IBM DATA SCIENCE SPACEX LAUNCH PROJECT

P.SINDHURA

HTTPS://GITHUB.COM/SINDHURAPISUPATI/IBM-DATA-SCIENCE-PORTFOLIO

27/12/2023

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 result

Introduction

Project background

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.

Problems you want to find answers

- Determine the cost of a launch 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

DATA METHODOLOGY

DATA COLLECTION

DATA WRANGLING

EDA WITH DATA VISUALIZATION

EDA WITH SQL

INTERACTIVE VIZUAL ANALYTICS(FOLIUM&PLOTLY DASH)

PREDICTIVE ANALYSIS

Methodology

- **Data collection methodology:**

  1.SpaceX API

  2.Web scrapping Falcon 9 and Falcon Heavy launch records from Spacex Wikipedia (https://en.wikipedia.org/wiki/List_of_Falcon/_9/_and_Falcon_Heavy_launches

- **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; Tuned models using GridSearchCV

Data Collection

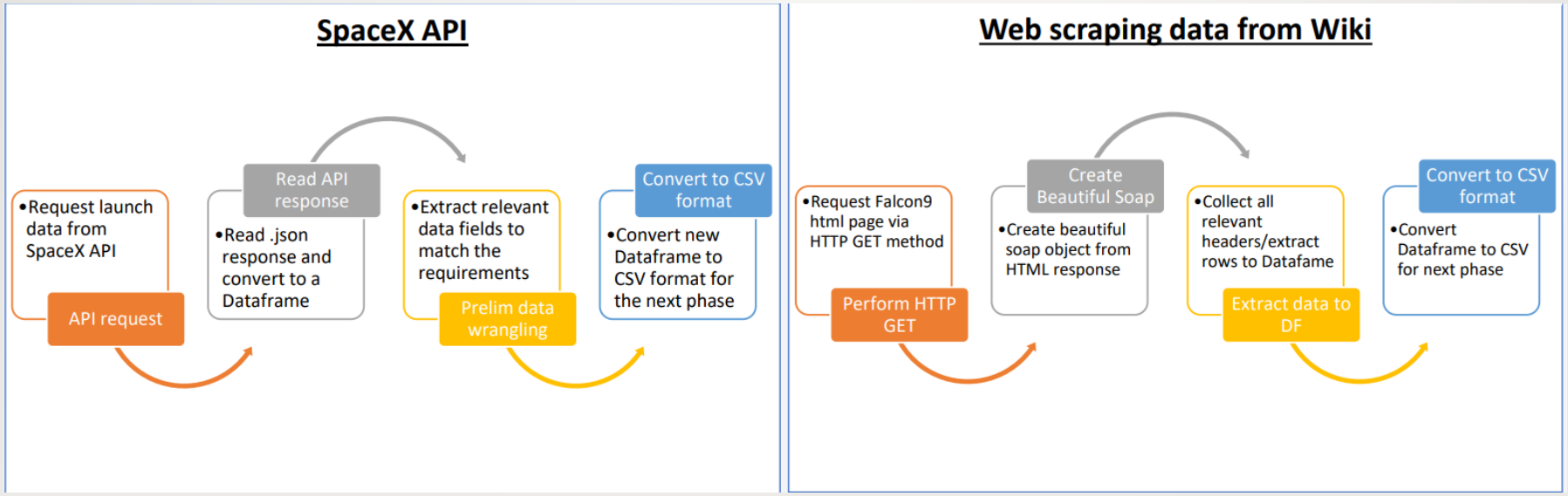
Data collection is the process of gathering data from available sources. This data can be structured, unstructured, or semi-structured. For this project, data was collected via SpaceX API

Space X API Data Columns:

FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins,Reused, Legs,LandingPad, Block, ReusedCount, Serial, Longitude, Latitude

Wikipedia Webscrape Data Columns:

Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing,Date,Time



Data Collection-Spacex API

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}
```

4. 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)
```

