IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Vikrant Shrirame
27 June 2024

Outline

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

Executive Summary

Our research aims to analyze data from SpaceX Falcon 9, sourced from various channels, and apply Machine Learning models to predict the outcome of first stage landings. This knowledge can guide other space agencies in their decision to compete with SpaceX.

- **Summary of methodologies:**
    1. Data Collection: We gathered data through API calls and web scraping.
    2. Data Transformation: To prepare the data, we applied wrangling techniques.
    3. Exploratory Data Analysis (EDA): Using SQL queries and visualizations, we explored the data.
    4. Geospatial Analysis: An interactive map was created using Folium to assess launch site proximity.
    5. Predictive Modeling: Finally, we developed a predictive model to evaluate the likelihood of successful first stage landings for Falcon 9.
- **Summary of all results:**
    1. Big Data: Efficiently handle large datasets.
    2. Cloud Solutions: Leverage scalability and storage.
    3. Machine Learning: Explore AI’s role in data science.

Introduction

- **Project background and context:**
    1. As private space travel achieves recent milestones, the space industry is increasingly becoming part of the mainstream and accessible to the general population. However, the cost of launching remains a significant hurdle for new entrants in the space race.
    2. SpaceX stands out from its competitors due to its ability to reuse the first stage. With each SpaceX launch costing approximately 62 million dollars, this unique advantage contrasts with other competitors who spend around 165 million dollars or more per launch.
    3. Recent successes in private space travel have propelled the space industry into the mainstream, making it more accessible to the public. Despite this progress, the high cost of launching remains a major obstacle for new players entering the space race.
- **Problems you want to find answers:**
    1. Assessing First Stage Landing Success: Investigate whether SpaceX Falcon 9's first stage will land successfully. Explore the likelihood of successful landings based on various parameters (e.g., launch site, payload mass, booster version).
    2. Parameter Impact and Correlations: Analyze how different factors influence landing outcomes. Examine correlations between launch sites and success rates.

Section 1

Methodology

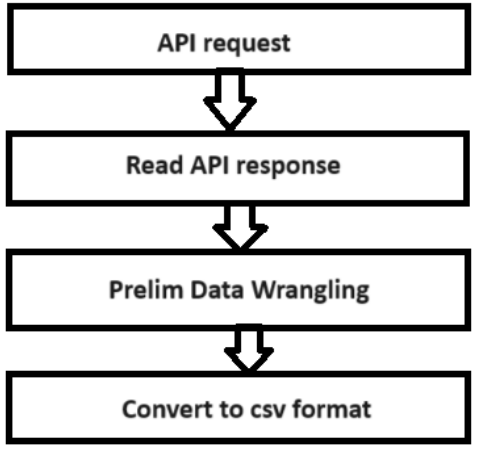
Methodology

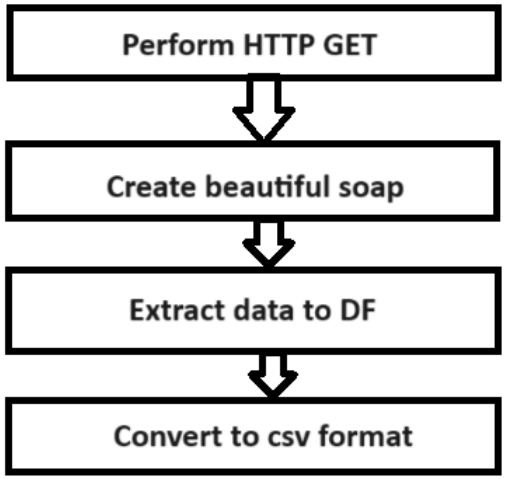
Executive Summary

- Data collection methodology:
  - Data was collected using two main methods: first, by accessing the SpaceX API, and second, by web scraping Falcon 9 and Falcon Heavy launch records from Wikipedia.
- Perform data wrangling
  - To prepare the data for modeling, we determined labels for training the supervised models.
  - Mission outcomes were converted into training labels, where 0 represents unsuccessful and 1 represents 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
  - To perform predictive analysis using classification models, we created a 'class' column, standardized and transformed the data, split it into train and test sets, and evaluated various algorithms (Logistic regression, SVM, decision tree, and KNN) using the test data.

Data Collection

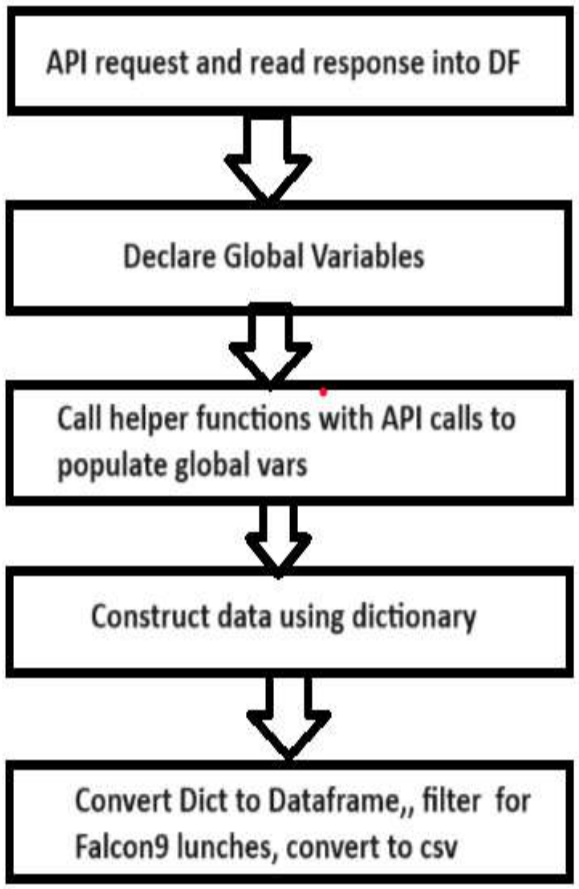
- Describe how data sets were collected.
  - Data collection involves systematically gathering observations or measurements. Whether you're conducting research for business, government, or academic purposes, data collection provides firsthand knowledge and original insights into your research problem
- You need to present your data collection process use key phrases and flowcharts





Data Collection – SpaceX API

Github



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

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

[7]: response = requests.get(spacex_url)

     Check the content of the response

[8]: print(response.content)
```

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

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

3. Retrieve relevant data using helper functions for columns with IDs (e.g., where the 'rocket' column serves as an identification number). The following functions can be used:

1. getBoosterVersion(data)
2. getLaunchSite(data)
3. getPayloadData(data)
4. getCoreData(data)

```
[23]: # Call getBoosterVersion
getBoosterVersion(data)

      the list has now been update

[24]: BoosterVersion[0:5]

[24]: ['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']

      we can apply the rest of the functions here:

[25]: # Call getLaunchSite
getLaunchSite(data)

[26]: # Call getPayloadData
getPayloadData(data)

[27]: # Call getCoreData
getCoreData(data)
```

Data Collection – SpaceX API

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

```
[28]: 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:

```
[29]: # Create a data from launch_dict
df_launch = pd.DataFrame(launch_dict)
```

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

Data Collection - Scraping

Github

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

```
[4]: 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

Create a `BeautifulSoup` object from the HTML `response`

```
[11]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data,"html.parser")
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
[12]: # Use soup.title attribute
tag_object = soup.title
print ("tag object:", tag_object)
```

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')
```

```
[15]: column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
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)
```

Data Collection - Scraping

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

```
[17]: launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
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. Populate the launch_dict with launch records obtained from the table rows. To parse the HTML data, make use of the following helper functions.

