IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Anandan Raju
March 31st 2023

Outline

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

Executive Summary

The goal of this research is to analyze SpaceX Falcon 9 data collected through various sources and employ Machine Learning models to predict the success of first stage landing that provides other space agencies the ability to decide if they bid against SpaceX.

- **Summary of methodologies**
  - Following concepts and methods were used to collect and analyze data, build and evaluate machine learning models, and make predictions:
    - Collect data through API and Web scraping
    - Transform data through data wrangling
    - Conduct exploratory data analysis with SQL and data visuals
    - Build an interactive map with folium to analyze launch site proximity
    - Build a dashboard to analyze launch records interactively with Plotly Dash
    - Finally, build a predictive model to predict if the first stage of Falcon 9 will land successfully
- **Summary of all results**
  - This report will share results in various formats such as:
    - Data analysis results
    - Data visuals, interactive dashboards
    - Predictive model analysis results

Introduction

- **Project background and context**
  - With the recent successes in private space travel, space industry is becoming more and more mainstream and accessible to general population. Cost of launch continues to remain a key barrier for new competitors to enter the space race
  - SpaceX with its first stage reuse capabilities offers a key advantage against its competitors. Each SpaceX launch costs around 62 million dollar and SpaceX can reuse stage 1 for future launches. This provides SpaceX a unique advantage where other competitors are spending around 165 mission plus for each launch
- **Problems you want to find answers**
  - Determine if the first stage of SpaceX Falcon 9 will land successfully
  - Impact of different parameters/variables on the landing outcomes (e.g., launch site, payload mass, booster version, etc.)
  - Correlations between launch sites and success rates

Section 1

Methodology

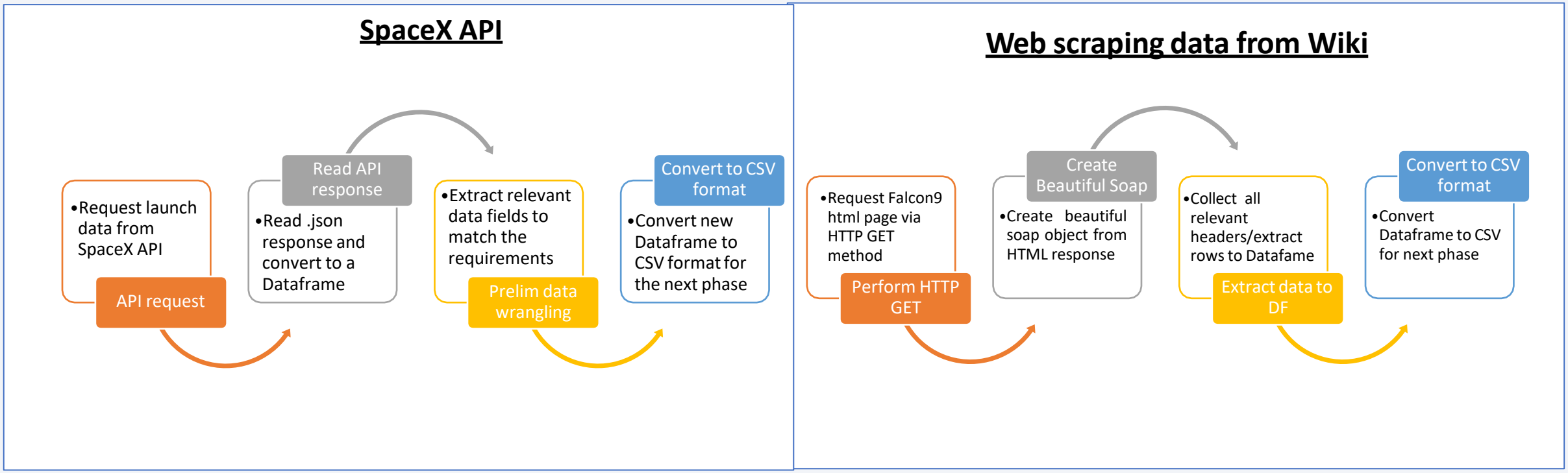
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX API
  - Web scrap Falcon 9 and Falcon Heavy launch records from Wikipedia (link)
- Perform data wrangling
  - Determined labels for training the supervised models by converting mission outcomes in to training labels (0-unsuccessful, 1-successful)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Created a column for 'class'; standardized and transformed data; train/test split data; find best classification algorithm (Logistic regression, SVM, decision tree, & KNN) using test data

Data Collection

- Data collection is the process of gathering data from available sources. This data can be structured, unstructured, or semi-structured. For this project, data was collected via SpaceX API and Web scrapping Wiki pages for relevant launch data.



Data Collection - SpaceX API

GitHub

1. API Request and read response into DF
2. Declare global variables
3. Call helper functions with API calls to populate global vars
4. Construct data using dictionary
5. Convert Dict to Dataframe, filter for Falcon9 launches, covert to CSV

1. Create API GET request, normalize data and read in to a Dataframe:

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

response = requests.get(spacex_url)

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

2. Declare global variable lists that will store data returned by helper functions with additional API calls to get relevant data

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

3. Call helper functions to get relevant data where columns have IDs (e.g., rocket column is an identification number)
   - getBoosterVersion(data)
   - getLaunchSite(data)
   - getPayloadData(data)
   - getCoreData(data)

4. Construct dataset from received data & combine columns into a dictionary:

```
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}
```

5. Create Dataframe from dictionary and filter to keep only the Falcon9 launches:

```
# Create a data from launch_dict
df_launch = pd.DataFrame(launch_dict)

# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df_launch[df_launch['BoosterVersion']!= 'Falcon 1']

data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

Data Collection - Scraping

GitHub

1. Perform HTTP GET to request HTML page
2. Create Beautiful Soap object
3. Extract column names from HTML table header
4. Create Dictionary with keys from extracted column names
5. Call helper functions to fill up dict with launch records
6. Convert Dictionary to Dataframe

1. Create API GET method to request Falcon9 launch HTML page

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

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

2. Create Beautiful Soap object

```
soup = BeautifulSoup(html_data,"html.parser")
```

3. Find all the tables on the Wiki page and extract relevant column names from the HTML table header

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

```
column_names = []

# Apply find_all() function with `th` element on firs
# Iterate each th element and apply the provided extr
# Append the Non-empty column name (`if name is not N
colnames = soup.find_all('th')
for x in range (len(colnames)):
    name2 = extract_column_from_header(colnames[x])
    if (name2 is not None and len(name2) > 3):
        column_names.append(name2)
```

4. Create an empty Dictionary with keys from extracted column names:

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each vc
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

5. Fill up the launch_dict with launch records extracted from table rows.
   - Utilize following helper functions to help parse HTML data