GitHub

Data Collection-Web Scraping

1. Perform HTTP GET to request HTML page
2. Create Beautiful Soup 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 Soup 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)
```

GitHub

Data Wrangling

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(contd.)

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

GitHub

EDA with Data Visualization

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

Bar Chart: Commonly used to compare the values of a variable at a given point in time. Plotted following Bar chart to visualize:

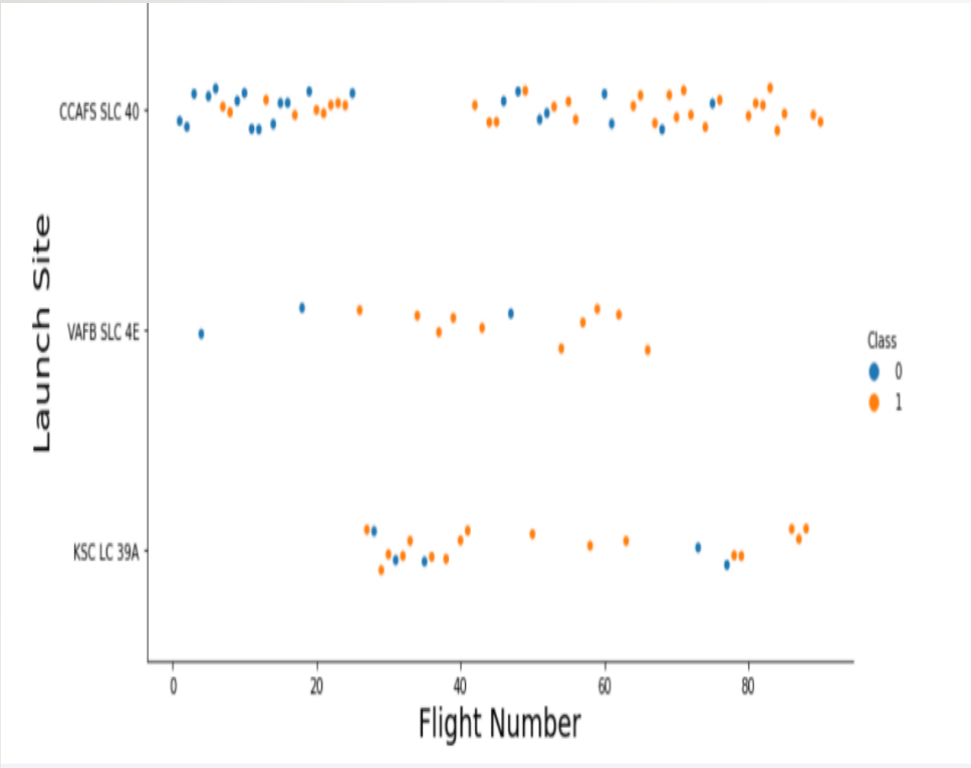
- Relationship between success rate of each orbit type

Line Chart: Commonly used to track changes over a period of time. It helps depict trends over time. Plotted following Line chart to observe:

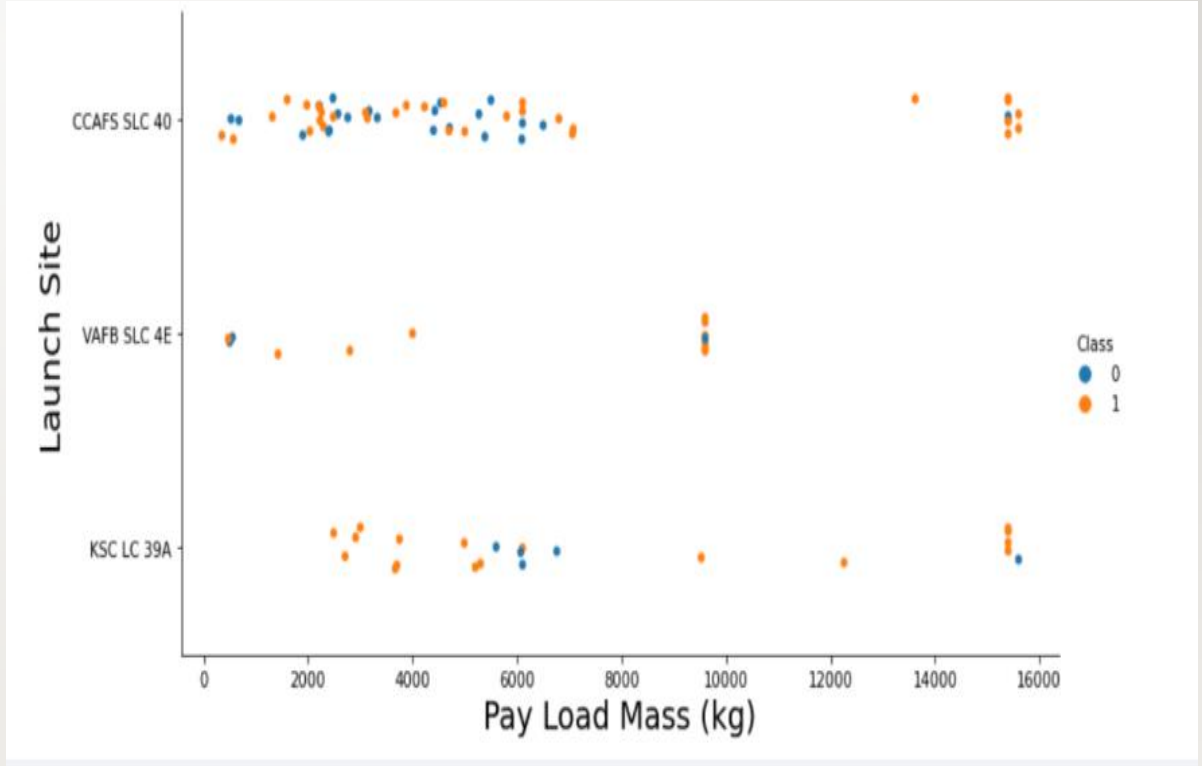
- Average launch success yearly trend

Scatter Plot
1.Relationship between Flight Number and Launch Site
2.Relationship between Payload Mass and 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

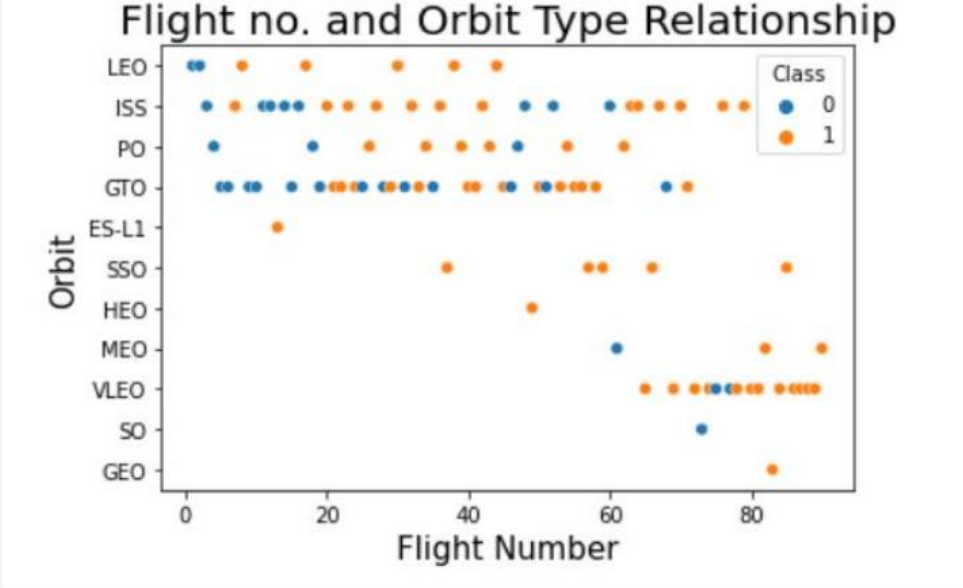


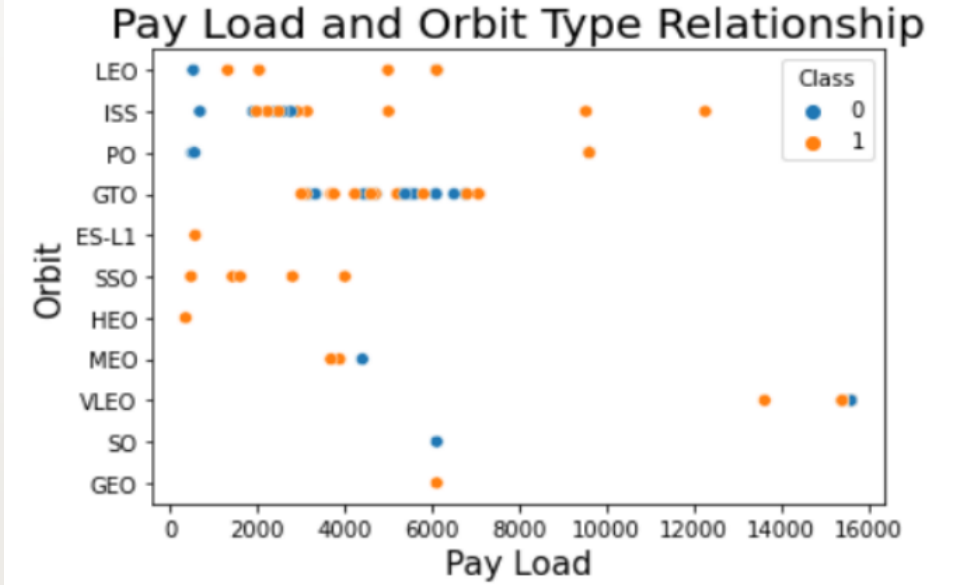
