IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

- Methodologies were used in the project:
  - ❖ Data collection via API
  - ❖ Data collection Webscraping
  - ❖ Data Wrangling
  - ❖ Data Analysis by using SQL
  - ❖ Visualization of Data analysis
  - ❖ Predication by using Machine Learning
- Summary of all results
  - ❖ Exploratory data analysis result
  - ❖ Predictive analysis result
  - ❖ Interactive dashboards for visualization of data analisys

Introduction

- Target of this project is investigation how successful were launched of Falcon 9 rocket by company Space X to determine if their technology can be used for launch of rockets and calculate cost of launch

- Problems we want to solve:
    - To decide which method will be opimal for estimation of launch
    - To determine the best parameters for successful landing
    - To determine cost of each launch

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

  There are some ways to collect data for project:

  - REST API technology
  - Web scrapping from Wiki and using BeautifulSoup package

- Perform data wrangling
  - Filter data for the Falcon 9 only and one-hot encoding for categorical features
- 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 predictive analysis used Line Regression, Decision Tree, Logistic Regression etc models

Data Collection

- Describe how data sets were collected.

  There are some ways to collect data for project:

  - REST API technology (api.spacexdata.com/v4/launches/past)
  - Web scrapping from Wiki and using BeautifulSoup for cleaning data

- You need to present your data collection process use key phrases and flowcharts

❖ In order to use REST API for collection data we used the Python libraries:

the json function to extract data, json_normalize() to transformed result and write it to the pandas dataframe.

To finalize of collection data was checked for missing values and updated where necessary.

❖ In order to use web scrapping we used BeautifulSoup library to parse data extractedas HTML table and convert to the pandas dataframe

Data Collection – SpaceX API

- Data collection using REST API:

❖ Get requet for rocket launch data

❖ Convert the json result using json_normalize

❖ Cleaning data (remove rows with multiply cores, payloads, convert data_utc to data_type format)

- Github URL: https://github.com/Inka611/Data-Science-Capstone-Project1/blob/3a7ff313ca3a342f3a0374954536c783f7801419/jupyter-labs-spacex-data-collection-api-final.ipynb

Get requests for rocket launch data using Rest API:

```
spacex_url=https://api.spacexdata.com/v4/launches/past
response = requests.get(spacex_url)
```

Use json_normalize meethod to convert the json result into a dataframe

```
r1 = response.json()
data = pd.json_normalize(r1)
```

Take a subset of our dataframe keeping only the features we want, remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a single rocket.

```
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]
```

extract the single value in the list and replace the feature.

```
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])
```

convert the date_utc to a datetime datatype and then extracting the date leaving the time

```
data['date'] = pd.to_datetime(data['date_utc']).dt.date
```

Data Collection - Scraping

- Data collection using Web scraping:

❖ By using HTTP GET method to request the Falcon9 Launch HTML page

❖ Create a BeautifulSoup object from the HTML response

❖ Create a data frame by parsing the launch HTML tables

❖ Export to csv

- Add the GitHub URL https://github.com/lnka611/Data-Science-Capstone-Project1/blob/fcbf3f19f9b0693a05c9e0b166fdf67f689fc7bd/jupyter-labs-webscraping_final.ipynb

```
static_url = https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922
# use requests.get() method with the provided static_url
# assign the response to a object
response=requests.get(static_url)
```

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

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
  # get table row
   for rows in table.find_all("tr"):
   if rows.th:
       if rows.th.string:
          flight_number=rows.th.string.strip()
          flag=flight_number.isdigit()
```

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- The Data Wranling stage is used to perform Exploratory Data Analysis (EDA), encoding categorical features and determine training and test data sets
- Data wrangling process includes following steps:

1. Calculate the number of launches on each site and the number and occurrence of each orbit:

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

2. Create a landing outcome label from Outcome column:

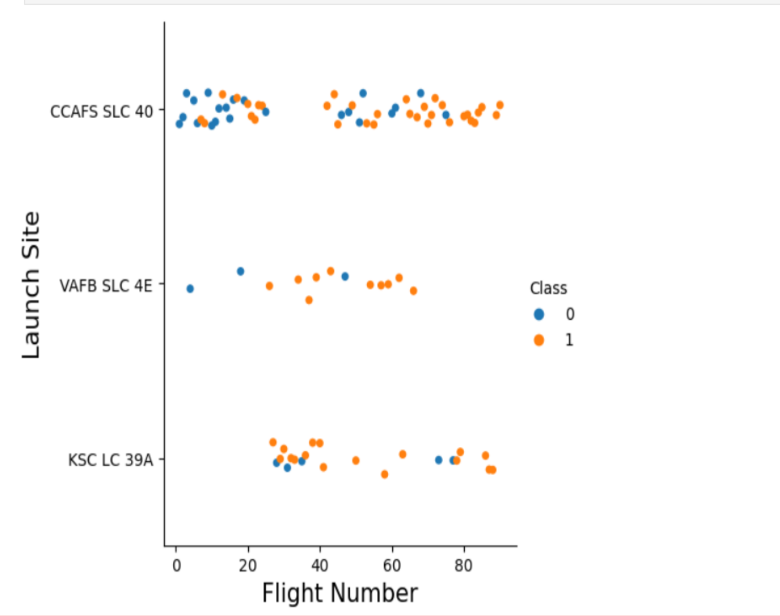
```
landing_class=[]
for oj in df['Outcome']:
    if oj in set(bad_outcomes):
        landing_class.append(0)
    else:
        landing_class.append(1)
df['Class']=landing_class
```

3. Export datra set to csv

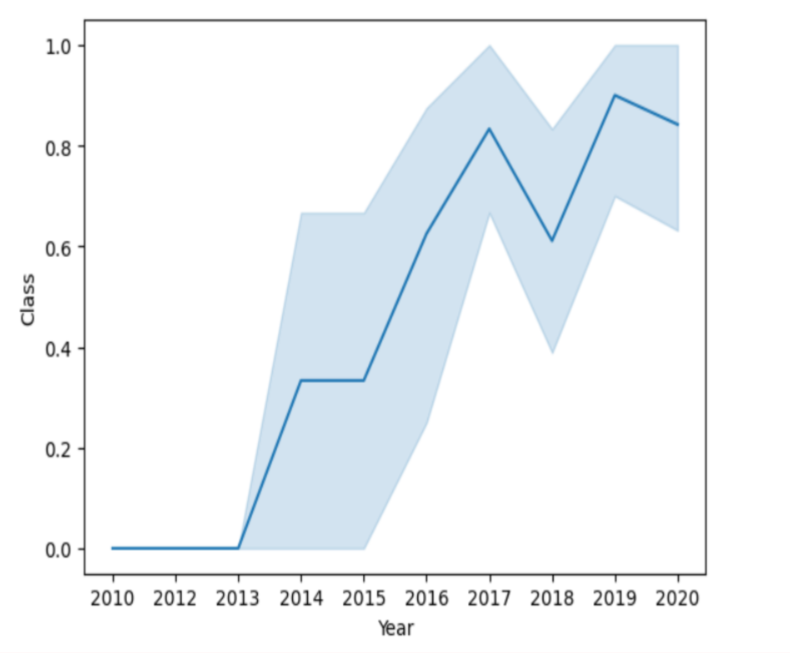
```
df.to_csv("dataset_part_2.csv",
index=False)
```

- GitHub URL :https://github.com/Inka611/Data-Science-Capstone-Project1/blob/1b2c9a31b9c5aa413d8659720db49c19a13c528a/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

EDA with Data Visualization



For visualization of the relationship between Flight Number and Launch Site we used the scatterplot



The line chart with x axis to be the extracted year and y axis to be the success rate demonstrates average success rate for years

- GitHub URL:https://github.com/Inka611/Data-Science-Capstone-Project1/blob/f7d9941b8b250324304ebba7643a03c99da63592/jupyter-labs-eda-dataviz.ipynb.jupyterlite_Final.ipynb

EDA with SQL

- SQL queries were used to performed multiply queries:

❖ **Unique launch sites in the space mission**

❖ **Total payload mass carried by boosters launched by NASA (CRS)**

❖ **Average payload mass carried by booster version F9 v1.1**

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

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

❖ **Total number of successful and failure mission outcomes**

❖ **List the names of the booster_versions which have carried the maximum payload mass**

❖ **Rank the count of landing outcomes**

- Git Hub link to notebook: https://github.com/Inka611/Data-Science-Capstone-Project1/blob/f7d9941b8b250324304ebba7643a03c99da63592/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- To build interactive Map with Folium were added objects:
  - ❖ Circles (to mark each Launch Site)
  - ❖ Markers (to mark each Launch site)
  - ❖ PolyLine (to demonstrate distance between Launching place and nearest city)

  Objects were marked by color according to status of launch (failure or successful)

- In order to determine color of the line the new column 'marker_color' calculated by formula spacex_df['class'].apply(lambda x: 'green' if x==1 else 'red') was added to the data frame

- Distance was calculated by formula:

```python
def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

Git Hub link:https://github.com/Inka611/Data-Science-Capstone-Project1/blob/08808110a447370eb893a451e2568937465f1752/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- The interactive dashboard was built by using Plotly Dash in this project
- The dashboards contained pie chart to show total launches for each Launch Site and Scatter plots to show relationship between Outcome and Payload Mass for booster version
- On the dashboard user will have possibility to choose Launch Site from drop-down list and max and min for Payload
- GitHub URL of completed Plotly Dash lab https://github.com/Inka611/Data-Science-Capstone-Project1/blob/9e9cde4cc0029fddf9d7346c1e52beab90fae593/spacex_dash_app.py

Predictive Analysis (Classification)

1. Bulding model:
- Load dataset by using Pandas and Numpy
- Split entire data to training and test dataset
- create a GridSearchCV different kind(Logical Regresion, Decision Tree, SVC, KNN) and fit it (mdl.fit(X_train, Y_train))

2. Evaluating Model:
-calculate accuracy for test data (mdl.score(X_test, Y_test))
-find best hyperparameters
-create confusion matrix

3.Improve and find the best model:
-create report by models
-choose model with the best accuracy

	Accuracy	Parameters	Test accuracy
Logistic Regression	0.866667	{'C': 1, 'penalty': 'l2', 'solver': 'lbfgs'}	0.739130
SVM	0.883333	{'C': 1.0, 'gamma': 1.0, 'kernel': 'sigmoid'}	0.739130
Decision Tree	0.940476	{'criterion': 'entropy', 'max_depth': 12, 'max...	0.739130
KNN	0.883333	{'algorithm': 'auto', 'n_neighbors': 5, 'p': 1}	0.695652

- GitHub URL https://github.com/Inka611/Data-Science-Capstone-Project1/blob/2b5c493ea75842e4d56ed3d7d5f77be7f9892545/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

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

Section 2

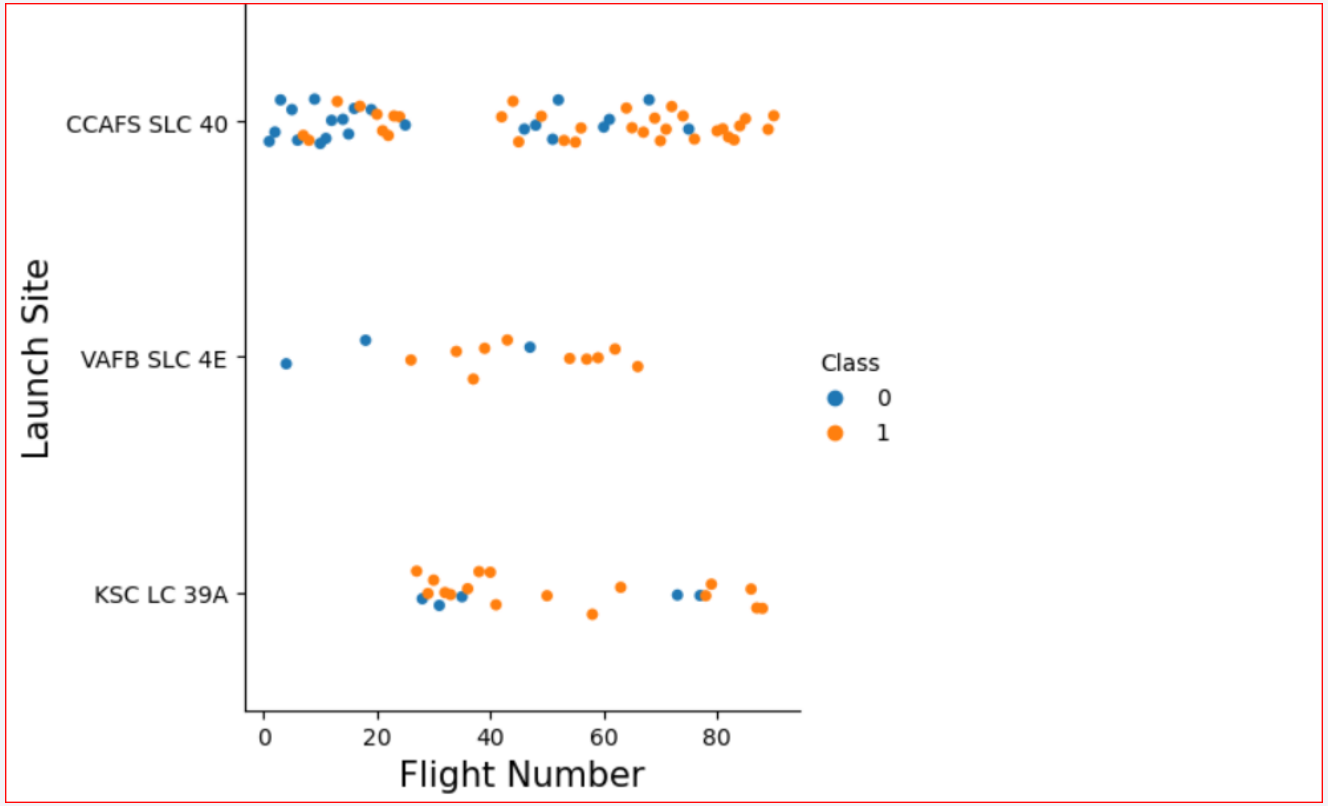
Insights drawn from EDA

Flight Number vs. Launch Site

- The scatter plot of Flight Number vs. Launch Site shows the larger the flight amount at a launch site, the greater the success rate at a launch site

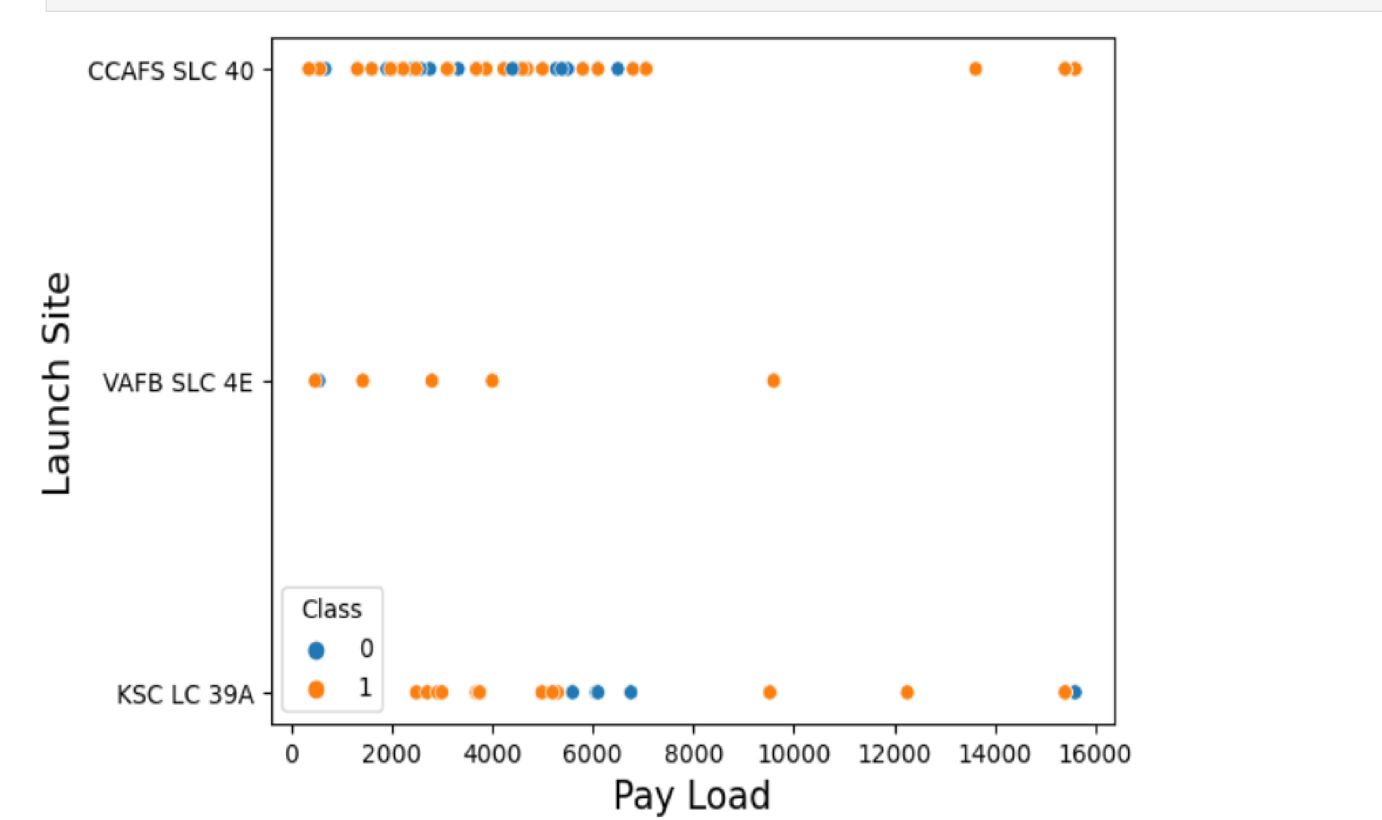
- Successful launch is launch with Class = 1 (orange circle),

