Photo: https://balkangreenenergynews.com/

IBM Developer
SKILLS NETWORK

Analysis of SpaceX Launch Data

John Isaac Enriquez

January 20, 2022

Outline

Executive Summary

Introduction

Methodology

Results

Conclusion

Appendix

IBM Developer
SKILLS NETWORK

Section 1

Executive Summary

Executive Summary

Methodology

The SpaceX launch data were collected and analyzed using various data science methods. Data collection is performed using web scraping and API requests. Preliminary data exploration was performed using SQL queries. Location data were visualized using Folium maps. A dashboard containing selected data features was also developed. The data was then imported and processed in Python Pandas and the features that correlate to successful first stage rocket landing were selected. Four classification methods were used to develop machine learning models that will predict the outcome of every launch.

Results

There are four launch sites, one of which is in Los Angeles and three are in Florida. Location data shows that the launch sites are in close proximity to the coast. The features that correlates to the launch outcome includes the launch date, launch site, payload weight, and orbit type. The developed machine learning models are able to predict the outcome of the launch with accuracy scores of 0.833 for the logistic regression, support vector machine, and K nearest neighbor, and 0.778 for the decision tree.

Photo: https://www.cbsnews.com/

Section 2

Introduction

Introduction

SpaceX is an American aerospace manufacturer, space transportation services, and communications corporation in Hawthorne, California. SpaceX manufactures the Falcon 9 launch vehicle that has a reusable first stage. This feature makes SpaceX Falcon 9 services about million dollar cheaper compared to other providers.

The objective of this project is to predict whether the first stage of Falcon 9 will successfully land. This determines the cost of the launch which is a valuable information for a competitor who wants to bid against SpaceX for a rocket launch. This will be done by gathering, analyzing, and modelling SpaceX launch data such as rocket used, payload delivered, launch specifications, landing specifications, and landing outcome which is available through SpaceX REST API.

The general objective of this project is to answer the following questions:

What features determines the launch outcome?

What machine learning method best predicts the outcome?

Photo: https://www.discovermagazine.com/

IBM Developer
SKILLS NETWORK

Section 3

Methodology

Methodology

Executive Summary

Data collection methodology:

Data was collected using API requests and web scraping

Perform data wrangling

Data was processed using Python Pandas

Perform exploratory data analysis (EDA) using visualization and SQL

EDA was performed using SQL queries and visualized using Matplotlib and Seaborn graphs

Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

Classification models were built using Python Scikit Learn module. The data set was divided into training and testing sets. The features were preprocessed using StandardScaler function. The parameters were tuned using GridSearchCV. Finally, the accuracy of each model was evaluated using the score function.

Data Collection

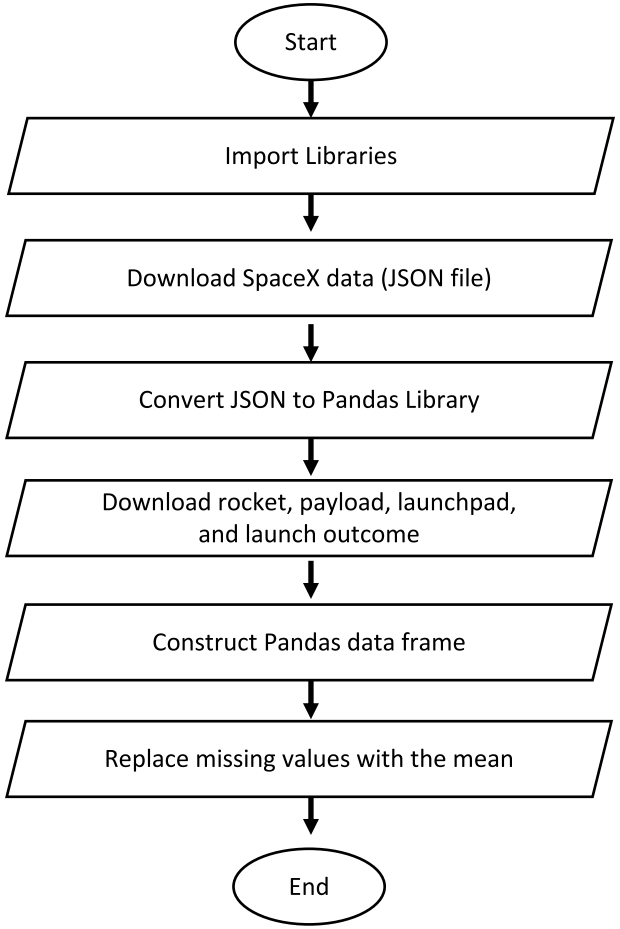
SpaceX launch data were collected from the SpaceX REST API and SpaceX wiki page. These data were converted into a Pandas library where they were processed for further analysis. The details of the data collection method are shown in the next section.

Data Collection – SpaceX API

1. Import Requests, Pandas, Numpy, and Datetime libraries
2. Use Requests to download SpaceX launch history data from REST API
   `requests.get(spacex_url)`
3. Convert the downloaded JSON file to Pandas library
   `pd.json_normalize(file.json())`
4. Collect the rocket, payload, launchpad, and the launch outcome data from the appropriate REST API
   getBoosterVersion(data), getLaunchSite(data), getPayloadData(data), getCoreData(data)
5. Construct Pandas data frame
```
launch_dict = {'FlightNumber': list(data['flight_number’]),
'Date': list(data['date’]), 'BoosterVersion':BoosterVersion, …
data_falcon = pd.DataFrame.from_dict(launch_dict)
```
6. Check for missing values and replace missing values with the mean
```
data_falcon9.isnull().sum()
mean = data_falcon9['PayloadMass'].mean()
data_falcon9['PayloadMass'].fillna(value=mean, inplace=True)
```

The completed SpaceX API calls notebook can be found on this GitHub URL:

https://github.com/johnisaacenriquez/learning/blob/main/IBM_Data_Science_Capstone_Project/Data%20Collection%20API.ipynb



Data Collection - Scraping

1. Import Requests, BeautifulSoup, and Pandas libraries
2. Use Requests to download and create an object from the Falcon 9 Launch Wiki page

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

3. Create a BeautifulSoup object and use find_all function to parse elements with type ‘table’

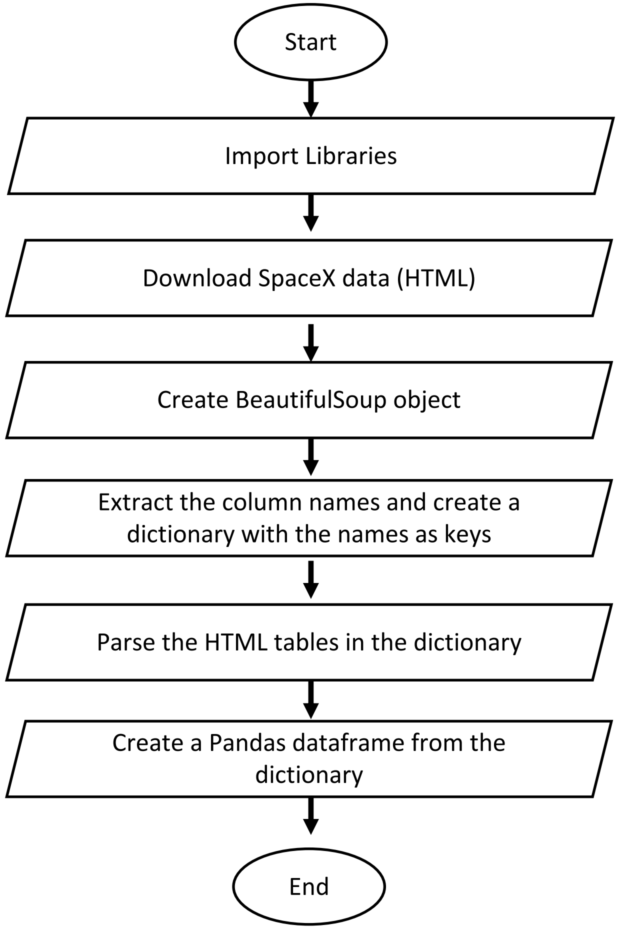
```
soup = BeautifulSoup(page.text, ‘html5lib’)
html_tables = soup.find_all(‘table’)
first_launch_table = html_tables[2]
```

4. Printing the first_lauch_table will show that the column names are embedded in table header elements <th>. Select ‘th’ elements using find_all and extract the columns using the extract_column_from_header() function.

