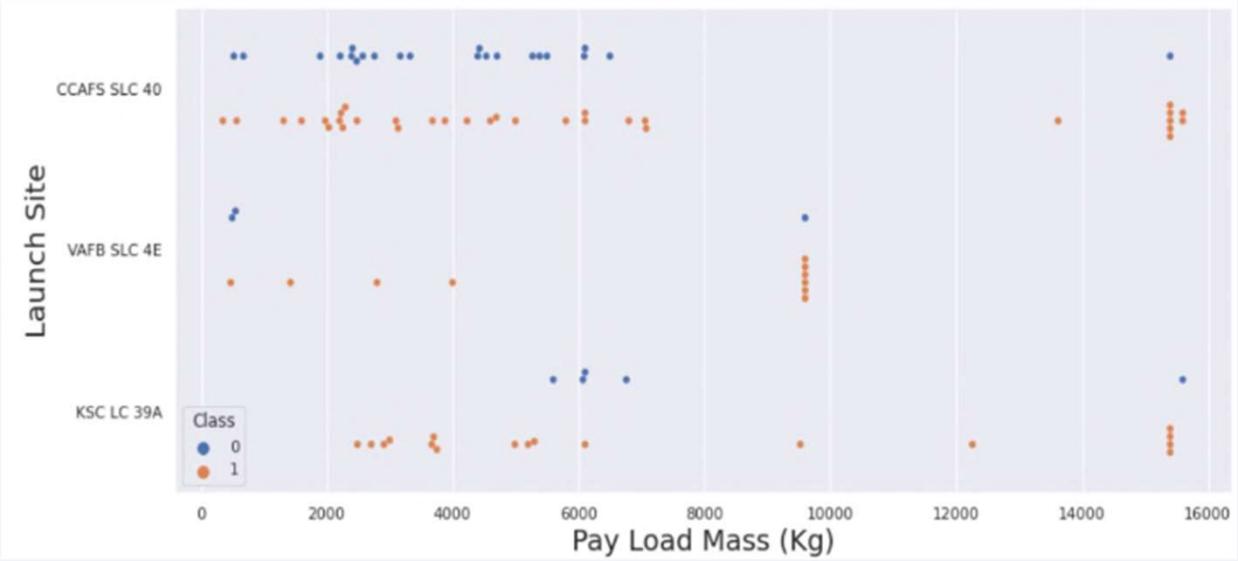
Flight Number vs. Launch Site

- scatter plot shows that most successful launches took off from CCAFS site
- Other two sits show generally a higher successful launch with an increasing number of flights



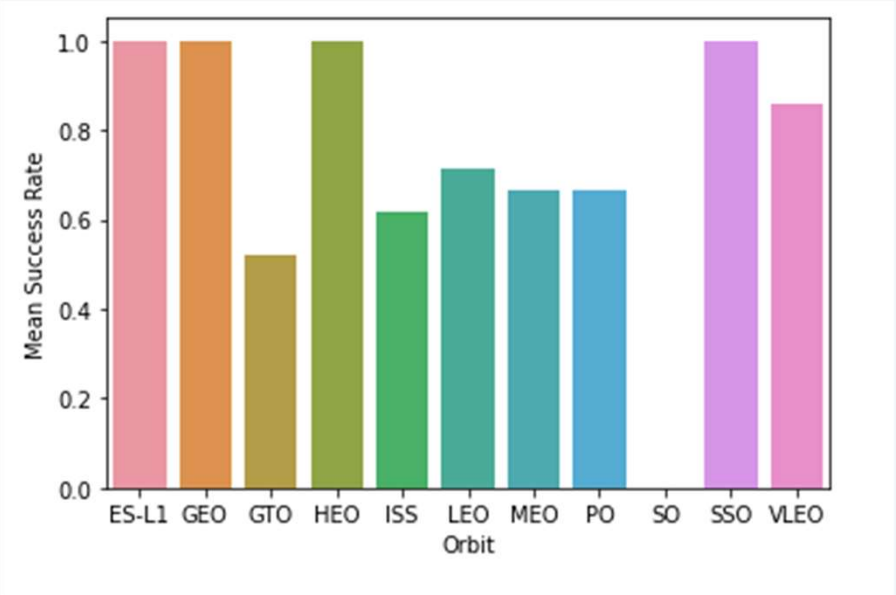
Payload vs. Launch Site

- No strong correlation successful launch correlation with launch site nor payload
- It appears there are overall more successful launches above 8000kgs