```
def date_time(table_cells):
```

```
def booster_version(table_cells):
```

```
def landing_status(table_cells):
```

```
def get_mass(table_cells):
```

6. Convert launch_dict to Dataframe:

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

Data Wrangling

GitHub

- Conducted Exploratory Data Analysis (EDA) to find patters in data and define labels for training supervised models
- The data set contained various mission outcomes that were converted into Training Labels with 1 meaning the booster successfully landed and 0 meaning booster was unsuccessful in landing. Following landing scenarios were considered to create labels:
  - True Ocean means the mission outcome was successfully landed to a specific region of the ocean
  - False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean
  - RTLS means the mission outcome was successfully landed to a ground pad
  - False RTLS means the mission outcome was unsuccessfully landed to a ground pad
  - True ASDS means the mission outcome was successfully landed on a drone ship
  - False ASDS means the mission outcome was unsuccessfully landed on a drone ship

Data Wrangling - cont'd

GitHub

1. Load dataset in to Dataframe
2. Find patterns in data
3. Create landing outcome label

1. Load SpaceX dataset (csv) in to a Dataframe

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appd
art_1.csv")
```

2. Find data patterns:
   i. Calculate the number of launches on each site

```
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40     55
KSC LC 39A       22
VAFB SLC 4E      13
```

   ii. Calculate the number and occurrence of each orbit `df['Orbit'].value_counts()`

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
GEO       1
HEO       1
SO        1
ES-L1     1
```

   iii. Calculate number/occurrence of mission outcomes per orbit type

```
landing_outcomes = df['Outcome'].value_counts()
```

3. Create a landing outcome label from Outcome column in the Dataframe

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

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

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0

EDA with Data Visualization

GitHub

As part of the Exploratory Data Analysis (EDA), following charts were plotted to gain further insights into the dataset:

1. Scatter plot:
   - Shows relationship or correlation between two variables making patterns easy to observe
   - Plotted following charts to visualize:
     - Relationship between Flight Number and Launch Site
     - Relationship between Payload and Launch Site
     - Relationship between Flight Number and Orbit Type
     - Relationship between Payload and Orbit Type

2. Bar Chart:
   - Commonly used to compare the values of a variable at a given point in time. Bar charts makes it easy to see which groups are highest/common and how other groups compare against each other. Length of each bar is proportional to the value of the items that it represents
   - Plotted following Bar chart to visualize:
     - Relationship between success rate of each orbit type

3. Line Chart:
   - Commonly used to track changes over a period of time. It helps depict trends over time.
   - Plotted following Line chart to observe:
     - Average launch success yearly trend

EDA with SQL

GitHub

- To better understand SpaceX data set, following SQL queries/operations were performed on an IBM DB2 cloud instance:

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was achieved.
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes
8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
9. List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
10. 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

Build an Interactive Map with Folium

GitHub

- Folium interactive map helps analyze geospatial data to perform more interactive visual analytics and better understand factors such location and proximity of launch sites that impact launch success rate.
- Following map object were created and added to the map:
  - Mark all launch sites on the map. This allowed to visually see the launch sites on the map.
    - Added 'folium.circle' and 'folium.marker' to highlight circle area with a text label over each launch site.
  - Added a 'MarkerCluster()' to show launch success (green) and failure (red) markers for each launch site.
  - Calculated distances between a launch site to its proximities (e.g., coastline, railroad, highway, city)
    - Added 'MousePosition() to get coordinate for a mouse position over a point on the map
    - Added 'folium.Marker()' to display distance (in KM) on the point on the map (e.g., coastline, railroad, highway, city)
    - Added 'folium.Polyline()' to draw a line between the point on the map and the launch site
    - Repeated steps above to add markers and draw lines between launch sites and proximities – coastline, railroad, highway, city)
- Building the Interactive Map with Folium helped answered following questions:
  - Are launch sites in close proximity to railways?
    - YES
  - Are launch sites in close proximity to highways?
    - YES
  - Are launch sites in close proximity to coastline?
    - YES
  - Do launch sites keep certain distance away from cities?
    - YES

Build a Dashboard with Plotly Dash

GitHub

- Built a Plotly Dash web application to perform interactive visual analytics on SpaceX launch data in real-time. Added Launch Site Drop-down, Pie Chart, Payload range slide, and a Scatter chart to the Dashboard.
    1. Added a Launch Site Drop-down Input component to the dashboard to provide an ability to filter Dashboard visual by all launch sites or a particular launch site
    2. Added a Pie Chart to the Dashboard to show total success launches when 'All Sites' is selected and

       show success and failed counts when a particular site is selected
    3. Added a Payload range slider to the Dashboard to easily select different payload ranges to identify visual patterns
    4. Added a Scatter chart to observe how payload may be correlated with mission outcomes for selected site(s). The color-label Booster version on each scatter point provided missions outcomes with different boosters
- Dashboard helped answer following questions:
    1. Which site has the largest successful launches? KSCLC-39A with 10
    2. Which site has the highest launch success rate? KSCLC-39A with 76.9% success
    3. Which payload range(s) has the highest launch success rate? 2000 - 5000 kg
    4. Which payload range(s) has the lowest launch success rate? 0-2000 and 5500 - 7000
    5. Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate? FT

Predictive Analysis (Classification)

GitHub

1. Read dataset into Dataframe and create a 'Class' array
2. Standardize the data
3. Train/Test/Split data in to training and test data sets
4. Create and Refine Models
5. Find the best performing Model

1. Load SpaceX dataset (csv) in to a Dataframe and create NumPy array from the column class in data

```
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object
et_part_2.csv")
```

```
Y = data['Class'].to_numpy()
```

2. Standardize data in X then reassign to variable X using transform

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

3. Train/test/split X and Y in to training and test data sets.

```
# Split data for training and testing data sets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split
( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

4. Create and refine Models based on following classification Algorithms: (below is LR example)

i. **Create Logistic Regression object and then create a GridSearchCV object**

ii. **Fit train data set in to the GridSearchCV object and train the Model**

```
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)
```

ii. **Find and display best hyperparameters and accuracy score**

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

iii. **Check the accuracy on the test data by creating a confusion matrix**

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

iv. **Repeat above steps for Decision Tree, KNN, and SVM algorithms**

3. Find the best performing model

```
Model_Performance_df = pd.DataFrame({'Algo Type': ['Logistic Regression', 'SVM','Decision Tree','KNN'],
'Accuracy Score': [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_],
'Test Data Accuracy Score': [logreg_cv.score(X_test, Y_test), svm_cv.score(X_test, Y_test),
tree_cv.score(X_test, Y_test), knn_cv.score(X_test, Y_test)]})
```

```
i = Model_Performance_df['Accuracy Score'].idxmax()
print('The best performing alogrithm is '+ Model_Performance_df['Algo Type'][i]
+ ' with score ' + str(Model_Performance_df['Accuracy Score'][i]))
```

```
The best performing alogrithm is Decision Tree with score 0.875
```

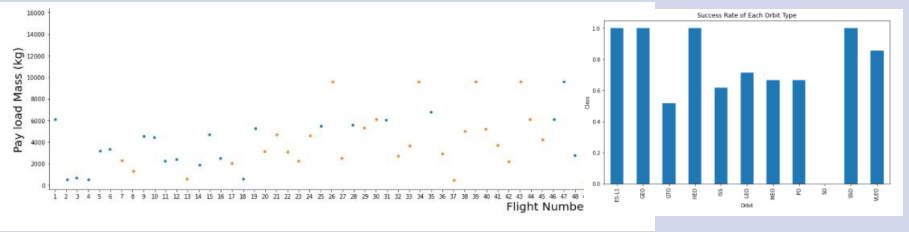
	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