For launch site ‘VAFB SLC 4E’, there are no rockets launched for payload greater than 10,000 kg • Percentage of successful launch (Class=1) increases for launch site ‘VAFB SLC 4E’ as the payload mass increases • There is no clear correlation or pattern between launch site and payload mass



Scatter Plot
1.Relationship between Flight Number and Orbit Type
2.Relationship between Payload Mass and Orbit Type

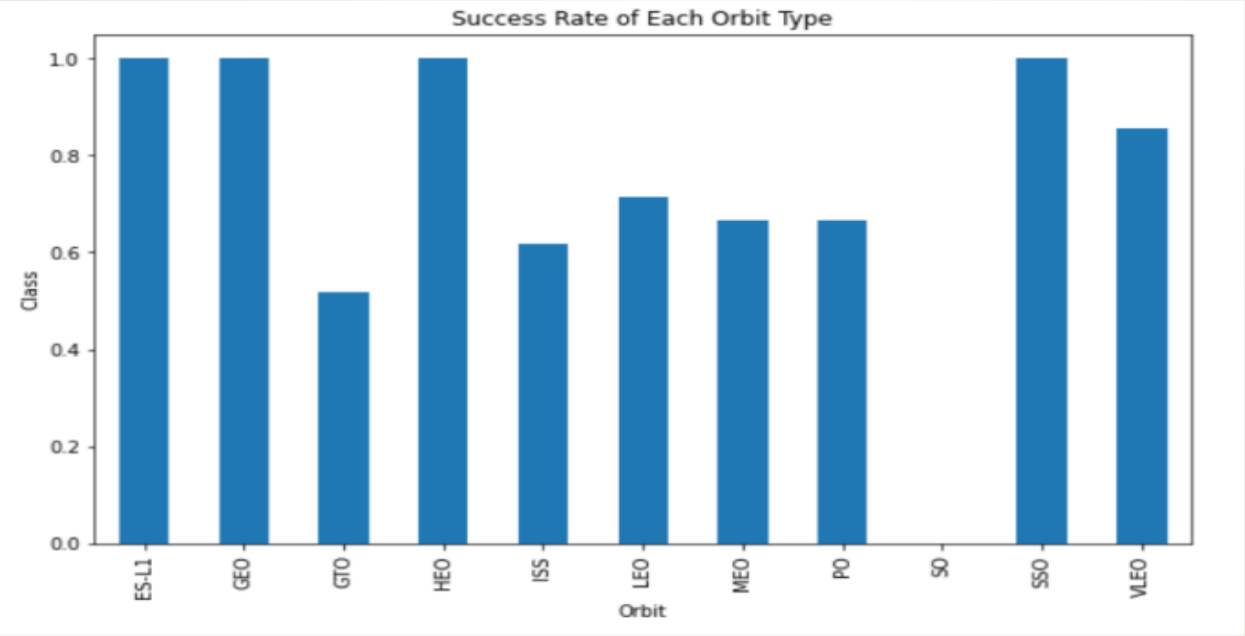
- For orbit VLEO, first successful landing (class=1) doesn’t occur until 60+ number of flights
- For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers
- There is no relationship between flight number and orbit for GTO
- Successful landing rates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





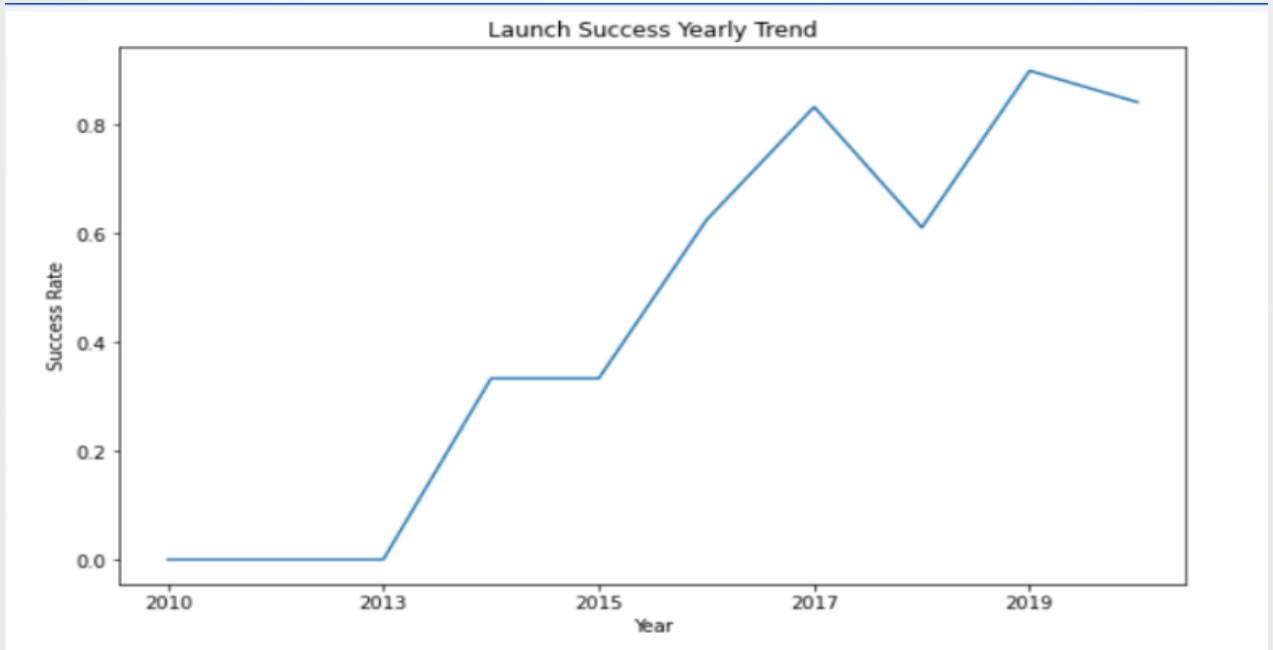
Bar Chart
Relationship between success rate of each orbit type

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



Line Chart
Average 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



EDA With SQL

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

- Loaded data set into IBM DB2 Database.
- Queried using SQL Python integration.
- Queries were made to get a better understanding of the dataset.
- Queried information about launch site names, mission outcomes, various pay load sizes of customers and booster versions, and landing outcomes

Task 1

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