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

5. Create an empty dictionary with keys from the extracted columns

```
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
...
```



Data Collection - Scraping

6. Parse the launch HTML tables to the launch dictionary . Use the provided helper functions if appropriate.

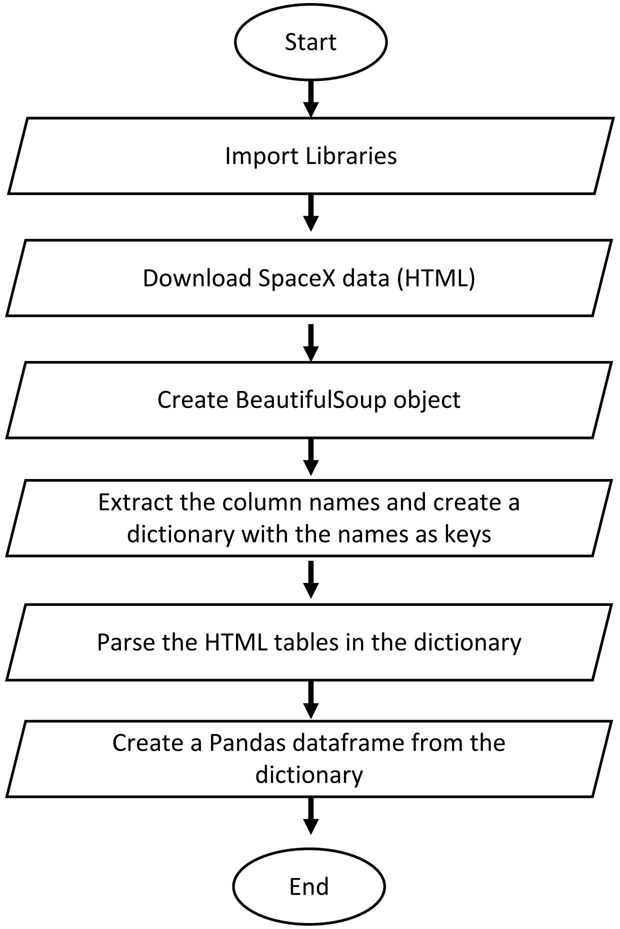
```
for table_number,table in
enumerate(soup.find_all('table',"wikitable plainrowheaders
collapsible")):
      for rows in table.find_all("tr"):
               ...
               date = datatimelist[0].strip(',')
               launch_dict["Date"].append(date)
               time = datatimelist[1]
               launch_dict["Time"].append(time)
               launch_site = row[2].a.string
               launch_dict["Launch site"].append(launch_site)
               payload_mass = get_mass(row[4])
               launch_dict["Payload mass"].append(payload_mass)
               ...
```

7. Delete keys in launch dictionary with None values. Finally, create a dataframe from the launch dictionary

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

The completed web scraping notebook can be found on this GitHub URL:

https://github.com/johnisaacenriquez/learning/blob/main/IBM_Data_Science_Capstone_Project/Data%20Collection%20Webscraping.ipynb



Data Wrangling

Exploratory data analysis was performed on SpaceX Launch dataframe to find some patterns in the data and determine the label for the landing outcome column.

1. Import Pandas and Numpy ibraries

2. Import SpaceX Launch csv file into Pandas

```
df=pd.read_csv("https://cf-courses-
data.s3.us.cloud-object-
storage.appdomain.cloud/IBM-DS0321EN-
SkillsNetwork/datasets/dataset_part_1.csv")
```

3. Identify the amount of missing table values and the types of data

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

4. Calculate the number of launches on each launch site

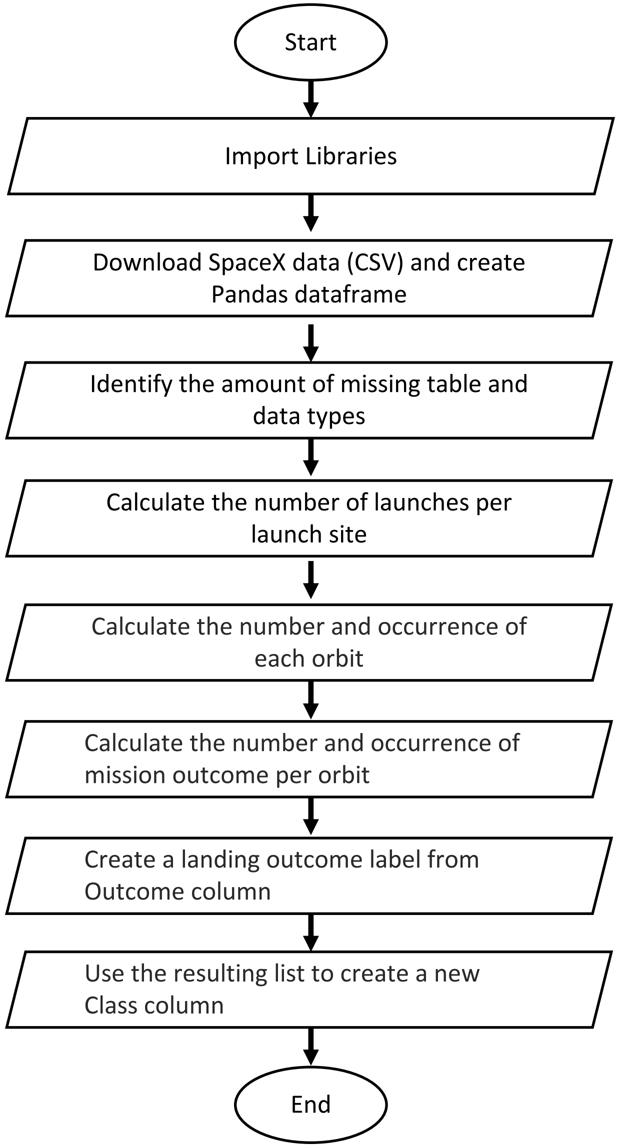
```
df[['LaunchSite']].value_counts()
```

5. Calculate the number and occurrence of each orbit

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

6. Calculate the number and occurrence of mission outcome per orbit

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



Data Wrangling

7. The ‘True’ and ‘False’ values means successful and unsuccessful landings, respectively. Create a list of unsuccessful (bad) outcome types.

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

8. Create a landing outcome label from Outcome column, setting 0 and 1 for unsuccessful and successful outcomes, respectively.

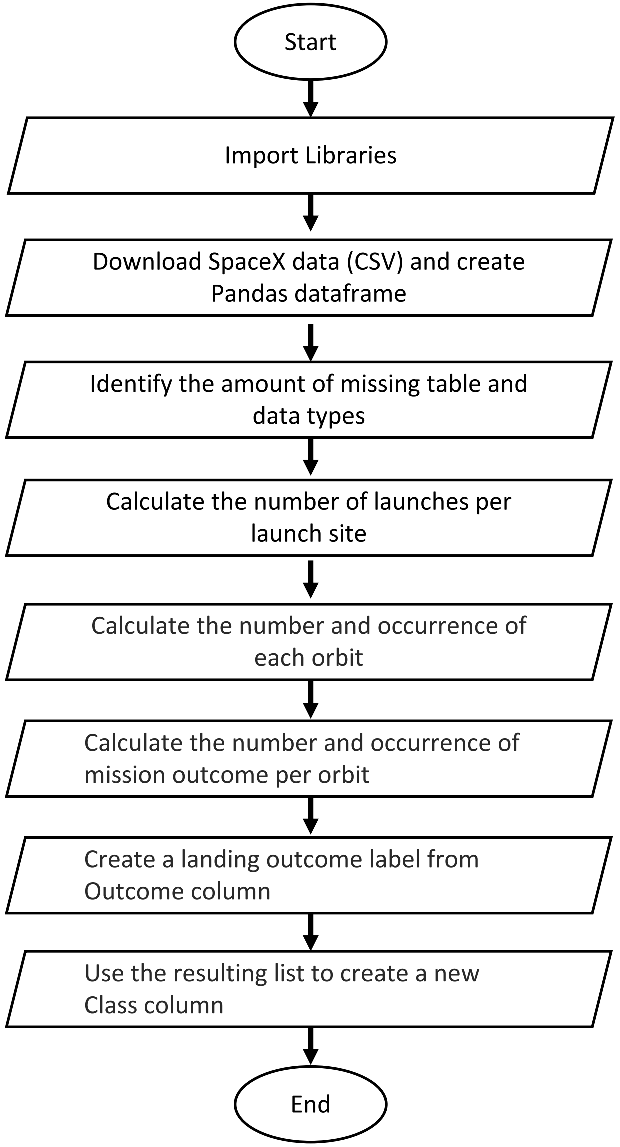
```
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
      landing_class.append(0)
    else:
      landing_class.append(1)
```

9. Create a new column ’Class’ on the dataframe from the landing_class list.

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

The completed data wrangling notebook can be found on this GitHub URL:

https://github.com/johnisaacenriquez/learning/blob/main/IBM_Data_Science_Capstone_Project/Data%20Wrangling.ipynb



EDA with Data Visualization

Scatterplots and categorical scatterplots were used to investigate the relationship between the following variables:

1. Payload vs Flight Number (catplot)
2. Flight Number vs Launch Site (catplot)
3. Payload vs Launch Site (catplot)
5. Flight Number vs Orbit (scatter plot)
6. Payload vs Orbit (scatter plot)

The success rate on each orbit is graphed using bar chart.

Finally, the success rate as a function of time (year) was plotted using a line chart.

The completed EDA with data visualization notebook can be found on this GitHub URL:

https://github.com/johnisaacenriquez/learning/blob/main/IBM_Data_Science_Capstone_Project/EDA%20with%20Data%20Visualization.ipynb

Build an Interactive Map with Folium

Folium was used to mark launch sites on a map, showing which of these sites has successful and unsuccessful landings.