Results

Following sections and slides explain results for:

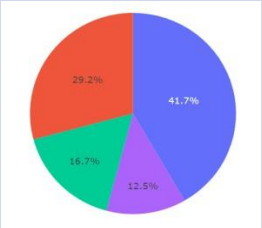
Exploratory data analysis results

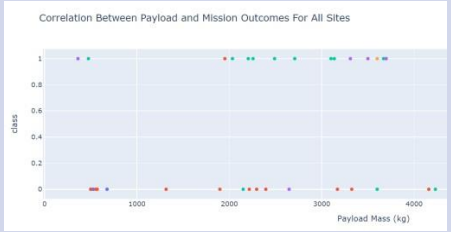
- Samples:



Interactive analytics demo in screenshots

- Samples





Predictive analysis results

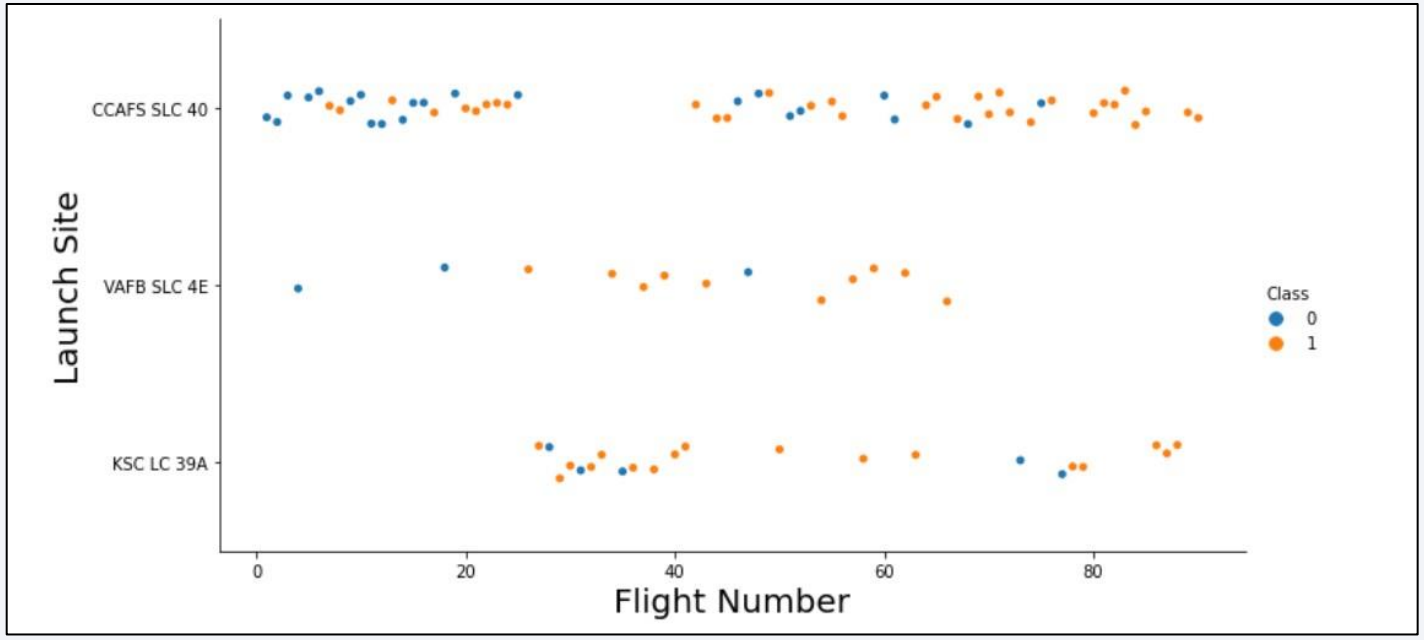
- Samples

	Algo Type	Accuracy Score
2	Decision Tree	0.903571
3	KNN	0.848214
1	SVM	0.848214
0	Logistic Regression	0.846429

