Data Science and Machine Learning Capstone Project

Baye Teshager Asmamaw

July 2022

IBM Developer

SKILLS NETWORK

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Special Insight
- Conclusion
- Appendix

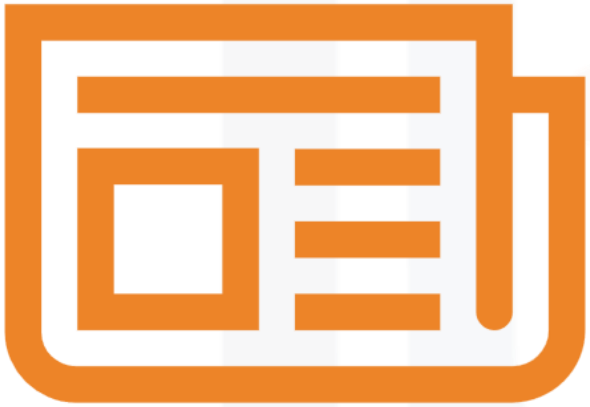
EXECUTIVE SUMMARY

- This projects predicts the success probability of the first launch of a SpaceX Falcon 9 rocket using machine learning algorithms
- Data is collected through API request and Web Scraping
- Methodology
  - Data wrangling
  - EDA with visualization and SQL
  - Building interactive map with Folium
  - Deploying predictive analysis (Classification)
- Results indicated that as flights increase, success rate increases
- Prediction accuracy of 83% obtained from all models deployed can be evidence that there is plenty of room to compete with SpaceX if a company intends to.

INTRODUCTION

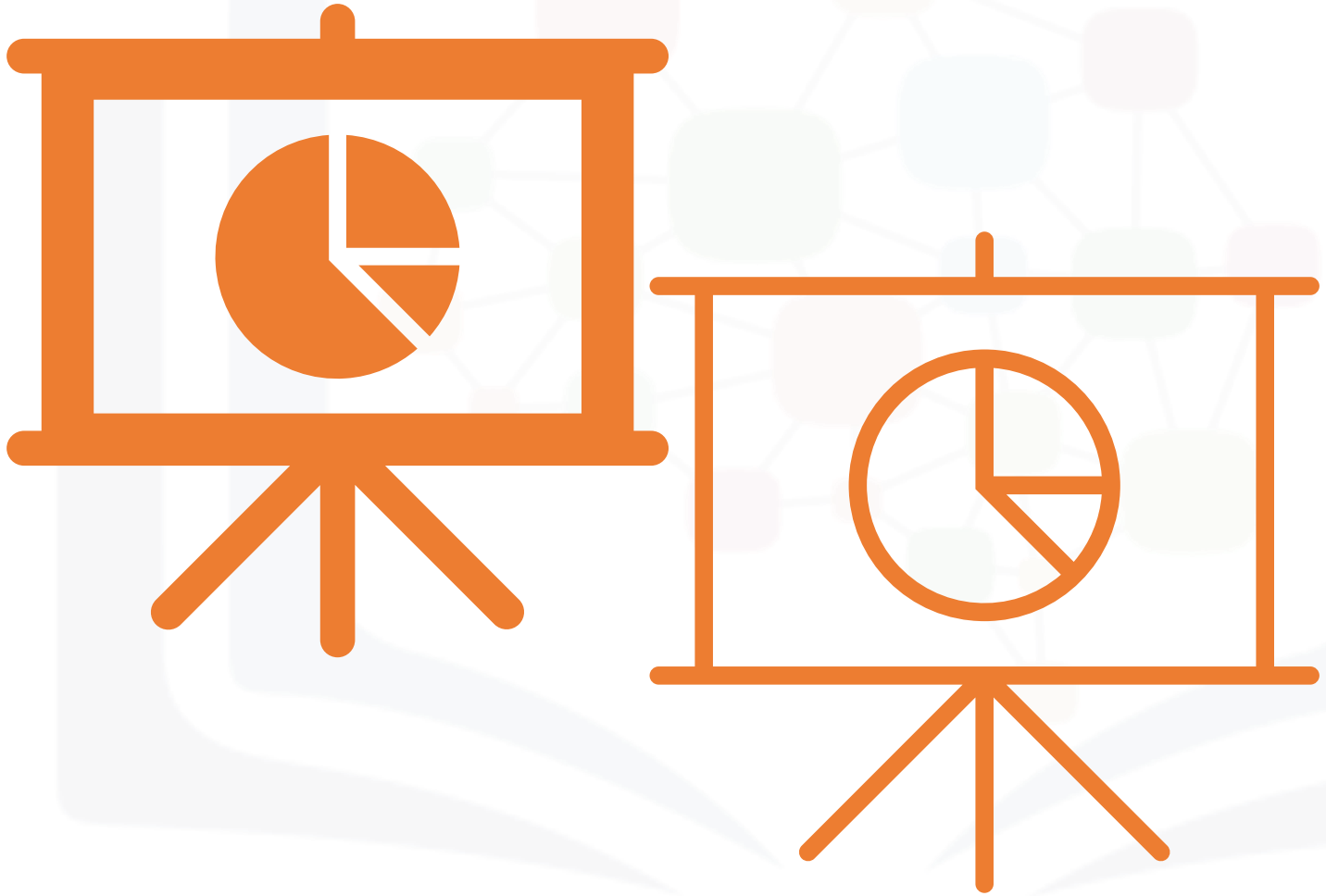
- Several companies are attempting to make space travel affordable for everyone
- SpaceX is the most successful, and one of the reasons is that their rocket launch is ***relatively inexpensive***
- SpaceX advertises Falcon 9 rocket launches on its website, with a cost of only 62 million dollars
- The savings is because SpaceX can ***reuse the first stage***
- Therefore, if we can determine if the first stage will land, we can determine the cost of a launch
- This project tries to determine the success of the first stage
  - analyzing data available from the SpaceX company website
  - deploying machine learning algorithms