- Markers and circles were created to mark the launch sites
- Cluster of markers were created to mark successful launches (green) and unsuccessful launches (red)
- The distances between a launch site and nearby railway, highway, and city were calculated. The distances were represented by lines and labeled using markers

The completed Interactive Visual Analytics with Folium notebook can be found on this GitHub URL:

https://github.com/johnisaacenriquez/learning/blob/main/IBM_Data_Science_Capstone_Project/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

Interactive visual analytics is performed by building a Plotly Dash Application. A pie chart that shows the success rate of each launch site, and the relative success rate between each launch site is added. The type of pie chart is selected using a drop down object. In addition, a scatter chart that shows the success of each booster version for every launch site is added. A slider that controls the payload range for the booster versions is also added.

The completed Plotly Dash lab Python code can be found on this GitHub URL:

> https://github.com/johnisaacenriquez/learning/blob/main/IBM_Data_Science_Capstone_Project/spacex_dash_app.py

Predictive Analysis (Classification)

Logistic regression, support vector machine, decision tree, and K nearest neighbor classification models were built from scikit-learn library. The features were preprocessed using StandardScaler. The data were split into training and testing data sets. For each classification model, GridSearchCV was used to search over and choose the best parameter values. Each parameter were used to fit the training data. The models were then used to make predictions using the test data. The results of the prediction were presented using confusion matrix. The accuracy of each model was calculated using Score function.

The completed Plotly Dash lab Python code can be found on this GitHub URL:

https://github.com/johnisaacenriquez/learning/blob/main/IBM Data Science Capstone Project/SpaceX Machine%20Learning%20Prediction Part 5.ipynb

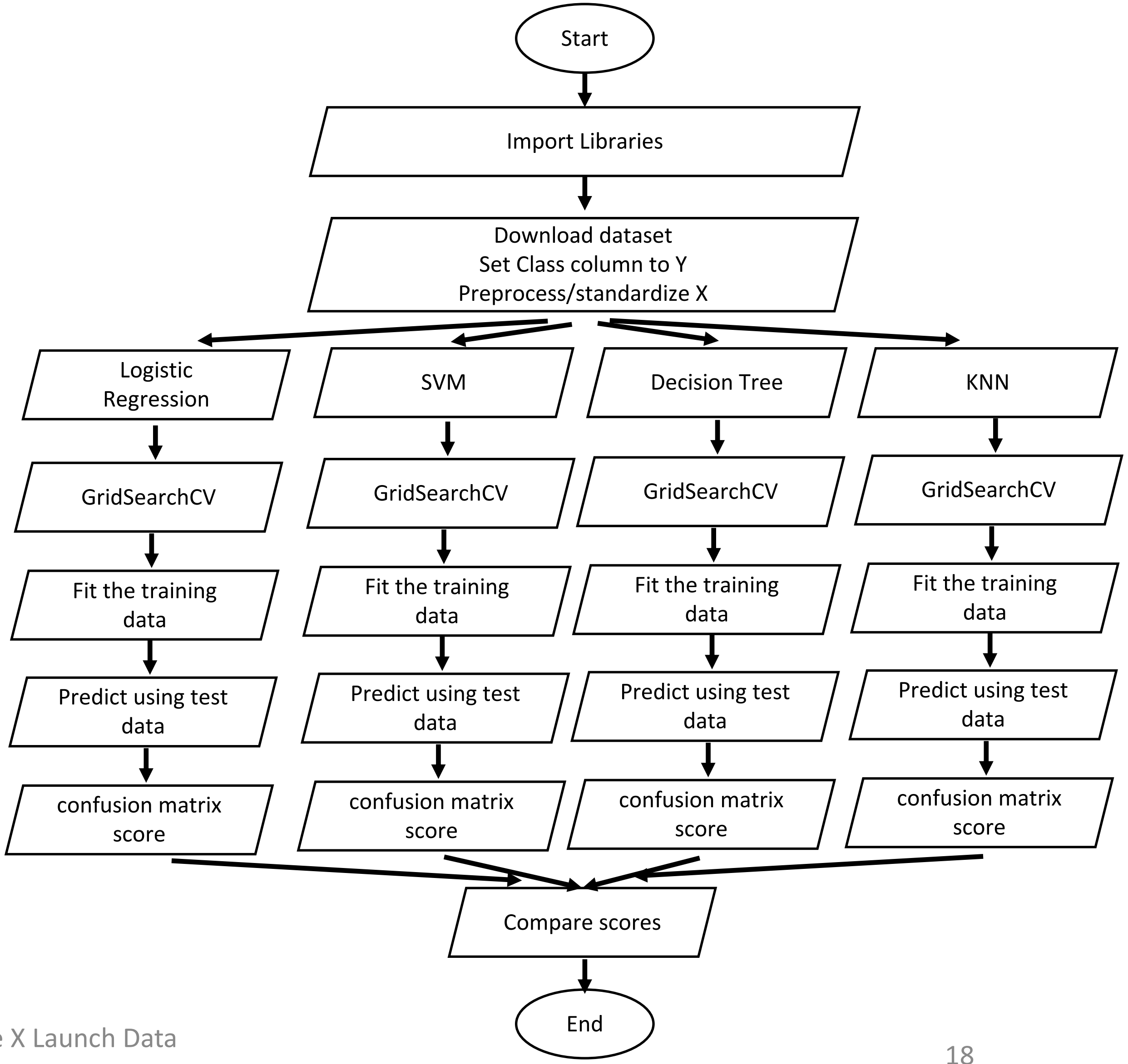


Photo: https://www.universetoday.com/

IBM Developer
SKILLS NETWORK

Section 4

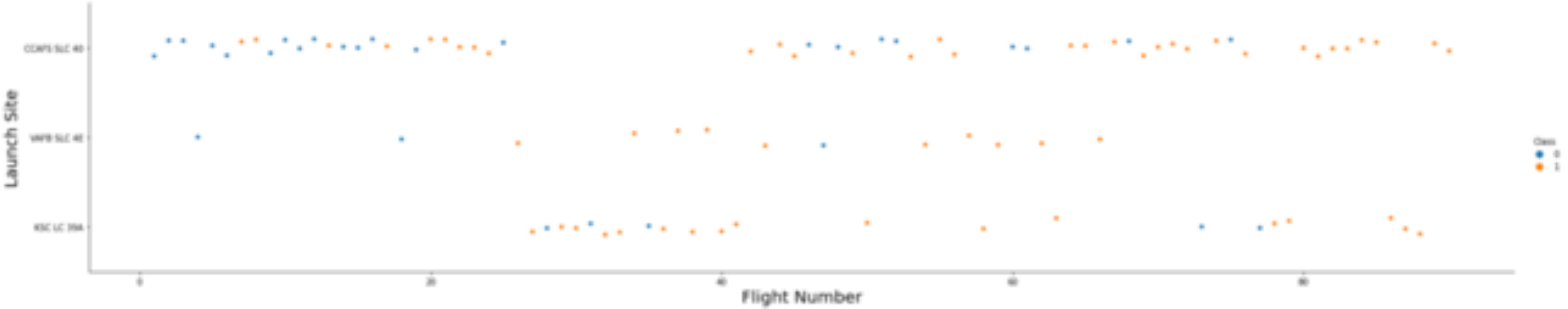
Results

Photo: https://www.forbes.com/

IBM Developer
SKILLS NETWORK

EDA with Visualization

1. Flight Number vs. Launch Site



The figure shows the scatter plot of the flight number vs launch site. Successful and unsuccessful landing of the first stage rocket were represented by blue and orange data points, respectively.

At earlier SpaceX operations (Flight number < 20), the rockets were launched mostly from CCAFS SLC launch site and the success rate were fairly mixed. Two rockets were launched from VAFB SLC but both were unsuccessful. At succeeding SpaceX operations (Flight number > 20), the launches from VAFB SLC and KSC LC were mostly successful. While the launches at CCAFS SLC have mixed result, it has been all successful for the most recent SpaceX operations (Flight number < 80).

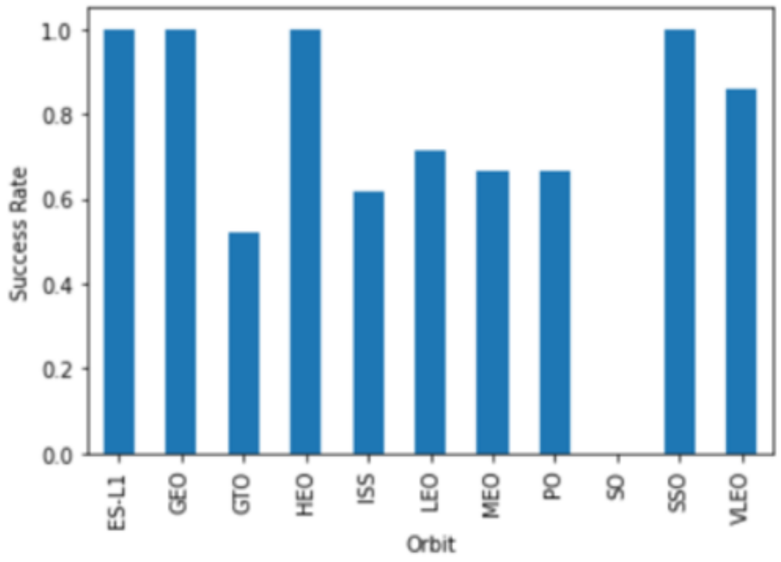
In general, most of the launches above Fight number 60 have been successful for all launch sites.