```
%sql select DISTINCT Launch_site from SPACEXTBL
```

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

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

Task 2

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

```
%sql select * from SPACEXTBL where Launch_Site like 'KSC%' LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Task 3

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

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

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

sum
45596

Task 4

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

```
%sql select avg(payload_mass__kg_) as Average from SPACEXTBL where booster_version like 'F9 V1.1'
```

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

Average
2928.4

Task 5

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

```
%sql select min(Date) as Date from SPACEXTBL where Landing_Outcome like "Success (drone ship)"
```

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

Date
2016-04-08

Task 6

List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000

```
%sql select booster_version from SPACEXTBL where (mission_outcome like 'Success') AND(Landing_outcome like "Success (ground
```

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

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1

Task 7

List the total number of successful and failure mission outcomes

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

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

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

Task 8

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

```
%sql select booster_version from SPACEXTABLE WHERE payload_mass__kg_=(SELECT max(payload_mass__kg_) from SPACEXTABLE)
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

Task 9

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

Note: SQLLite does not support monthnames. So you need to use substr(Date,6,2) for month, substr(Date,9,2) for date, substr(Date,0,5),='2017' for year.

[152...

```
%sql SELECT substr(Date,6,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, Landing_Outcome FROM SPACEXTBL where Landing_Outcc
```

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

=[152...

month	Date	Booster_Version	Launch_Site	Landing_Outcome
02	2017-02-19	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
05	2017-05-01	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
06	2017-06-03	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
08	2017-08-14	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
09	2017-09-07	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
12	2017-12-15	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

Task 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

161...