METHODOLOGY



- Data collection
  - API requests
  - Web Scraping
- Data wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)

RESULTS



Data Collection by API request

Sending API Request and obtaining response

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [36]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [37]: response = requests.get(spacex_url)
```

Check the content of the response

```
In [38]: print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://
images2.imgbox.com/3c/0e/T8iJcSN3_o.png","large":"https://images2.imgbox.com/40/e3/GypSkayF_o.png"},"reddit":{"campaign":
null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://w
ww.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spacex-inaugural-fal
con-1-rocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17T0
0:00:00.000Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":fal
se,"failures":[{"time":33,"altitude":null,"reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and
loss of vehicle","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995
de566f86","flight_number":1,"name":"FalconSat","date_utc":"2006-03-24T22:30:00.000Z","date_unix":1143239400,"date_loca
l":"2006-03-25T10:30:00+12:00","date_precision":"hour","upcoming":false,"cores":[{"core":"5e9e289df35918033d3b2623","flig
ht":1,"gridfins":false,"legs":false,"reused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"la
ndpad":null}],"auto_update":true,"tbd":false,"launch_library_id":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reus
ed":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/4
f/e3/I0lkuJ2e_o.png","large":"https://images2.imgbox.com/be/e7/iNqsqVYM_o.png"},"reddit":{"campaign":null,"launch":nul
l,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/wa
tch?v=Lk4zQ2wP-Nc","youtube_id":"Lk4zQ2wP-Nc","article":"https://www.space.com/3590-spacex-falcon-1-rocket-fails-reach-or
bit.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":null,"static_fire_date_unix":nul
l,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":301,"altitude":289,"reas
on":"harmonic oscillation leading to premature engine shutdown"}],"details":"Successful first stage burn and transition t
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Filtering dataset for Falcon 9 launches

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
In [55]: # Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df['BoosterVersion'] == 'Falcon 9']
```

Now that we have removed some values we should reset the FlgihtNumber column

```
In [60]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.head()
```

Out[60]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	S
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B

GitHub link (click and follow)

https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/1.%20Data%20Collection%20%20with%20API.ipynb

Data Collection by Web Scraping

Request the Falcon9 Launch Wiki page from its URL

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
In [5]:
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url).text
```

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

```
In [6]:
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data, 'html5lib')
```