2. Payload vs. Launch Site

The figure shows the scatter plot of the flight number vs pay load mass. Successful and unsuccessful landing of the first stage rocket were represented by blue and orange data points, respectively.

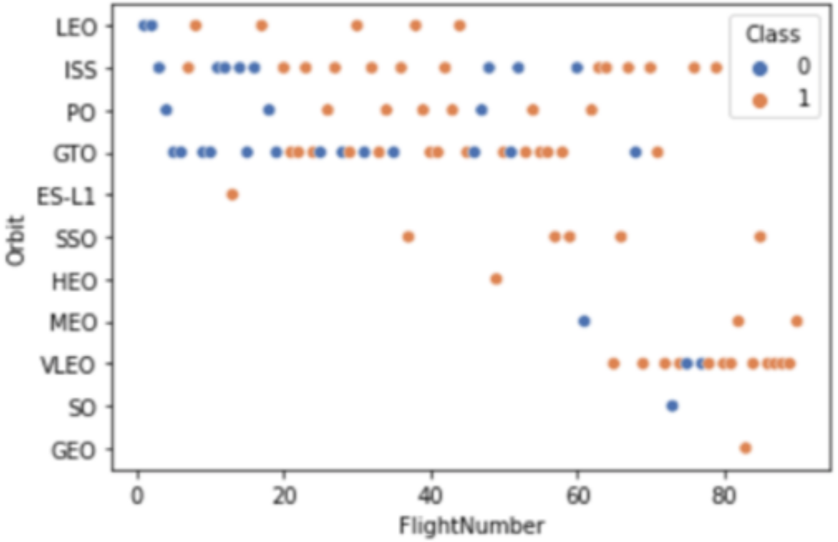
Most of the rockets with payloads of exactly 10000 kg were launched from VAFB SLC. Payloads heavier than 10000 kg were launched from either CCAFS SLC or KSK LC. While most of the rockets were launched from CCAFS SLC, rockets with payloads of between ~8000 kg to ~13000 kg were not launched from this site.

3. Success Rate vs. Orbit Type



The bar chart shows the success rate of recovering the first stage rocket for each orbit. Rocket launched to the ES-L1, GEO, HEO, SSO, and VLEO orbits have been all successful. LEO, MEO, PO, ISS, and GTO have success rates of between 0.5-0.7. A rocket launched to SO has resulted to failure.

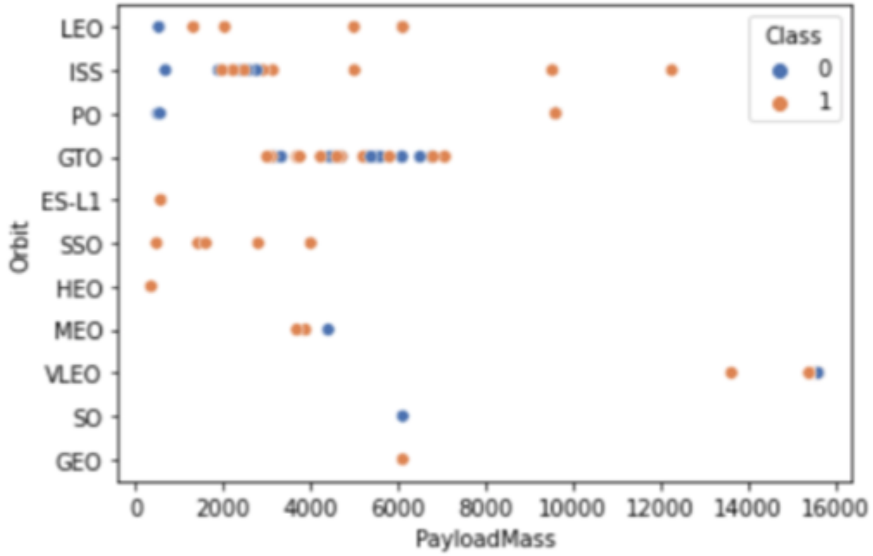
4. Flight Number vs. Orbit Type



The figure shows the scatter plot of the flight number vs orbit type. Successful and unsuccessful landing of the first stage rocket were represented by blue and orange data points, respectively.

Most rockets have been launched to ISS and GTO orbit. Rockets have been launched to ISS orbit from earlier flights to the most recent flights. The frequency of rocket launch to GTO has decreased recently. Since flight number 60, significant number of launches were aimed to VLEO orbit, most of which are successful.

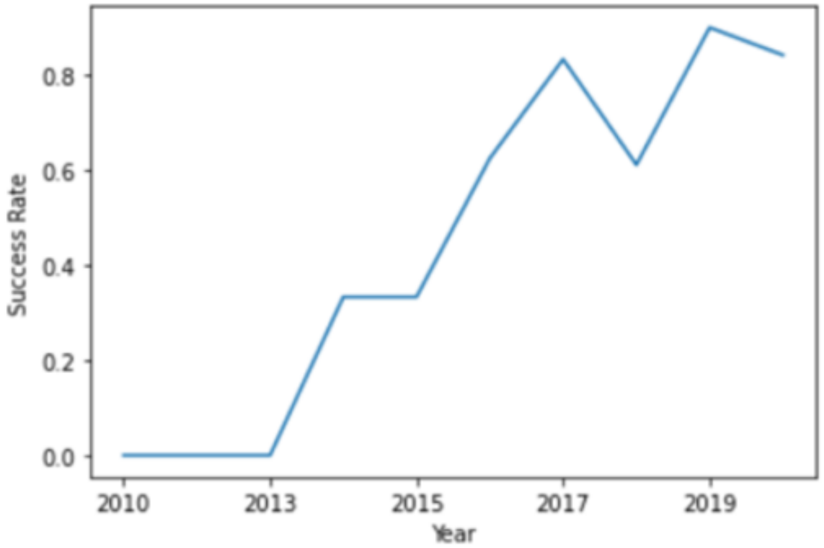
5. Payload vs. Orbit Type



The figure shows the scatter plot of the payload mass vs orbit type. Successful and unsuccessful landing of the first stage rocket were represented by blue and orange data points, respectively.

Heavy payloads (10000 kg or more) were launched mostly to ISS and VLEO orbits. The payloads launched to ISS range from the lightest to the heaviest, while only heavy payloads were launched to VLEO. The payload mass of rocket launched to GTO is convoluted to values between 3000 – 8000 kg.

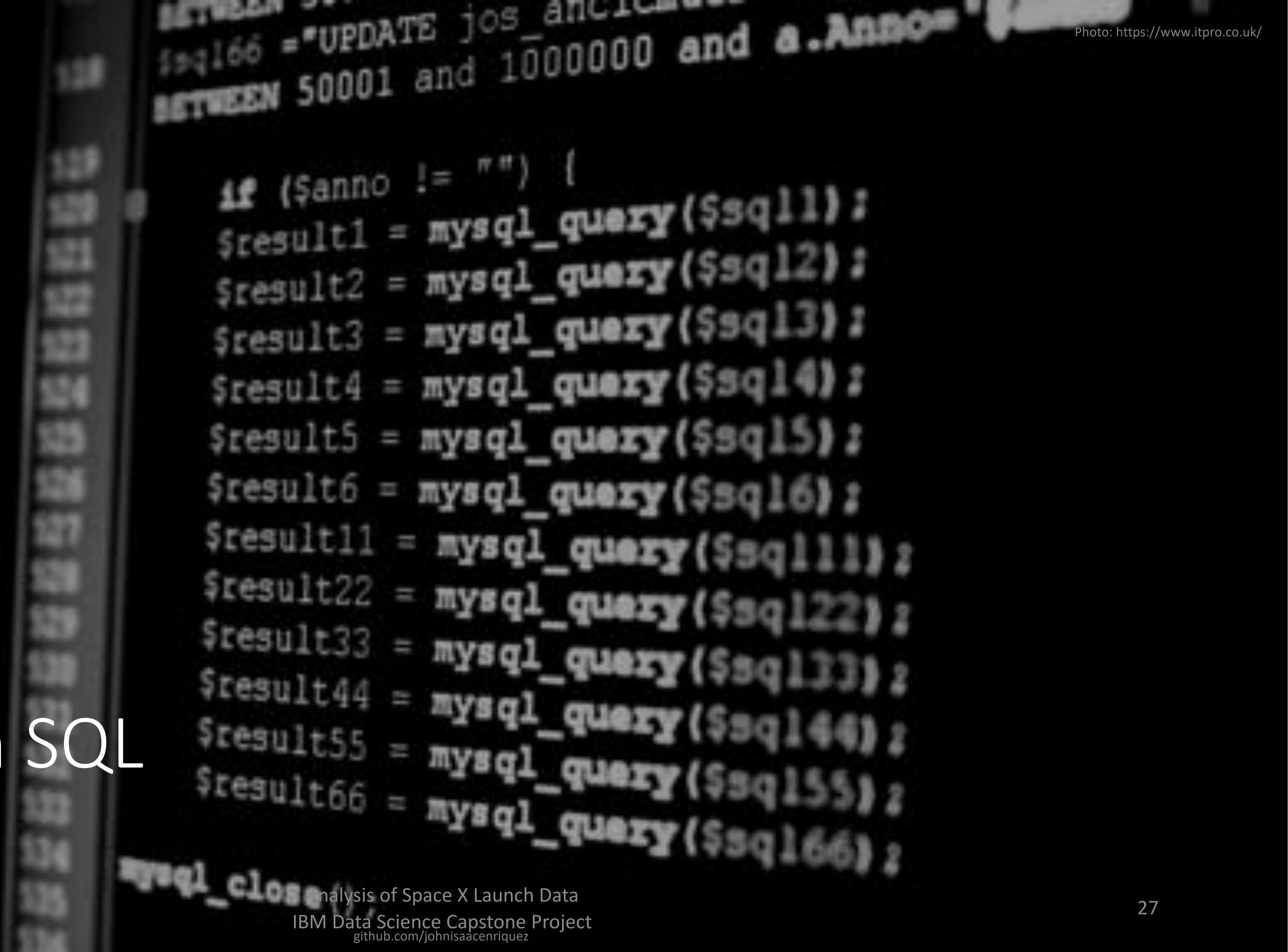
6. Launch Success Yearly Trend



The figure shows the yearly trend of the launch success. From 2010-2013, all attempts to land the first stage rocket results to failure. Successful launches were recorded between 2013-2014. The success rate had been stagnant from 2014 to 2015 (~0.3), but increased significantly from 2015 to 2017 (0.8). A decrease in the success rate was recorded in 2018 (0.6) but it jumps back up to around 0.9 in 2019.

