Winning Space Race with Data Science

JOHN AKINWANDE ELEWA

FEBRUARY 26, 2024

Outline

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

Executive Summary

Summary of Methodologies

- Data Collection through API
- Data Collection – Web Scrapping
- Data Collection - Data Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

Summary of Results

- Exploratory Data Analysis Results
- Interactive Analytics
- Predictive Analytics Results
- Conclusion

Introduction

Project Background and Context

The **SpaceX Data Science Project** aims to leverage data analytics and machine learning techniques to enhance space exploration and rocket technology. As SpaceX continues to revolutionize space travel, understanding and optimizing data-driven processes become critical. This project operates at the intersection of data science, engineering, and space exploration.

Problems to find Address

- Rocket Performance Optimization
- Predictive Maintenance
- Mission Success Rate Enhancement
- Space Debris Mitigation
- Improving on Crew Health and Safety
- Potential for Increased Market Share

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected via REST API with Web Scraping Methodology via Wikipedia
- Performed data wrangling:
  - Data was cleaned, Missing values addressed, Categorized with One-hot-Encoding
- Performed exploratory data analysis (EDA) using visualization and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models:
  - Used to build, tune, evaluate classification models

Data Collection

- Data was collected via 2 methods: *get.request* to the SpaceX – API; then,
- Decoded the response content as a json() file;
- Turned data into a pandas dataframe;
- Requested data using GET request; and parse data;
- Filtered the data for only Falcon9 data and information;

```
# Calculate the mean value of PayloadMass column

# Replace the np.nan values with its mean value
PayloadMass = pd.DataFrame(data_falcon9['PayloadMass'].values.tolist()).mean(1)
print(PayloadMass)
```

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

0]:  'https://api.spacexdata.com/v4/launches/past'

1]:  response = requests.get(spacex_url)
     response

1]:  <Response [200]>
```

Check the content of the response

```
2]:  print(response.content)
```

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNe
static_json_url

'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/AI
on'
```

We should see that the request was successfull with the 200 status response code

```
response.status_code

200
```

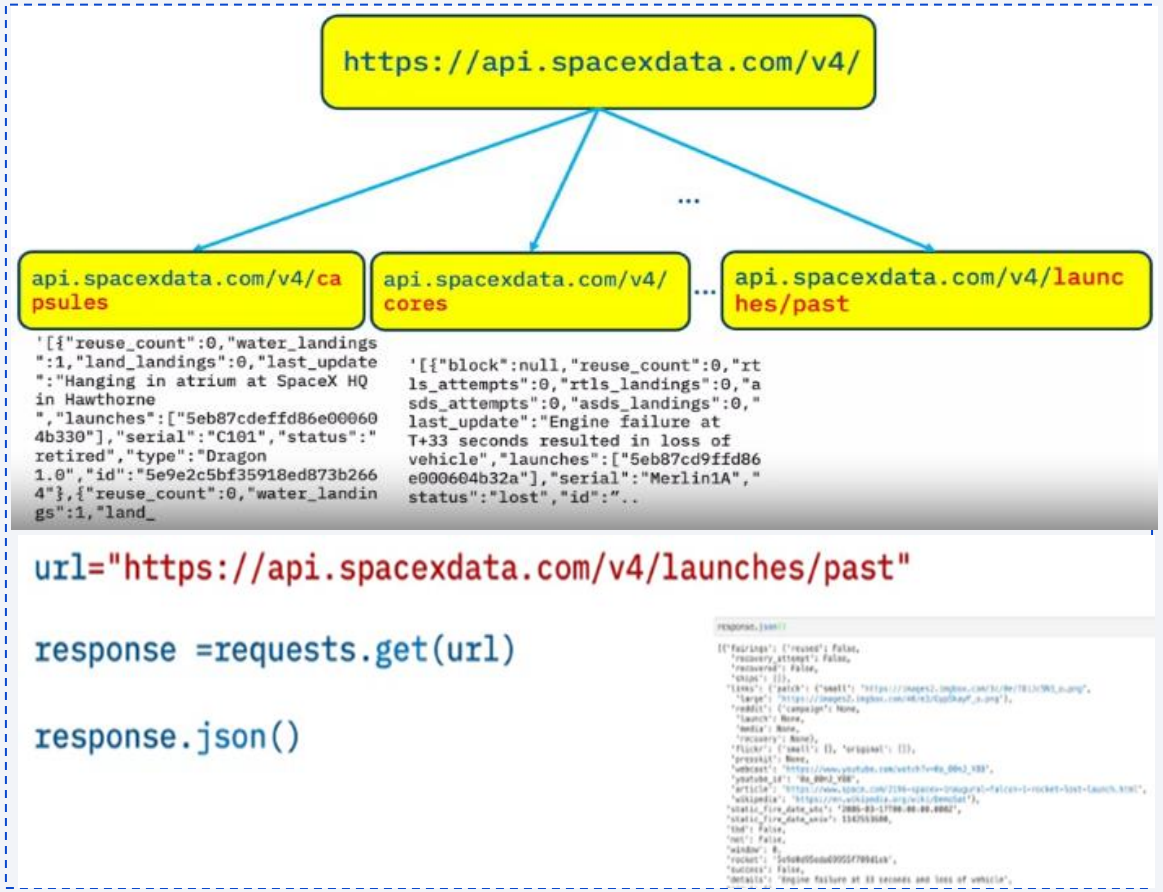
Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_norı`

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

Data Collection – SpaceX API

- Data collection with SpaceX API via GET request, filtered data, only for Falcon9 launches and dealt with missing values.