```
%sql select Landing_Outcome,count(*) as count_outcome from SPACEXTBL where DATE between '2010-06-04' and '2017-03-20' group
```

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

161...

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

Build an Interactive Map with Folium

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.

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

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

SpaceX Falcon9 - Launch Sites Map

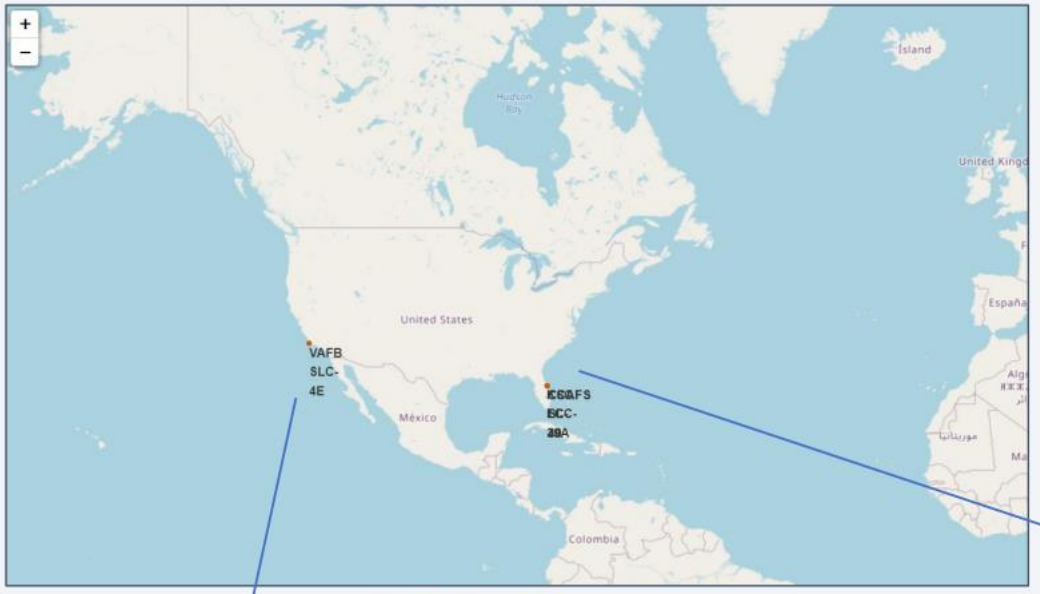


Fig 1 – Global Map

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

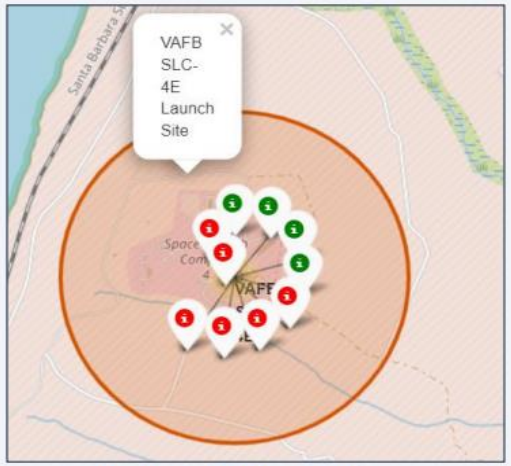


Fig 2 – VAFB Launch Site with success/failed markers

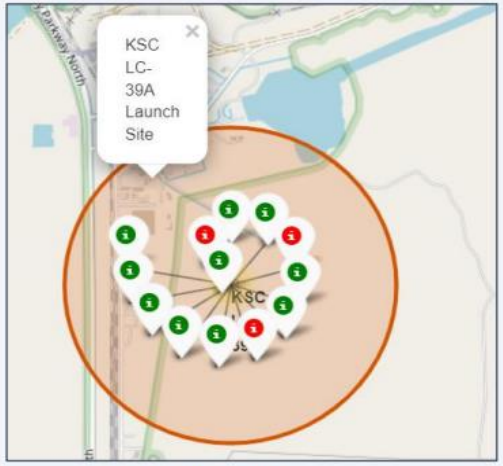