- Failure launch is launch with Class = 0 (blue circle)



Payload vs. Launch Site

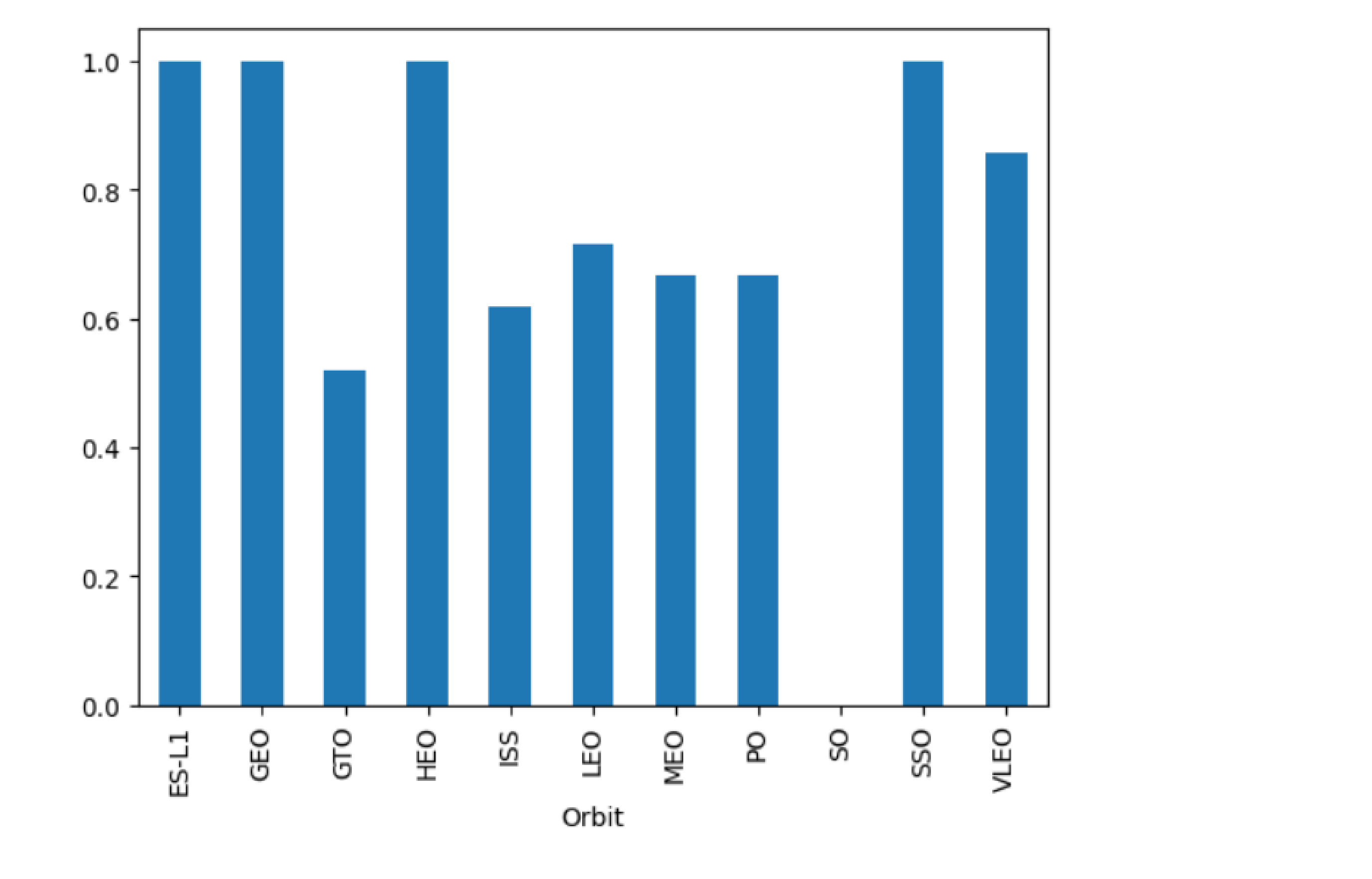
- This scatter plot shows dependence between possibility of successful launch and payload mass for different launch site
- It shows that possibility of successful launch increases for payload mass greater than 8000



Success Rate vs. Orbit Type

- The bar chart for the success rate of each orbit type

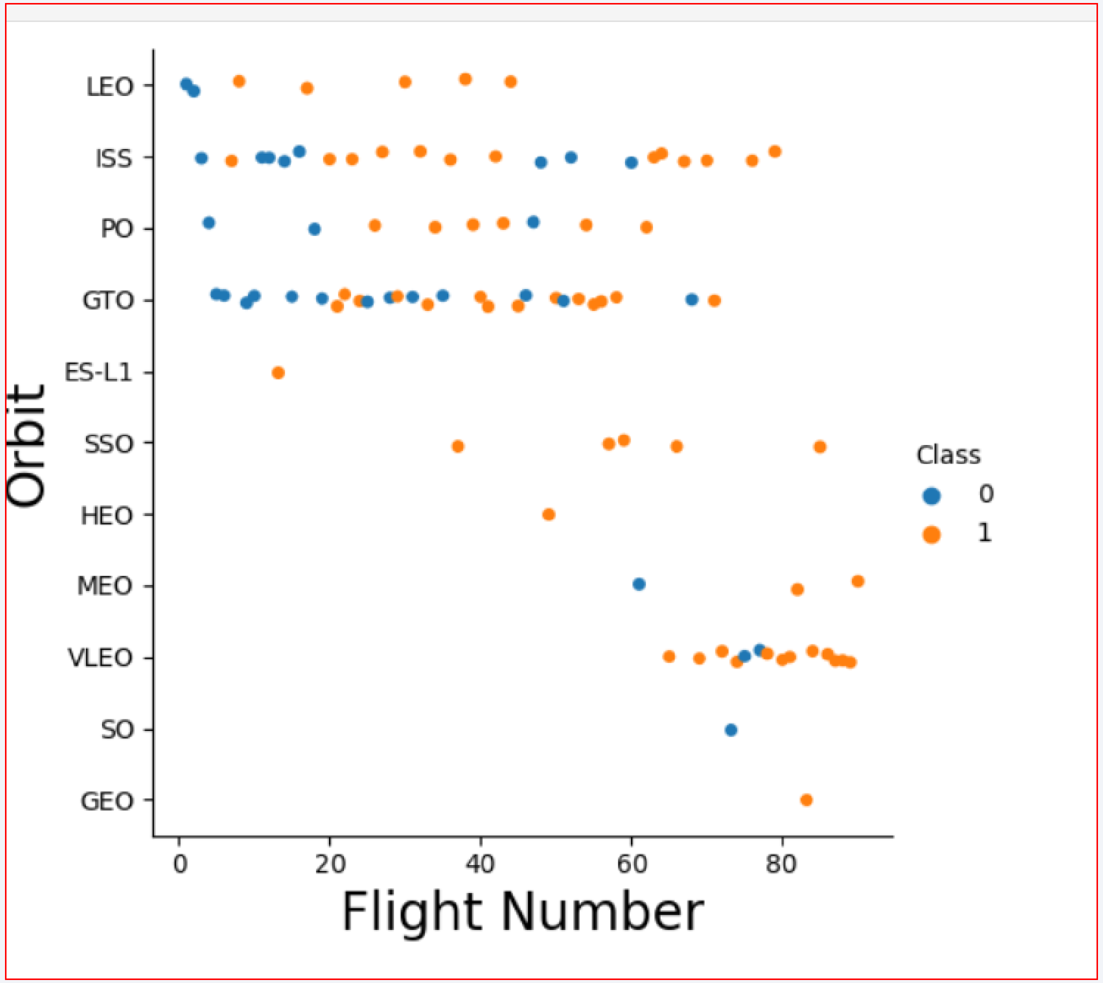
- It shows that the most successful launches was for orbit type ‘ES-L1’, ‘GEO’, ‘HEO’.