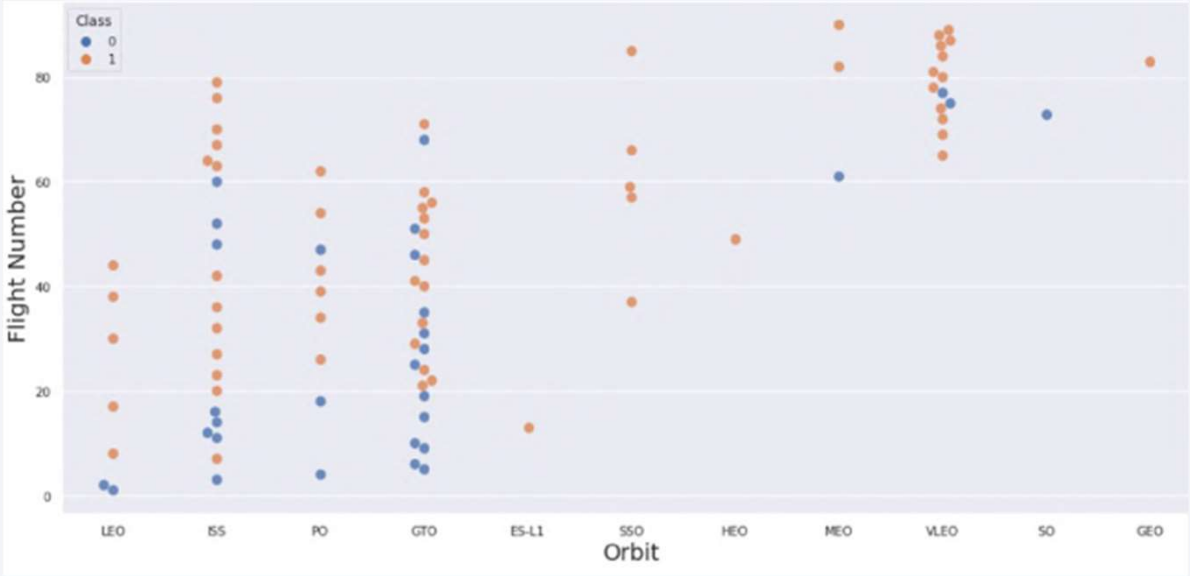
Success Rate vs. Orbit Type

- 100% success rates:
  - in Orbit ES, GEO, SSO, HEO
  - But ES, HEO, GEO have only 1 datapoint
- 0% success rate in orbit SO. Which has only 1 data point and is statistically not significant