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

```
In [7]:
# Use soup.title attribute
soup.title
```

```
Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Fill in the parsed launch record values into launch_dict, and create a dataframe

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
In [17]:
df=pd.DataFrame(launch_dict)
df.head()
```

Out[17]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA (COTS)\nNRO	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

GitHub link (click and follow)

https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/2.%20Data%20Collection%20with%20Web%20Scraping.ipynb

IBM Developer

SKILLS NETWORK

Data Wrangling

Identify missing values and data types

Identify and calculate the percentage of the missing values in each attribute

```
In [3]: df.isnull().sum()/df.count()*100
```

```
Out[3]: FlightNumber      0.000
        Date              0.000
        BoosterVersion    0.000
        PayloadMass       0.000
        Orbit             0.000
        LaunchSite        0.000
        Outcome           0.000
        Flights           0.000
        GridFins          0.000
        Reused            0.000
        Legs              0.000
        LandingPad       40.625
        Block             0.000
        ReusedCount       0.000
        Serial            0.000
        Longitude         0.000
        Latitude          0.000
        dtype: float64
```

Identify which columns are numerical and categorical:

```
In [4]: df.dtypes
```

```
Out[4]: FlightNumber      int64
        Date             object
        BoosterVersion   object
        PayloadMass     float64
        Orbit            object
        LaunchSite       object
        Outcome          object
        Flights           int64
        GridFins           bool
        Reused             bool
        Legs               bool
        LandingPad       object
        Block           float64
        ReusedCount       int64
        Serial           object
        Longitude       float64
        Latitude        float64
        dtype: object
```

Calculating number of occurrences and mission outcome

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

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

TASK 3: Calculate the number and occurence of mission outcome per orbit type

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

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

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

```
In [8]: for i,outcome in enumerate(landing_outcomes.keys()):
            print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

```
In [9]: bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
        bad_outcomes
```

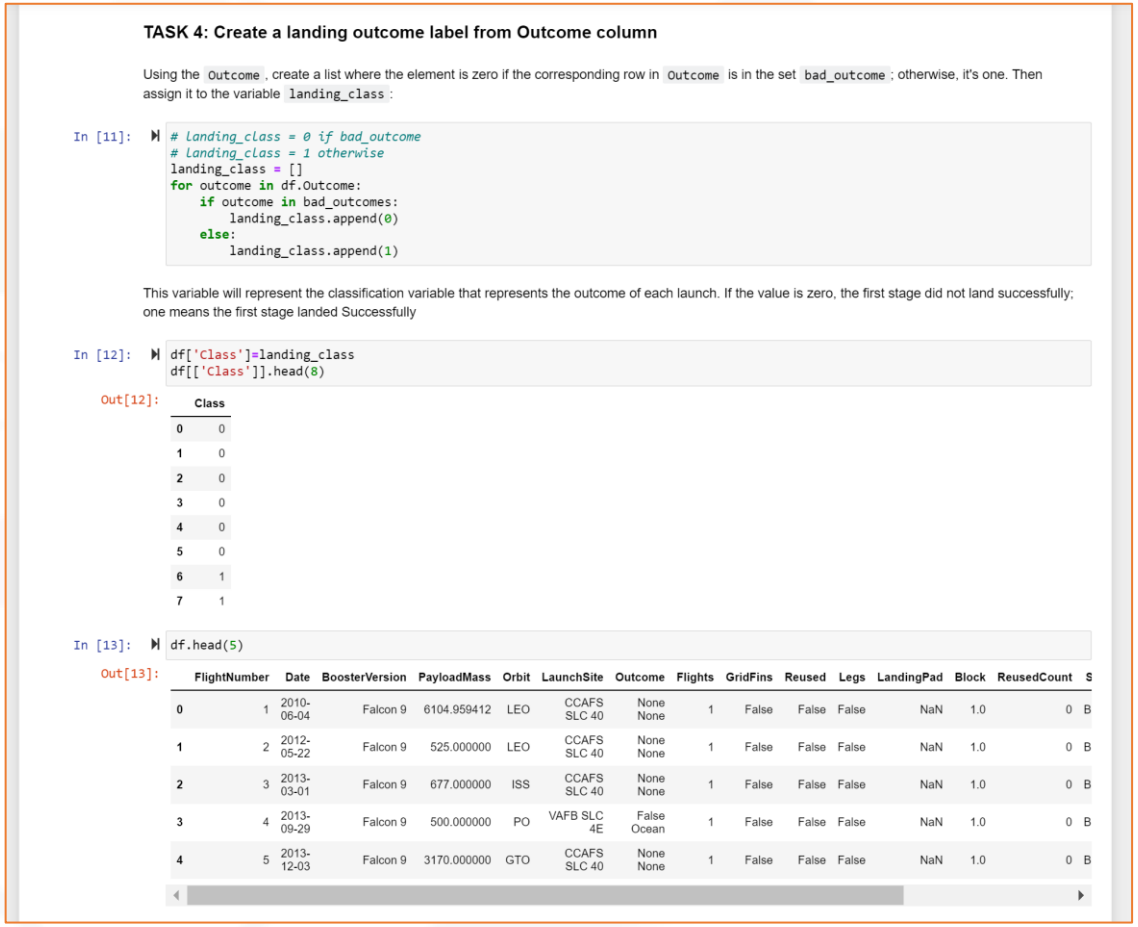
```
Out[9]: {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