Flight Number vs. Orbit Type

- This screenshot shows the scatter point of Flight number vs. Orbit type

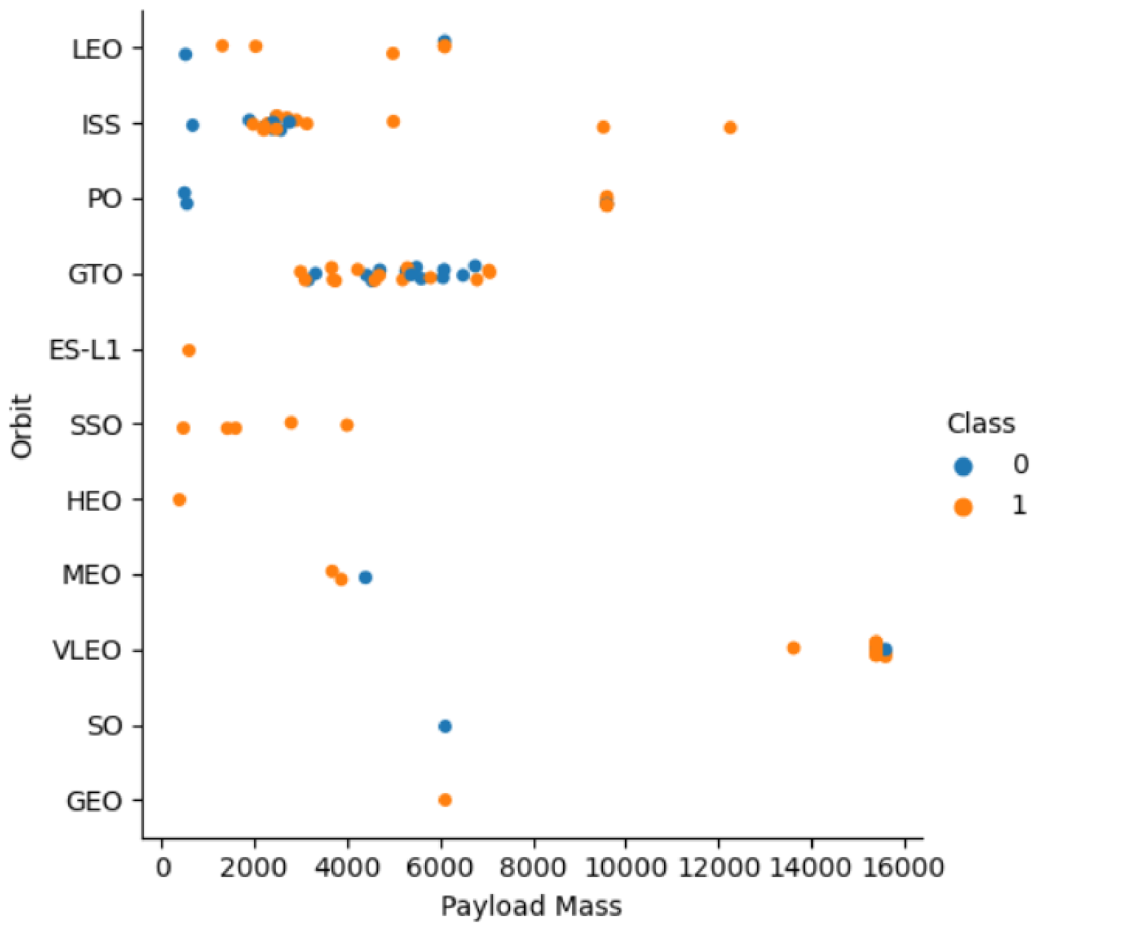
- The scatter plot shows dependence frequency of successful launches on Flight number and Orbit type



Payload vs. Orbit Type

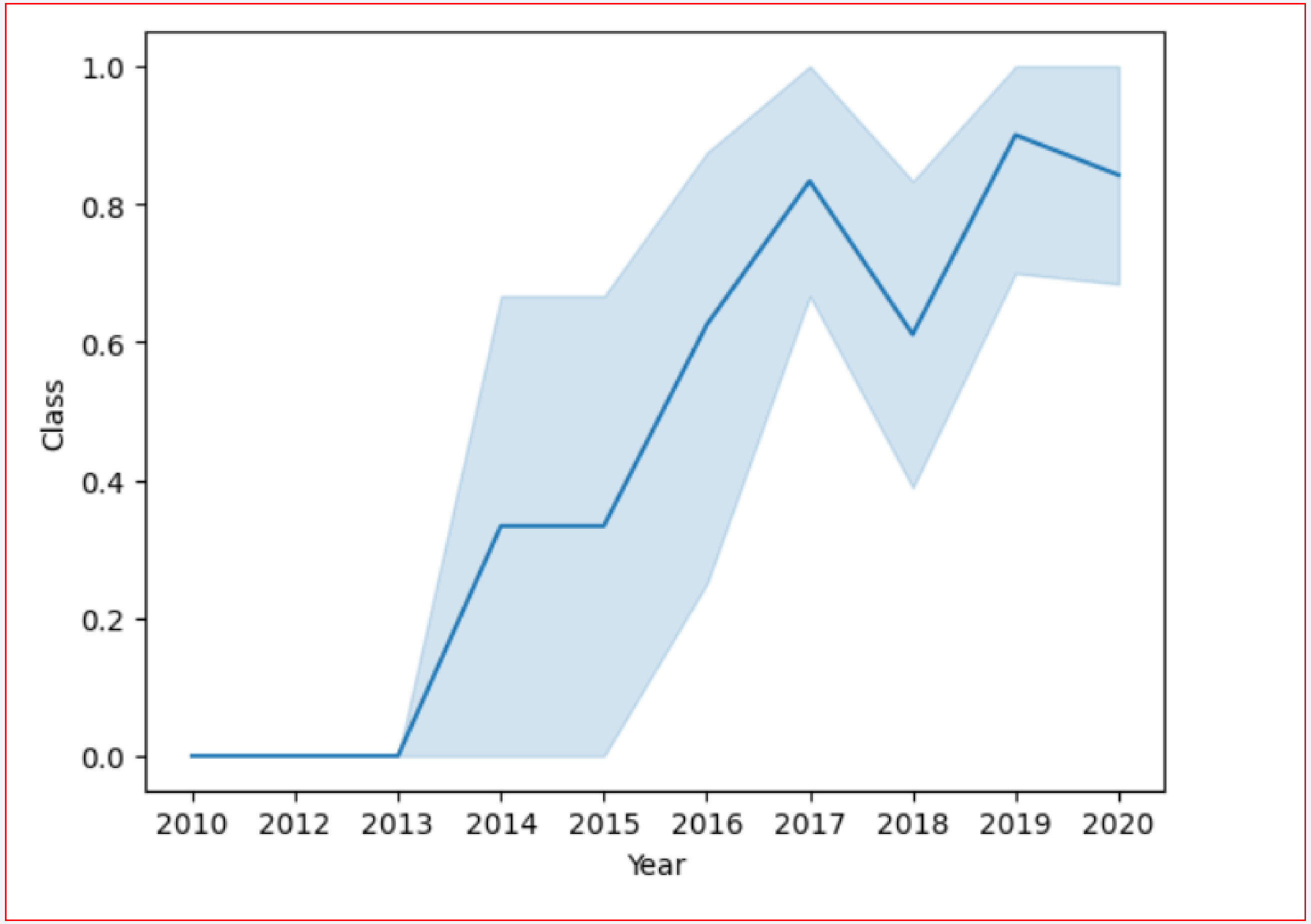
- This screenshot shows the scatter point of payload vs. orbit type

- The scatter plot shows dependence frequency of successful launches on payload and orbit type



Launch Success Yearly Trend

- The line chart demonstrates average success rate by years
- It shows that success rate was growing since 2013 year through 2017, but after 2017 year success rate decreased sharply. Restoring of growth of success rate was began in 2018 year



All Launch Site Names

- Find the names of the unique launch sites

```
result = cur.execute('SELECT DISTINCT Launch_Site from SPACEXTBL').fetchall()
result
```

```
[('CCAFS LC-40',), ('VAFB SLC-4E',), ('KSC LC-39A',), ('CCAFS SLC-40',)]
```

- In order to extract data was used Python package sqlite3.
- To get list of launch sites names was used SELECT DISTINCT statement. It returns list of tuples. The first element of each tuple is launch site name

Launch Site Names Begin with 'CCA'

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

```
result1 = cur.execute('SELECT * from SPACEXTBL WHERE "Launch_Site" LIKE "CCA%" LIMIT 5').fetchall()
result1
```

- To get the first 5 records was used SELECT statement with condition ‘LIKE CCA%’ and ‘LIMIT 5’. It returns list of tuples as on screenshot

```
('2010-04-06', '18:45:00', 'F9 v1.0  B0003', 'CCAFS LC-40', 'Dragon Spacecraft Qualification Unit', 0, 'LEO', 'SpaceX', 'Success', 'Failure (parachute)')
('2010-08-12', '15:43:00', 'F9 v1.0  B0004', 'CCAFS LC-40', 'Dragon demo flight C1, two CubeSats, barrel of Brouere cheese', 0, 'LEO (ISS)', 'NASA (COTS) NRO', 'Succe
ss', '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')
('2012-08-10', '00:35:00', 'F9 v1.0  B0006', 'CCAFS LC-40', 'SpaceX CRS-1', 500, 'LEO (ISS)', 'NASA (CRS)', 'Success', 'No attempt')
('2013-01-03', '15:10:00', 'F9 v1.0  B0007', 'CCAFS LC-40', 'SpaceX CRS-2', 677, 'LEO (ISS)', 'NASA (CRS)', 'Success', 'No attempt')
```

- Each element of tuple contain all fields of the SPACEXTBL table

Total Payload Mass

- To calculate the total payload carried by boosters from NASA was used sql query with condition for the Customer field

```
result2 = cur.execute('SELECT SUM(PAYLOAD_MASS__KG_), Customer from SPACEXTBL WHERE "Customer"="NASA (CRS)"\
                    GROUP BY CUSTOMER').fetchall()
result2
```

- The query returns the tuple where the first element is total payload and the second is name of Customer

```
[(45596, 'NASA (CRS)')]
```