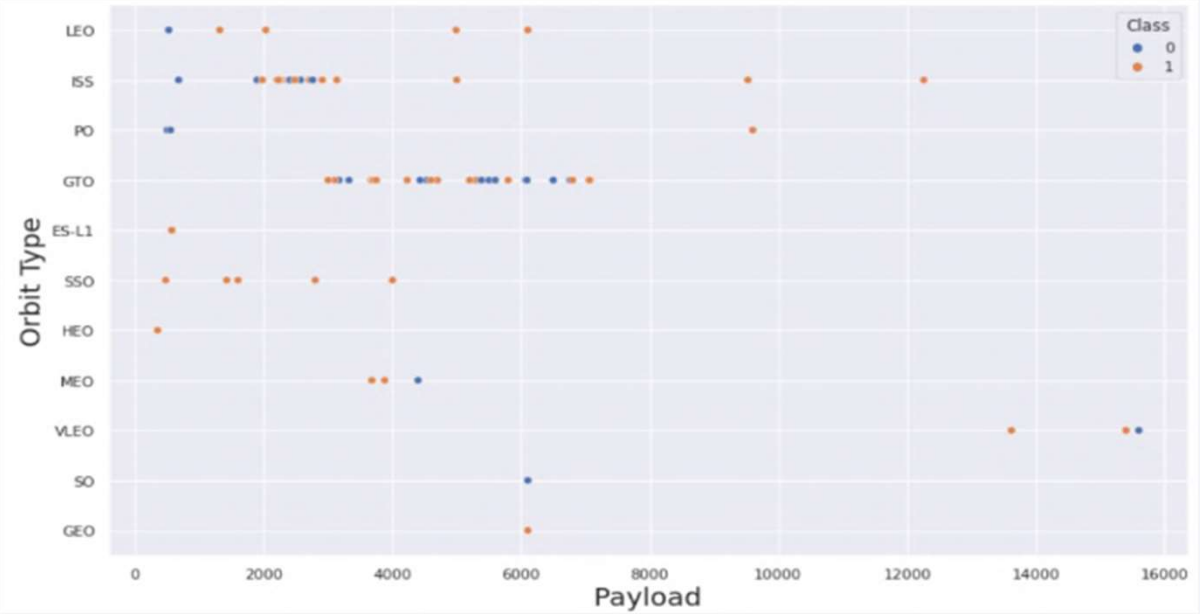
Flight Number vs. Orbit Type

- Generally a trend of higher success rates with high flight numbers. GTO seems to be an exception
- Some orbits have only 1 or few datapoints and have to be considered separately
- Approx. 50% of orbits have very few flights



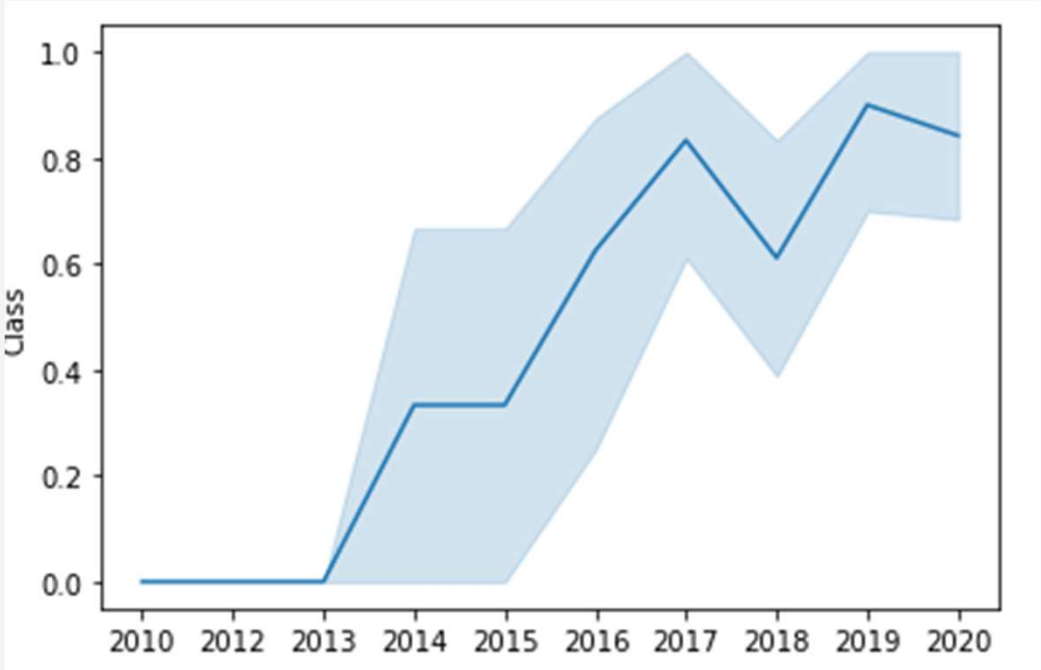
Payload vs. Orbit Type

- Clear separation of low and higher payload flights at around 8000kg
- Only few flights above 8000kg
- SSO orbit is the only orbit with more than one datapoint and 100% success
- Large payloads appear to be more successful in certain orbits (e.g. LEO, ISS)