GitHub link (click and follow)

https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/3.%20Data%20Wrangling.ipynb

Data Wrangling

Calculating label outcome



TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

In [11]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df.Outcome:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

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
6	1
7	1

In [13]:
```
df.head(5)
```

Out[13]:

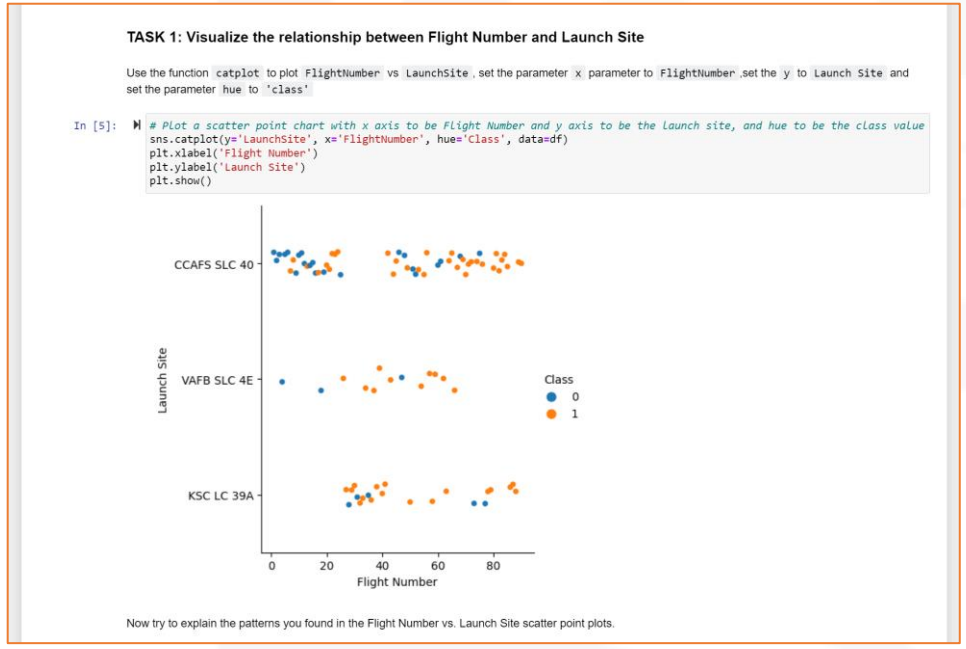
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	S
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B

GitHub link (click and follow)