```
[9]: def date_time(table_cells):
         """
         This function returns the data
         Input: the  element of a table
         """
         return [data_time.strip() for d

     def booster_version(table_cells):
         """
         This function returns the boost
         Input: the  element of a table
         """
         out=''.join([booster_version fo
         return out

     def landing_status(table_cells):
         """
         This function returns the landi
         Input: the  element of a table
         """
         out=[i for i in table_cells.str
         return out


     def get_mass(table_cells):
         mass=unicodedata.normalize("NFK
```

6. Convert launch_dict to Dataframe

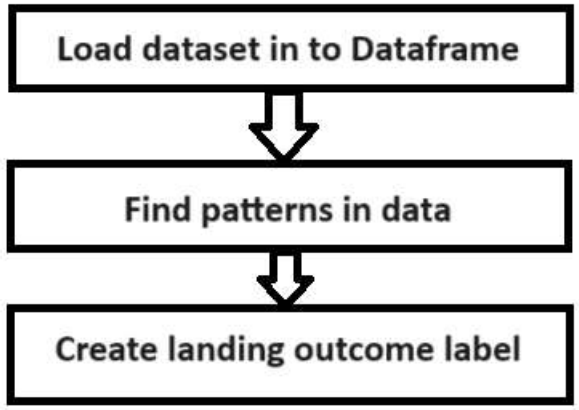
```
[21]: df=pd.DataFrame(launch_dict)
```

Data Wrangling

Github

- I performed Exploratory Data Analysis (EDA) to identify patterns in the data and define labels for training supervised models.
- The dataset included various mission outcomes, which were transformed into training labels. A label of '1' indicated successful booster landings, while '0' represented unsuccessful landings. The following landing scenarios were considered for label creation:
    1. 'True Ocean': The mission outcome successfully landed in a specific region of the ocean.
    2. 'False Ocean': The mission outcome was unsuccessful in landing in a specific ocean region.
    3. 'RTLS': Successful landing on a ground pad.
    4. 'False RTLS': Unsuccessful landing on a ground pad.
    5. 'True ASDS': Successful landing on a drone ship.
    6. 'False ASDS': Unsuccessful landing on a drone ship.

Data Wrangling



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

```
In [2]: df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
        df.head(10)
```

2. Find data patterns:
(A) Calculate the number of launches on each site

```
In [5]: # Apply value_counts() on column LaunchSite
        df['LaunchSite'].value_counts()
```

```
Out[5]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dtype: int64
```

(B) Calculate the number and occurrence of each orbit

```
In [6]: # Apply value_counts on Orbit column
        df['Orbit'].value_counts()
```

```
Out[6]: GTO      27
        ISS      21
        VLEO     14
        PO        9
        LEO       7
        SSO       5
        MEO       3
        GEO       1
        HEO       1
        SO        1
        ES-L1     1
        Name: Orbit, dtype: int64
```

(C) Calculate number/occurrence of mission outcomes per orbit type

```
In [8]: # landing_outcomes = values on Outcome column
        landing_outcomes = df['Outcome'].value_counts()
```

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

```
In [11]: # 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)
```

```
In [12]: df['Class']=landing_class
         df[['Class']].head(8)
```

Out[12]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0

EDA with Data Visualization

Github

During the Exploratory Data Analysis (EDA), we created the following charts to gain deeper insights into the dataset:

1. **Scatter Plot:** A scatter plot reveals relationships or correlations between two variables, making patterns easy to observe.

   We visualized the following relationships:

   - Flight Number vs. Launch Sit
   - Payload vs. Launch Site
   - Flight Number vs. Orbit Type
   - Payload vs. Orbit Type

2. Bar Chart:

- Bar charts are commonly used to compare the values of a variable at a specific point in time.
- We plotted a bar chart to visualize the relationship between the success rate of each orbit type.

3. Line Chart:

- Line charts track changes over time and help depict trends.
- We created a line chart to observe the average launch success trend on a yearly basis.

EDA with SQL

Github

- **The SpaceX dataset was thoroughly analyzed using SQL queries and operations on an IBM DB2 cloud instance. The following tasks were performed:**

1. Extracted the unique launch site names from the space mission data.
2. Retrieved 5 records where launch sites start with the prefix ‘CCA.’
3. Calculated the total payload mass carried by boosters launched by NASA (CRS).
4. Computed the average payload mass carried by booster version F9 v1.1.
5. Identified the date of the first successful landing outcome on a ground pad.
6. Listed the names of boosters that achieved success on a drone ship and had a payload mass between 4000 and 6000.
7. Summarized the total number of successful and failed mission outcomes.
8. Utilized a subquery to find booster versions that carried the maximum payload mass.
9. Compiled a list of failed landing outcomes on drone ships, along with their booster versions and launch site names for the year 2015.
10. Ranked the count of landing outcomes (e.g., Failure on drone ship or Success on ground pad) between June 4, 2010, and March 20, 2017, in descending order.

Build an Interactive Map with Folium

Github

- **The Folium interactive map is a powerful tool for analyzing geospatial data, enabling more engaging visual analytics and a deeper understanding of factors such as location and proximity that impact launch success rates. Let's break down the key steps involved in creating and enhancing this map:**

1. Map Creation: We generated a map object to visualize launch sites.
2. Marking Launch Sites: All launch sites were marked on the map, providing a clear visual representation.
3. Highlighting Areas: We used 'folium.circle' and 'folium.marker' to highlight circular areas around each launch site, accompanied by text labels.
4. Launch Outcome Markers: A 'MarkerCluster()' was employed to display success (green) and failure (red) markers for each launch site.
5. Distance Calculations: Distances between launch sites and nearby features (such as coastline, railroad, highway, and city) were calculated.
6. Mouse Position Tracking: The 'MousePosition()' feature allowed us to obtain coordinates for any point on the map.
7. Distance Display: 'folium.Marker()' was used to show distances (in kilometers) on the map for specific points (e.g., coastline, railroad, highway, city).
8. Connecting Points: 'folium.Polyline()' drew lines connecting points of interest to their corresponding launch sites.
9. Repeat Process: We repeated the above steps to connect launch sites with nearby features (coastline, railroad, highway, city).
10. Answering Questions: The interactive map built with Folium provided answers to important questions:
    - Are launch sites close to railways? Yes.
    - Are launch sites close to highways? Yes.
    - Are launch sites close to the coastline? Yes.
    - Do launch sites maintain a certain distance from cities? Yes.

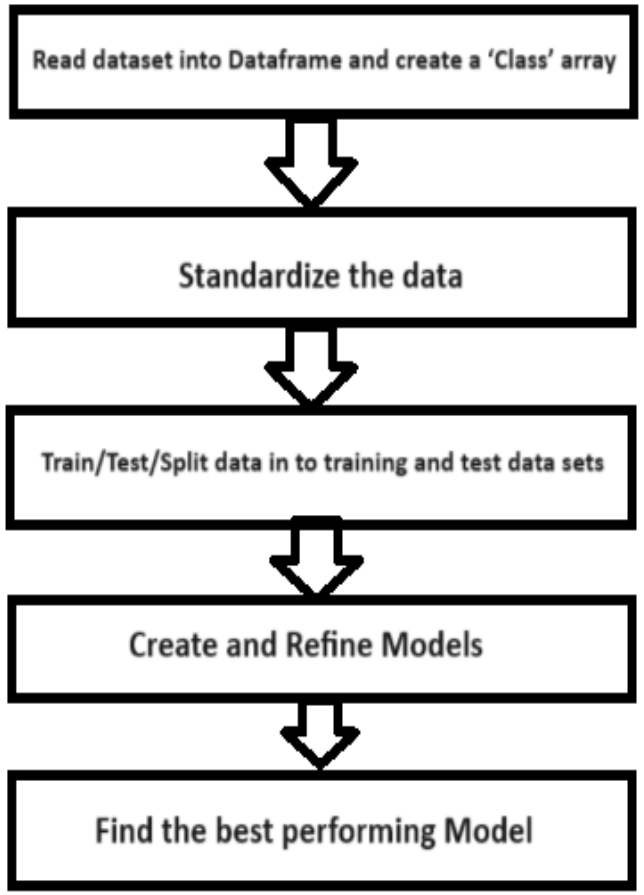
Build a Dashboard with Plotly Dash

Github

- We developed a real-time Plotly Dash web application for interactive visual analytics of SpaceX launch data. The application includes several components:
    1. Launch Site Dropdown: Users can filter visualizations by selecting all launch sites or a specific site.
    2. Pie Chart: When ‘All Sites’ is chosen, the pie chart displays total successful launches. For specific sites, it shows success and failure counts.
    3. Payload Range Slider: Users can easily explore different payload ranges to identify visual trends.
    4. Scatter Chart: This chart examines the correlation between payload and mission outcomes for selected site(s). Each scatter point is color-labeled by booster version.
- The dashboard provides answers to the following questions:
    1. Which site has the highest number of successful launches? KSC LC-39A (10 launches).
    2. Which site has the highest launch success rate? KSC LC-39A (76.9% success rate).
    3. Which payload range(s) have the highest launch success rate? 2000–5000 kg.
    4. Which payload range(s) have the lowest launch success rate? 0–2000 kg and 5500–7000 kg.
    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. Load SpaceX dataset (csv) in to a Dataframe and create NumPy array from the column class in data