Average Payload Mass by F9 v1.1

- To calculate the average payload mass carried by booster version F9 v1.1 was used following query

```
result3 = cur.execute('SELECT AVG(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE "Booster_Version"="F9 v1.1"').fetchall()
result3
```

- The query returns tuple where the first element is average payload mass

```
[(2928.4,)]
```

First Successful Ground Landing Date

- To find the dates of the first successful landing outcome on ground pad was used the following query with using Min function for Date field

```
result4 = cur.execute('SELECT MIN(Date) from SPACEXTBL WHERE "Landing_Outcome"="Success (ground pad)"').fetchall()
result4
```

- The query returns tuple where the first element is the first date

```
[('2015-12-22',)]
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- To get list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 was used the following query

```
result5 = cur.execute('SELECT Booster_Version from SPACEXTBL WHERE ("Landing_Outcome"="Success (drone ship)") \
AND (PAYLOAD_MASS__KG_ between 4000 and 6000)').fetchall()
for qi in result5:
    print(qi)
```

- The query returns list of tuples where the first element is Booster_Version

```
('F9 FT B1022',)
('F9 FT B1026',)
('F9 FT  B1021.2',)
('F9 FT  B1031.2',)
```

Total Number of Successful and Failure Mission Outcomes

- To calculate the total number of successful and failure mission outcomes was used the query with 'UNION' statement

```
result_outcome = cur.execute('SELECT COUNT(*), "Success" from SPACEXTBL WHERE ("Landing_Outcome" LIKE "Success%")\
                             Union\
                             SELECT COUNT(*), "Failure" from SPACEXTBL WHERE ("Landing_Outcome" LIKE "FAILURE%")').fetchall()
result_outcome
```

- The query returns list of tuples where the first element is number of outcomes, the second type of outcome: "Failure" or "Success"

```
(10, 'Failure')
(61, 'Success')
```

Boosters Carried Maximum Payload

- To get list the names of the booster which have carried the maximum payload mass is used query with subquery, where subquery returns value of maximum payload

```
result_max_boos = cur.execute('SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_=\
                    (SELECT MAX(PAYLOAD_MASS__KG_) from SPACEXTBL)').fetchall()
result_max_boos
```

The Query returns list of tuples where the first element is booster version, the second is payload mass

```
[('F9 B5 B1048.4', 15600),
 ('F9 B5 B1049.4', 15600),
 ('F9 B5 B1051.3', 15600),
 ('F9 B5 B1056.4', 15600),
 ('F9 B5 B1048.5', 15600),
 ('F9 B5 B1051.4', 15600),
 ('F9 B5 B1049.5', 15600),
 ('F9 B5 B1060.2 ', 15600),
 ('F9 B5 B1058.3 ', 15600),
 ('F9 B5 B1051.6', 15600),
 ('F9 B5 B1060.3', 15600),
 ('F9 B5 B1049.7 ', 15600)]
```

2015 Launch Records

- To get list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 is used following query

```
resmon = cur.execute('SELECT SUBSTR(Date, 6,2) as Month, Landing_Outcome, BOOSTER_VERSION, \
          LAUNCH_SITE FROM SPACEXTBL WHERE Date LIKE "2015%"').fetchall()

resmon
```

- The query returns list of tuple where each tuple contains month, status of outcome, booster version and launch site

```
[('10', 'Failure (drone ship)', 'F9 v1.1 B1012', 'CCAFS LC-40'),
 ('11', 'Controlled (ocean)', 'F9 v1.1 B1013', 'CCAFS LC-40'),
 ('02', 'No attempt', 'F9 v1.1 B1014', 'CCAFS LC-40'),
 ('04', 'Failure (drone ship)', 'F9 v1.1 B1015', 'CCAFS LC-40'),
 ('04', 'No attempt', 'F9 v1.1 B1016', 'CCAFS LC-40'),
 ('06', 'Precluded (drone ship)', 'F9 v1.1 B1018', 'CCAFS LC-40'),
 ('12', 'Success (ground pad)', 'F9 FT B1019', 'CCAFS LC-40')]
```

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

```
resrank = cur.execute('SELECT COUNT(*), Landing_Outcome FROM SPACEXTBL  WHERE DATE between "2010=06-04" and "2017-03-20" \
                      GROUP BY Landing_Outcome ORDER BY COUNT(*) DESC')
resrank.fetchall()
```

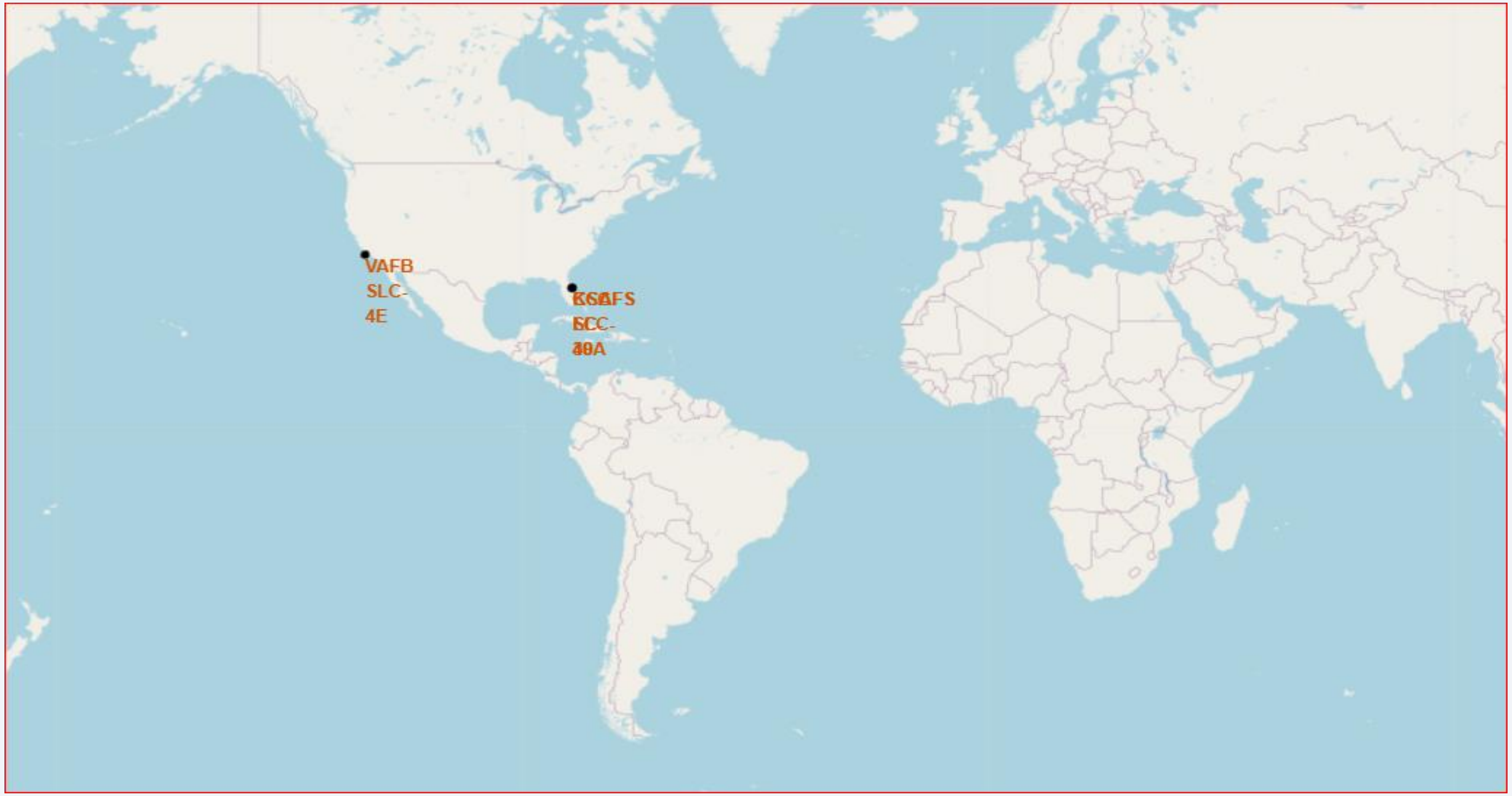
- The query returns the list of tuples ordered by descendence