Launch Success Yearly Trend

- Overall trend of increasing success over time
- With a reduced success rate in 2018 (more investigation required)
- Overall the trend is asymptotically approaching a 100% success rate



All Launch Site Names

- DISTINCT is used to ID unique launch sites

```
%sql select distinct launch_site from SPACEXTBL
```

```
 * ibm_db_sa://ndz14400:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
   sqlite:///SpaceEx.sqlite
Done.
```

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

Launch Site Names Begin with 'CCA'

- LIKE is used to query for CCA sites
- LIMIT to be used for limiting the number of records

```
%sql select * from SPACEXTBL where launch_site like 'CCA%'
```

```
 * ibm_db_sa://ndz14400:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
   sqlite:///SpaceEx.sqlite
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

Total Payload Mass

- SUM is used to sum up all payload mass for NASA as a costumer

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://ndz14400:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
   sqlite:///SpaceEx.sqlite
Done.
```

1
45596

Average Payload Mass by F9 v1.1

- AVG is used to calculate the average payload for all flights for F9 v.1.1%
- To find only F9 v1.1 LIKE was used

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where booster_version like 'F9 v1.1%'
```

```
 * ibm_db_sa://ndz14400:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
   sqlite:///SpaceEx.sqlite
Done.
   1
2534
```

First Successful Ground Landing Date

- MIN was used to identify the smallest date WHERE mission outcome was successful

```
%sql select min(date) from SPACEXTBL where MISSION_OUTCOME = 'Success'
```

```
 * ibm_db_sa://ndz14400:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
   sqlite:///SpaceEx.sqlite
Done.
        1
2010-06-04
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- WHERE used to query for successful landings only
- AND, > and < was used to query for payload mass larger 4000 smaller 6000

```
%sql select booster_version from SPACEXTBL where MISSION_OUTCOME = 'Success' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
```

```
 * ibm_db_sa://ndz14400:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
   sqlite:///SpaceEx.sqlite
Done.
```

booster_version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 FT B1032.2
F9 B4 B1040.2
F9 B5 B1046.2

Total Number of Successful and Failure Mission Outcomes

- COUNT to count number of missions from SPACEXTBL
- GROUP by mission outcome which is failure, success and success (payload status unclear)

```
%sql select mission_outcome, count(*) from SPACEXTBL group by mission_outcome
```

```
 * ibm_db_sa://ndz14400:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
   sqlite:///SpaceEx.sqlite
Done.
```

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

Boosters Carried Maximum Payload

- Subquery with WHERE was used (with MAX) to query for maximum carried payload mass per booster version.

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

```
 * ibm_db_sa://ndz14400:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
   sqlite:///SpaceEx.sqlite
Done.
```

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

2015 Launch Records

- Query for landing outcome, booster version, launch site using WHERE DATA LIKE to return only 2015 launches

```
%sql select landing__outcome, booster_version, launch_site from SPACEXTBL where date like '2015%'
```

```
* ibm_db_sa://ndz14400:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
  sqlite:///SpaceEx.sqlite
Done.
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Controlled (ocean)	F9 v1.1 B1013	CCAFS LC-40
No attempt	F9 v1.1 B1014	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
No attempt	F9 v1.1 B1016	CCAFS LC-40
Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40
Success (ground pad)	F9 FT B1019	CCAFS LC-40

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