```
[4]: from js import fetch
     import io

     URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv"
     resp1 = await fetch(URL1)
     text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
     data = pd.read_csv(text1)
```

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

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

```
[10]: # students get this
      transform = preprocessing.StandardScaler()
      X= preprocessing.StandardScaler().fit(X).transform(X)
      X[0:5]
```

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

```
[11]: # 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)
(A) Create Logistic Regression object and then create a GridSearchCV object

(B) Fit train data set in to the GridSearchCV object and train the Model

```
[16]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
      LR = LogisticRegression()
      logreg_cv = GridSearchCV(LR, parameters,cv=10)
      logreg_cv.fit(X_train, Y_train)
```

Predictive Analysis (Classification)

(C) Find and display best hyperparameters and accuracy score

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

(D) Check the accuracy on the test data by creating a confusion matrix

```
[19]: yhat=logreg_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```

(E) Repeat above steps for Decision Tree, KNN, and SVM algorithms

5. Find the best performing model

```
[40]: 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)]})
      Model_Performance_df.sort_values(['Accuracy Score'], ascending = False, inplace=True)
      Model_Performance_df
```

[40]:

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.666667
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

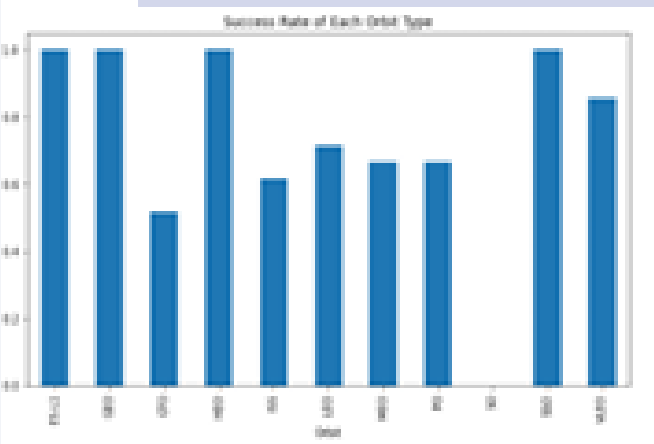
```
[41]: i = Model_Performance_df['Accuracy Score'].idxmax()
      print('The best performing alogrithm is '+ Model_Performance_df['Algo Type'][i]
      + ' with score ' + str(Model_Performance_df['Accuracy Score'][i]))
```