Section 2

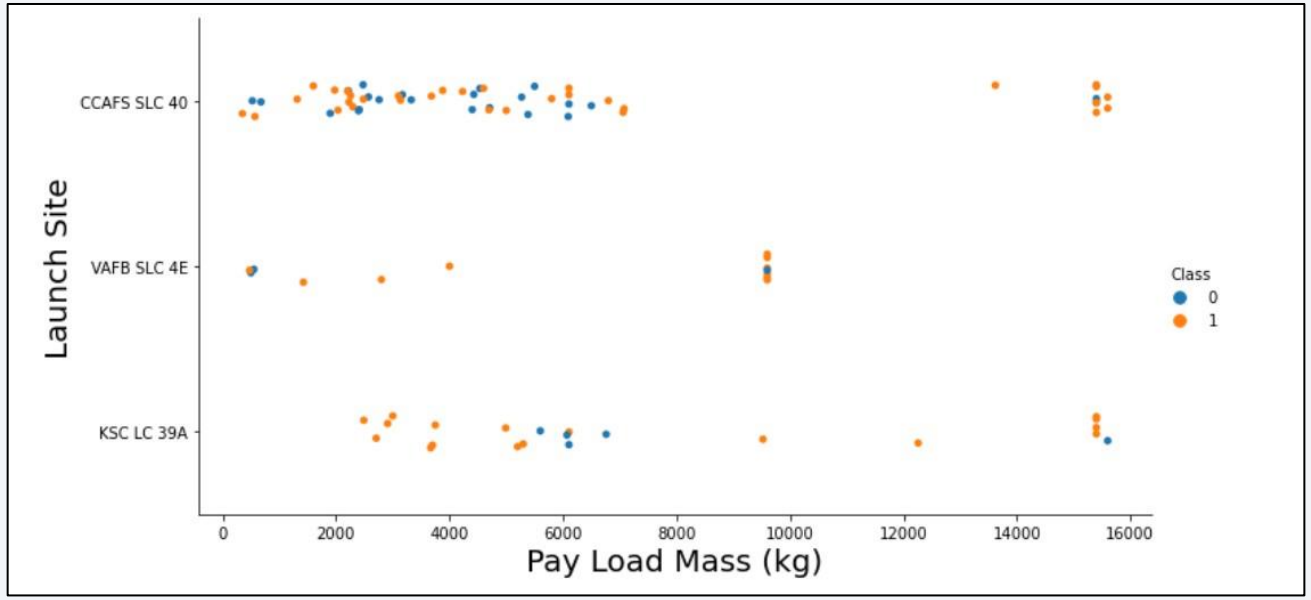
Insights drawn from EDA

Flight Number vs. Launch Site



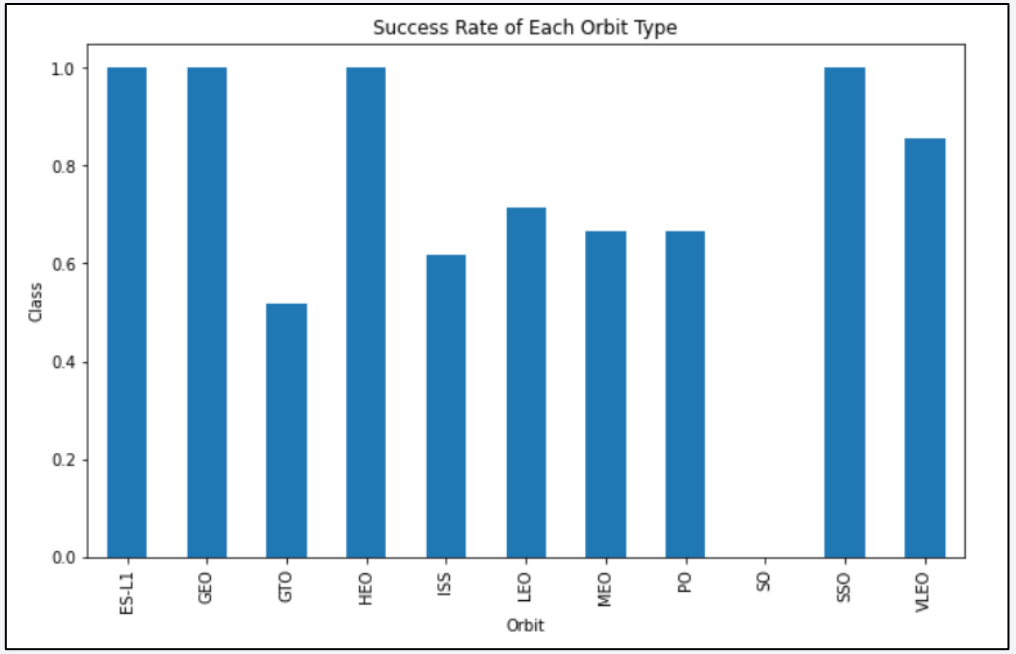
- Success rates (Class=1) increases as the number of flights increase
- For launch site 'KSC LC 39A', it takes at least around 25 launches before a first successful launch

Payload vs. Launch Site



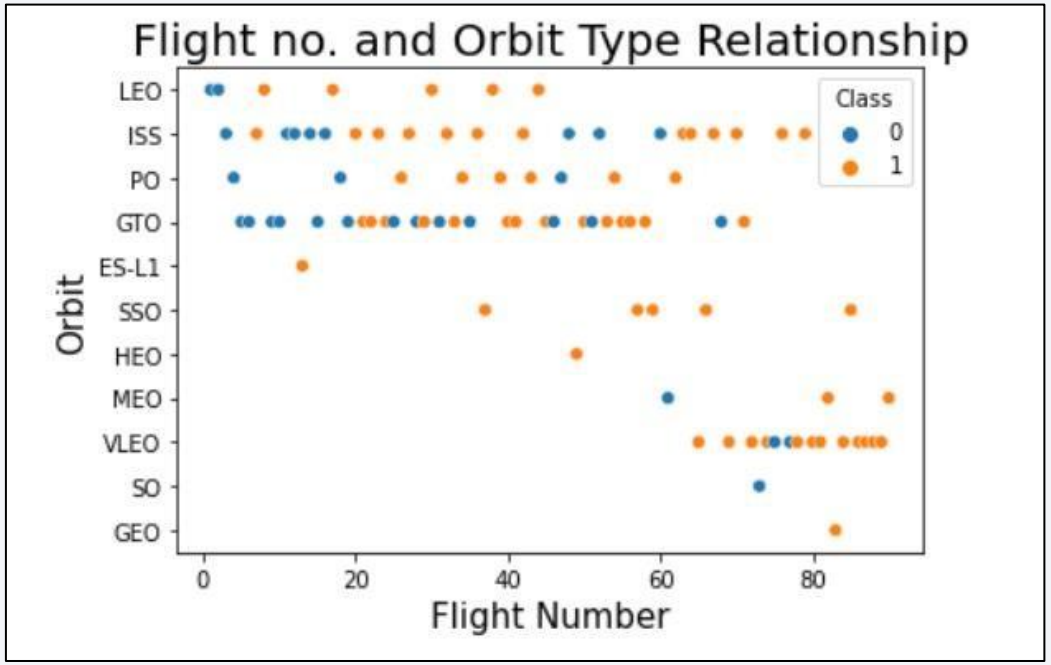
- For launch site ‘VAFB SLC 4E’, there are no rockets launched for payload greater than 10,000 kg
- Percentage of successful launch (Class=1) increases for launch site ‘VAFB SLC 4E’ as the payload mass increases
- There is no clear correlation or pattern between launch site and payload mass

Success Rate vs. Orbit Type



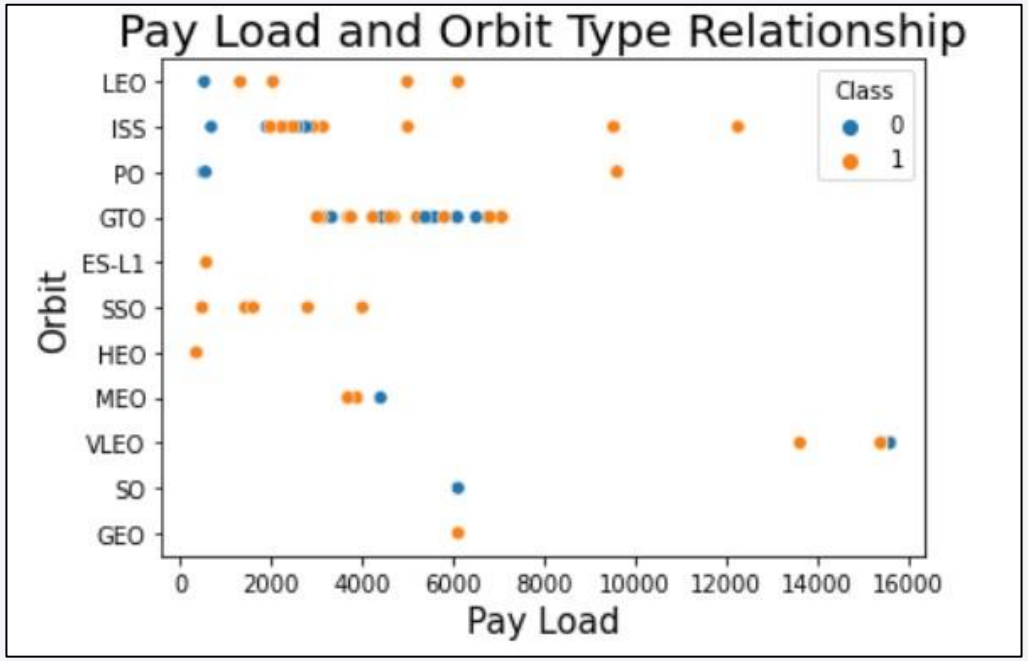
- Orbits ES-LI, GEO, HEO, and SSO have the highest success rates
- GTO orbit has the lowest success rate