- GitHub URL link to the completed SpaceX API calls notebook: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/jupyter-labs-spacex-data-collection-api.ipynb for an external reference and peer-review purpose



Data Collection - Scraping

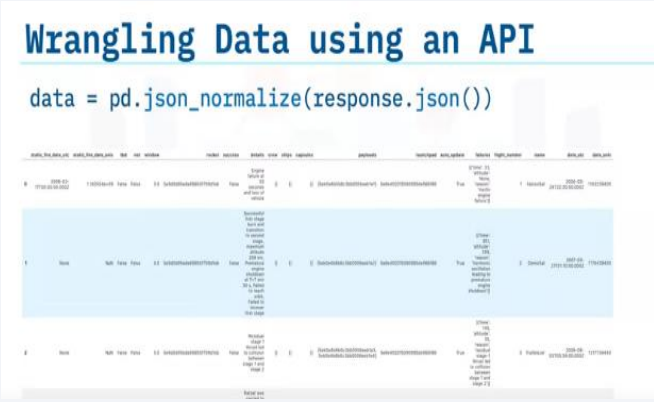
- Process:
  - Use of helper functions to process web scraped HTML table
  - Use of BeautifulSoup for Scraping Falcon9 records data.
  - Used BeautifulSoup to extract columns variable names from HTML table
- GitHub URL of completed web scraped notebook: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/Completed-jupyter-labs-webscraping.ipynb

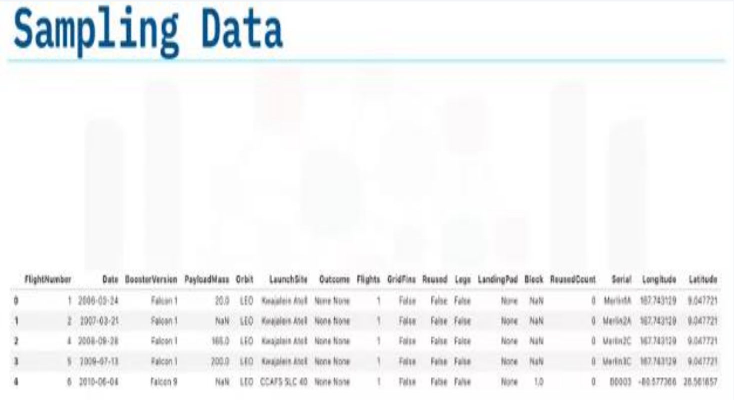
```python
# Use soup.title attribute
# Print the extracted column titless
import requests
from bs4 import BeautifulSoup

# Example URL (replace with your desired webpage)
url_to_scrape = "https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches"
response = requests.get(url_to_scrape)

if response.status_code == 200:
    # Create a BeautifulSoup object from the HTML content
    soup = BeautifulSoup(response.text, 'html.parser')

    # Print the title of the webpage
    title = soup.title
    print("Webpage title:", title.string)
else:
    print(f"Error: Unable to retrieve the webpage. Status code: {response.status_code}")
```

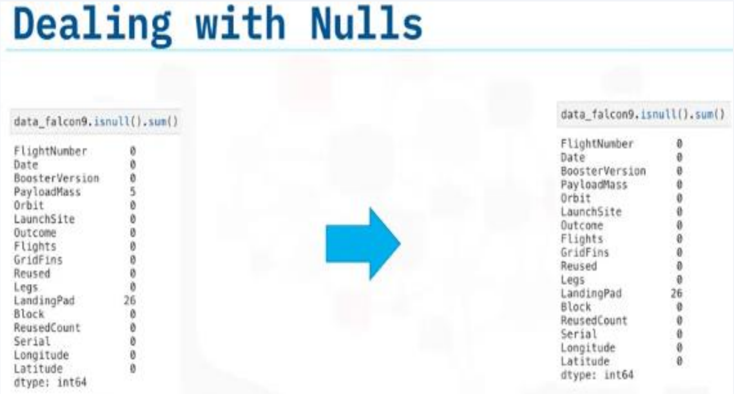
```
Webpage title: List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

Data Wrangling

- How data were processed:
  - Loaded SpaceX Dataset;
  - Calculated percentage of missing values;
  - Calculated the number and occurrences, launches of each site, of each orbit, of mission outcomes, and landing outcomes and exported the results to .csv file.
- GitHub URL of completed data wrangling related notebook:
- https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/Completed-labs-jupyter-spacex-Data%20wrangling.ipynb

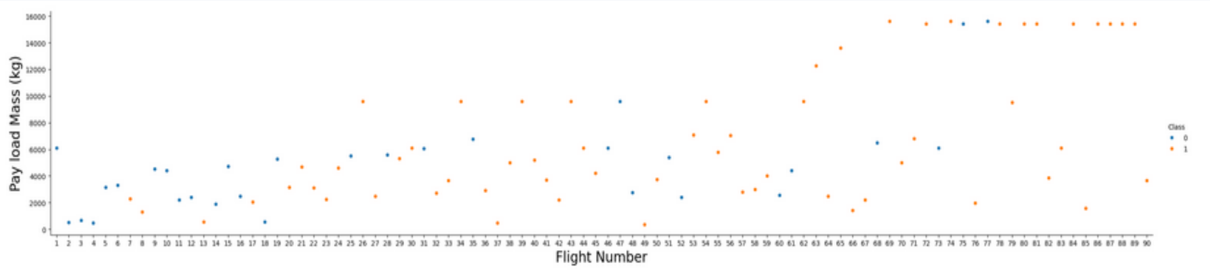






EDA with Data Visualization

- Summary of charts plotted and why used those charts:
    - Flight number and Payload;
    - Flight number and Launch Site;
    - Payload and Launch Site;
    - Success rate of each orbit type and flight number and orbit type.
    - These were done to see how these variables evolve together and how they affect the launch outcomes.



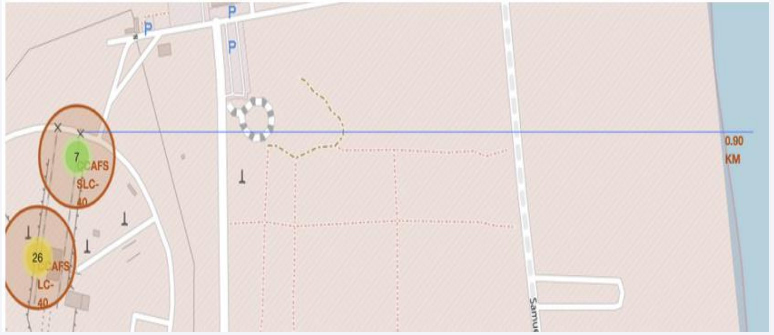
- GitHub URL of your completed EDA with data visualization notebook: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/Completed-jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Summary of the SQL queries performed:
  - Names of Unique launch sites;
  - Average payload mass accrued by booster; ver. F9 v1.1;
  - Date of first successful landing outcome in ground pad was achieved;
  - Names of the boosters which have success in drone ship;
  - List of total number of successful and failure mission outcomes;
  - List of records which will display the month names, failure landing_outcomes in drone ship
- Add the GitHub URL of your completed EDA with SQL notebook: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/Completed-jupyter-labs-eda-sql.ipynb

Build an Interactive Map with Folium

- Summary of map objects - markers, circles, lines, created and added to a folium map:
  - Mark all launch sites on a map;
  - Mark the success/failed launches for each site on the map;
  - Calculated the distances between a launch site to its proximities;



- Objects interactive visual analytics dissemination of information