Fig 3 – KSC LC-39A success/failed markers

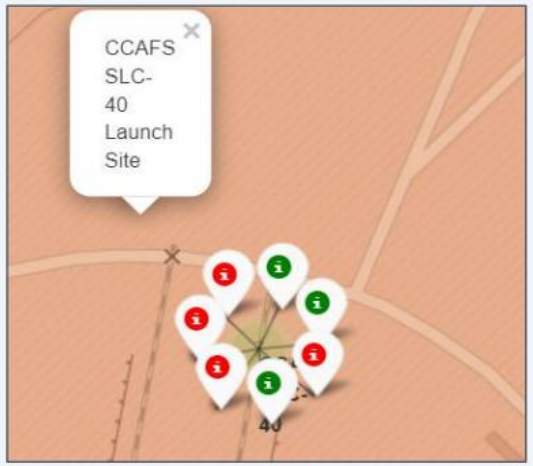


Fig 4 – CCAFS SLC-40 success/failed markers

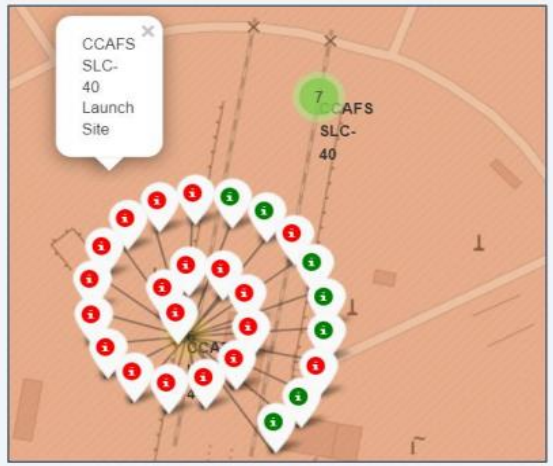


Fig 5 – CCAFS SLC-40 success/failed markers

SpaceX Falcon9 – Launch Site to proximity Distance Map

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

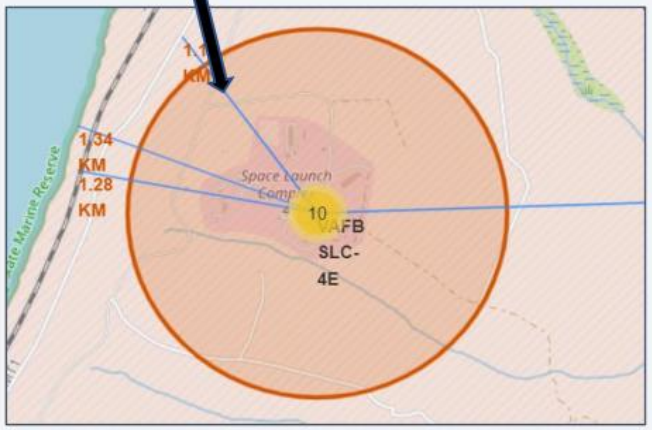


Fig 2 – Zoom in for sites – coastline, railroad, and highway

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)

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.

Build a Dashboard with Plotly Dash

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? KSC LC-39A with 10

2. Which site has the highest launch success rate? KSC LC-39A with 76.9% success

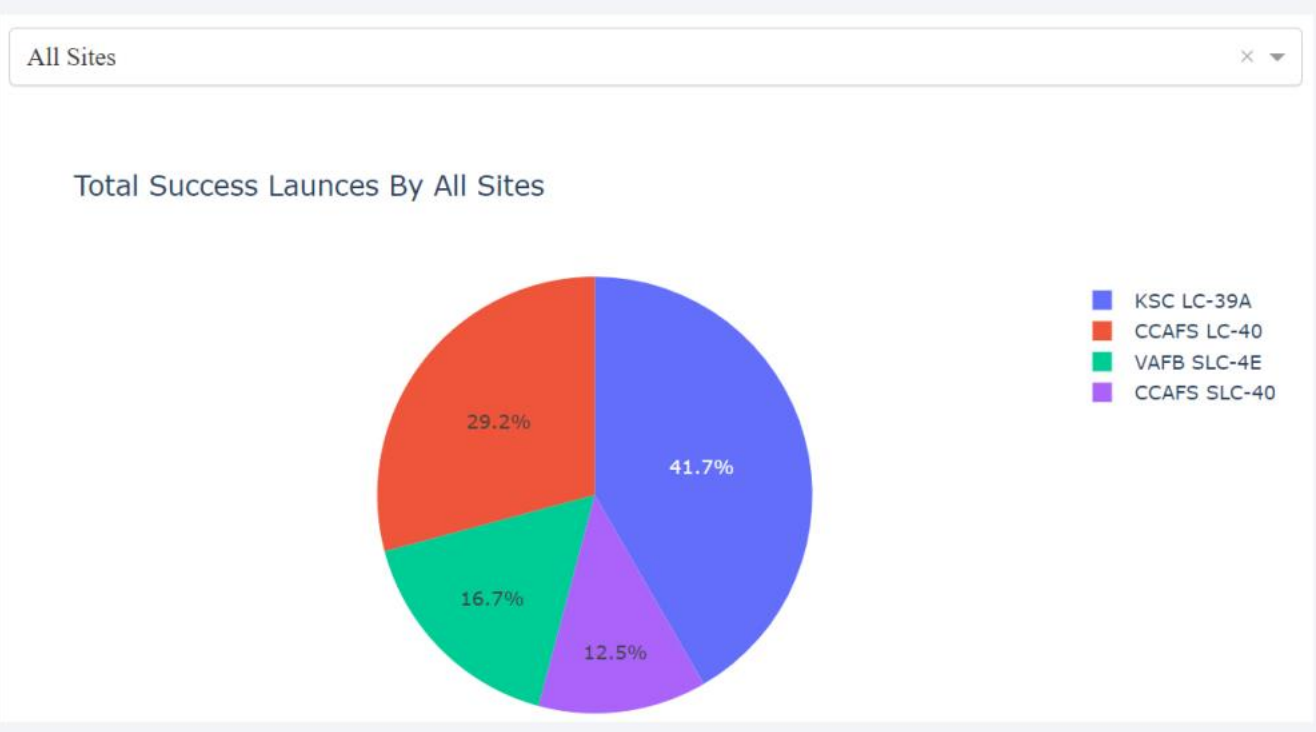