Flight Number vs. Orbit Type



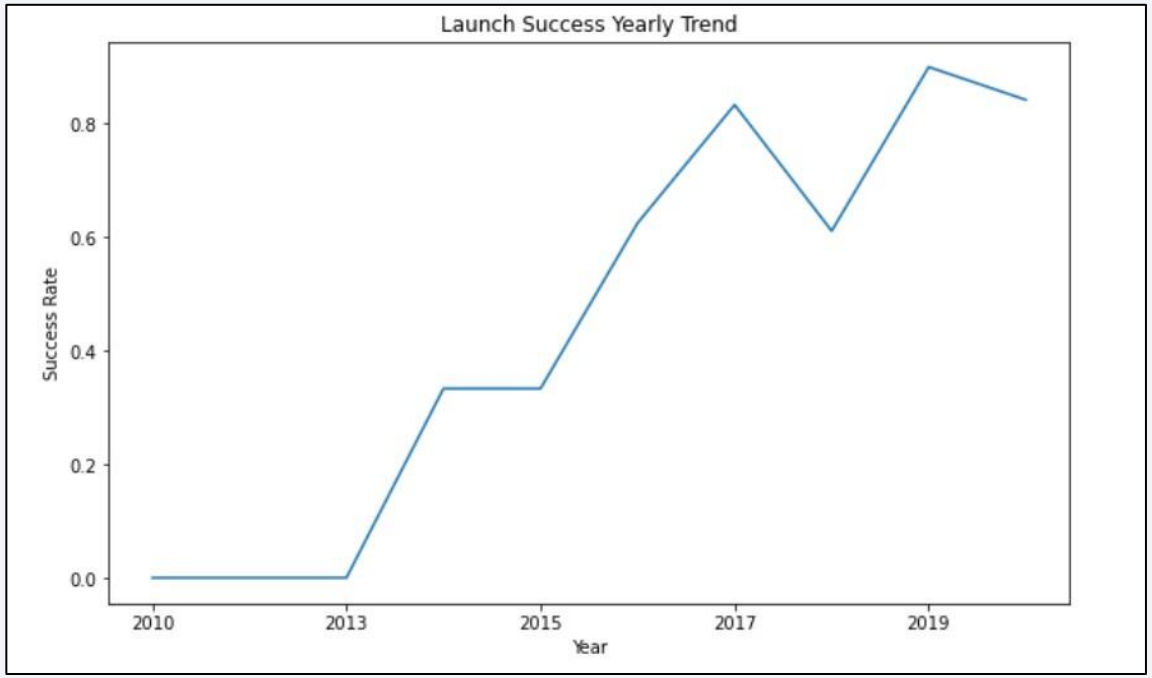
- For orbit VLEO, first successful landing (class=1) doesn't occur until 60+ number of flights
- For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers
- There is no relationship between flight number and orbit for GTO

Payload vs. Orbit Type



- Successful landing rates (Class=1) appear to increase with pay load for orbits LEO, ISS, PO, and SSO
- For GEO orbit, there is not clear pattern between payload and orbit for successful or unsuccessful landing

Launch Success Yearly Trend



- Success rate (Class=1) increased by about 80% between 2013 and 2020
- Success rates remained the same between 2010 and 2013 and between 2014 and 2015
- Success rates decreased between 2017 and 2018 and between 2019 and 2020

All Launch Site Names

- Query:

  select distinct launch_site from spacextbl

- Description:
  - ‘distinct’ returns only unique values from the queries column (Launch_Site)
  - There are 4 unique launch sites

- Result:

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

Launch Site Names Begin with 'CCA'

- Query:

```
select * from spacextbl where Launch_Site LIKE 'CCA%' limit 5;
```

- Description:
  - Using keyword ‘Like’ and format ‘CCA%’, returns records where ‘Launch_Site’ column starts with “CCA”.
  - Limit 5, limits the number of returned records to 5

- Result:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Query:

```
select sum(PAYLOAD_MASS__KG_) from spacextbl where Customer = 'NASA (CRS)'
```

- Description:
  - ‘sum’ adds column ‘PAYLOAD_MASS_KG’ and returns total payload mass for customers named ‘NASA (CRS)’

- Result:

45596

Average Payload Mass by F9 v1.1

- Query:

```
select avg(PAYLOAD_MASS__KG_) from spacextbl where Booster_Version LIKE 'F9 v1.1'
```

- Description:
  - ‘avg’ keyword returns the average of payload mass in ‘PAYLOAD_MASS_KG’ column where booster version is ‘F9 v1.1’

- Result:

2928

First Successful Ground Landing Date

- Query:

```
select min(Date) as min_date from spacextbl where Landing__Outcome = 'Success (ground pad)';
```

- Description:
  - ‘min(Date)’ selects the first or the oldest date from the ‘Date’ column where first successful landing on group pad was achieved
  - Where clause defines the criteria to return date for scenarios where ‘Landing_Outcome’ value is equal to ‘Success (ground pad)’
- Result:

min_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Query:

```
select Booster_Version from spacextbl where (PAYLOAD_MASS__KG_> 4000 and PAYLOAD_MASS__KG_ < 6000)
and (Landing__Outcome = 'Success (drone ship)');
```

- Description:
  - The query finds the booster version where payload mass is greater than 4000 but less than 6000 and the landing outcome is success in drone ship
  - The ‘and’ operator in the where clause returns booster versions where both conditions in the where clause are true

- 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

- Query:

```
select Mission_Outcome, count(Mission_Outcome) as counts from spacextbl group by Mission_Outcome
```

- Description:
  - The ‘group by’ keyword arranges identical data in a column in to group
  - In this case, number of mission outcomes by types of outcomes are grouped in column ‘counts’

- Result:

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

Boosters Carried Maximum Payload

- Query:

```
select Booster_Version, PAYLOAD_MASS__KG_ from spacextbl where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextbl)
```

- Description:

  - The sub query returns the maximum payload mass by using keywork ‘max’ on the pay load mass column
  - The main query returns booster versions and respective payload mass where payload mass is maximum with value of 15600

- 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

2015 Launch Records

- Query:

```
select Landing__Outcome, Booster_Version, Launch_Site from spacextbl where Landing__Outcome = 'Failure (drone ship)' and year(Date) = '2015'
```

- Description:
  - The query lists landing outcome, booster version, and the launch site where landing outcome is failed in drone ship and the year is 2015
  - The 'and' operator in the where clause returns booster versions where both conditions in the where clause are true
  - The 'year' keywork extracts the year from column 'Date
  - The results identify launch site as 'CCAFS LC-40' and booster version as F9 v1.1 B1012 and B1015 that had failed landing outcomes in drop ship in the year 2015

- Result:

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

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

- Query:

```
select Landing__Outcome, count(*) as LandingCounts from spacextbl where Date between '2010-06-04' and '2017-03-20'
group by Landing__Outcome
order by count(*) desc;
```

- Description:
  - The ‘group by’ key word arranges data in column ‘Landing Outcome’ into groups
  - The ‘between’ and ‘and’ keywords return data that is between 2010-06-04 and 2017-03-20
  - The ‘order by’ keyword arranges the counts column in descending order
  - The result of the query is a ranked list of landing outcome counts per the specified date range

- Result:

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

Section 4

Launch Sites Proximities Analysis

SpaceX Falcon9 - Launch Sites Map

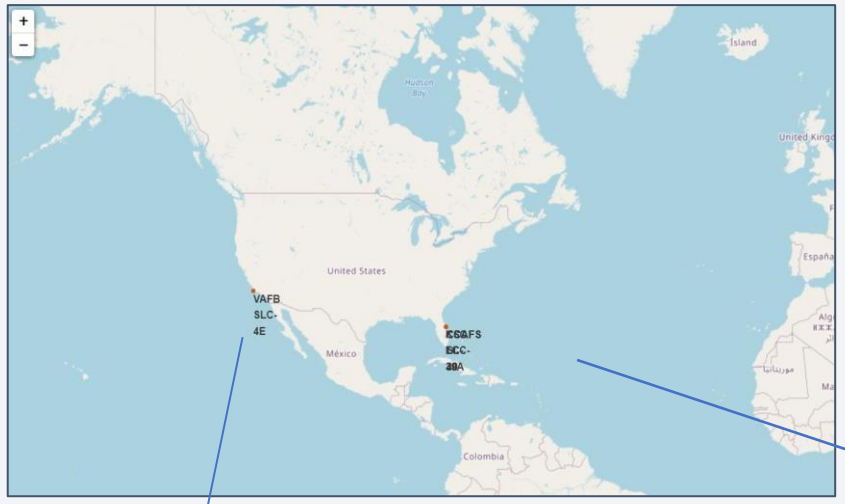


Fig 1 – Global Map

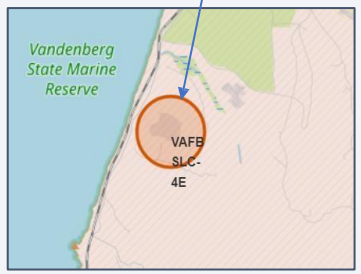


Fig 2 – Zoom 1

Figure 1 on left displays the Global map with Falcon 9 launch sites that are located in the United States (in California and Florida). Each launch site contains a circle, label, and a popup to highlight the location and the name of the launch site. It is also evident that all launch sites are near the coast.

Figure 2 and Figure 3 zoom in to the launch sites to display 4 launch sites:

- VAFB SLC-4E (CA)
- CCAFS LC-40 (FL)
- KSC LC-39A (FL)
- CCAFS SLC-40 (FL)

Fig 3 – Zoom 2

SpaceX Falon9 - Success/Failed Launch Map for all Launch Sites

Fig 1 – US map with all Launch Sites

- Figure 1 is the US map with all the Launch Sites. The numbers on each site depict the total number of successful and failed launches
- Figure 2, 3, 4, and 5 zoom in to each site and displays the success/fail markers with green as success and red as failed
- By looking at each site map, KSC LC-39A Launch Site has the greatest number of successful launches

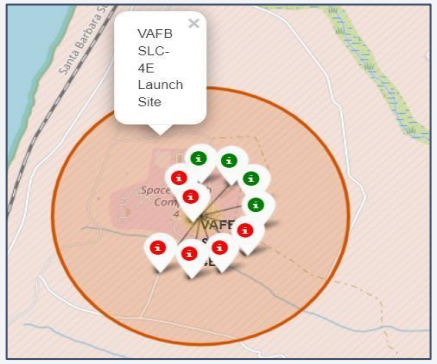


Fig 2 – VAFB Launch Site with success/failed markers

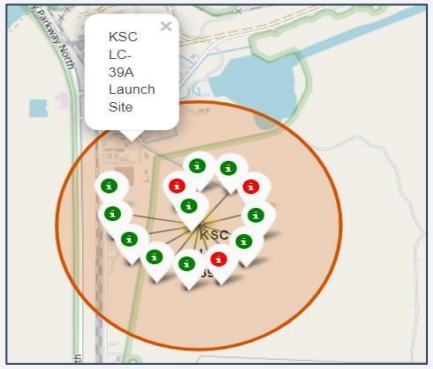


Fig 3 – KSC LC-39A success/failed markers

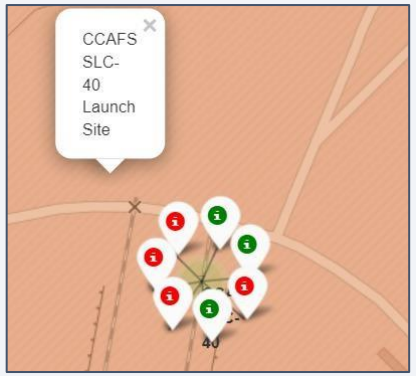


Fig 4 – CCAFS SLC-40 success/failed markers

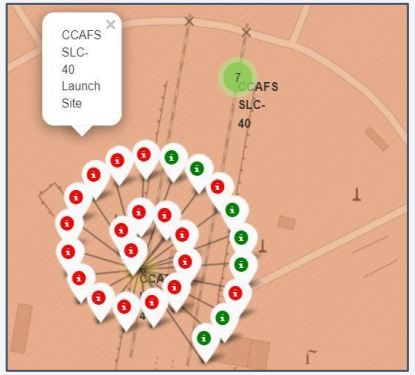


Fig 5 – CCAFS SLC-40 success/failed markers

SpaceX Falcon9 - Launch Site to proximity Distance Map

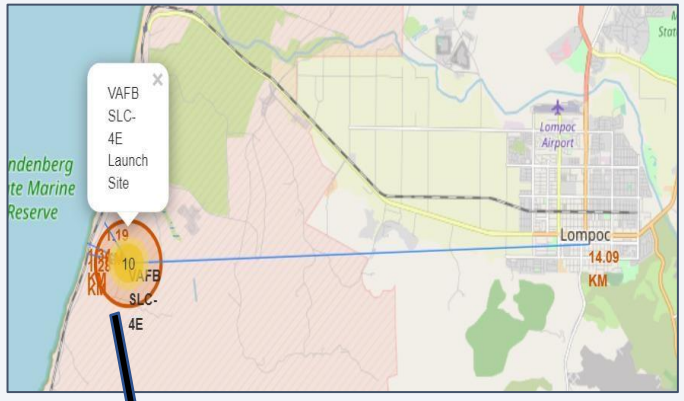


Fig 1 – Proximity site map for VAFB SLC-4E

Figure 1 displays all the proximity sites marked on the map for Launch Site VAFB SLC-4E. City Lompoc is located further away from Launch Site compared to other proximities such as coastline, railroad, highway, etc. The map also displays a marker with city distance from the Launch Site (14.09 km)