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

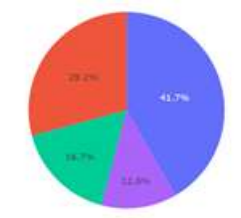
Results

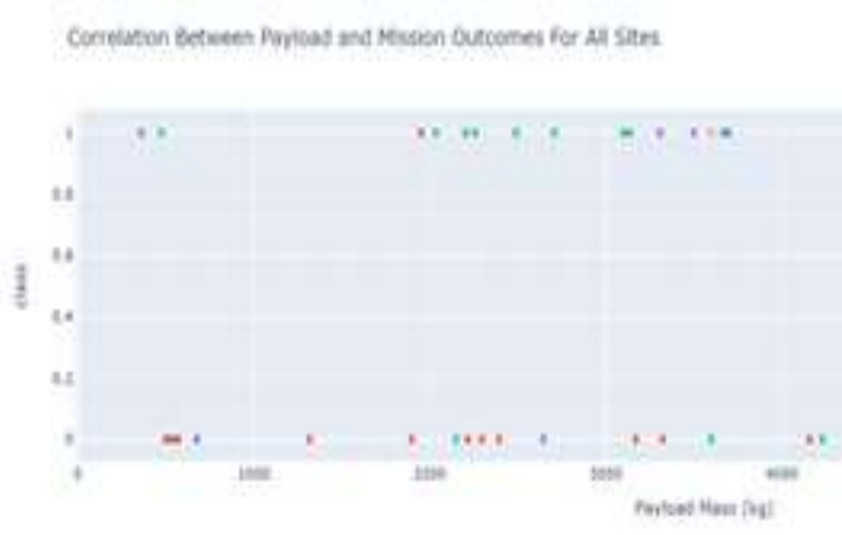
- Following sections and slides explain results for:

Exploratory data analysis results



Interactive analytics demo in screenshots





Predictive analysis results

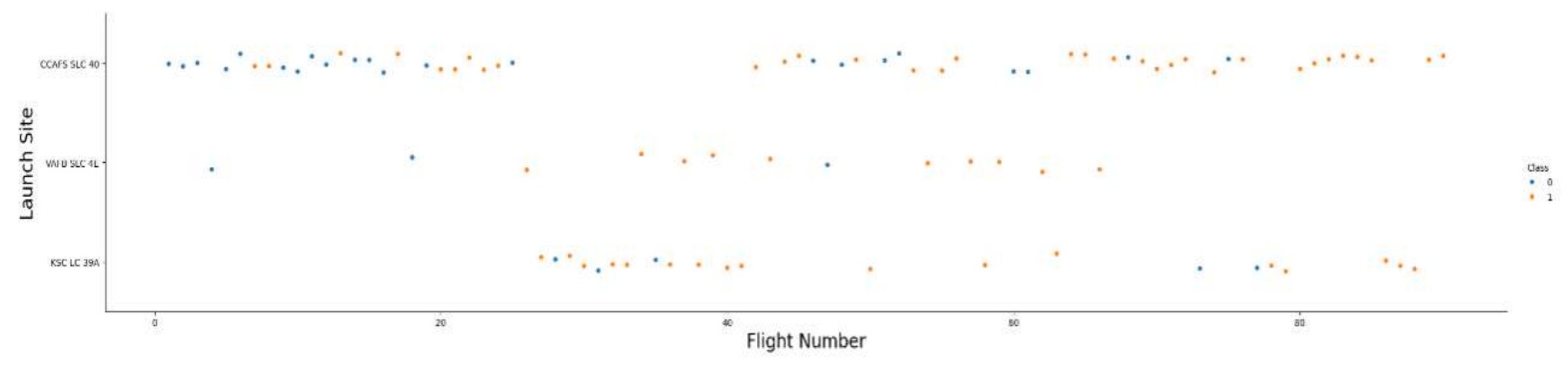
	Algo Type	Accuracy Score
2	Decision Tree	0.903571
3	KNN	0.848214
1	SVM	0.848214
0	Logistic Regression	0.846429

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

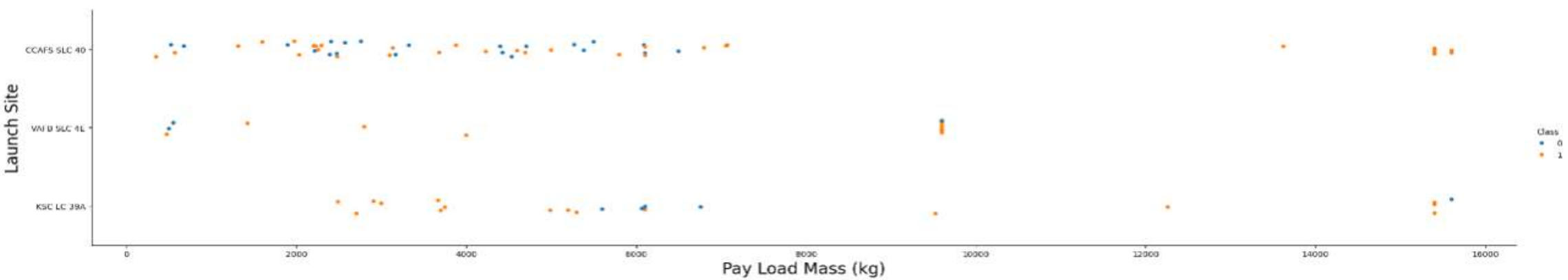
- Flight Number vs. Launch Site



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

Payload vs. Launch Site

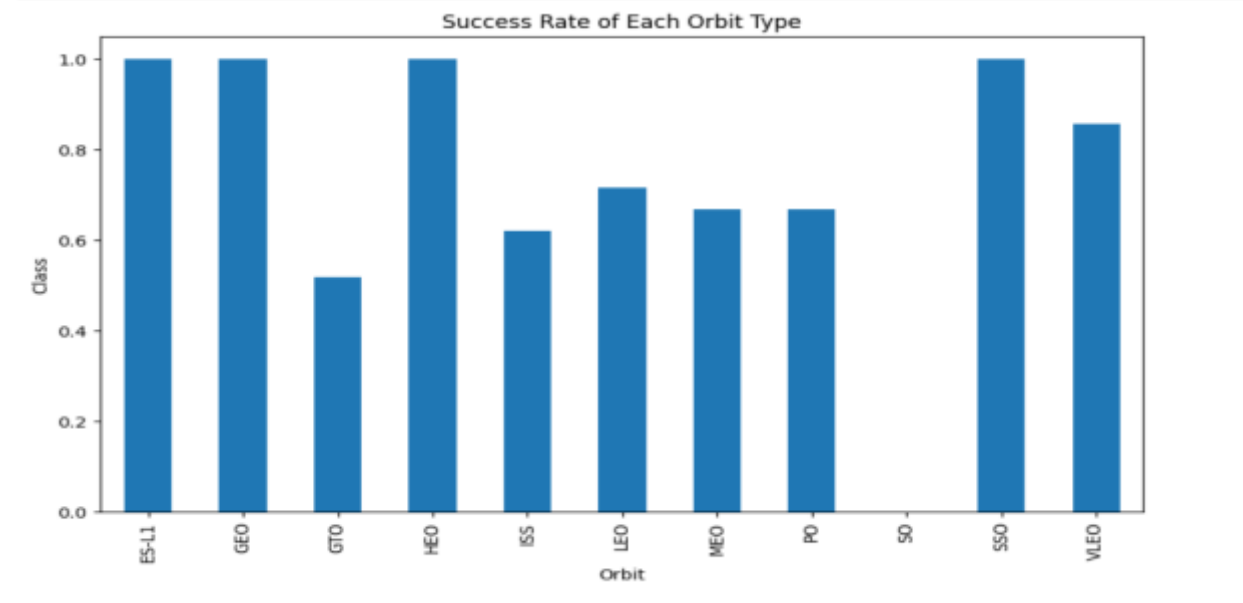
- scatter plot of Payload vs. Launch Site



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

Success Rate vs. Orbit Type

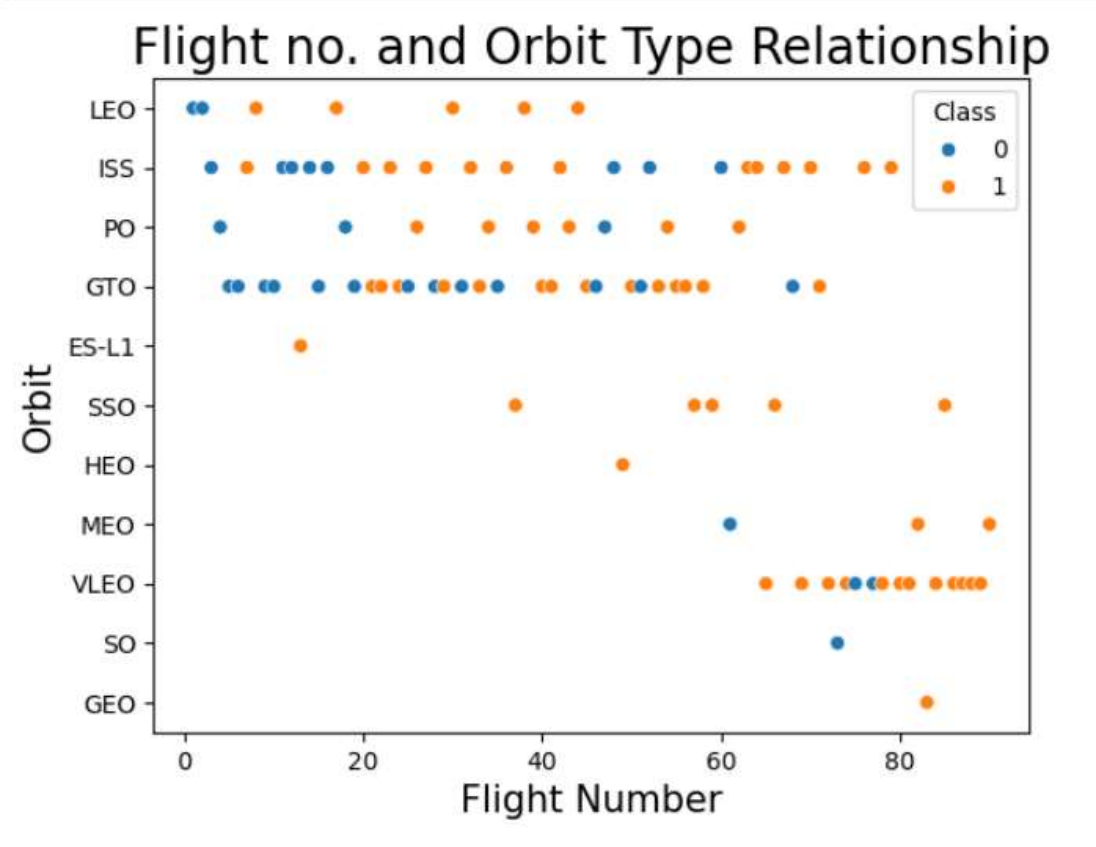
- Bar chart for the success rate of each orbit type



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

Flight Number vs. Orbit Type

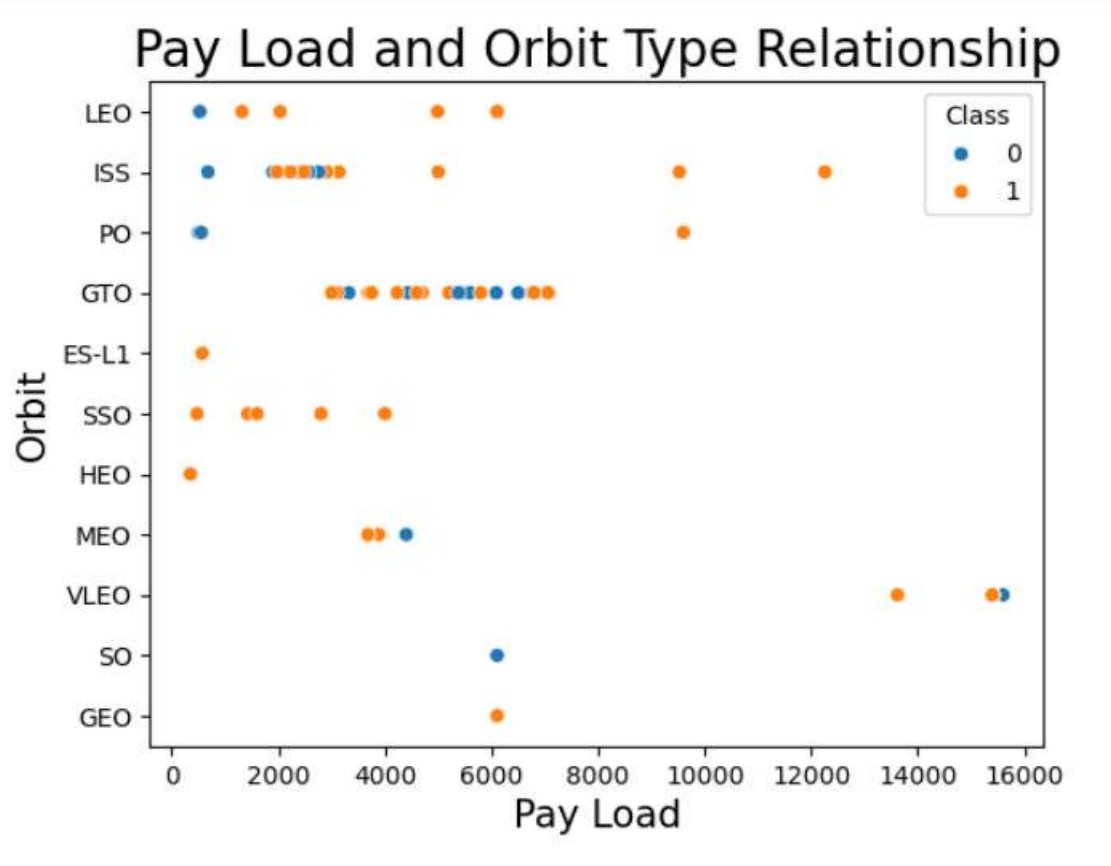
- Scatter point of Flight number vs. Orbit type



1. For orbit VLEO, first successful landing (class=1) doesn't occur until 60+ number of flights
2. For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers 22
3. There is no relationship between flight number and orbit for GTO

Payload vs. Orbit Type

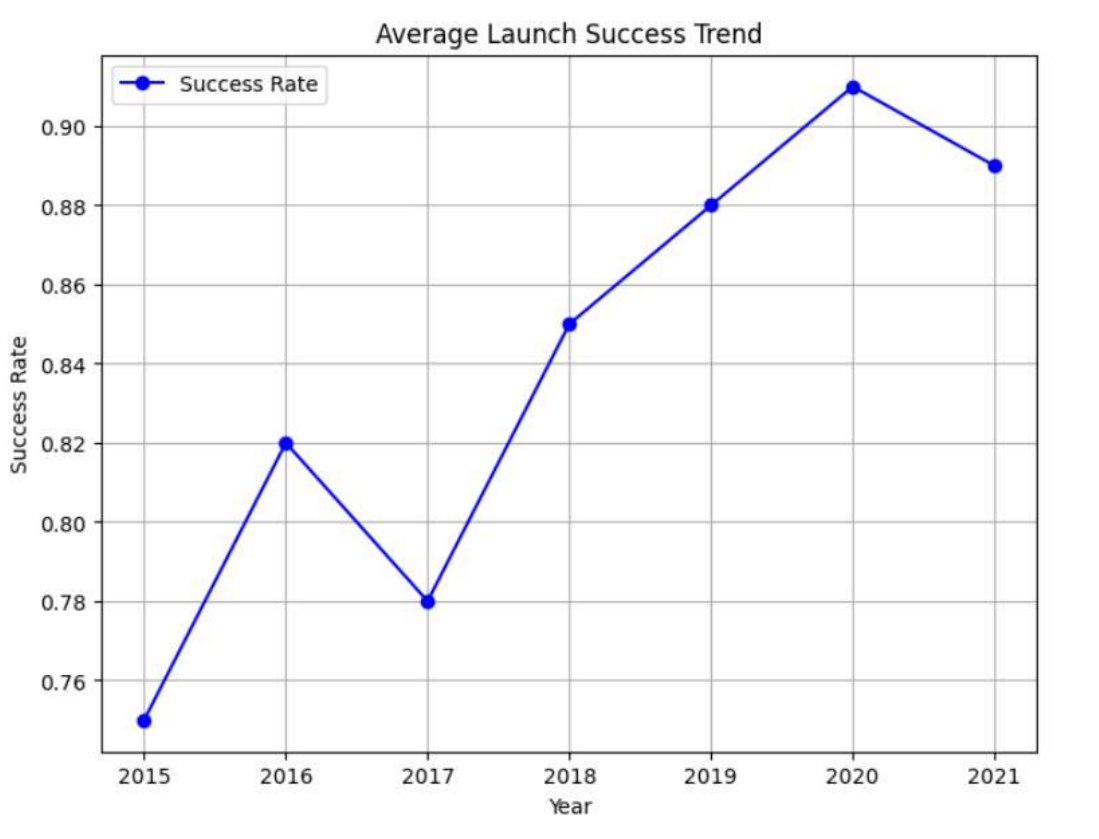
- Scatter point of payload vs. orbit type



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

Launch Success Yearly Trend

- Line chart of yearly average success rate



1. Success rate (Class=1) increased by about 80% between 2013 and 2020
2. Success rates remained the same between 2010 and 2013 and between 2014 and 2015
3. Success rates decreased between 2017 and 2018 and between 2019 and 24 2020

All Launch Site Names

- Find the names of the unique launch sites

Task 1

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

[10]:
```
%%sql