```
%sql select landing__outcome, count(*) from (select * from SPACEXTBL where date between '2010-06-04' and '2017-03-20') group by landing_outcome order
```

- I did not succeed in answering that question

Section 3

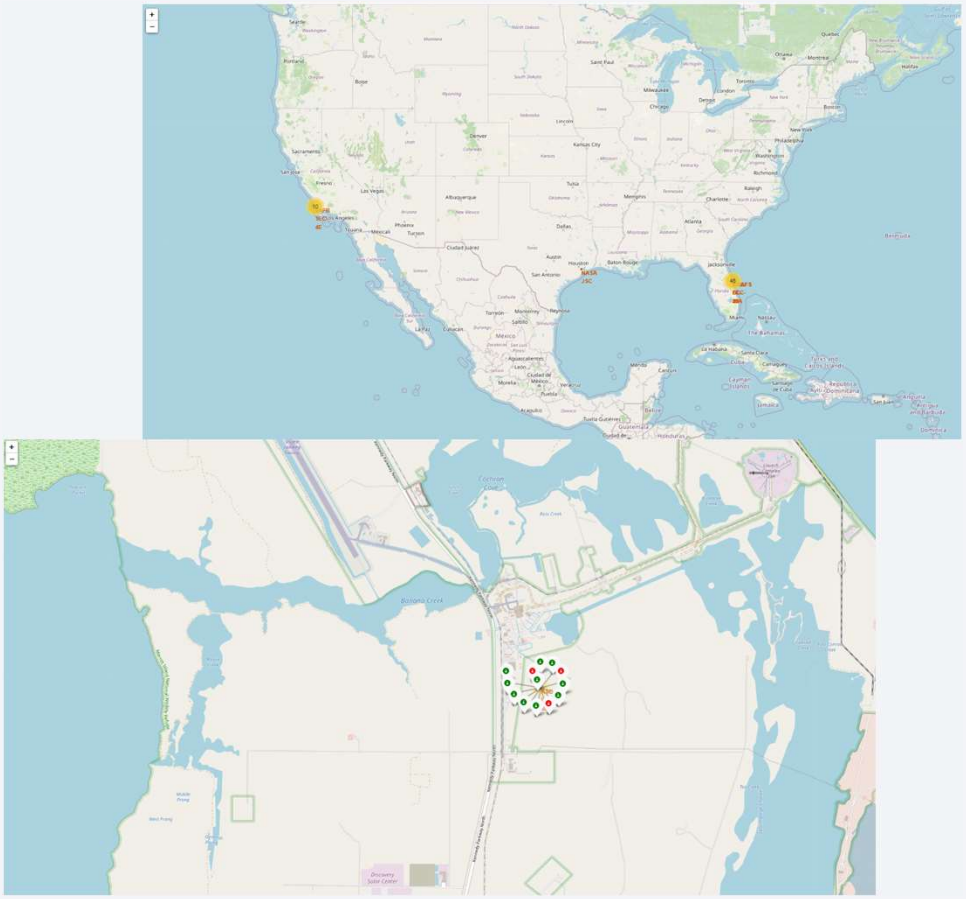
Launch Sites Proximities Analysis

All Marked Launch Sites Map within US

- Only LA, Houston and Florida launch sites within the US
- Site name and marker are shown