```
[26]:
# Initializing the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)

# Adding a Circle object for each launch site based on its coordinate (Lat, Long) values
launch_sites_dict = launch_sites_df.set_index('Launch Site').T.to_dict('list')
launch_sites_dict

[26]: {'CCAFS LC-40': [28.56230197, -80.57735648],
 'CCAFS SLC-40': [28.56319718, -80.57682003],
 'KSC LC-39A': [28.57325457, -80.64689529],
 'VAFB SLC-4E': [34.63283416, -120.6107455]}
```

```
[71]:
distance_highway = calculate_distance(launch_site_lat, launch_site_lon, closest_highway[0], closest_highway[1])
print('distance_highway =',distance_highway, ' km')
distance_railroad = calculate_distance(launch_site_lat, launch_site_lon, closest_railroad[0], closest_railroad[1])
print('distance_railroad =',distance_railroad, ' km')
distance_city = calculate_distance(launch_site_lat, launch_site_lon, closest_city[0], closest_city[1])
print('distance_city =',distance_city, ' km')

distance_highway = 0.5834695366934144  km
distance_railroad = 1.2845344718142522  km
distance_city = 51.43416999517233  km
```

- GitHub URL of your completed interactive map with Folium map: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/Completed-lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Summary of what plots/graphs and interactions *codes* added to a dashboard:
  - Dropdown list to enable Launch Site selection;
  - A pie chart to show the total successful launches count for all sites;
  - Added a slider to select payload range;
  - A scatter chart for the correlation between payload and launch success;

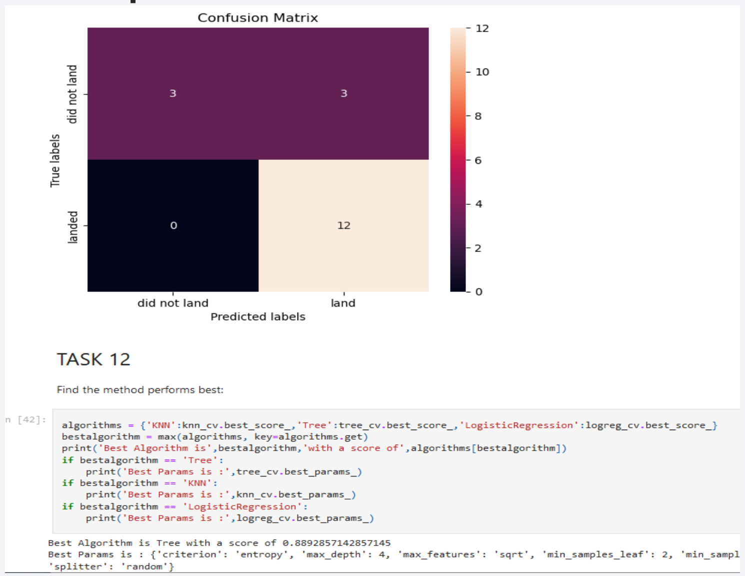
```python
# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-pay
@app.callback(
    Output(component_id='success-payload-scatter-chart', component_property='figure'),
    [Input(component_id='site-dropdown', component_property='value'), Input(component_id="
)
def get_scatter_chart(entered_site, payload_range):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        data = filtered_df[['Payload Mass (kg)', 'class', 'Booster Version Category']].cop
        data1 = data[(data['Payload Mass (kg)'] >= payload_range[0]) & (data['Payload Mass
        fig = px.scatter(data1, x="Payload Mass (kg)", y="class", color="Booster Version C
        return fig
    else:
        data = filtered_df.loc[filtered_df['Launch Site'] == entered_site, ['Payload Mass
        data1 = data[(data['Payload Mass (kg)'] >= payload_range[0]) & (data['Payload Mass
        fig = px.scatter(data1, x="Payload Mass (kg)", y="class", color="Booster Version C
        return fig
```

- Plots and interactions: for easy selections and comparison purposes:
  - Plots and Charts added to show the relationships of Outcomes with Payload Mass(kg);

- GitHub URL of your completed Plotly Dash lab: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Summary of how the classification model was built, evaluated and improved, to find the best performing model:
    - created a column for the class;
    - Standardized the data;
    - Data was split into training data and test data;
- Model development process:
    - Performed exploratory Data Analysis and determine Training Labels;
    - Built different Machine Learning models;
    - Found the best Hyperparameter for SVM, Classification Trees and Logistic Regression;
    - Found the method that performs best using test data;
- Add the GitHub URL of your completed predictive analysis: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/Completed-SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

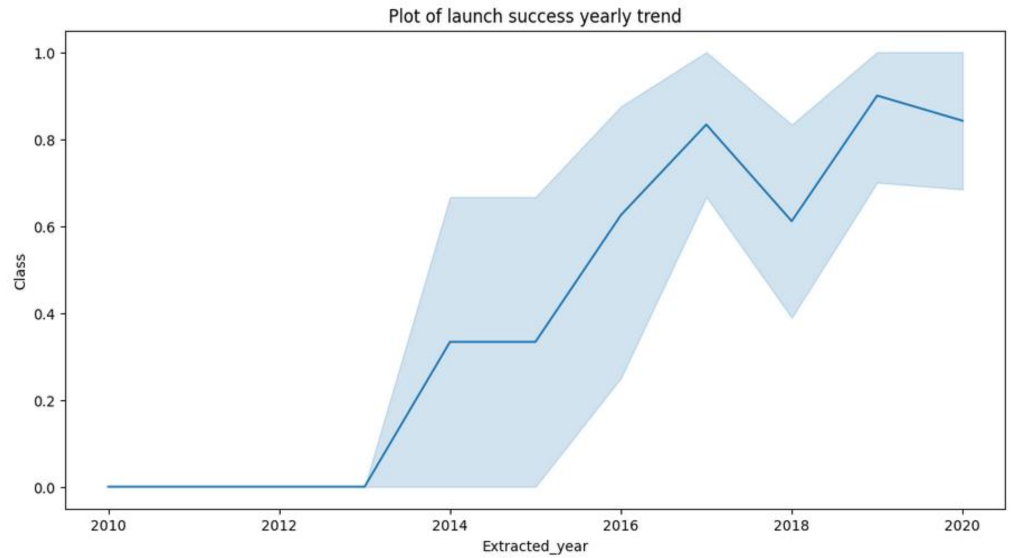


Results

- Exploratory data analysis results:

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

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```



- Interactive analytics demo in screenshots:

- Predictive analysis results

Find the method performs best:

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8892857142857145
Best Params is : {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5,
'splitter': 'random'}
```