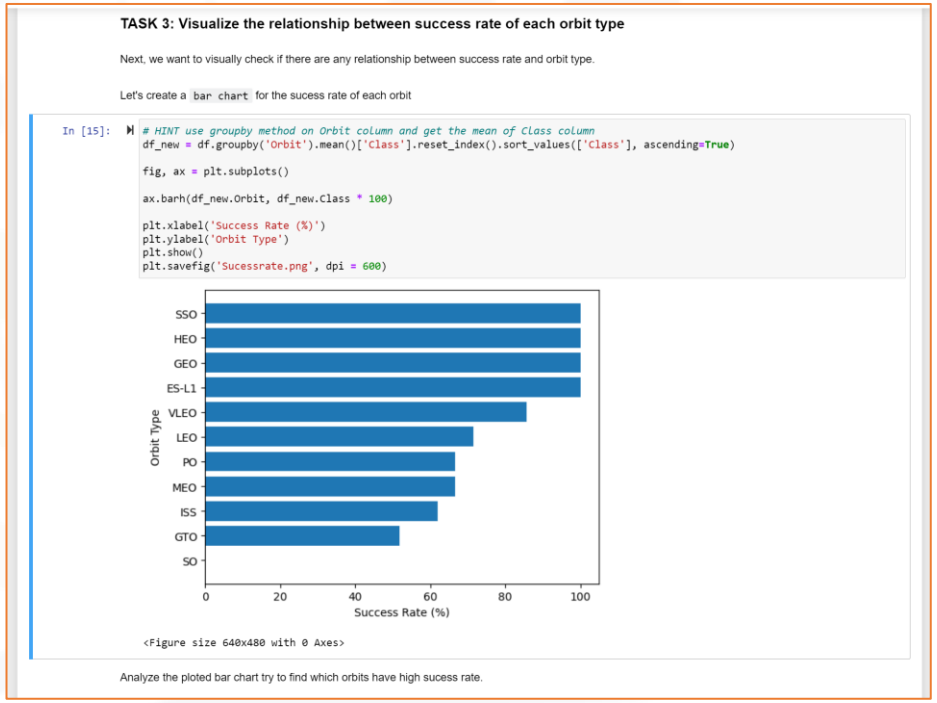
https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/3.%20Data%20Wrangling.ipynb

EDA with Data Visualization

Flight number vs. Launch rate



Success rate vs. Orbit type



GitHub link (click and follow)

https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/4.%20EDA%20with%20Visualization%20lab.ipynb