Launch Site map with Success/Failure Color Labels

- A Marker cluster was generated to show the number of launches per site
- Zooming in to a particular Florida site reveals the color coding of successful/unsuccessful launches

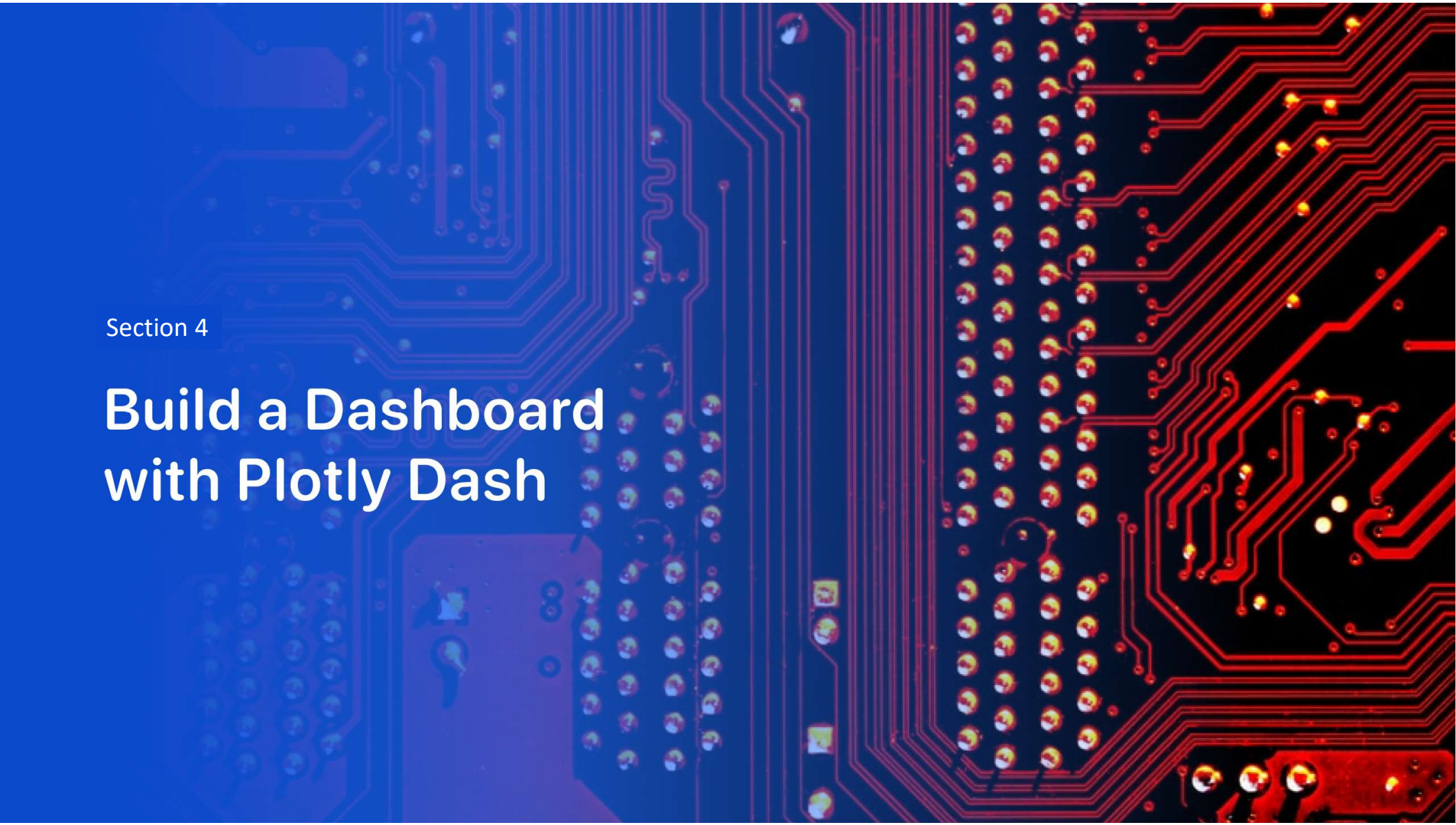


Distance to proximities

- Distance to other proximities were calculated (see code on right)
- Markerlines could not be generated