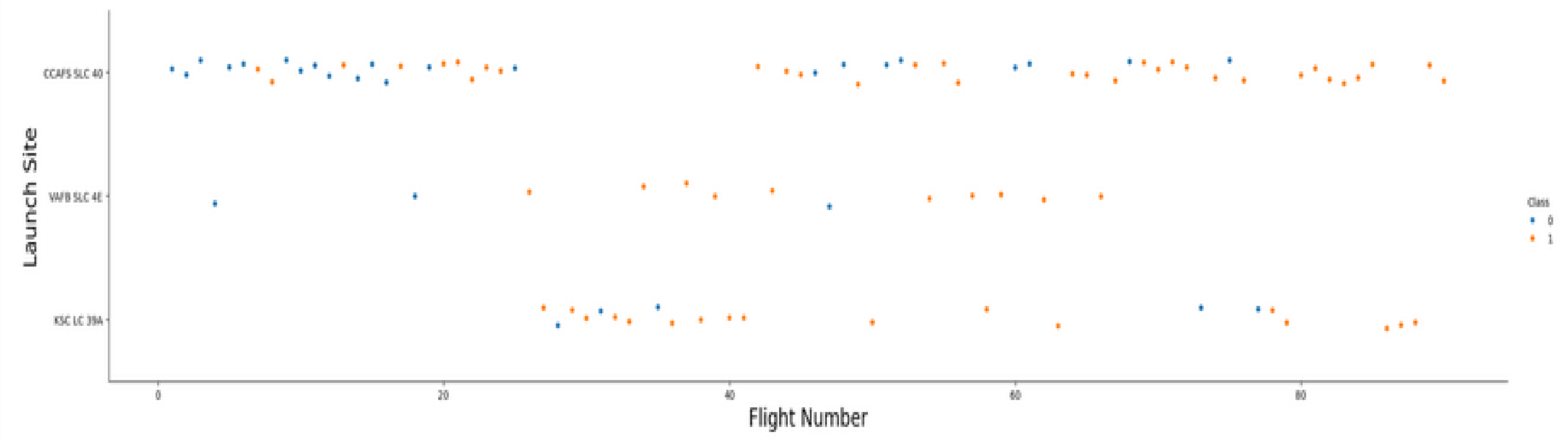
  - Best Algorithm is Tree with a score of 0.8892857142857145

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Showing a scatter plot of Flight Number vs. Launch Site:

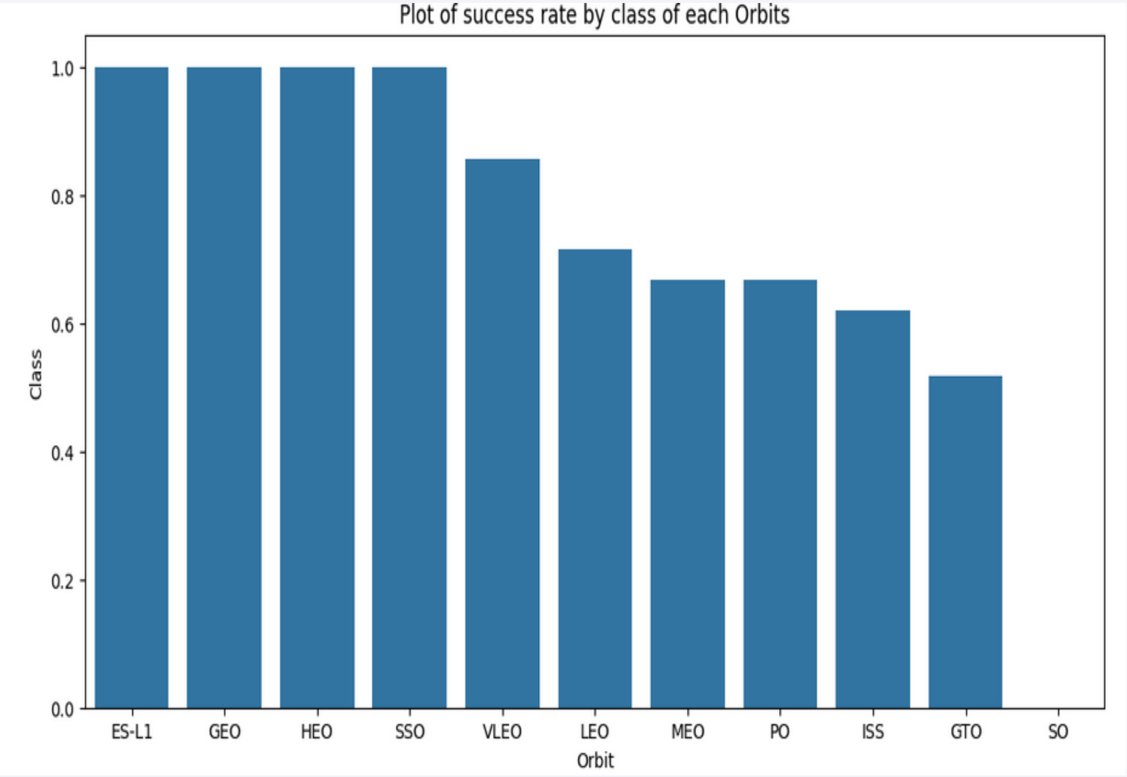


- The screenshot of the scatter show that the greater the Flight Number the greater the successes rate.