IBM Developer
SKILLS NETWORK

Photo: https://www.itpro.co.uk/



EDA with SQL

1. All Launch Site Names

Unique launch site names were queried from the data base using the command

```
%sql SELECT
UNIQUE(launch_site) FROM
SPACEXTBL
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, and LAUNCH_SITE is the column header.

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

2. Launch Site Names Begin with 'CCA'

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Five records where launch sites begin with 'CCA' were queried using the command

```
%sql SELECT * FROM SPACEXTBL WHERE (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, and LAUNCH_SITE is the column header. The '%' character in 'CCA%' is a wildcard character.

3. Total Payload Mass

The total payload carried by boosters from NASA were queried from the data base using the command

```
%sql SELECT SUM(payload_mass__kg_) AS sum_payload FROM SPACEXTBL WHERE (customer) = 'NASA (CRS)'
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, and PAYLOAD_MASS__KG is the column header of the payload mass column.

sum_payload
45596

4. Average Payload Mass by F9 v1.1

The average payload mass carried by booster version F9 v1.1 were queried from the data base using the command

```
%sql SELECT AVG(payload_mass__kg_) AS average_payload FROM SPACEXTBL WHERE (booster_version) = 'F9 v1.1'
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, PAYLOAD_MASS__KG is the column header of the payload mass column, and BOOSTER_VERSION is the column header for the name of boosters.

average_payload
2928

5. First Successful Ground Landing Date

The date of the first successful landing outcome on ground pad were queried from the data base using the command

```
%sql SELECT MIN(date) FROM SPACEXTBL WHERE
landing__outcome = 'Success (ground pad)'
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, and LANDING__OUTCOME is the column header for the landing outcomes.

1
2015-12-22

6. Successful Drone Ship Landing with Payload between 4000 and 6000 kg

The names of boosters which have successfully landed on drone ship and had payload greater than 4000 kg but less than 6000 kg were queried from the data base using the command

```
%sql SELECT booster_version FROM SPACEXTBL WHERE landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 AND 6000
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, LANDING__OUTCOME is the column header for the landing outcomes, and PAYLOAD_MASS__KG_ is the column header for the payload mass.

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

7. Total Number of Successful and Failure Mission Outcomes

The total number of successful and failure mission outcomes were queried from the data base using the command

```
%sql SELECT mission_outcome, COUNT(mission_outcome) AS outcome FROM SPACEXTBL GROUP BY mission_outcome
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, MISSION_OUTCOME is the column header for the mission outcomes

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

8. Boosters that Carried Maximum Payload

The names of the booster which have carried the maximum payload mass were queried from the data base using the command

```
%sql SELECT booster_version FROM SPACEXTBL WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL)
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, BOOSTER_VERSION is the column header for the names of the boosters, and PAYLOAD_MASS__KG is the column header for the the mass of the payloads

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

9. 2015 Launch Records

The failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 were queried from the data base using the command

```
%sql SELECT date, booster_version, launch_site, landing__outcome FROM SPACEXTBL WHERE landing__outcome = 'Failure (drone ship)' AND YEAR(date) = '2015'
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, LAUNCH_SITE, LANDING__OUTCOME, and YEAR(date) are the column headers for the launch sites, landing outcomes, and year of the launch date.

DATE	booster_version	launch_site	landing__outcome
2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

The count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20 were queried from the data base and ranked using the command