```
# find coordinate of the closet coastline
# e.g.,: Lat: 28.56367  Lon: -80.57163
# distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
distance_coastline = calculate_distance(28.563197,-80.576820, 28.56057,-80.56773)
distance_coastline
```

```
0.9348702016299324
```

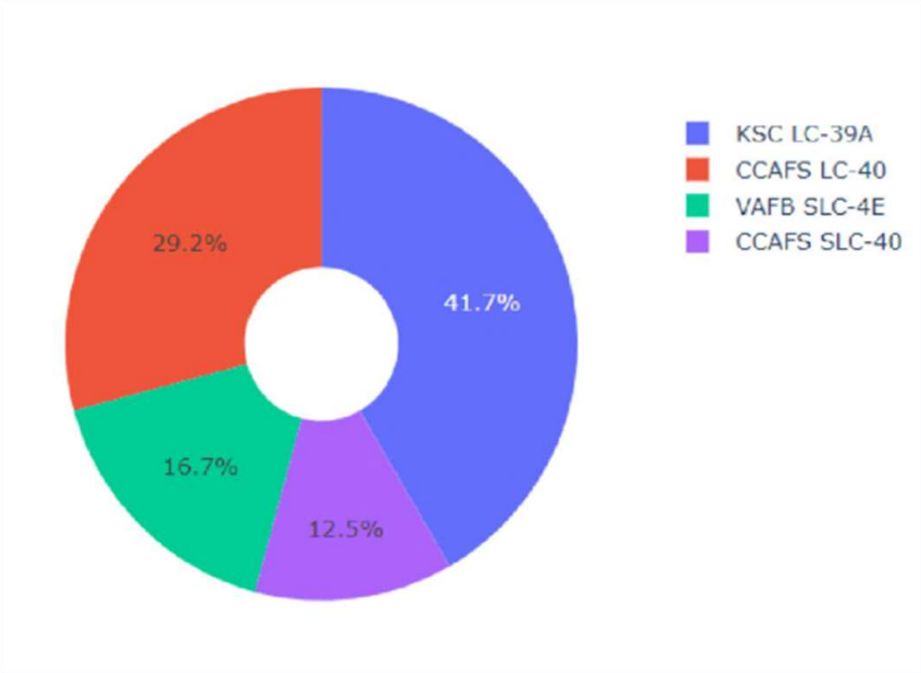


Section 4

Build a Dashboard with Plotly Dash

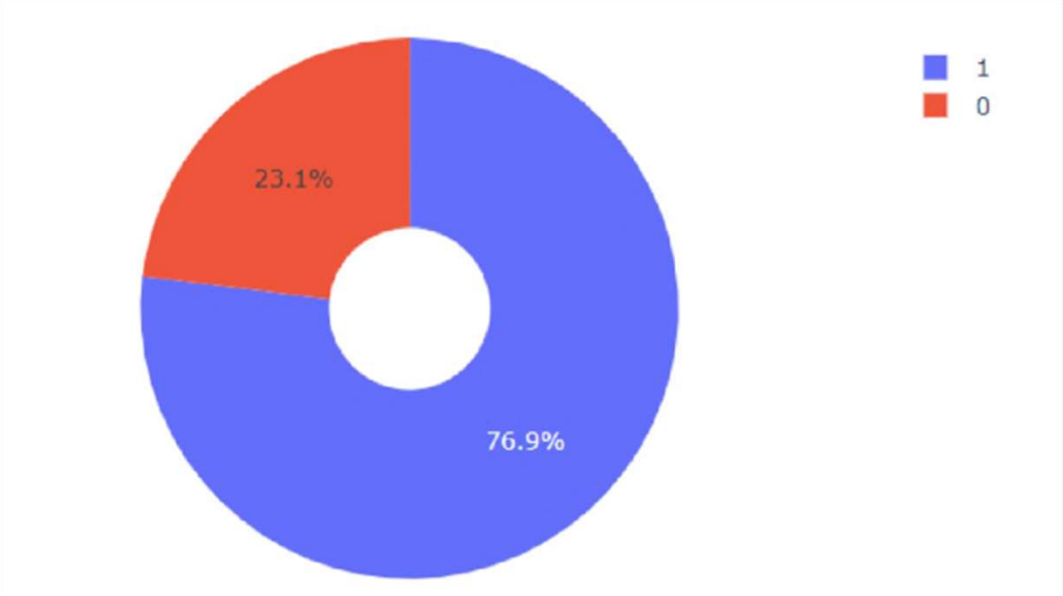
Success Rate vs. Launch Site

- KSC has highest number of launches



Launch Site with highest number of successful launches

- ~77% of launches from KSC LC-39A were successful



Payload vs. Launch Result Scatter Plot

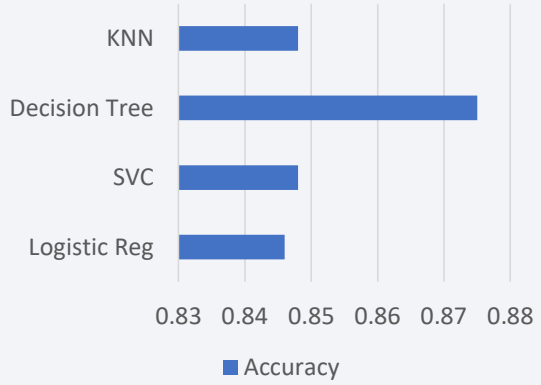
- Overall it was found that lower payloads in particular between 0-4000kg had a higher success rate than 4000-10000kg

Section 5

Predictive Analysis (Classification)

Classification Accuracy