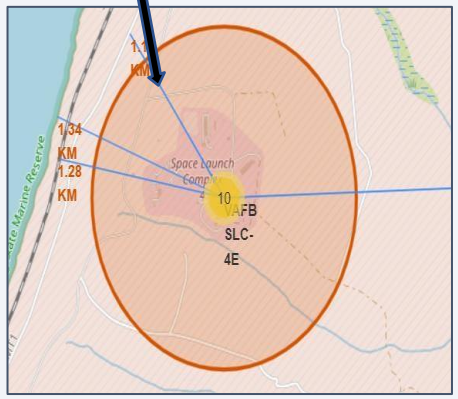
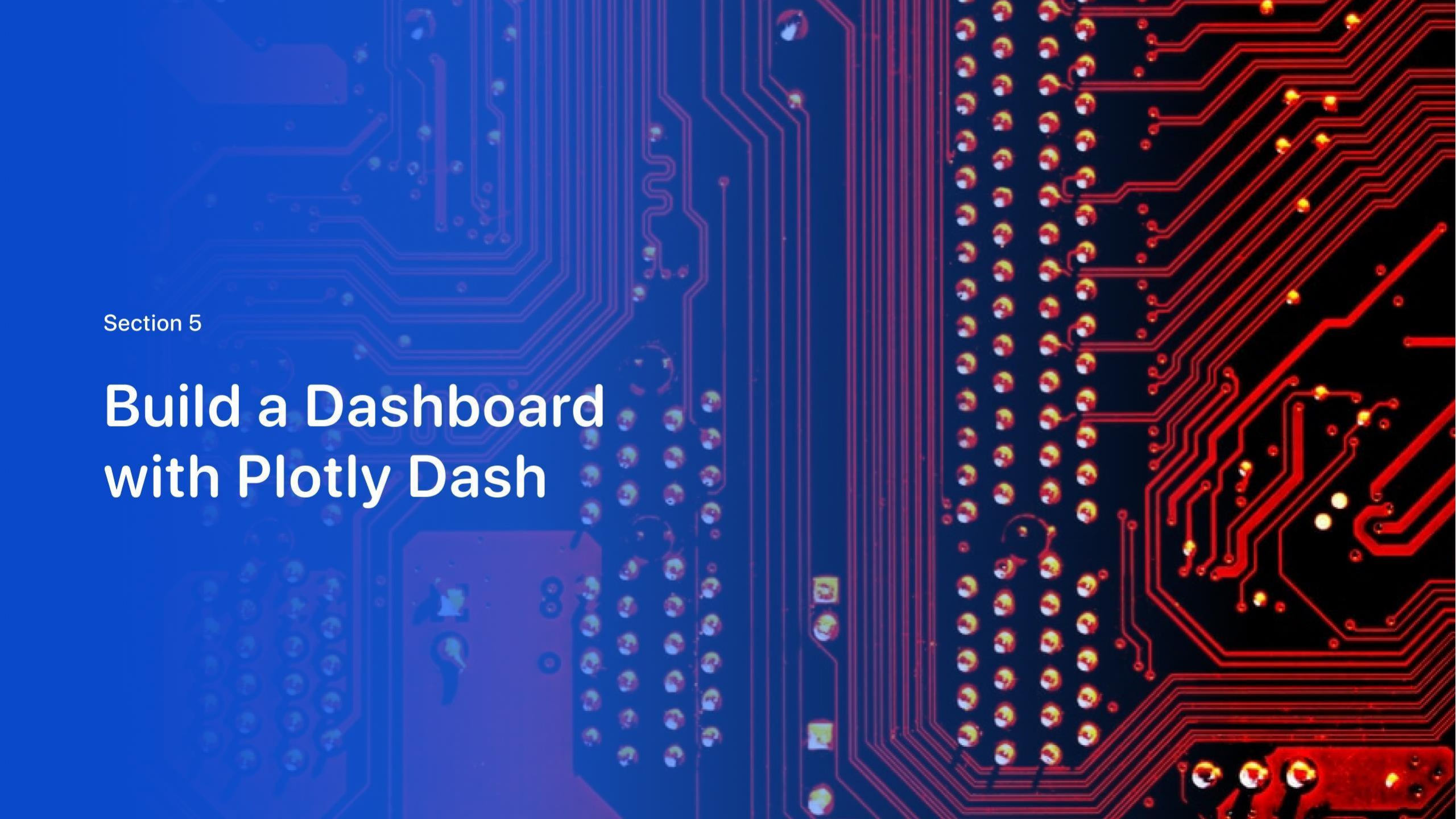


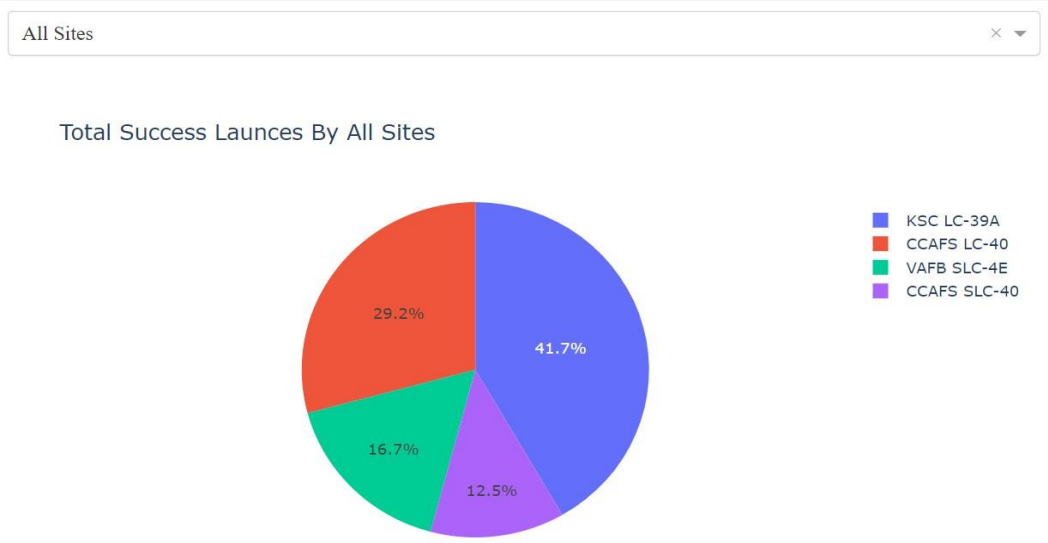
Fig 2 – Zoom in for sites – coastline, railroad, and highway

Figure 2 provides a zoom in view into other proximities such as coastline, railroad, and highway with respective distances from the Launch Site

In general, cities are located away from the Launch Sites to minimize impacts of any accidental impacts to the general public and infrastructure. Launch Sites are strategically located near the coastline, railroad, and highways to provide easy access to resources.