```
[(10, 'No attempt'),
 (5, 'Success (ground pad)'),
 (5, 'Success (drone ship)'),
 (5, 'Failure (drone ship)'),
 (3, 'Controlled (ocean)'),
 (2, 'Uncontrolled (ocean)'),
 (1, 'Precluded (drone ship)')]
```

Section 3

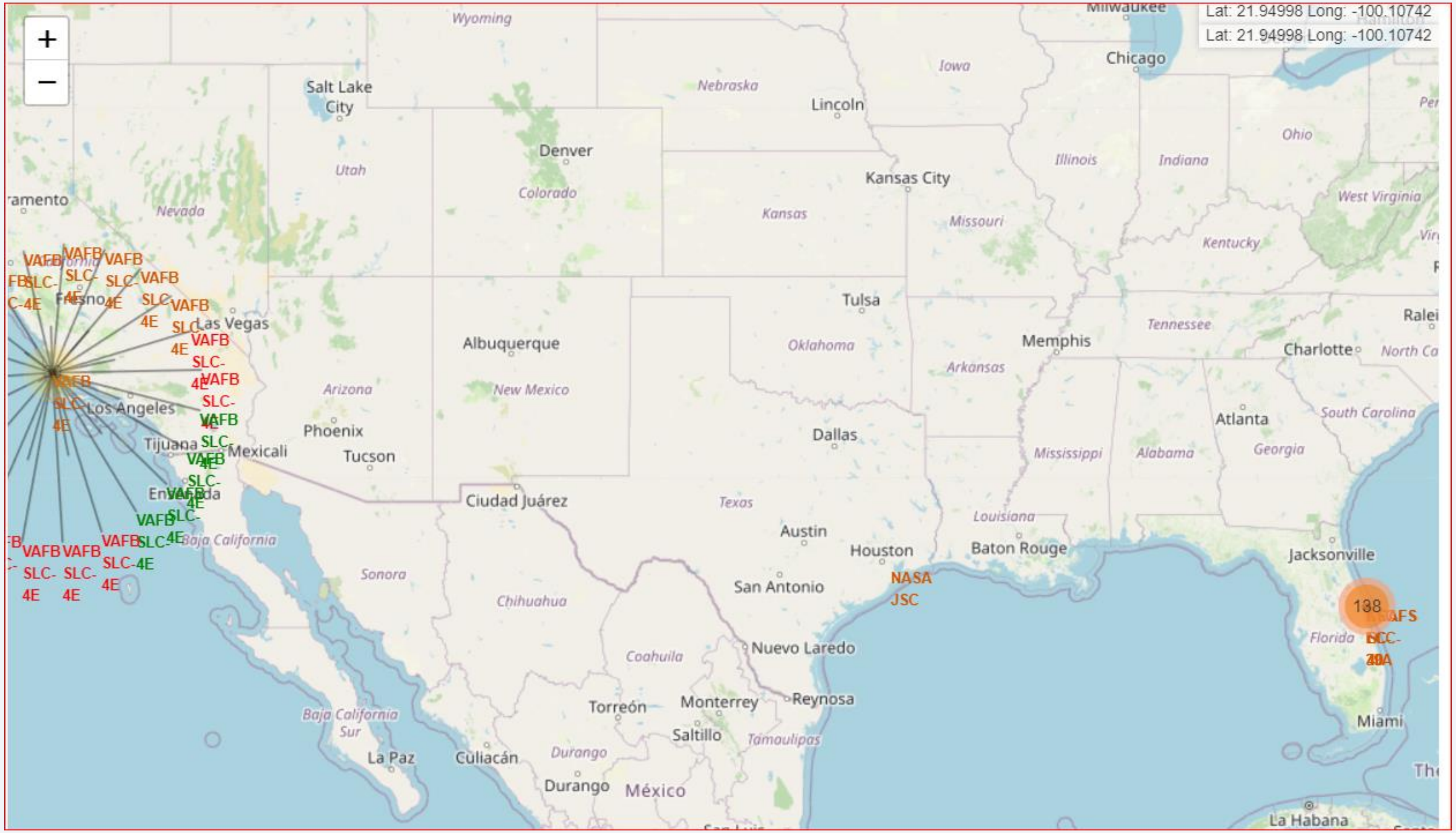
Launch Sites Proximities Analysis

All Launch Sites on a Map



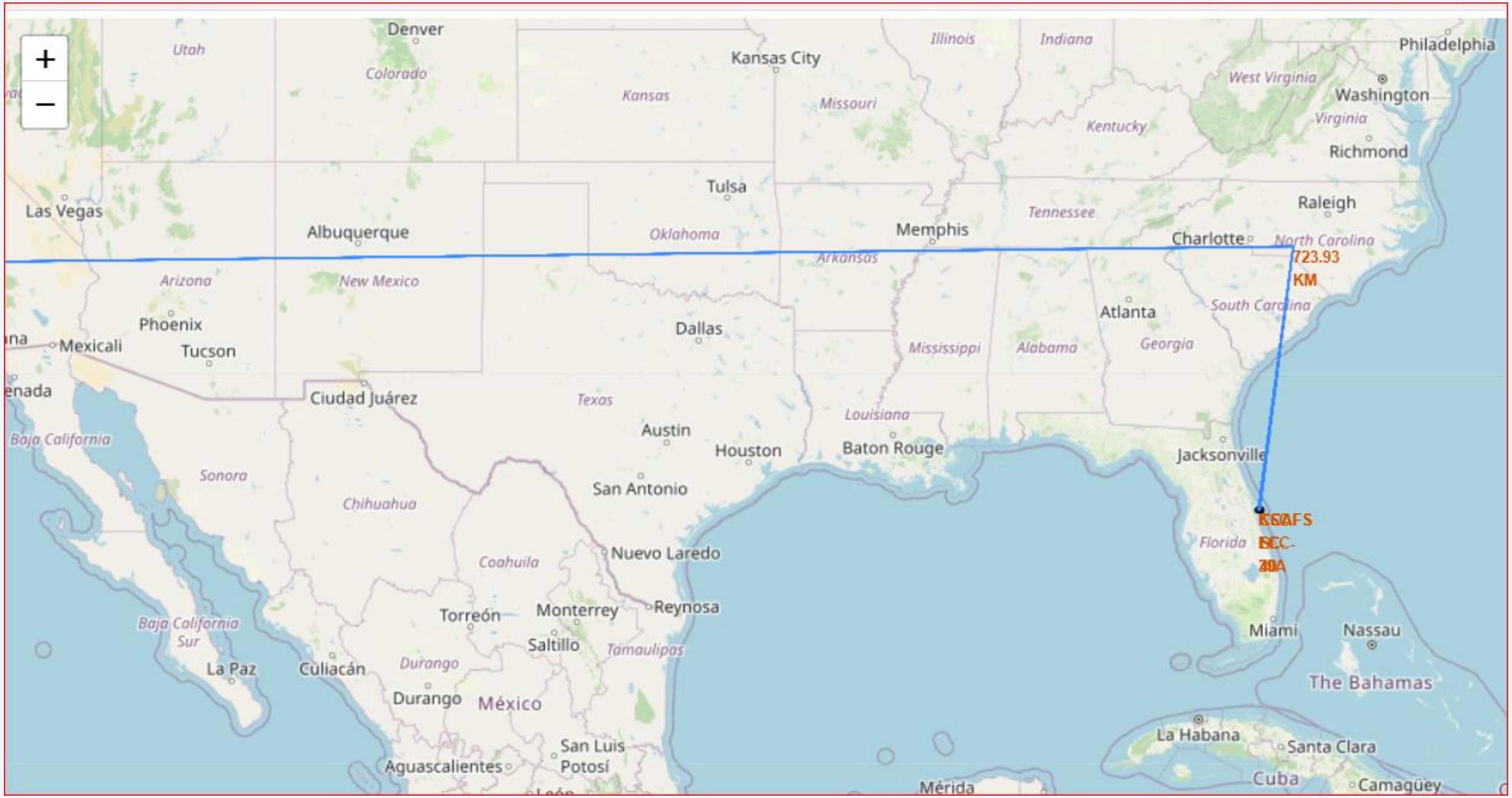
The screenshot contains the map with markerss for each Launch Site

Mark the success/failed launches



Successful launch on this screenshot is marked by green, failed - red

Distances between a launch site to its proximities

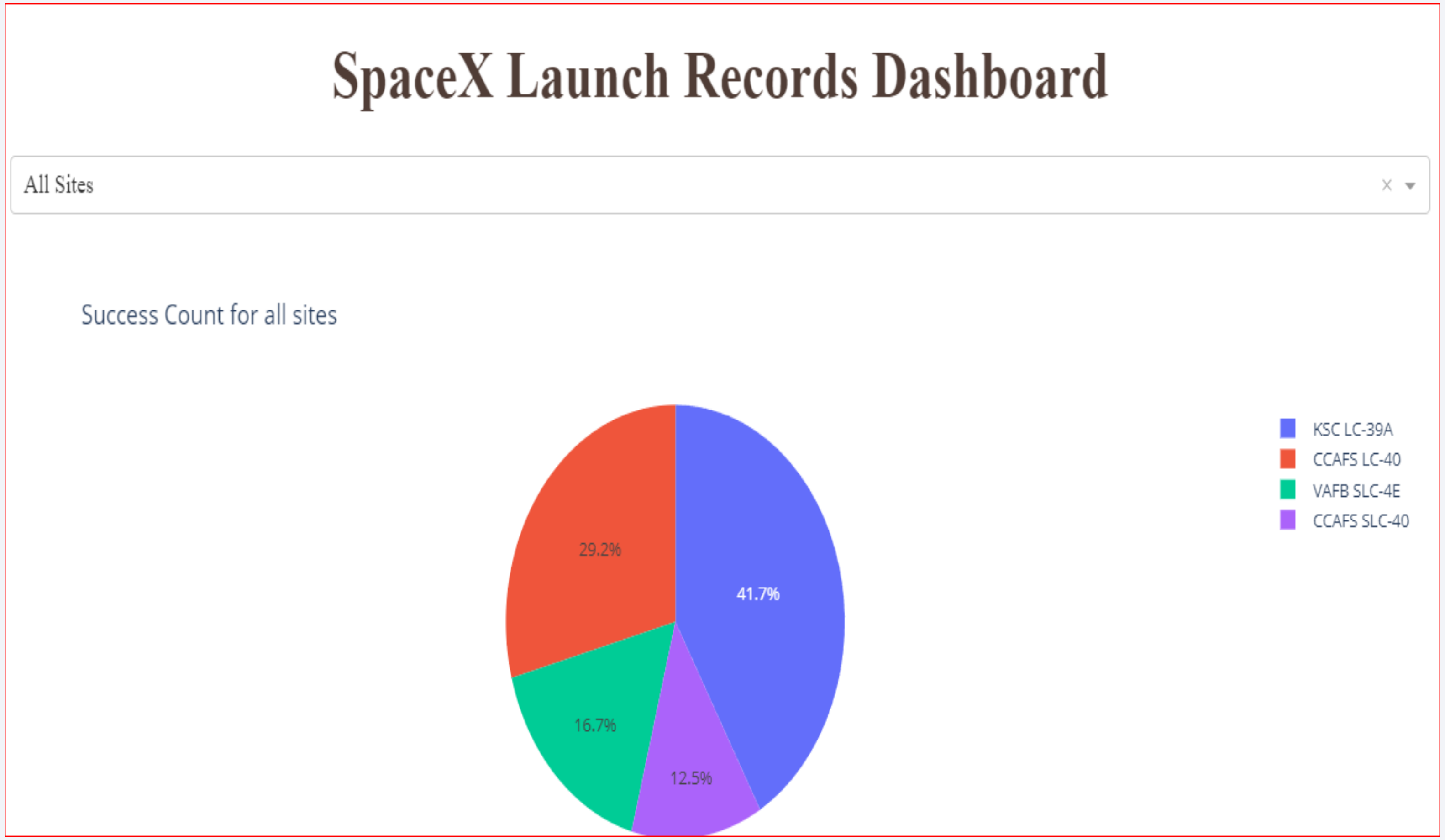


The poly line on this map shows distance to closest coasline

Section 4

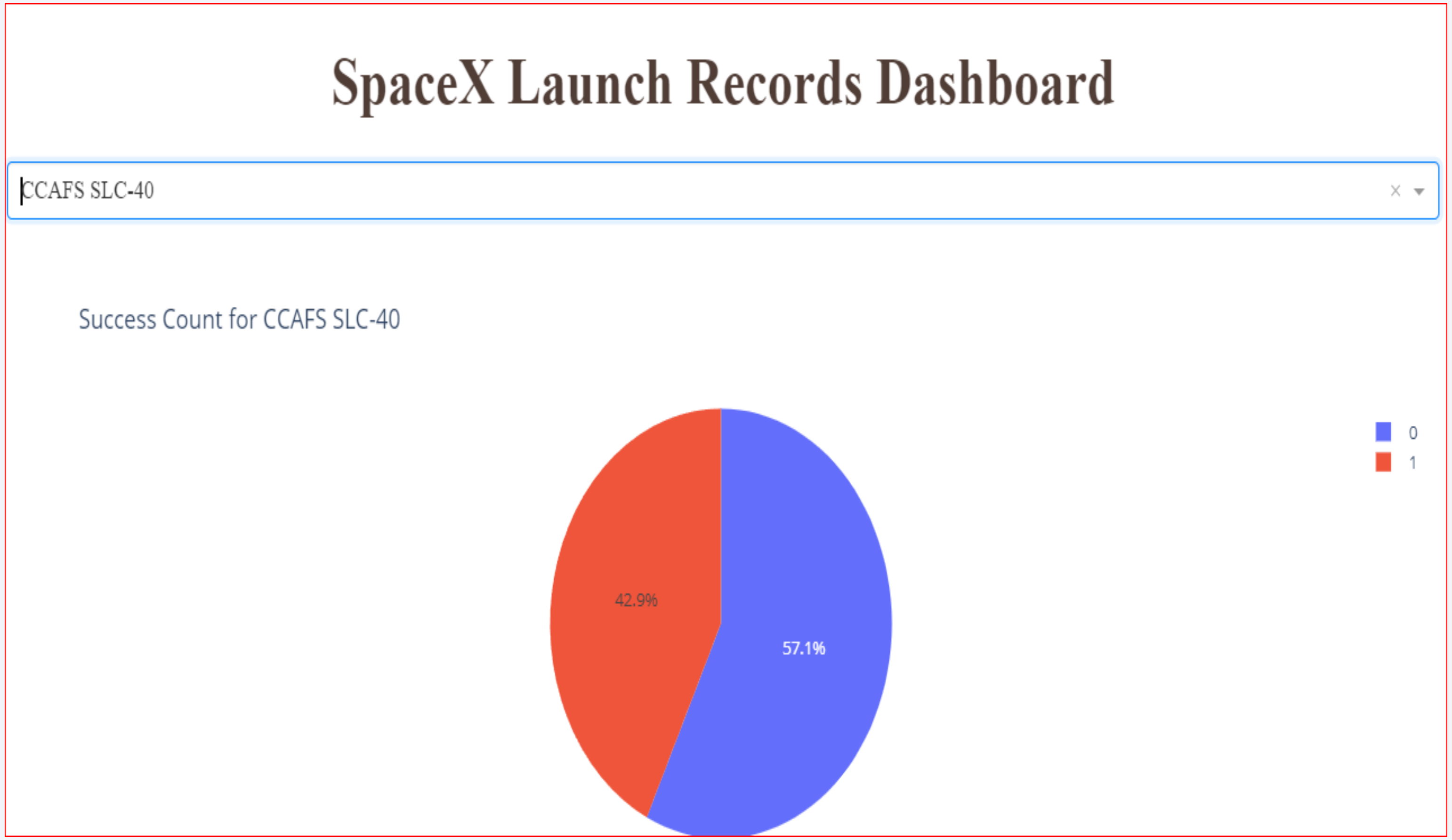
Build a Dashboard with Plotly Dash

Launch Success Count Pie Chart for All Sites



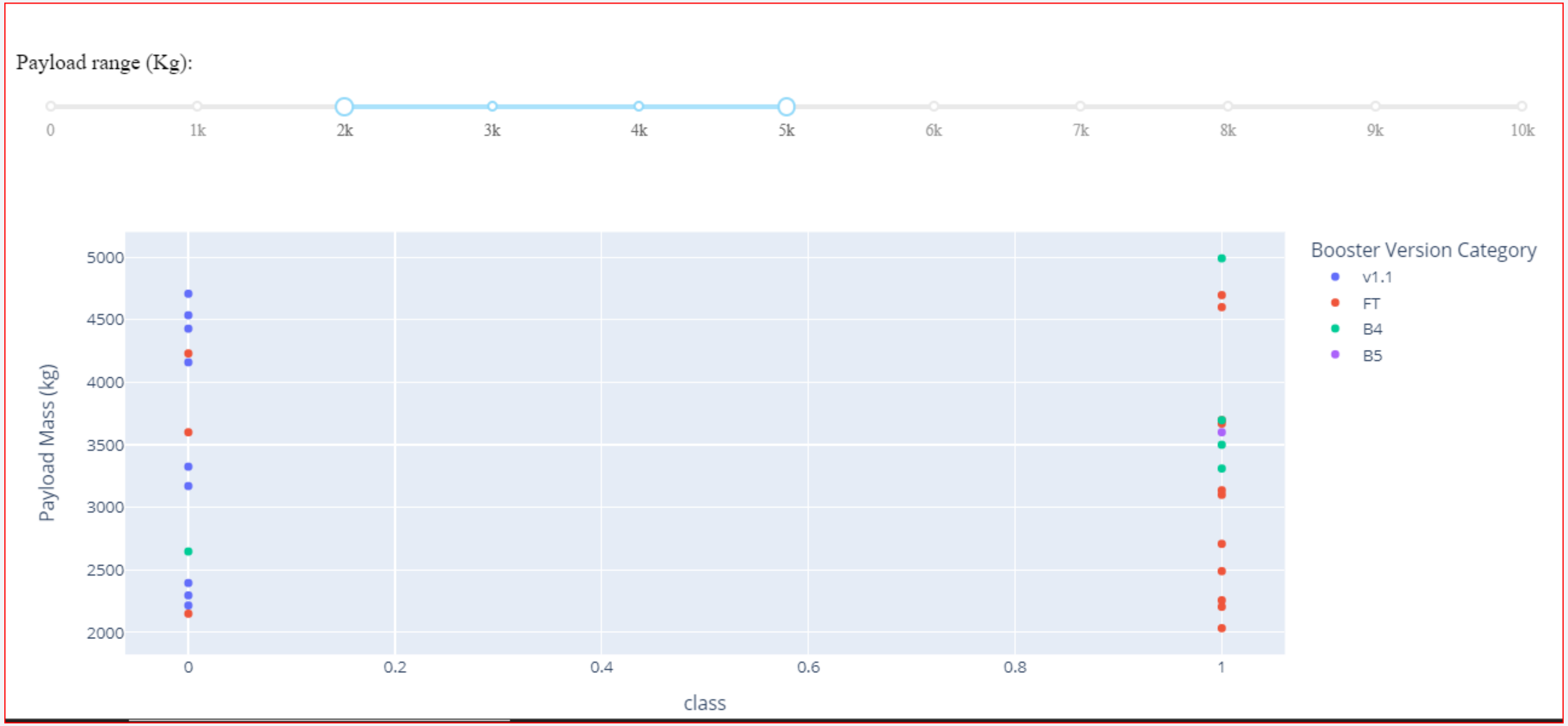
- This chart demonstrates success launches for all sites. It shows that KSC LC-39A had the most successful launches