select distinct Launch_Site from spacextbl
```

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

[10]:

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

Description :

1. ‘distinct’ returns only unique values from the queries column (Launch_Site)

2. There are 4 unique launch sites

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

Task 2

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

[11]:
```
%%sql

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

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

[11]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Description :

1. Using keyword ‘Like’ and format ‘CCA%’, returns records where ‘Launch_Site’ column starts with “CCA”.

2. Limit 5, limits the number of returned records to 5

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

Task 3

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

```
[12]: %%sql

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

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

[12]:

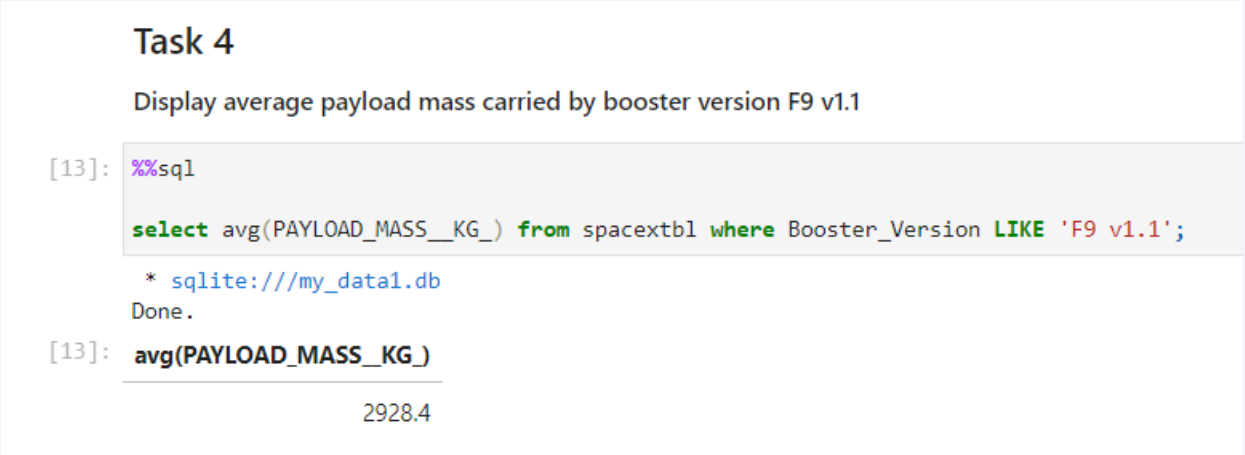
sum(PAYLOAD_MASS__KG_)
45596

Description :

1. ‘sum’ adds column ‘PAYLOAD_MASS_KG’ and returns total payload mass for customers named ‘NASA (CRS)’

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

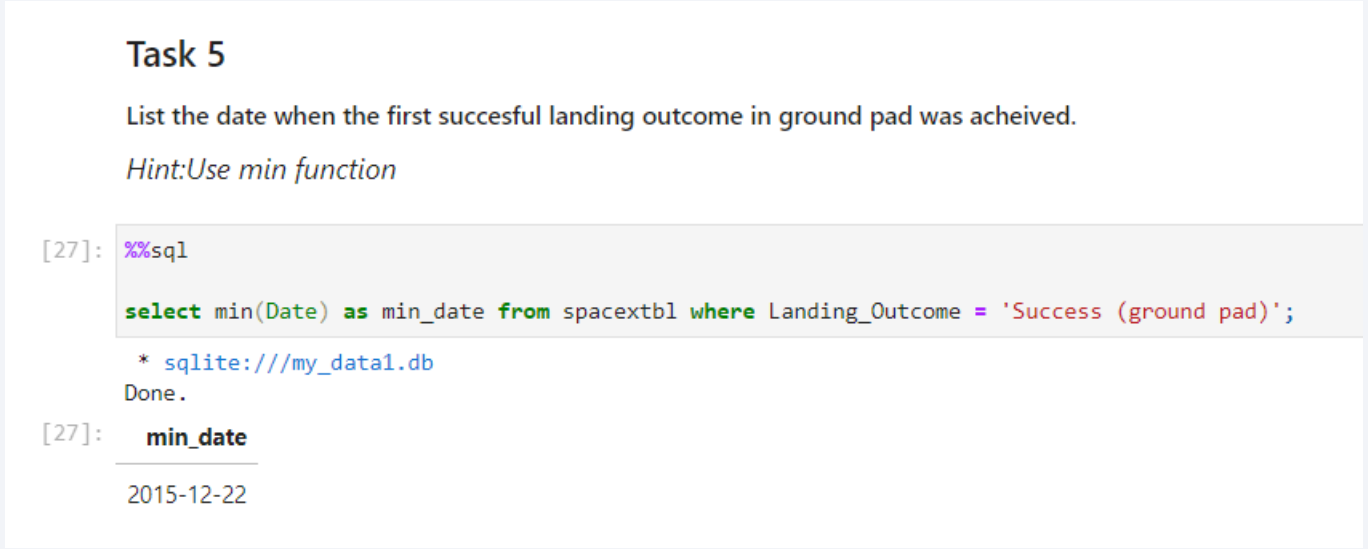


Description :

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

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad



Task 5

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

Hint:Use min function

[27]:
```
%%sql

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

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

[27]:

min_date
2015-12-22

Description :

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

Successful Drone Ship Landing with Payload between 4000 and 6000

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

Task 6

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

[29]:
```
%%sql

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

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

[29]:

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

Description :

1. 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
2. The ‘and’ operator in the where clause returns booster versions where both conditions in the where clause are true

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

[30]:
```
%%sql

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

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

[30]:

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

Description :

1. The ‘group by’ keyword arranges identical data in a column in to group.

2. In this case, number of mission outcomes by types of outcomes are grouped in column ‘counts’

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload

Task 8 ¶

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

[31]:
```
%%sql

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

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

[31]:

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

Description :

1. The sub query returns the maximum payload mass by using keywork ‘max’ on the pay load mass column

2. The main query returns booster versions and respective payload mass where payload mass is maximum with value of 15600

2015 Launch Records

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

[38]:
```
%%sql

SELECT Landing_Outcome, Booster_Version, Launch_Site
FROM spacextbl
WHERE Landing_Outcome = 'Failure (drone ship)' AND substr(Date, 0, 5) = '2015'
```

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

[38]:

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

Description :

1. The query lists landing outcome, booster version, and the launch site where landing outcome is failed in drone ship and the year is 2015
2. The ‘and’ operator in the where clause returns booster versions where both conditions in the where clause are true
3. The ‘year’ keywork extracts the year from column ‘Date
4. 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

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

- 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

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

[39]:
```
%%sql