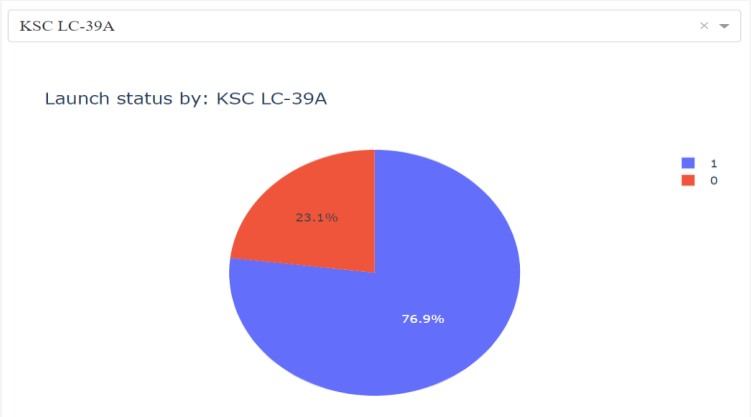

Section 5
Build a Dashboard with Plotly Dash

Launch Success Counts For All Sites



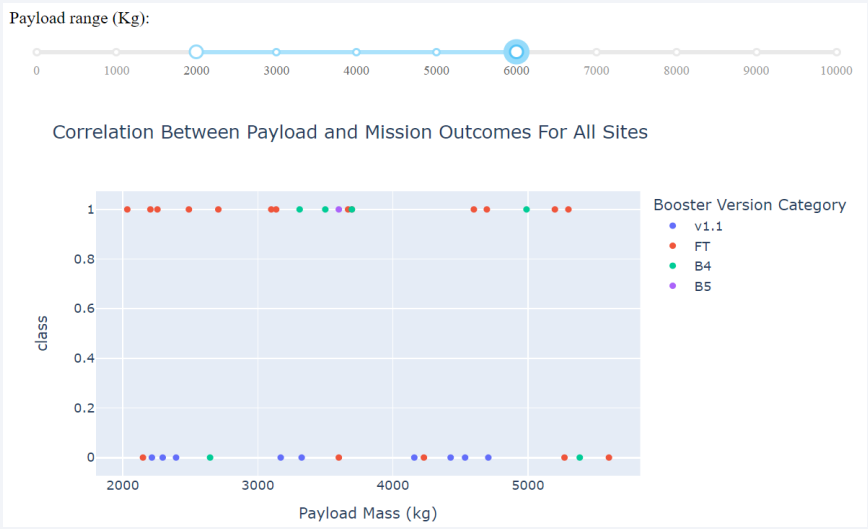
- Launch Site 'KSC LC-39A' has the highest launch success rate
- Launch Site 'CCAFS SLC-40' has the lowest launch success rate

Launch Site with Highest Launch Success Ratio



- KSC LC-39A Launch Site has the highest launch success rate and count
- Launch success rate is 76.9%
- Launch success failure rate is 23.1%

Payload vs. Launch Outcome Scatter Plot for All Sites

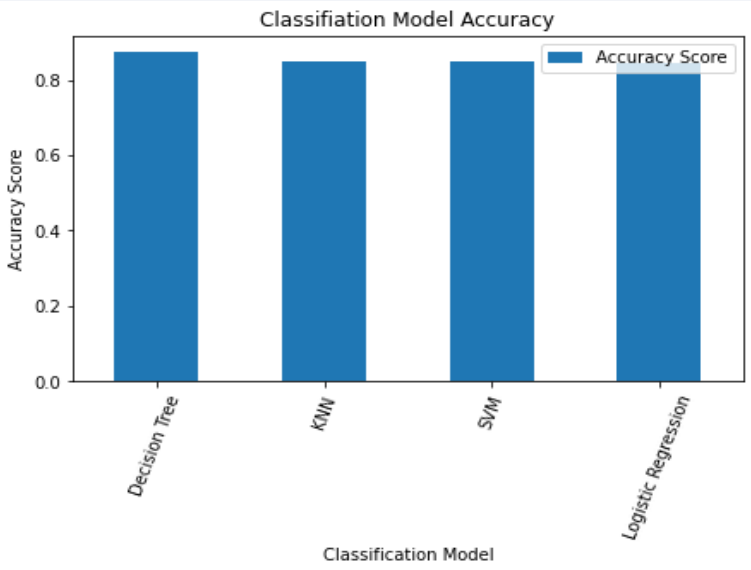


- Most successful launches are in the payload range from 2000 to about 5500
- Booster version category ‘FT’ has the most successful launches
- Only booster with a success launch when payload is greater than 6k is ‘B4’

Section 6

Predictive Analysis (Classification)

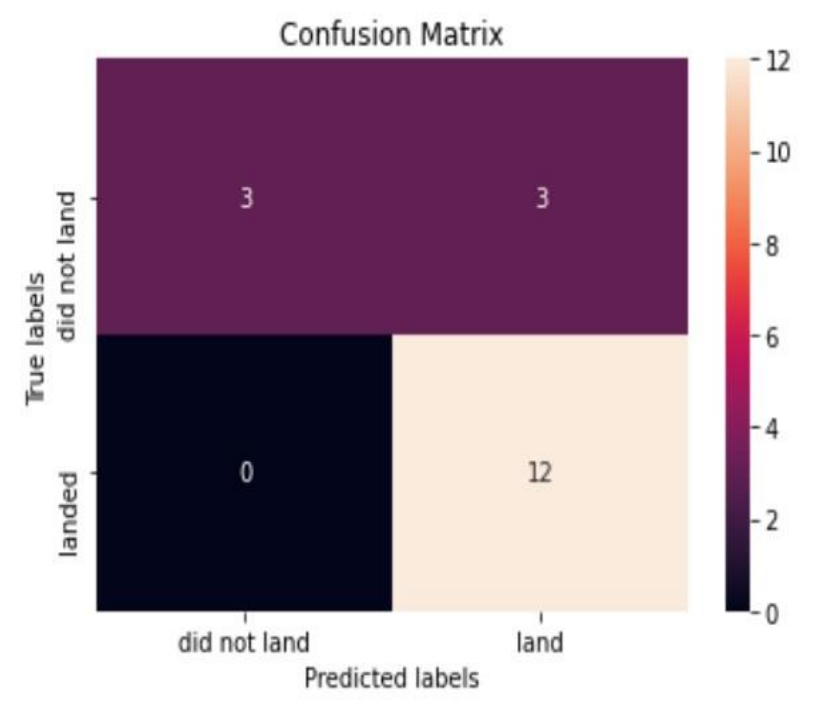
Classification Accuracy



	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

- Based on the Accuracy scores and as also evident from the bar chart, Decision Tree algorithm has the highest classification score with a value of .8750
- Accuracy Score on the test data is the same for all the classification algorithms based on the data set with a value of .8333
- Given that the Accuracy scores for Classfication algorithms are very close and the test scores are the same, we may need a broader data set to further tune the models

Confusion Matrix



- The confusion matrix is same for all the models (LR, SVM, Decision Tree, KNN)
- Per the confusion matrix, the classifier made 18 predictions
- 12 scenarios were predicted Yes for landing,and they did land successfully (True positive)
- 3 scenarios (top left) were predicted No for landing, and they did not land (True negative)
- 3 scenarios (top right) were predicted Yes for landing, but they did not land successfully (False positive)
- Overall, the classifier is correct about 83% of the time ((TP + TN) / Total) with a misclassification or error rate ((FP + FN) / Total) of about 16.5%

Conclusions

- As the numbers of flights increase, the first stage is more likely to land successfully
- Success rates appear go up as Payload increases but there is no clear correlation between Payload mass and success rates
- Launch success rate increased by about 80% from 2013 to 2020
- Launch Site 'KSC LC-39A' has the highest launch success rate and Launch Site 'CCAFS SLC-40' has the lowest launch success rate
- Orbits ES-L1, GEO, HEO, and SSO have the highest launch success rates and orbit GTO the lowest
- Lunch sites are located strategically away from the cities and closer to coastline, railroads, and highways
- The best performing Machine Learning Classfication Model is the Decision Tree with an accuracy of about 87.5%. When the models were scored on the test data, the accuracy score was about 83% for all models. More data may be needed to further tune the models and find a potential better fit.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!