Pie Chart for Highest Launch Success



- This chart demonstrates proportion between successful and failed launches for most successful site (CCAFS SLC-40)

Payload vs. Launch Outcome for all sites



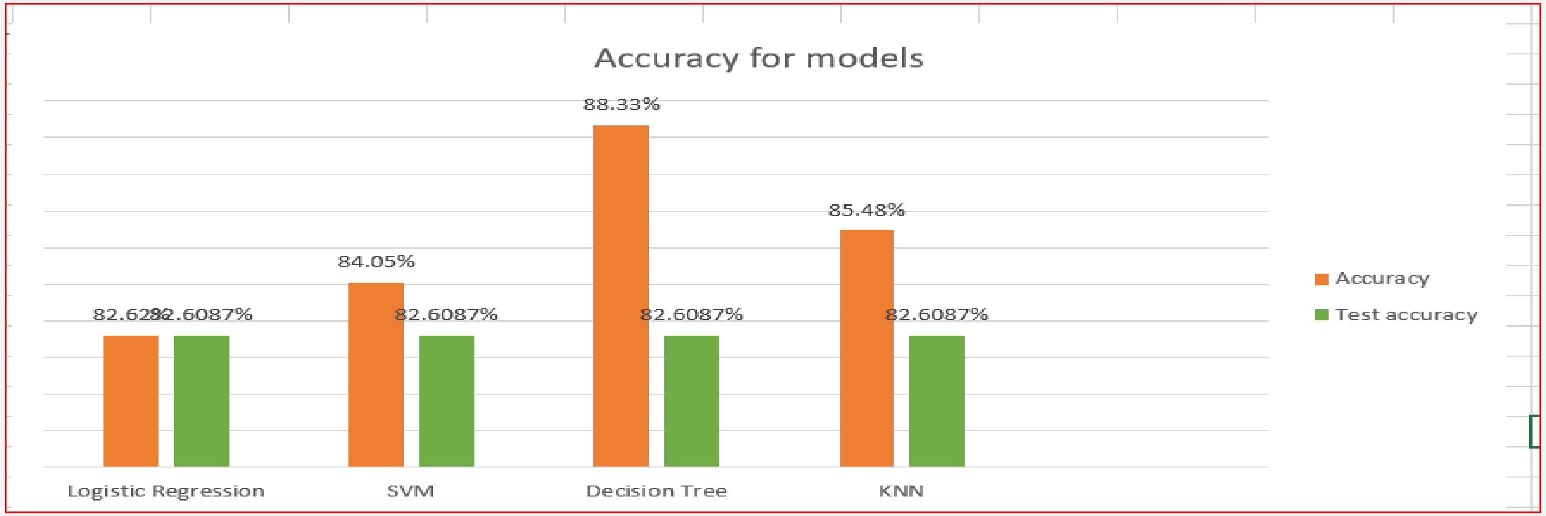
This scatterplot demonstrates all successful launches with payload between 2000 and 5000 by booster version categories

Section 5

Predictive Analysis (Classification)

Classification Accuracy

	Accuracy	Parameters	Test accuracy	Models
Logistic Regression	0.826190	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}	0.826087	Logistic Regression
SVM	0.840476	{'C': 1.0, 'gamma': 0.03162277660168379, 'kern...	0.826087	SVM
Decision Tree	0.883333	{'criterion': 'entropy', 'max_depth': 6, 'max_...	0.826087	Decision Tree
KNN	0.854762	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}	0.826087	KNN

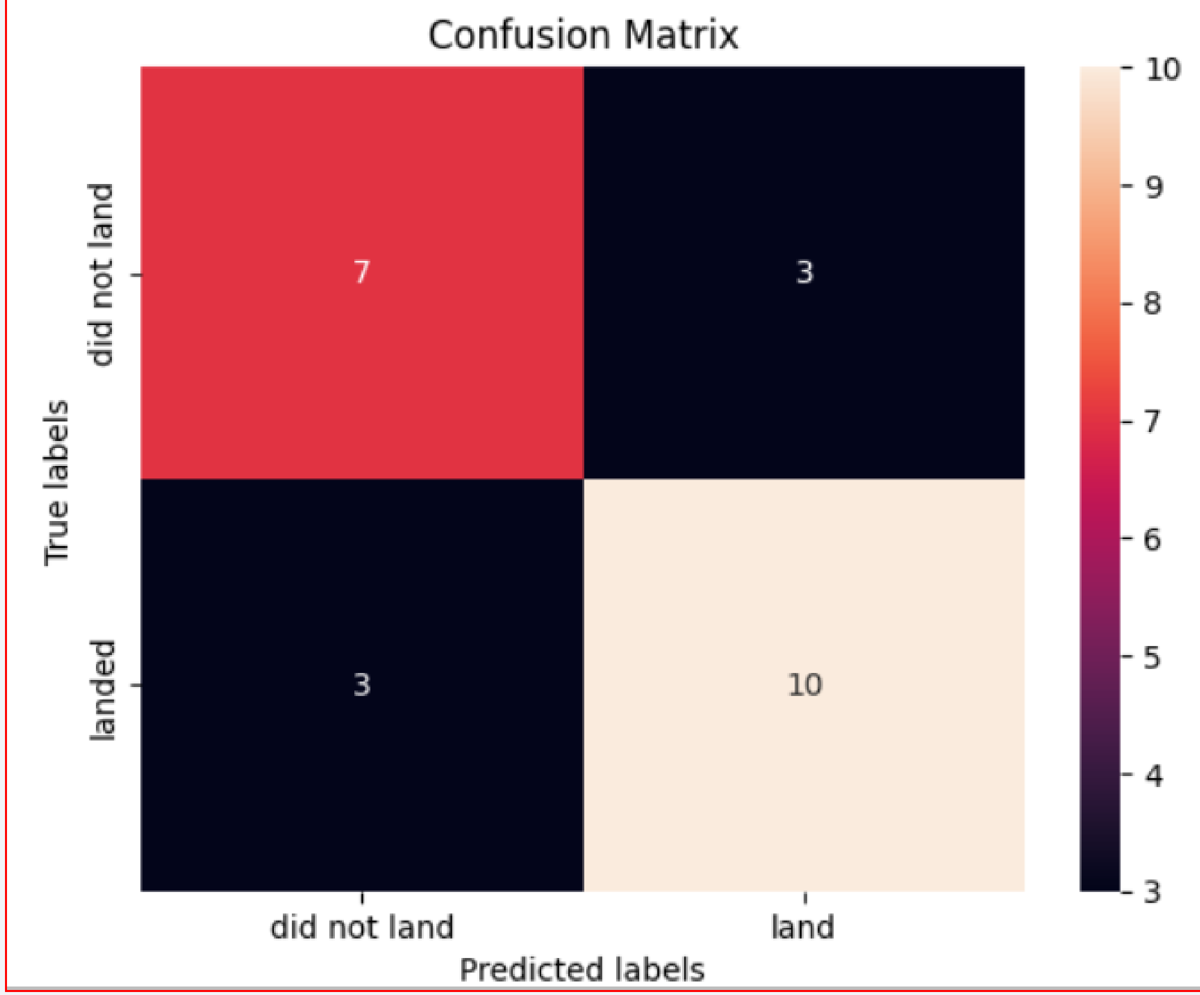


- Following table and Bar chart demonstrate best accuracy and parameters for each classification model (Logistic Regression, SVM, Decision Tree and KNN)

- According to these reports the model which has the highest classification accuracy is **Decision Tree**

Confusion Matrix

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



- A **Confusion matrix** is ***matrix*** used for evaluating the **performance of a classification model.** A good model should have high true positive (TP) rate and low true negative (TN)
- The Desicion Tree model has ***TP = 10, TN = 7***, and false positive rate and false negative rate as 3

Conclusions

- The best orbit types for launches are orbit ‘ES-L1’, ‘GEO’, ‘HEO’.
- The Success rate was growing since 2013 year through 2017, but after 2017 year success rate decreased sharply. Restoring of growth of success rate was began in 2018 year
- The best machine learning algorithm for classification is Decision Tree
- Total payload mass carried from NASA is 45596

Appendix

Thank you!