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;
```

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

[39]:

Landing_Outcome	LandingCounts
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

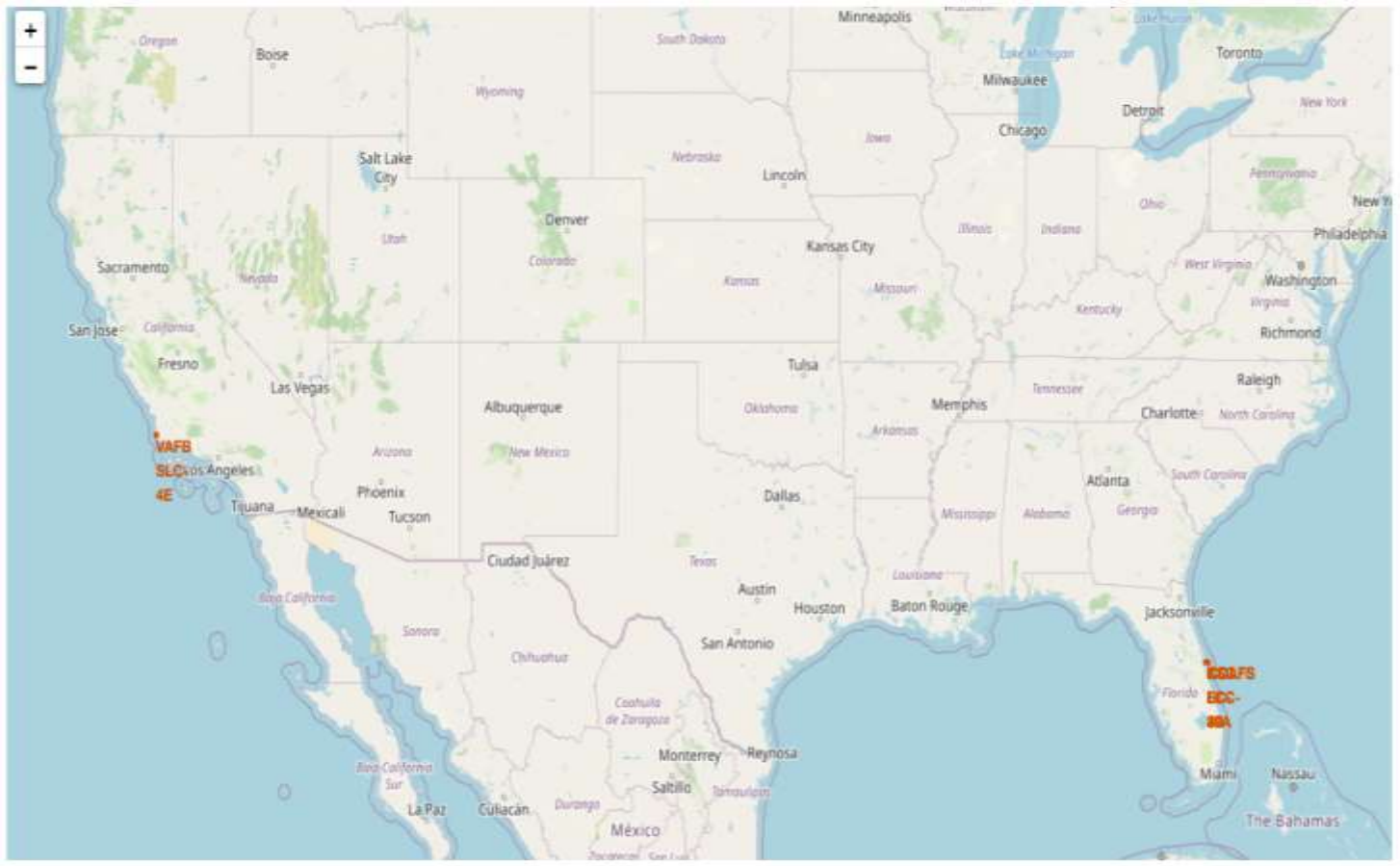
Description :

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

Section 3

Launch Sites Proximities Analysis

SpaceX Falcon9 – Launch Sites Map



1. Figure Displays the Global map with Falcon 9 launch sites that are located in the United States (in California and Florida).
2. Each launch site contains the name of the launch site. It is also evident that all launch sites are near the coast.
    - VAFB SLC-4E (CA)
    - CCAFS LC-40 (FL)
    - KSC LC-39A (FL)
    - CCAFS SLC-40 (FL)

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

Fig 1 – US map with all Launch Sites

1. 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.
2. Figure 2 zoom in to each site and displays the success/fail markers with green as success and red as failed.

Fig 2 – Launch Site with success/failed markers

SpaceX Falcon9 – Launch Site to proximity Distance Map

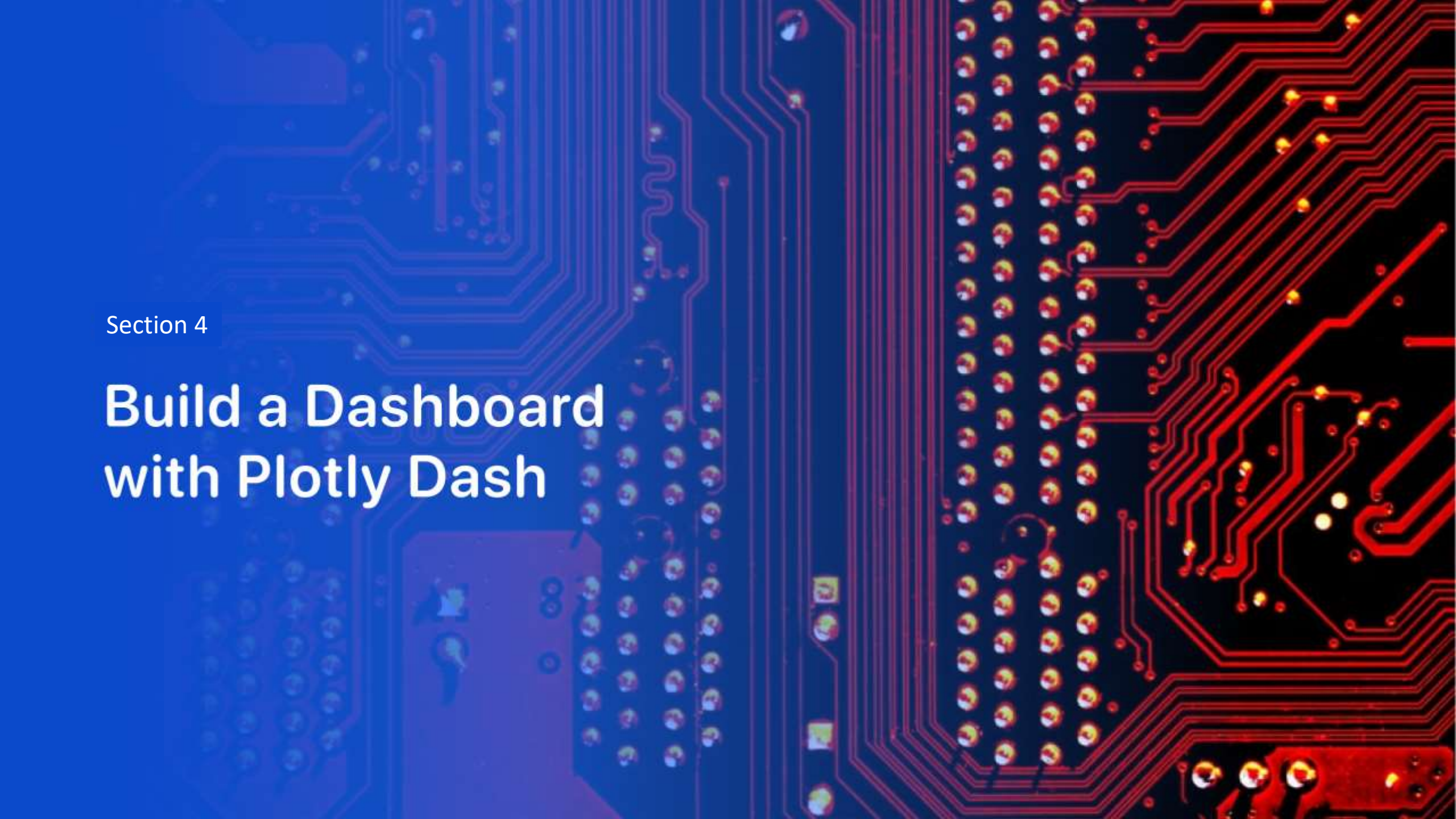
Fig 1 – Proximity site map

Figure 1 displays all the proximity sites marked on the map for Launch Site . 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.