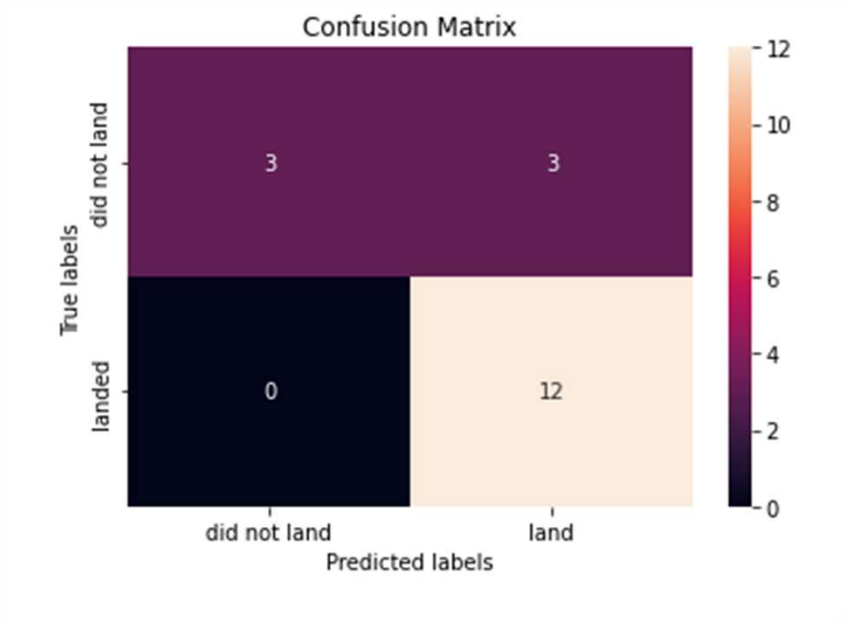
- The Decision Tree shows the highest model accuracy



	Model	Accuracy	Prediction score
0	LogisticRegression()	0.8464285714285713	0.8333333333333334
1	SVC()	0.8482142857142856	0.8333333333333334
2	DecisionTreeClassifier()	0.875	0.8333333333333334
3	KNeighborsClassifier()	0.8482142857142858	0.8333333333333334

Confusion Matrix

- The decision tree confusion matrix shows:
  - False positive is high, i.e. predicting 3 landed versus in reality they failed



Conclusions

- Decision Tree performed best against other models
- KSC has the highest number of launches with a 77% success rate
- Overall landing success rates increased steadily from 2013 onwards with a dip in 2018
- SSO orbit has the highest success rate with a significant amount of data points.
- Lower payloads (<4000kg) had a higher chance of success than higher payloads

Thank you!