```
%sql SELECT landing__outcome, COUNT(*)
AS count_launches FROM SPACEXTBL WHERE
DATE BETWEEN '2010-06-04' AND '2017-03-
20' GROUP BY landing__outcome ORDER BY
count_launches DESC
```

where %sql is the magic function for Jupyter notebook, SPACEXTBLE is the name of the table, LANDING__OUTCOME is the column header for the landing outcomes, and COUNT_LAUNCHES is the column header for representing total number of launches calculated from the total number of rows

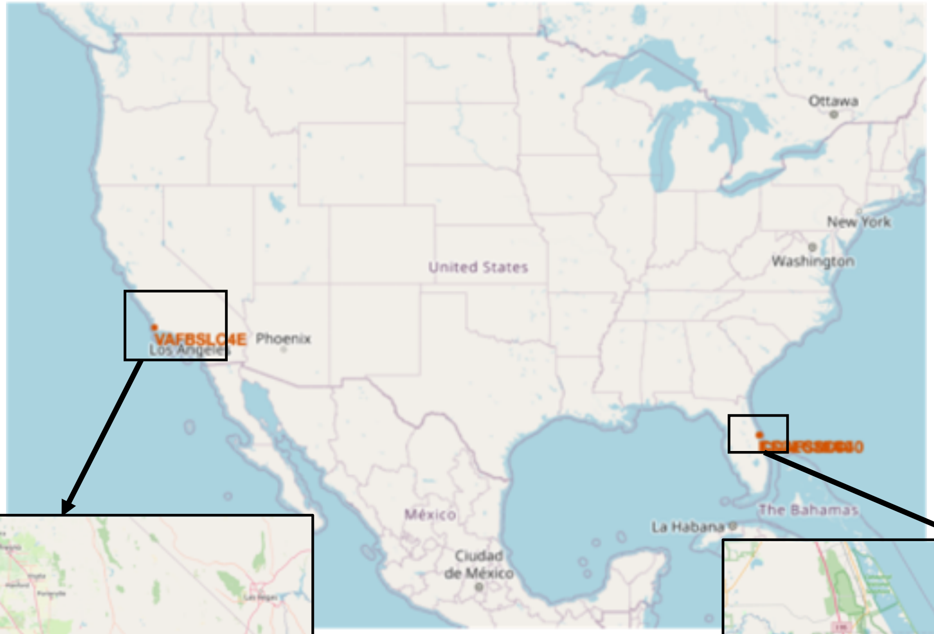
landing__outcome	count_launches
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Photto: https://spacecoastdaily.com/

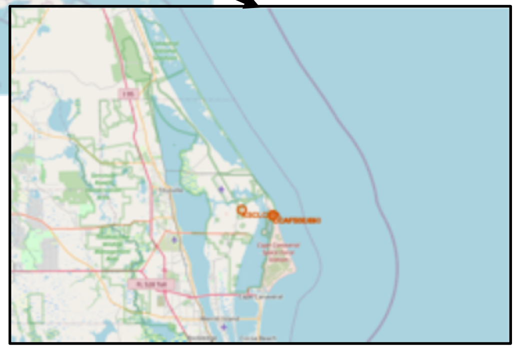
Launch Sites Proximities Analysis

1. Launch Site Locations

Launch sites were located either in Los Angeles or Florida. All launch sites are in close proximity to the coast and are located in the southernmost part of the United States.



VAFB SLC in Los Angeles

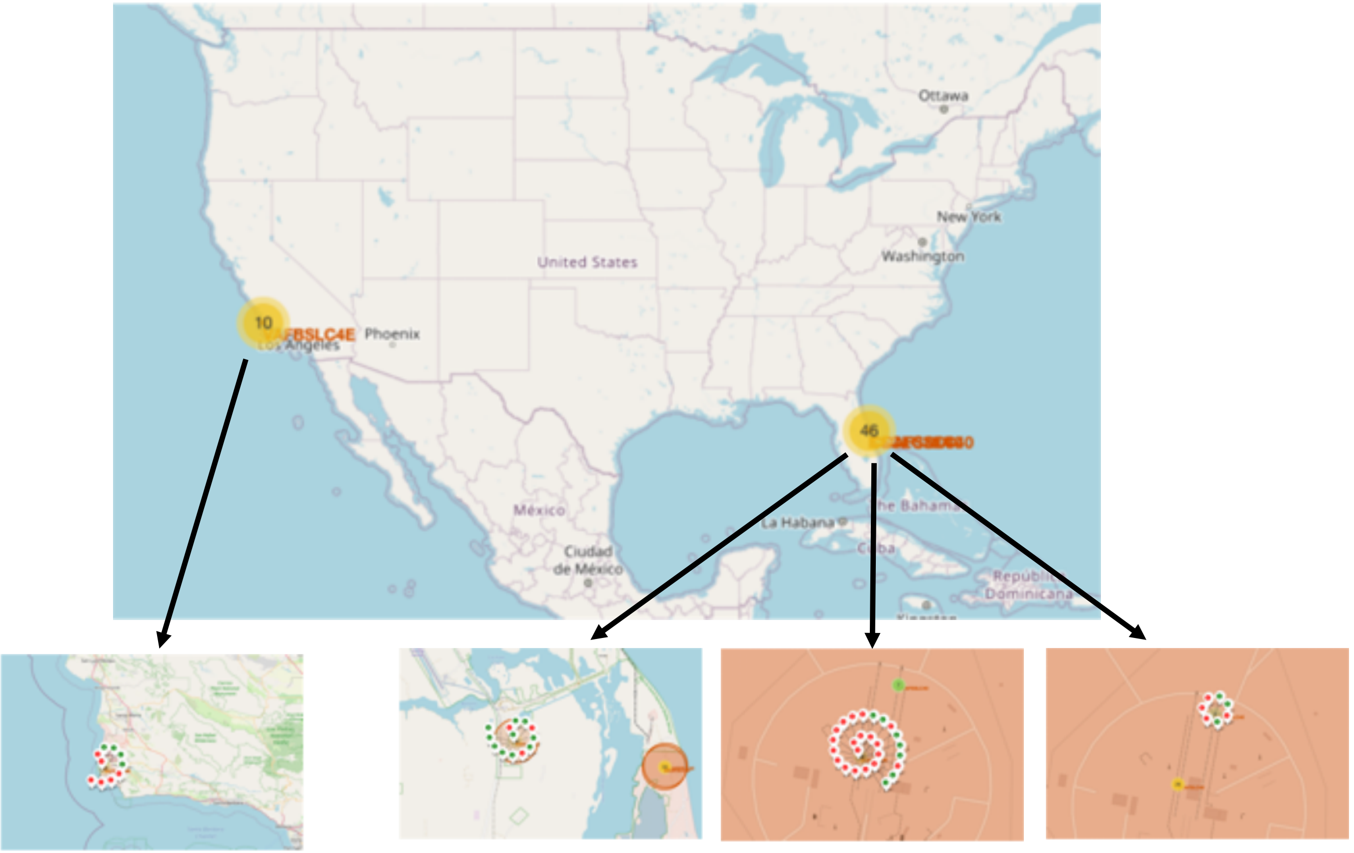


CCAFS LC, CCAFS SLC, and KSC LC in Florida

2. Launch Outcomes

The map shows the number of launches in each area. A total of 10 launches were done on VAFB SLC site in Los Angeles, while 46 were done on Florida, of which 13, 26, and 7 were from KSC LC, CCAFS SLC, and CCAFS LC launch sites, respectively.

The green and red markers on the map show the successful and failed launch outcomes. KSC LC has the best success-fail ratio of 10:3. It is followed by CCAFS LC, VAFB SLC, and CCAFS SLC with success-fail ratios of 3:4, 4:6, 7:19, respectively.



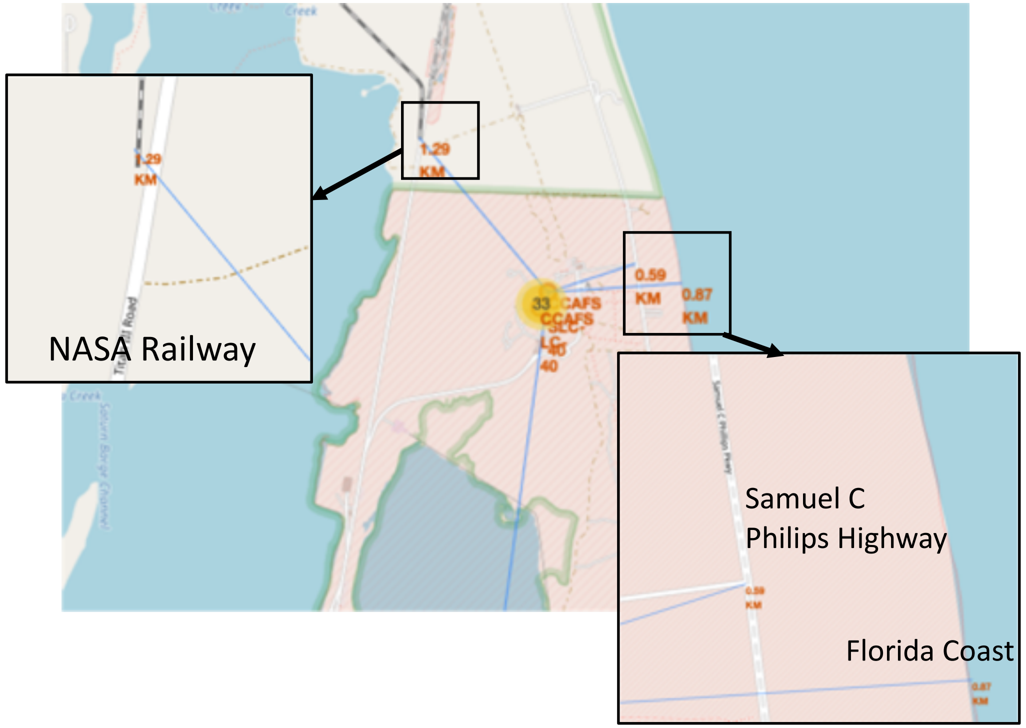
VAFB SLC

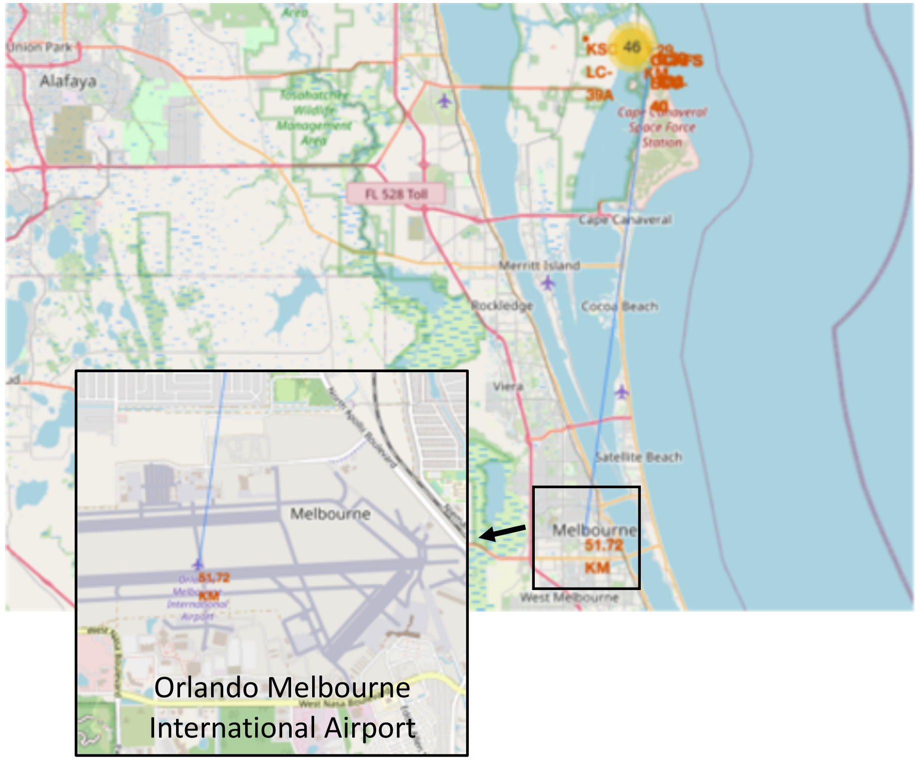
KSC LC

CCAFS SLC

CCAFS LC