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.

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

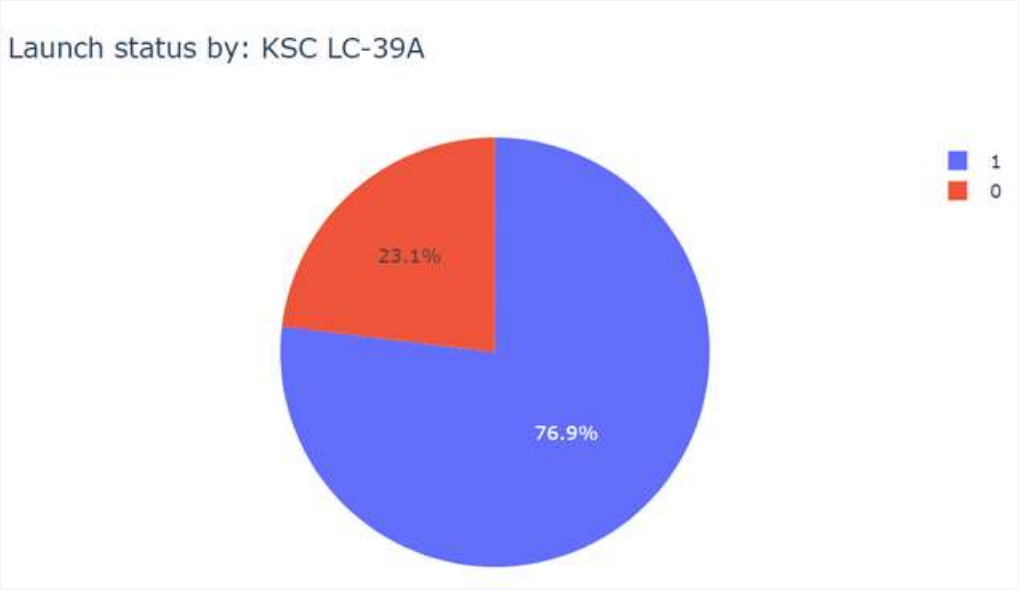

Section 4
Build a Dashboard with Plotly Dash

Launch Success Counts For All Sites

1. Launch Site ‘KSC LC-39A’ has the highest launch success rate

2. Launch Site ‘CCAFS SLC 40’ has the lowest launch success rate

Launch Site with Highest Launch Success Ratio



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

Payload vs. Launch Outcome Scatter Plot for All Sites

1. Most successful launches are in the payload range from 2000 to about 5500

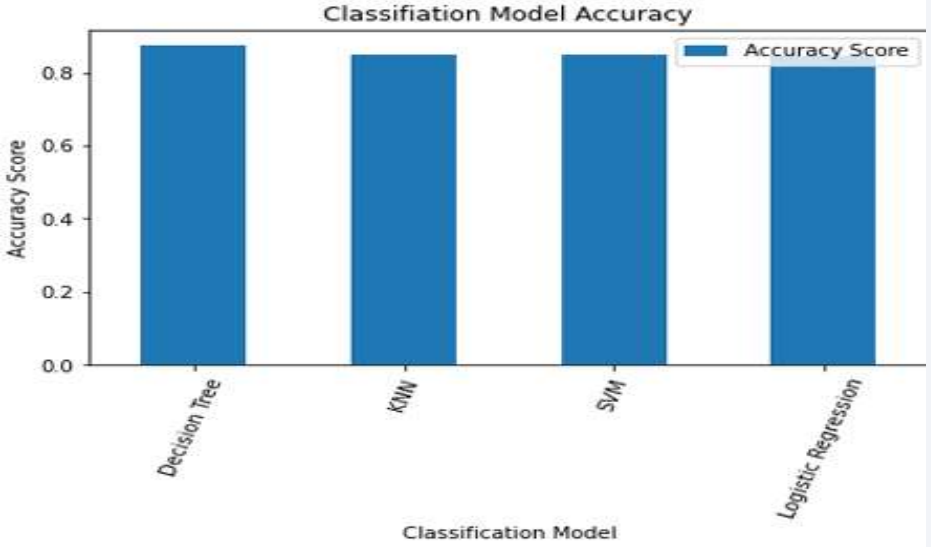
2. Booster version category ‘FT’ has the most successful launches

3. Only booster with a success launch when payload is greater than 6k is ‘B4’

Section 5

Predictive Analysis (Classification)

Classification Accuracy



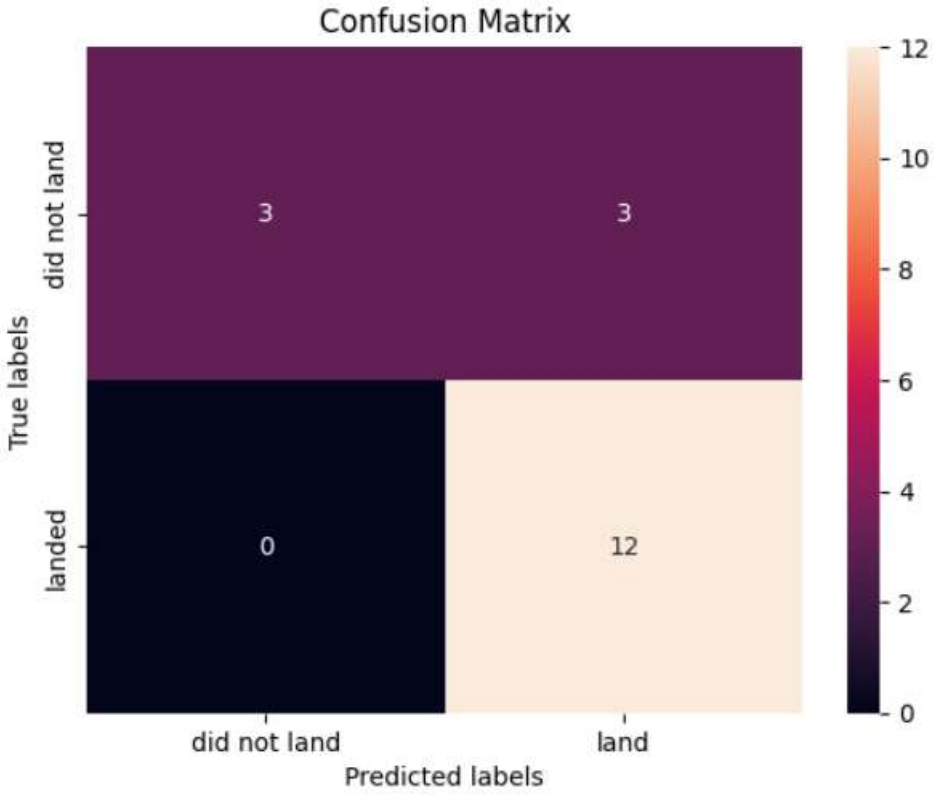
[40]:

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.666667
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

1. 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
2. Accuracy Score on the test data is the same for all the classification algorithms based on the data set with a value of .8333
3. Given that the Accuracy scores for Classifications 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