Success Rate vs. Orbit Type

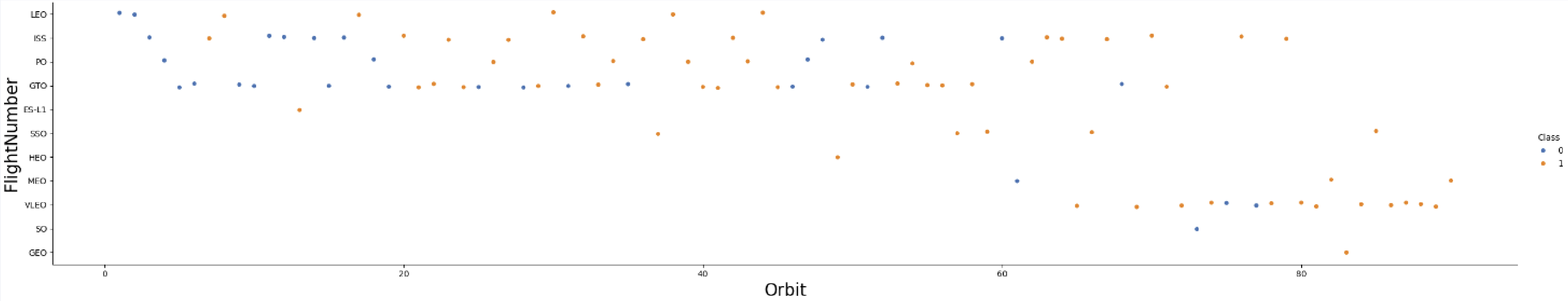
- Showing a bar chart for the success rate of each orbit type:

- The plot shows that ES-L1, GEO and HEO and SSO had the most success rates with VLEO not too far behind.



Flight Number vs. Orbit Type

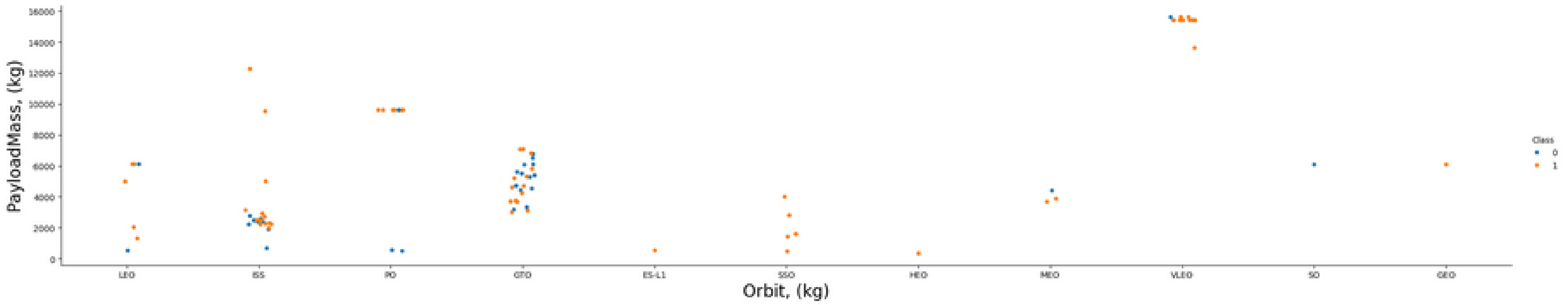
- Show a scatter point of Flight number vs. Orbit type:



- The plot shows that GTO, PO, ISS, and LEO orbits has the most flights. LEO has the most successful even though there seems to be no relationship between flight number and GTO orbit.

Payload vs. Orbit Type

- Showing a scatter point of payload vs. Orbit type:



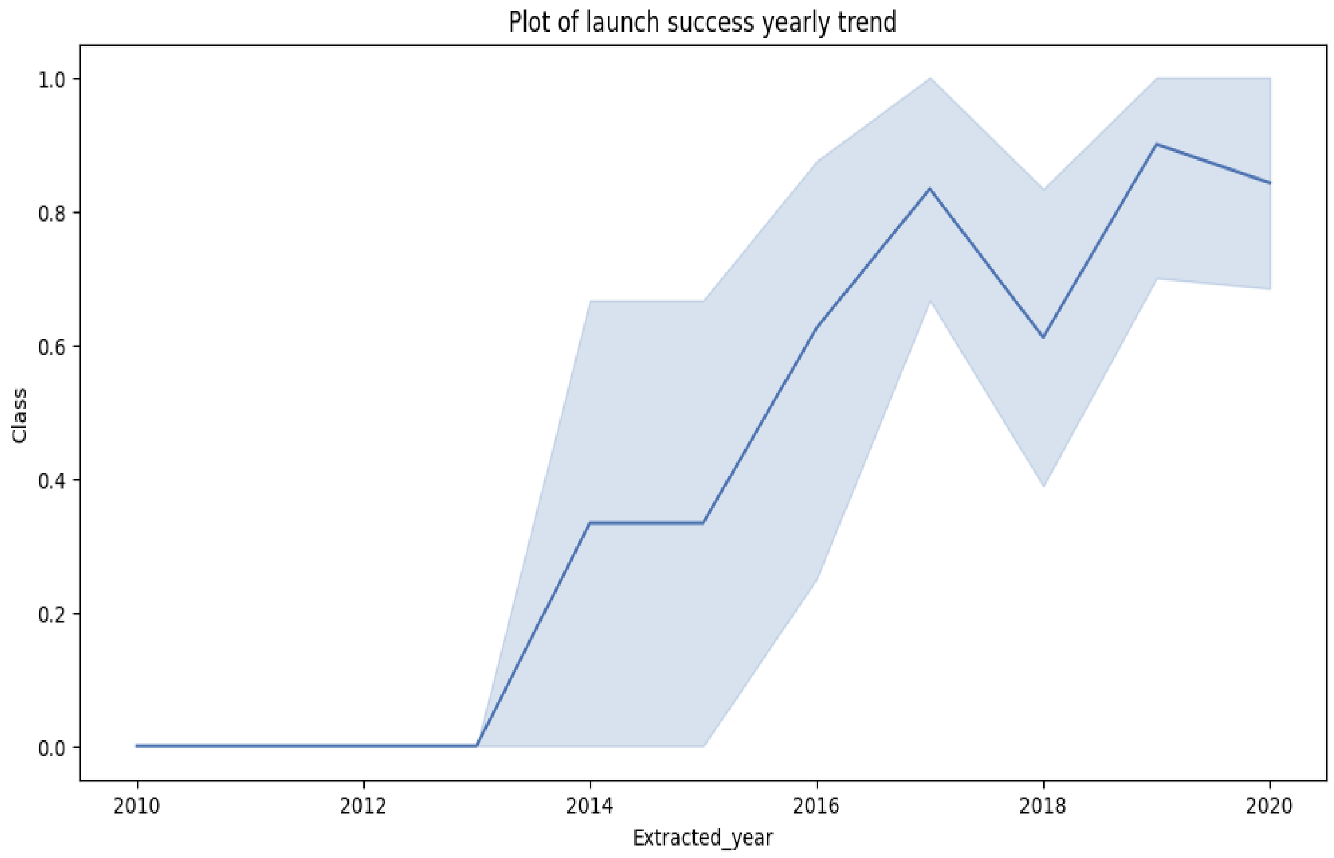
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

- The plot shows that with heavy payloads the successful landing or positive landing rate are more for PO, LEO, VLEO and ISS.