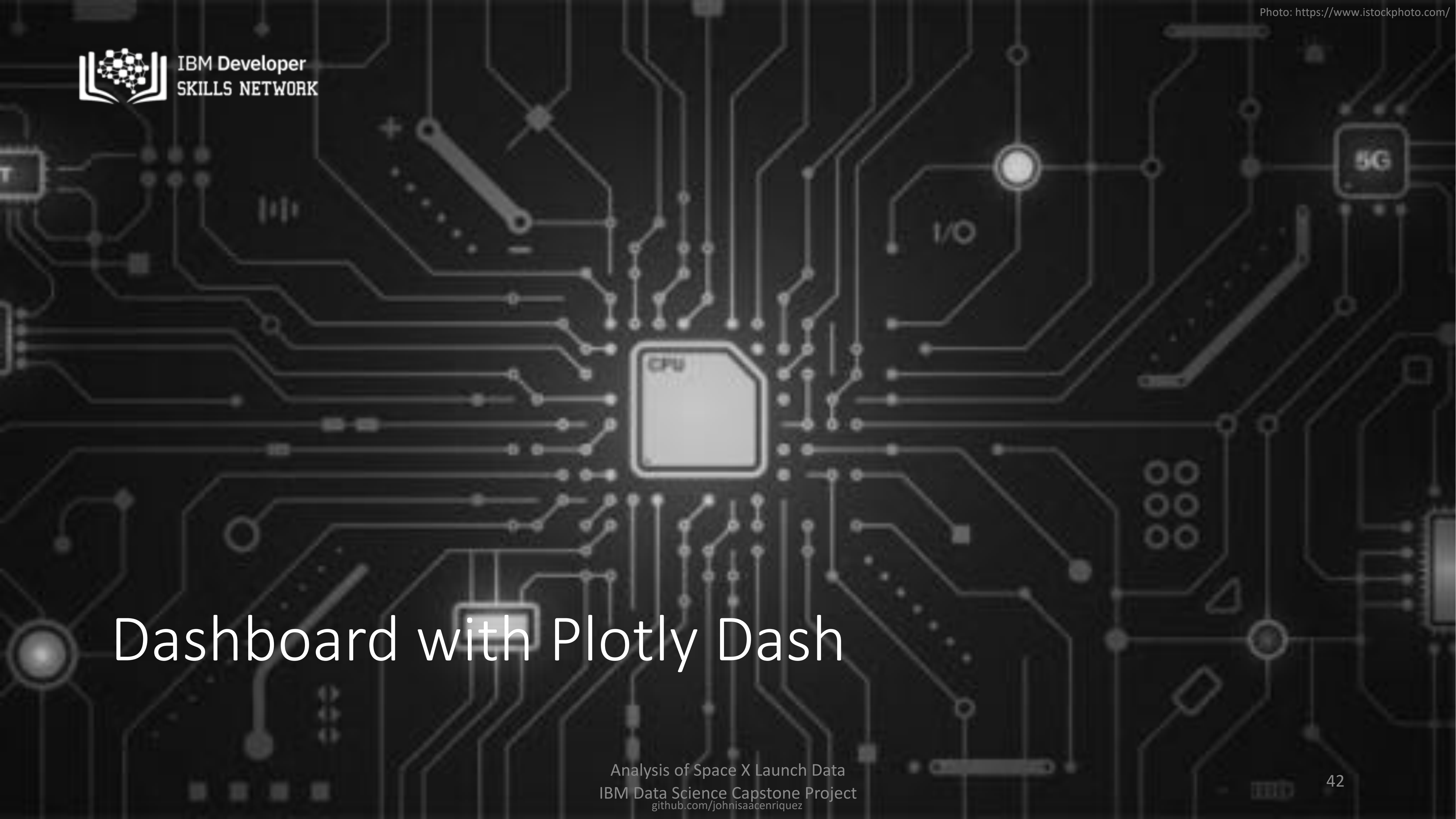
3. Proximity of CCAFS SLC Launch Site to Common Landmarks





The map shows the proximity of CCAFS SLC to some common landmarks. The figure on the left shows its proximity to the nearest railway (left inset) and highway (right inset). The distance to the coastline is also shown (right inset).

The figure on the right shows its proximity to the city of Melbourne. The inset shows that the proximity is measured to the Orlando Melbourne International Airport.

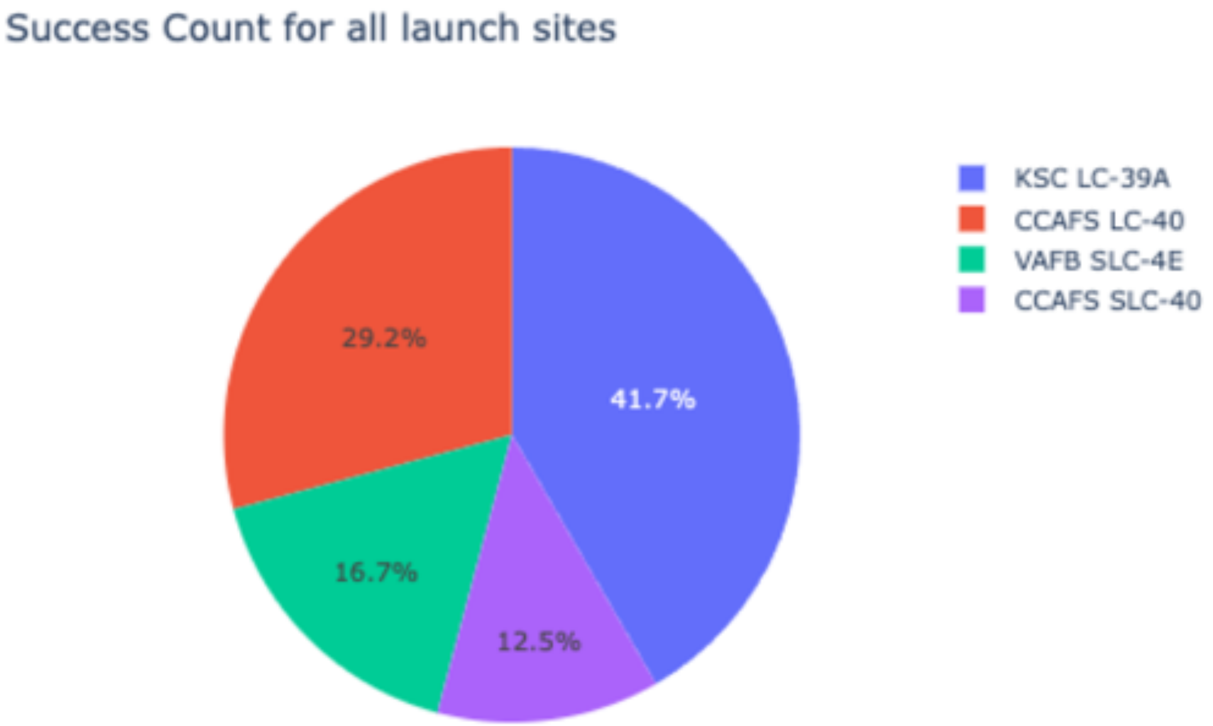


IBM Developer
SKILLS NETWORK

Dashboard with Plotly Dash

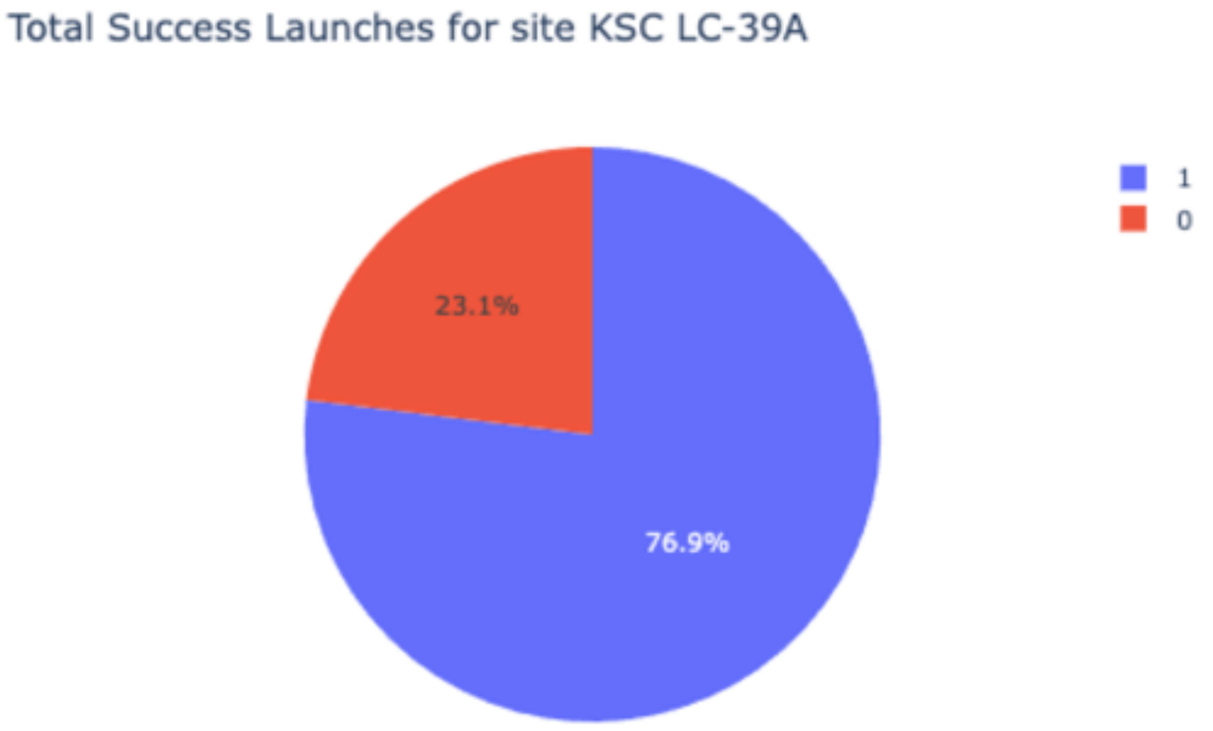
1. Launch Success Count for All Launch Sites

The pie graph shows the success count for all launch sites. KSC LC has the most success count, followed by CCAFS LC, VAFB SLC, and CCAFS SLC.



2. Total Success Launches for Site KSC LC-39A

The pie graph shows the success rate for the site KSC LC, where 0 and 1 corresponds to failure and success, respectively.

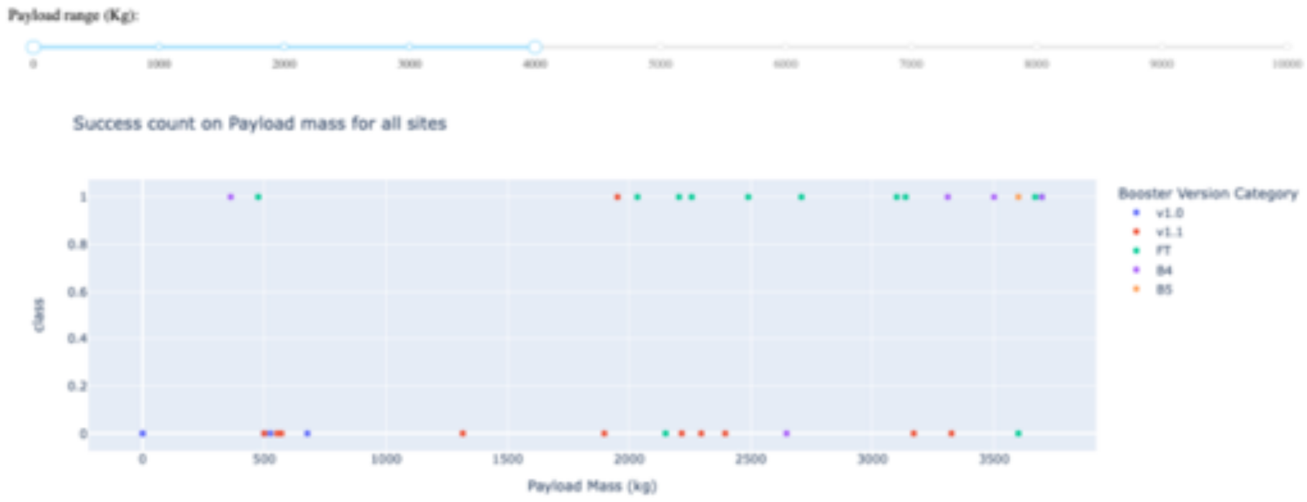


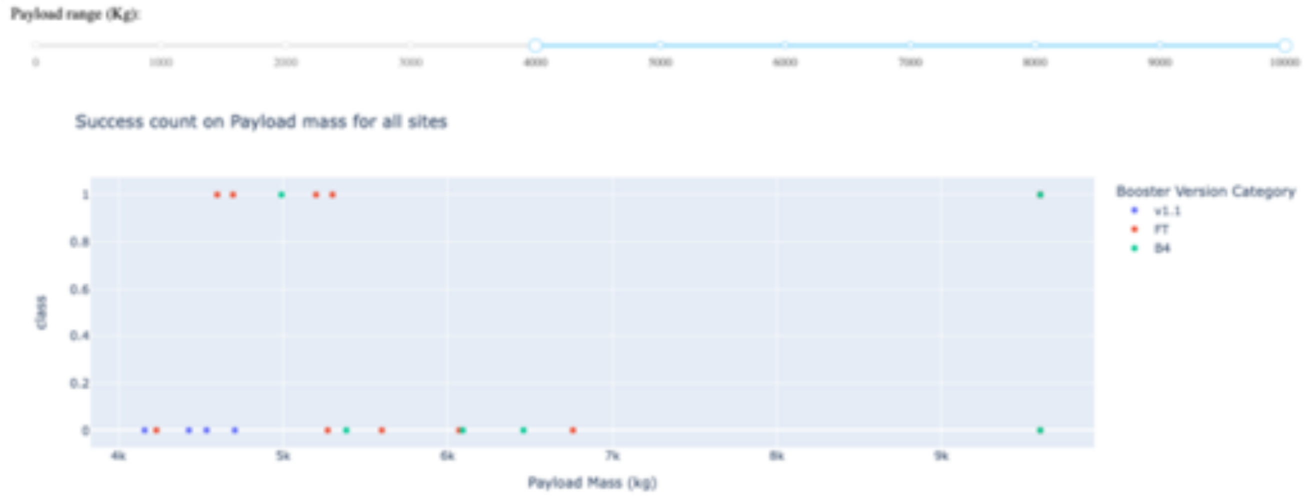
3. Success Count on Payload Mass for All Sites

The figure shows the success count for a given payload mass.

For payloads of less than 4000 kg, five booster versions were used, of which FT booster has the most number of successful outcomes.

For payloads more than 4000 kg, only three booster versions were used, and only FT and B4 boosters have recorded a successful outcome. For heavy payloads (> 9000 kg), only the B4 booster were used.



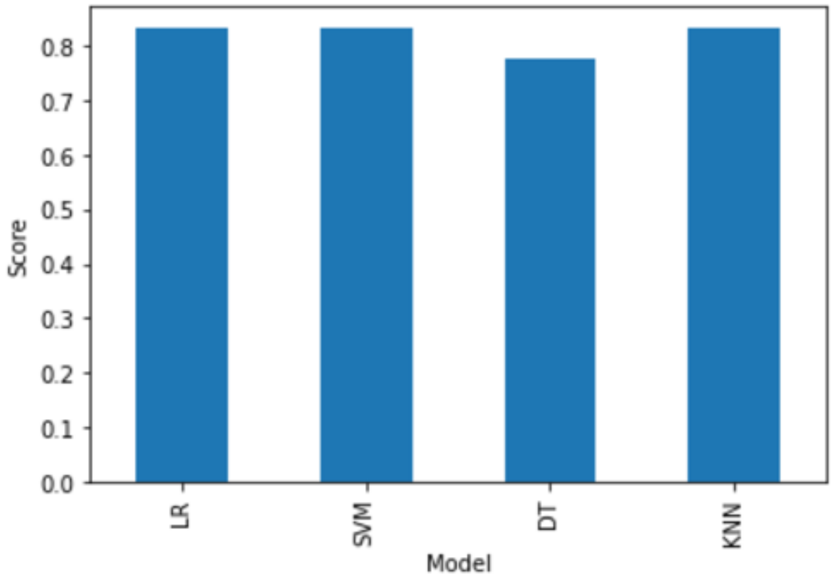


IBM Developer
SKILLS NETWORK

Predictive Analysis (Classification)

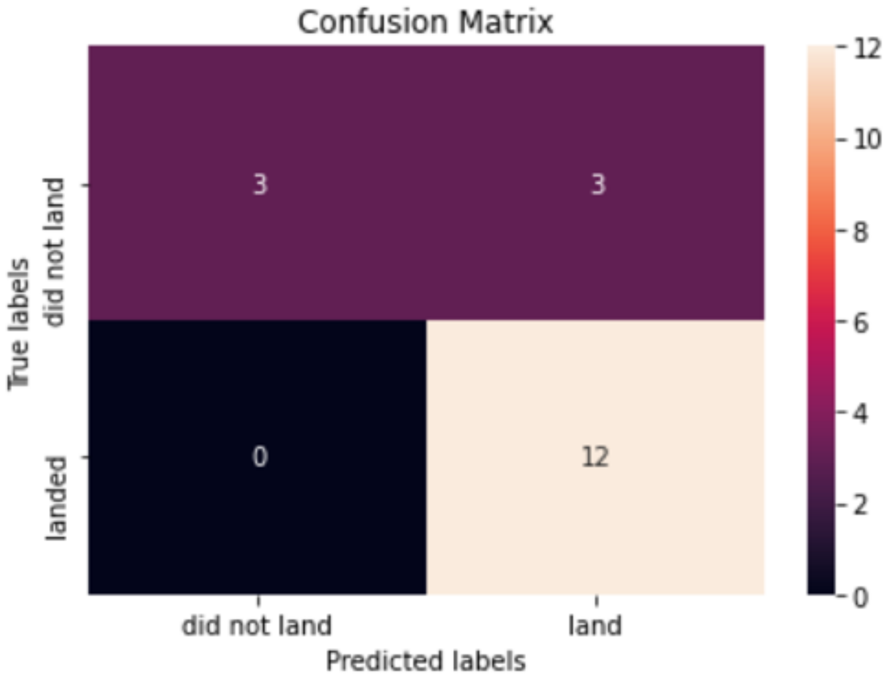