3. Which payload range(s) has the highest launch success rate? 2000 – 5000 kg

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

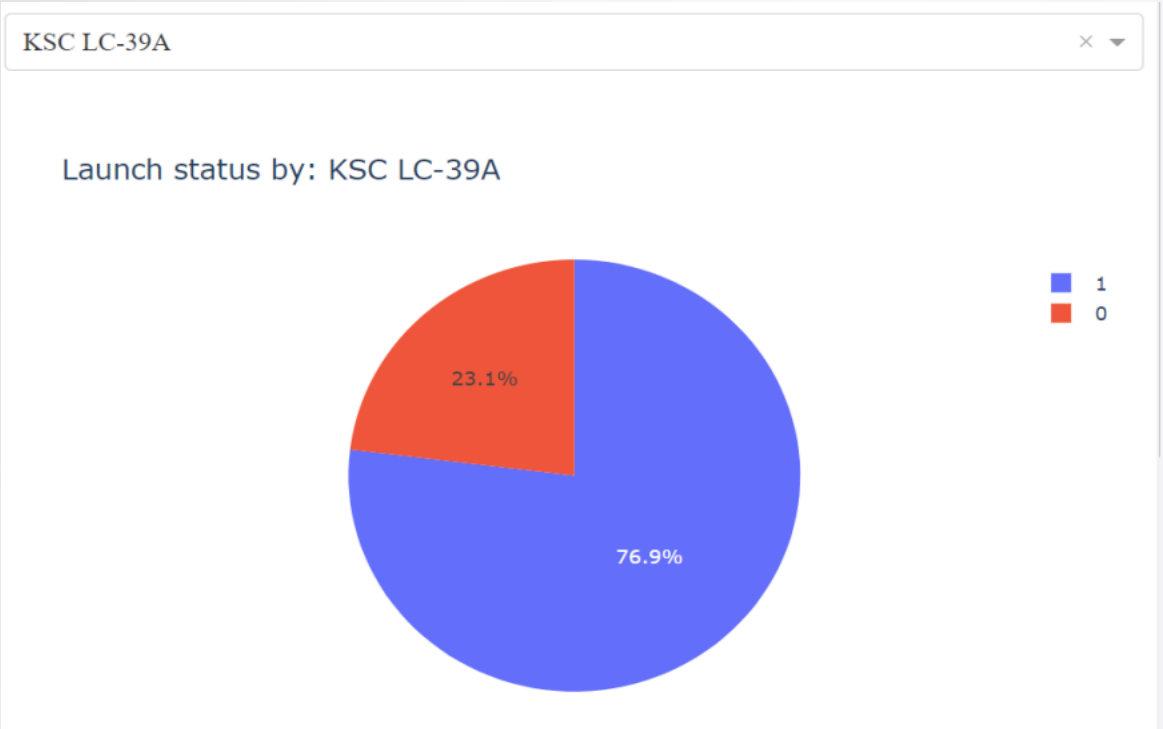
Launch Success Count for all Sites

- Launch Site ‘KSC LC-39A’ has the highest launch success rate
- Launch Site ‘CCAFS SLC40’ has the lowest launch success rate



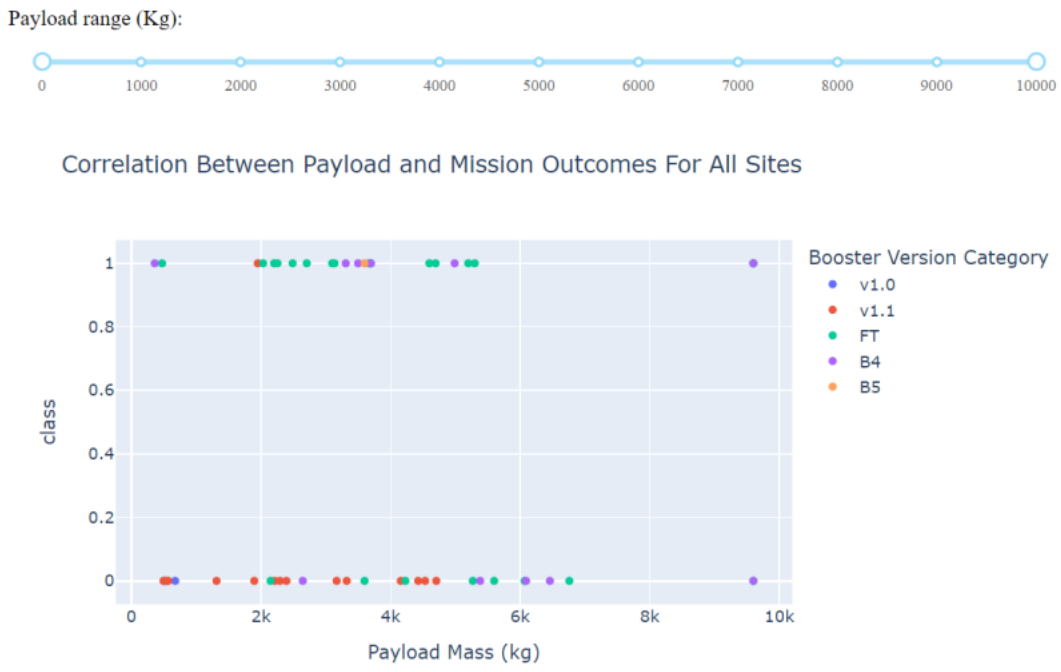
Launch Site with Highest Launch Success Ratio

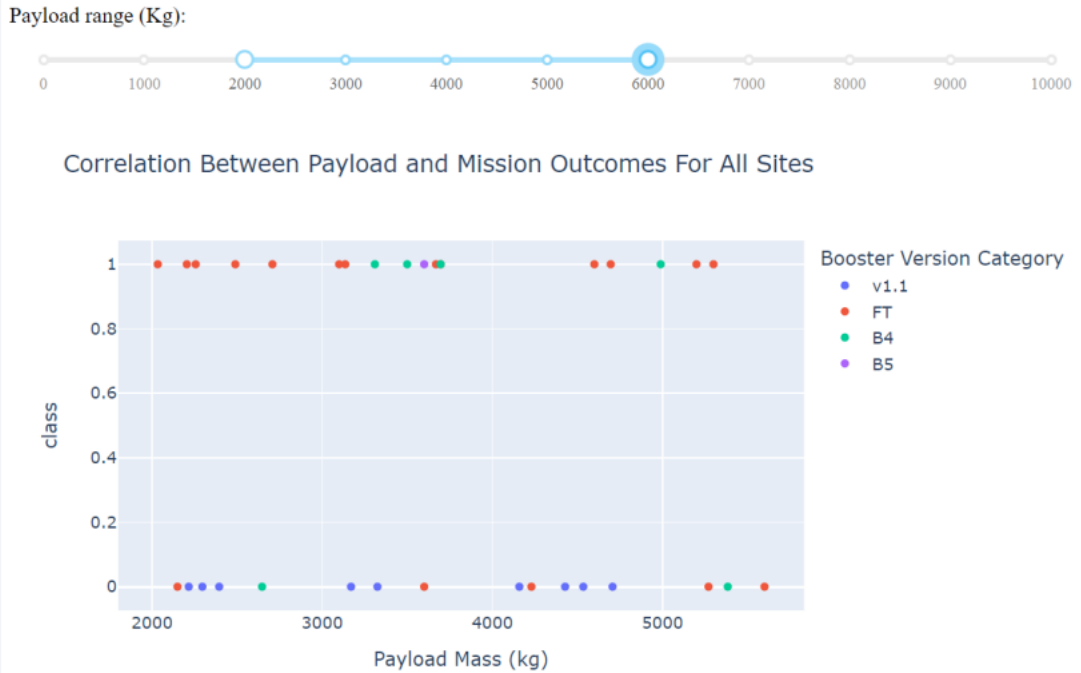
- KSC LC-39A Launch Site has the highest launch success rate and count
- Launch success rate is 76.9%
- Launch success failure rate is 23.1%



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’





Predictive Analysis(classification models)

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)
```