Launch Success Yearly Trend

- Showing a line chart of yearly average success rate:

- Plot shows launch success yearly trends climbs up from 2013 to 2020, with a slight dip in 2018.



All Launch Site Names

- Names of the **unique** launch sites:
  - Launch_Sites
  - CCAFS LC-40
  - VAFB SLC-4E
  - KSC LC-39A
  - CCAFS SLC-40
- SQL query result using DISTINCT:
  - This shows only the unique launch sites from the SpaceX data.

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

```
%sql SELECT DISTINCT "Launch_Site" as Launch_Sites FROM SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

- Showing 5 records where launch sites begin with `CCA`:

- Showing launch sites with the string "CCA", using SQL query:
  - SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5

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

```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	

Total Payload Mass

- Calculated total payload carried by boosters from NASA:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEXTBL WHERE customer = 'NASA (CRS)';
```

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

Total Payload Mass by NASA (CRS)
45596

- Showing the results of the total payload by booster from NASA:

  45596 kg.

Average Payload Mass by F9 v1.1

- Calculated average payload mass carried by booster version F9 v1.1:

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by booster version F9 v1.1" FROM SPACEXTBL WHERE booster_version =
```

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

Average Payload Mass by booster version F9 v1.1
2928.4

- Calculated payload mass = 2928.4 kg

First Successful Ground Landing Date

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

```
%sql SELECT MIN(DATE) AS "First succesful landing outcome in ground pad was acheived." FROM SPACEXTBL WHERE Landing_Outcome =
```

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

First succesful landing outcome in ground pad was acheived.
2015-12-22

- Date of first successful landing outcome on ground pas was: 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' \
AND payload_mass__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

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

- Query result of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000 – <u>*shown above.*</u>

Total Number of Successful and Failure Mission Outcomes

- Calculated total number of successful and failure mission outcomes:

```
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", sum(case when MISSION_O
```

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

Successful Mission	Failure Mission
100	1

- Showing results as:
  - Successful Mission = 100
  - Failure Mission = 1

Boosters Carried Maximum Payload

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

```
%sql SELECT DISTINCT BOOSTER_VERSION "Booster Version with Naximum Payload Mass" FROM
```

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

Booster Version with Naximum Payload Mass
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

- Presenting query result for 12 Booster Versions with successful Maximum Payload Mass:

2015 Launch Records

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

```
%sql SELECT substr(Date, 6, 2) AS Month, landing_outcome AS Failure_Landing_Outcomes, Booster_version, launch_site FROM SPACEX
```

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

Month	Failure_Landing_Outcomes	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
02	Controlled (ocean)	F9 v1.1 B1013	CCAFS LC-40
03	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

- Results of failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015, as shown above.

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

- Ranked 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:

```
%sql SELECT LANDING_OUTCOME as "Landing Outcomes", COUNT(LANDING_OUTCOME) AS "Total Count" FROM SPACEXTBL \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING_OUTCOME \
ORDER BY COUNT(LANDING_OUTCOME) DESC;
```

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

- “No Attempt” having 10, with Success and Failure having equal total counts of 5 each.

Landing Outcomes	Total Count
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

Section 3

Launch Sites Proximities Analysis

<Folium Map – Marked Launch Sites>

- Replaced title with an appropriate title: <u>Folium Map – Marked Launch Sites</u>

- Generated folium map: screenshot includes all launch sites' location markers on a global map

- Folium Map showing launch sites proximity to the equator and very close to proximity to the coast of Florida and California.

Folium Map: Marked Launch Sites- Color Labelled

- Replace title with an appropriate title:
- Folium Map – Marked Launch Sites Color Labelled

- Folium map with a proper screenshot showing the color-labeled launch outcomes on the map
- Color-labelled for easy identification of which launch sites have relatively high success rates:
  - Green Successful Launches,
  - Red – Failure Launches

Folium Map: Marked Launch Sites-Calculated Proximity Distance

- Replaced Folium map screenshot title with an appropriate title:

Folium Map: Marked Launch Sites – Calculated Proximity Distance

- Generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

- Showing calculated proximity distance of 0.9km

Section 4

Build a Dashboard with Plotly Dash

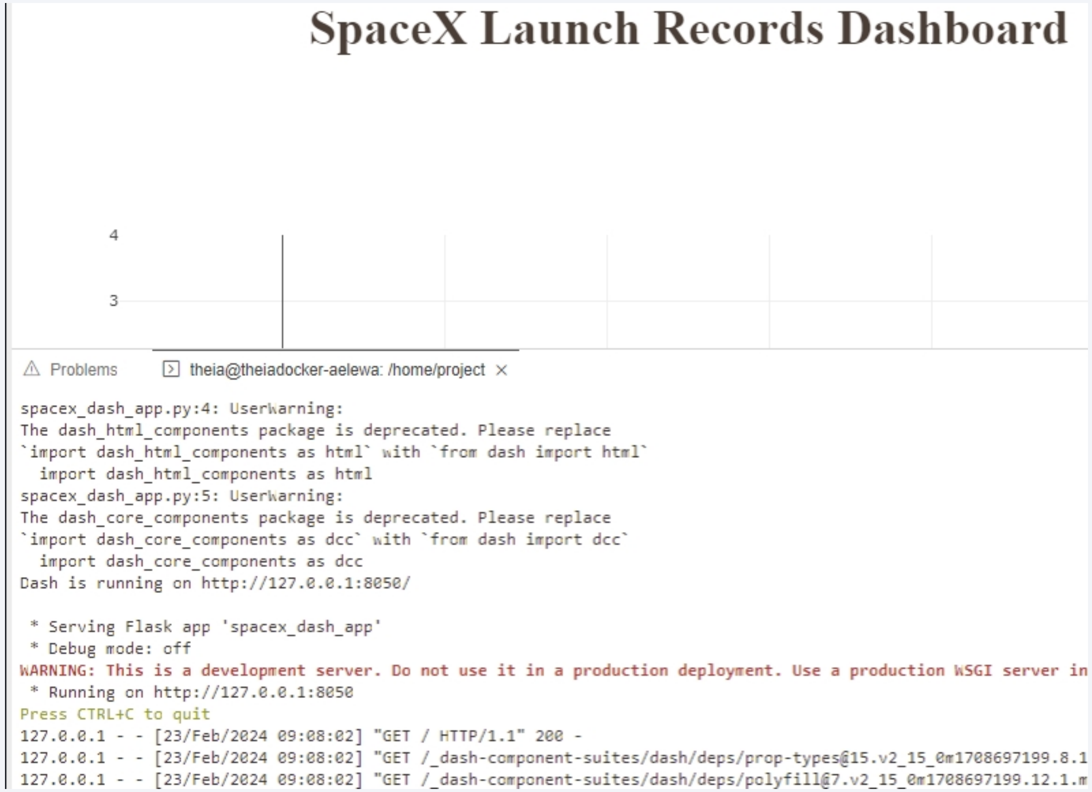
Plotly Dashboard: Plotly Dash

- Replaced title with an appropriate title:

  Plotly Dashboard: Launch Success Count All Sites

- Showing application interface "*Cloud IDE refused to generate full diagram due to "busy port: 8050"*

- Link to screenshot: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/Dasbard-Render.jpg



Plotly Dashboard: Launch Success Count – All Sites

- Replaced title with an appropriate title:

  Plotly Dashboard: Launch Success Count All Sites

- Showing application code screenshot of launch success count for all sites, for a pie chart: "*Cloud IDE refused to generate diagram*"

- Link to screenshot: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/TASK-1.jpg

```python
# Create a Dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(children=[
    html.H1('SpaceX Launch Records Dashboard',
            style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),

    # TASK 1: Add a dropdown list to enable Launch Site selection
    # The default select value is for ALL sites
    dcc.Dropdown(
        id='site-dropdown',
        options=[
            {'label': 'All Sites', 'value': 'ALL'},
            {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
            {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'},
            {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
            {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
        ],
        value='ALL',
        placeholder="Select a Launch Site here",
        searchable=True
    ),

    html.Br(),
    # Rest of your layout components...
])
```

Plotly Dashboard: Launch Site with Highest Success Ratio

- Replace title with an appropriate title:
- Plotly Dashboard: Launch Site with Highest Success Ratio
- Showing the *code* screenshot of the pie chart code for the launch site with highest launch success ratio
- Used callback Input and Output
- Used get.piechart and filtered “df” to get the launch site with the highest success ratio.
- Link: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/TASK-2.jpg

```
# TASK 2: Add a pie chart to show the total successful launches count for all sites
# If a specific launch site was selected, show the Success vs. Failed counts for the site
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
@app.callback(
    Output(component_id='success-pie-chart', component_property='figure'),
    Input(component_id='site-dropdown', component_property='value')
)
def get_pie_chart(entered_site):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        counts = filtered_df.groupby('Launch Site')['class'].sum().reset_index()
        fig = px.pie(counts, values='class', names='Launch Site', title='Total Success Launches by Site')
        return fig
    else:
        data = filtered_df.loc[filtered_df['Launch Site'] == entered_site, ['Launch Site', 'class']]
        counts = data['class'].value_counts(normalize=True)
        fig = px.pie(counts, values=counts.values, names=counts.index, title=f"Total Success Launches for site {entere
        return fig
```

Plotly Dashboard: Payload vs Launch Outcome

- Replaced title with an appropriate title:

Plotly Dashboard: Payload vs Launch Outcome

- Showing *code* screenshots of success Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

- Showing the important *code elements* and steps for which payload range or booster version have the largest success rate, etc.

- Link: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/TASK-4.jpg

```
# TASK 4: Add a scatter chart to show the correlation between payload and launch success
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])

# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as outpu
@app.callback(
    Output(component_id='success-payload-scatter-chart', component_property='figure'),
    [Input(component_id='site-dropdown', component_property='value'), Input(component_id="payload-slider", component_p
)
def get_scatter_chart(entered_site, payload_range):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        data = filtered_df[['Payload Mass (kg)', 'class', 'Booster Version Category']].copy()
        data1 = data[(data['Payload Mass (kg)'] >= payload_range[0]) & (data['Payload Mass (kg)'] <= payload_range[1])
        fig = px.scatter(data1, x="Payload Mass (kg)", y="class", color="Booster Version Category")
        return fig
    else:
        data = filtered_df.loc[filtered_df['Launch Site'] == entered_site, ['Payload Mass (kg)', 'class', 'Booster Ver
        data1 = data[(data['Payload Mass (kg)'] >= payload_range[0]) & (data['Payload Mass (kg)'] <= payload_range[1])
        fig = px.scatter(data1, x="Payload Mass (kg)", y="class", color="Booster Version Category")
        return fig

# Run the app
if __name__ == '__main__':
    app.run_server()
```

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Model showing the highest classification accuracy is the Decision Tree Classifier;
- The final score is: 0.8892857142857145

Find the method performs best:

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

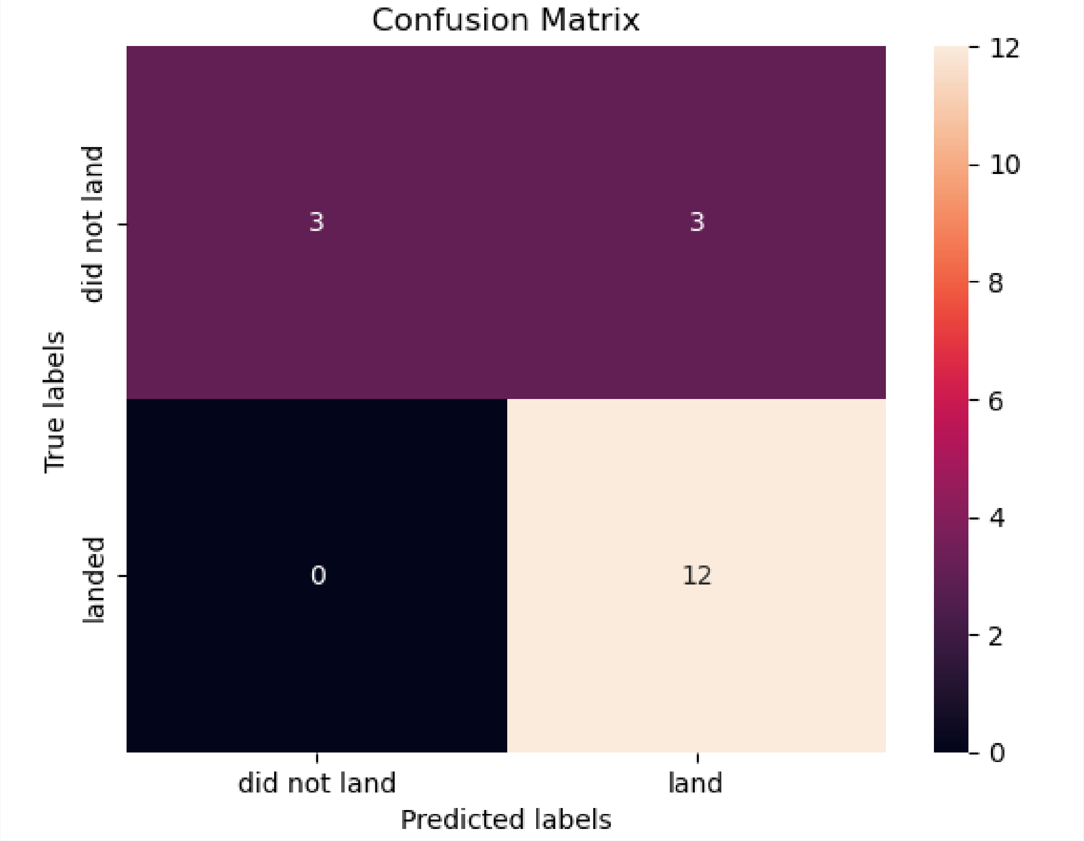
```
Best Algorithm is Tree with a score of 0.8892857142857145
Best Params is : {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix – Best Performing Model

- Showing the confusion matrix of the best performing model: –

- From the Decision Tree Classifier

- Showing the notable differences between the different classes. However False Positive exists for cases where unsuccessful landing were marked as successful.



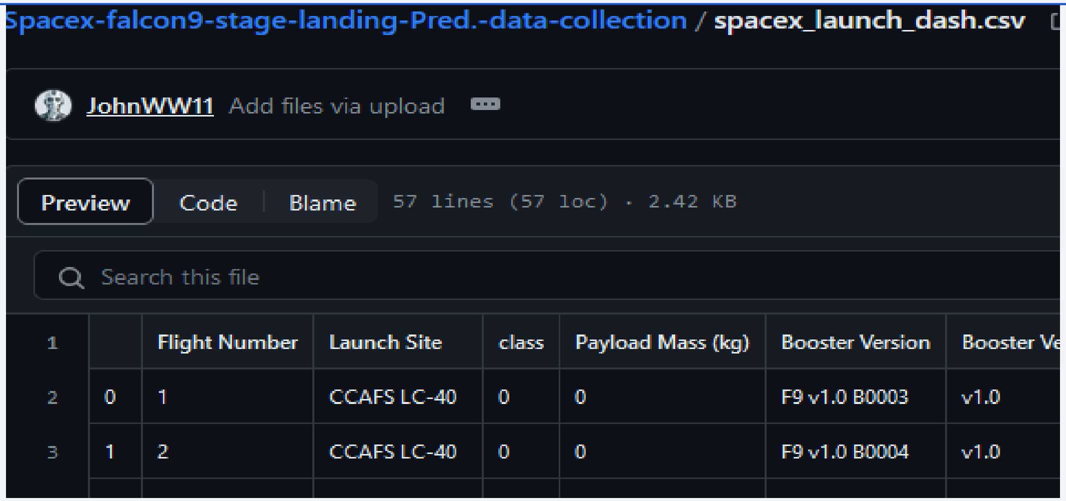
Conclusions

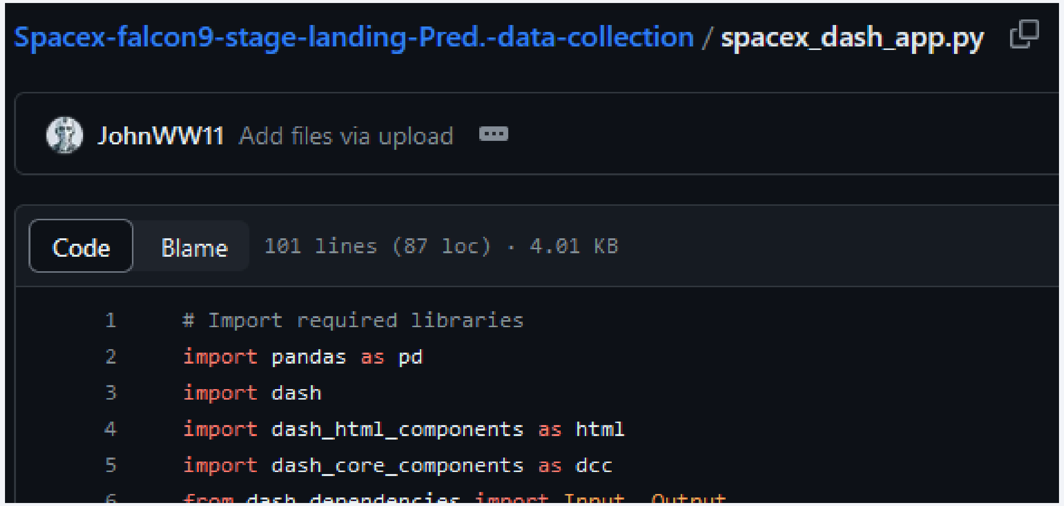
We can successfully conclude that:

- Point 1 - Larger flight amount seems to correspond with greater success rate;
- Point 2 – Orbit success rates corresponds with ES-L1, GEO, HEO, SSO, VLEO;
- Point 3 – The most successful launch site is KSC LC-39A of all other sites;
- Point 4 – The Decision Tree Classifier model work best in all of the models developed;
- Point 5 – Proximity to the equator and the ocean provides easier access for operational tasks;

Appendix

- SpaceX Launch Dash csv:
- Github link: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/spacex_launch_dash.csv
- SpaceX Launch Dash code snippet:
- Github link: https://github.com/JohnWW11/Spacex-falcon9-stage-landing-Pred.-data-collection/blob/main/spacex_dash_app.py





Thank you!