EDA with SQL

Query and result to count successful landing outcomes between 2010 and 2017

Task 10

Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

In [21]:
```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS total_number
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY total_number DESC
```

```
 * ibm_db_sa://ljv11121:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[21]:

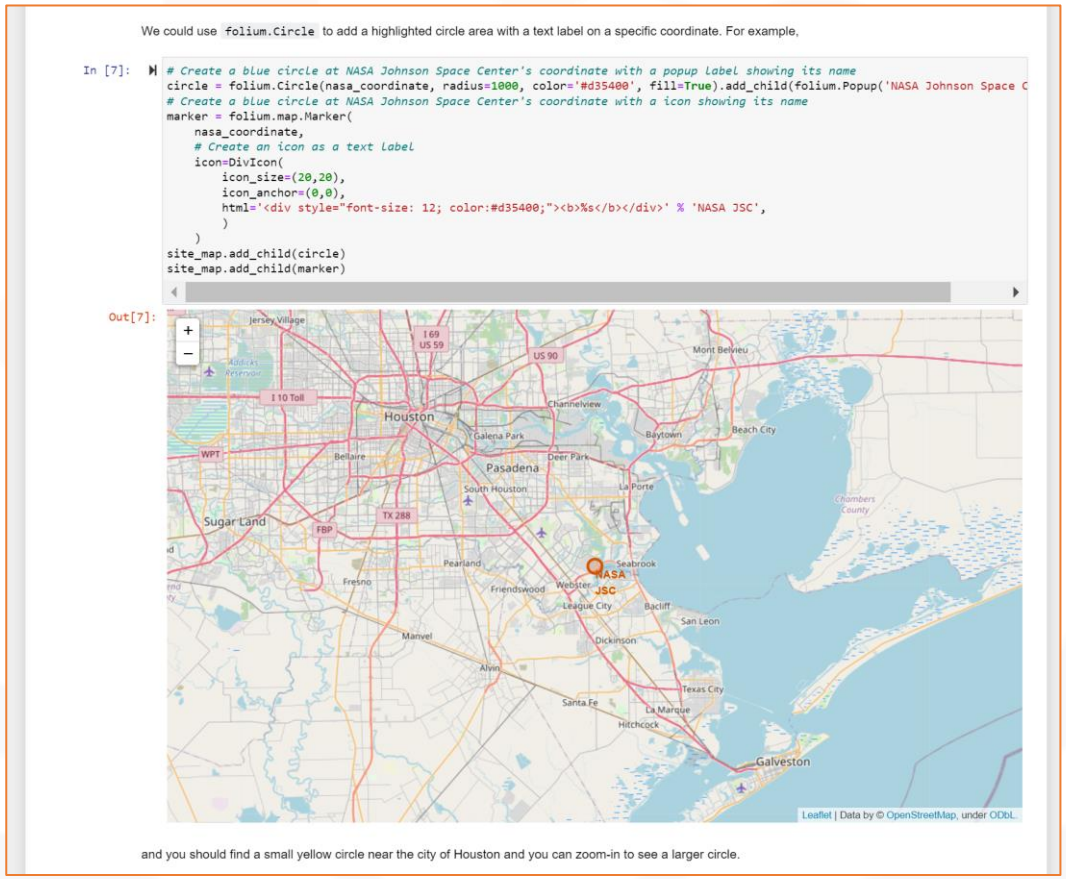
landing__outcome	total_number
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Success (ground pad)	4
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	1
Precluded (drone ship)	1

GitHub link (click and follow)

https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/5.%20EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

Locating launch site near Huston



GitHub link (click and follow)

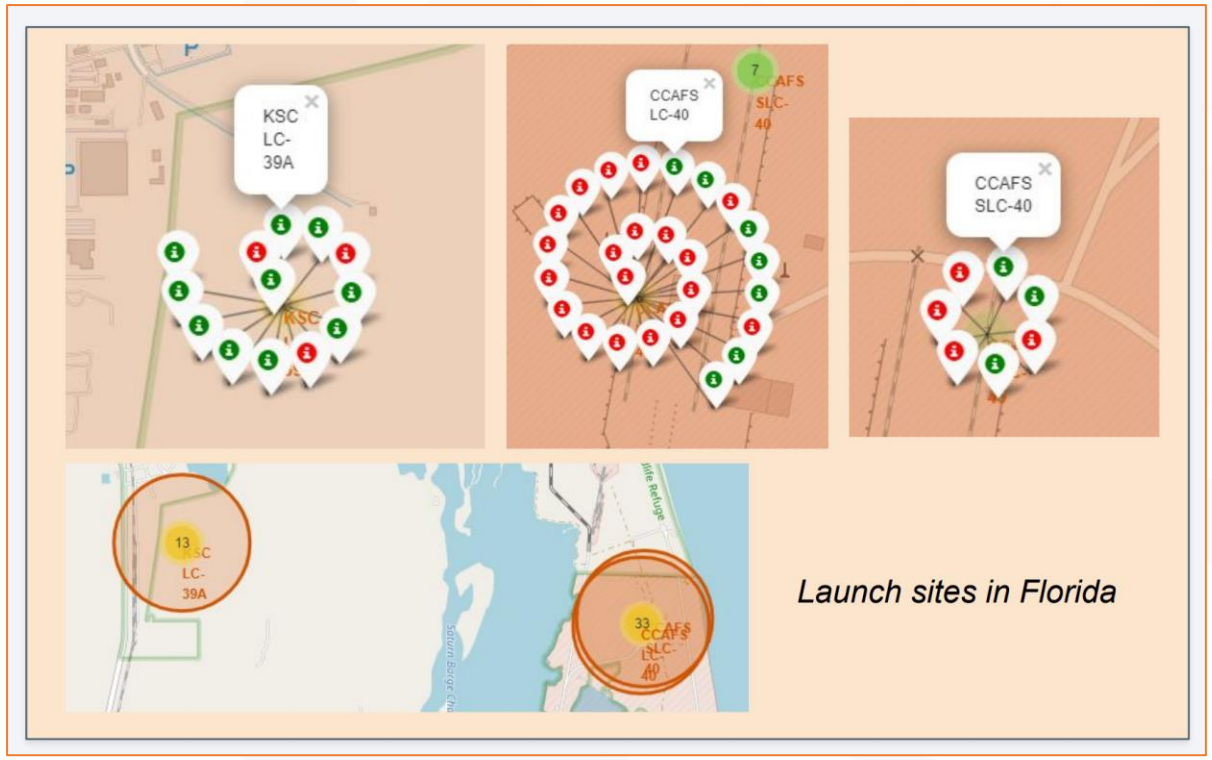
https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/6.%20Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

IBM Developer

SKILLS NETWORK