   iii. Find and display best hyperparameters and accuracy score

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

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

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

   v. 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]))
```

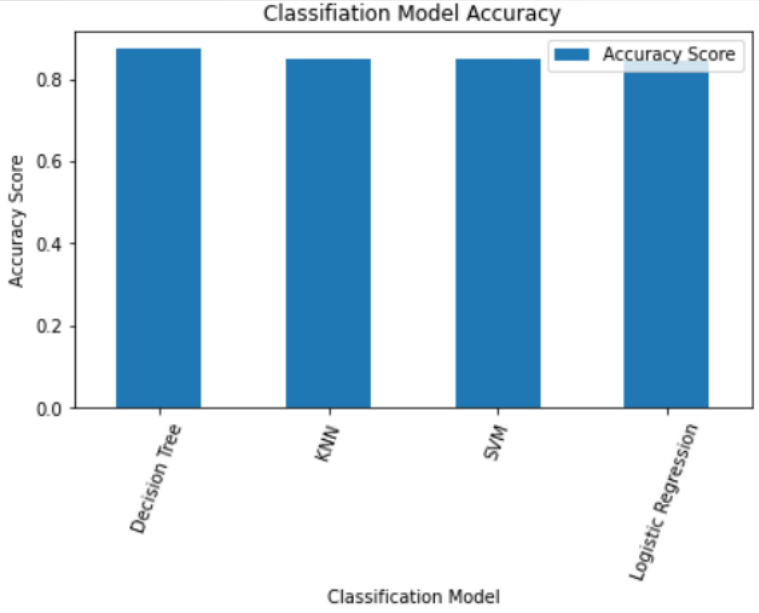
```
The best performing alogrithm is Decision Tree with score 0.875
```

	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

GitHub

Classification Accuracy

- 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 0.87321
- Accuracy Score on the test data is the same for all the classification algorithms based on the data set with a value of 0.8889
- 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



	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.873214	0.888889
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

Confusion Matrix

The confusion matrix is same for all the models (LR, SVM, Decision Tree, KNN)

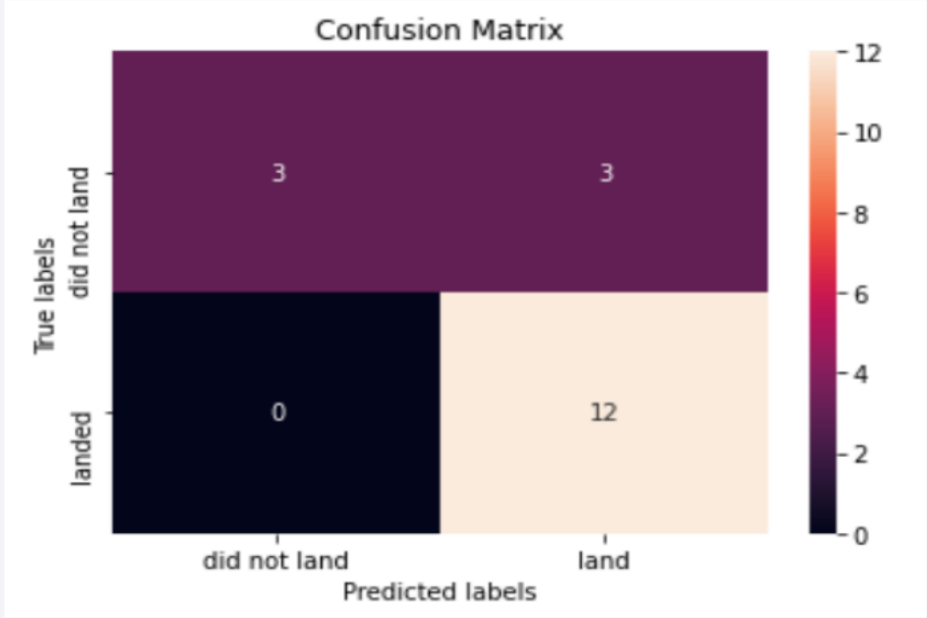
• Per the confusion matrix, the classifier made 18 predictions

• 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%

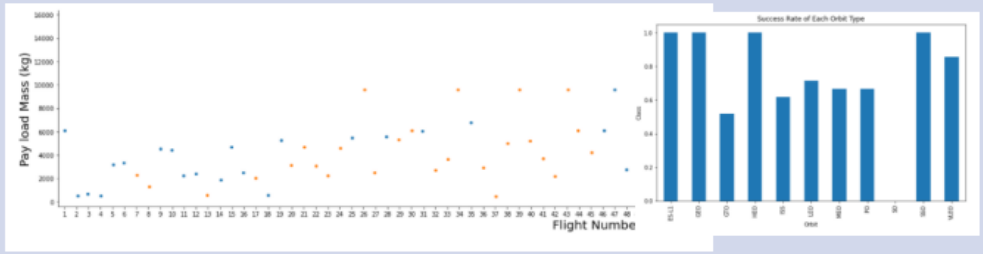


Results

Following sections and slides explain results for:

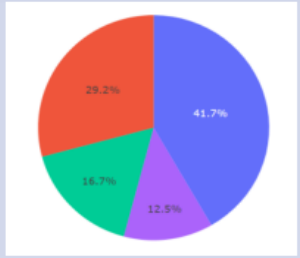
Exploratory data analysis results

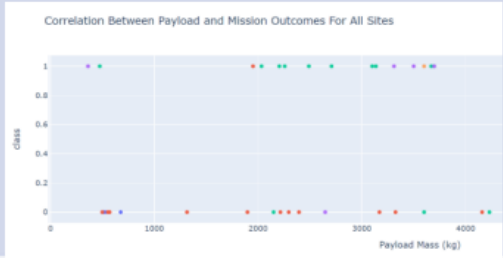
- Samples:



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

Conclusion

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 SLC40’ has the lowest launch success rate

Orbits ES-L1, GEO, HEO, and SSO have the highest launch success rates and orbit GTO the lowest

Launch 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 87.3%. 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

Instructors:
Rav Ahuja, Alex Aklson, Aije Egwaikhide, Svetlana Levitan, Romeo Kienzler, Polong Lin, Joseph Santarcangelo, Azim Hirjani, Hima Vasudevan, Saishruthi Swaminathan, Saeed Aghabozorgi, Yan Luo


THANK YOU