```
[39]: yhat = knn_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```



1. The confusion matrix is consistent across all models (Logistic Regression, SVM, Decision Tree, and KNN).
2. The classifier made a total of 18 predictions.
3. Out of these predictions:
   - 12 scenarios were correctly predicted as “Yes” for landing, and they did indeed land successfully (True positive).
   - 3 scenarios (top left) were correctly predicted as “No” for landing, and they did not land (True negative).
   - 3 scenarios (top right) were incorrectly predicted as “Yes” for landing, but they did not land successfully (False positive).
4. Overall, the classifier’s accuracy is approximately 83% ((True Positives + True Negatives) / Total), with a misclassification or error rate of about 16.5% ((False Positives + False Negatives) / Total).

Conclusions

1. **Flight Frequency and Landing Success:** As the number of flights increases, the likelihood of a successful first-stage landing also increases.
2. **Payload and Success Rates:** While success rates tend to rise with increasing payload mass, there is no clear linear correlation between payload mass and success rates.
3. **Improvement Over Time:** The launch success rate has seen a significant increase of approximately 80% from 2013 to 2020.
4. **Launch Sites and Success:** Among launch sites, ‘KSC LC-39A’ boasts the highest success rate, while ‘CCAFS SLC 40’ has the lowest.
5. **Orbit Types and Success:** Orbits such as ES-L1, GEO, HEO, and SSO exhibit the highest success rates, whereas GTO (Geostationary Transfer Orbit) has the lowest.
6. **Strategic Site Locations:** Launch sites are strategically positioned away from cities and closer to coastlines, railroads, and highways.
7. **Best Model Performance:** The Decision Tree model performs best, achieving an accuracy of approximately 87.5%. However, all models scored around 83% accuracy on the test data. Further data and tuning may enhance model performance

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!