1. Classification Accuracy

The bar chart shows the accuracy of logistic regression (LR), support vector machine (SVM), decision tree (DT), and K nearest neighbor (KNN) classification models calculated using the score function. LR, SVM, KNN are equally accurate in predicting the test data set (0.833), while DT is slightly less accurate (0.778).



2. Confusion Matrix

The confusion matrix for the KNN classifier is shown. There are 3 true negatives (did not land) and 12 true positive (land) predictions. Moreover, there are 0 false negative and 3 false positive. The KNN was able to distinguish between different classes. However, it has problems in giving false positive predictions.



Section 5

Conclusions

Conclusions

- SpaceX launch data were collected and analyzed to generate insights and build machine learning classification model that will predict the success of landing the first stage booster.
- All launch sites are in close proximity to the coasts, and located at the southernmost parts of United states.
- Launch date, launch site, payload weight, and orbit type are the features that correlate to the success of launch.
- In general, the success rate has been increasing every year.
- Machine learning classification models were built to predict the launch outcome. An accuracy score of 0.833 were obtained using logistic regression, support vector machine, and K nearest neighbor.

Appendix

Web scraping helper functions

```python
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
        return colunm_name
```

Photo: https://www-cdn.eumetsat.int/

IBM Developer SKILLS NETWORK

Thank you for your attention.