Build an Interactive Map with Folium

Locating launch site near Florida



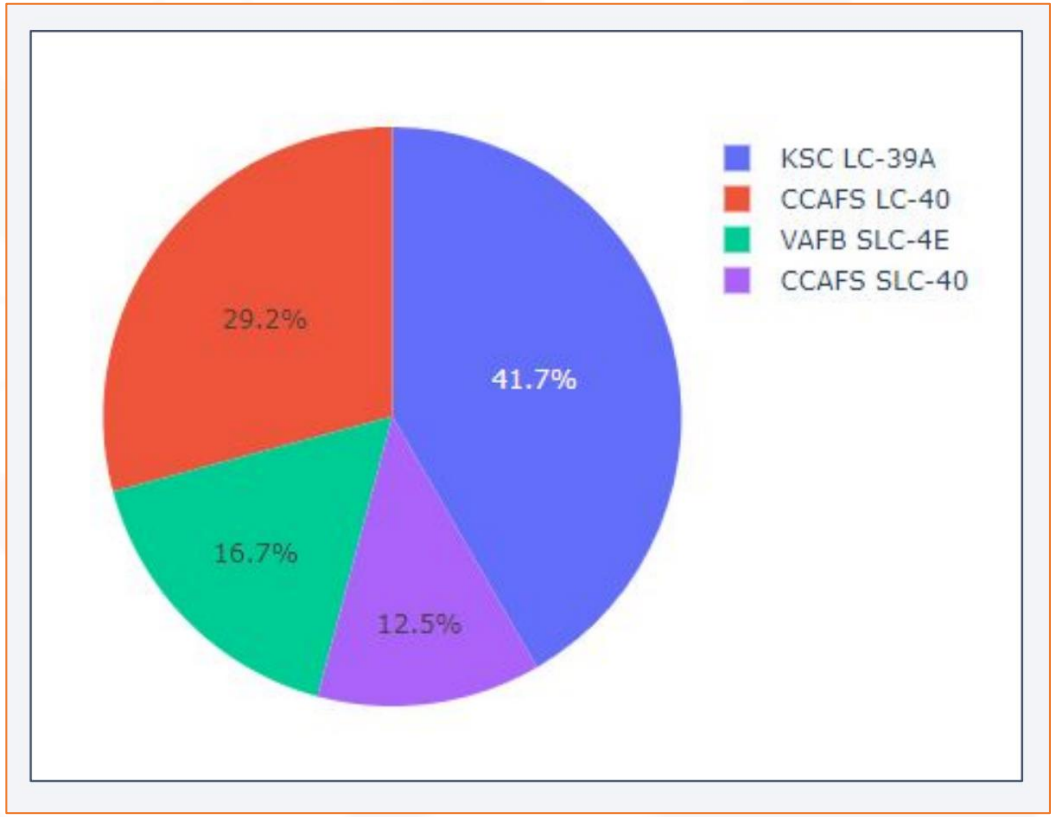
- By clicking on the marker clusters, successful landing (green) or failed landing (red) are displayed

GitHub link (click and follow)

https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/6.%20Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Building Dashboard with plotly dash

Total Success Launches By all sites



GitHub link (click and follow)

hthttps://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/2.%20Data%20Collection%20with%20Web%20Scraping.ipynb

Building Dashboard with plotly dash

Payload vs. Launch Outcome Scatter Plot

GitHub link (click and follow)

hthttps://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/2.%20Data%20Collection%20with%20Web%20Scraping.ipynb

Predictive Analysis (Classification)

Train, test data split and hyperparameter tuning

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

```
In [10]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

```
In [11]: Y_test.shape
Out[11]: (18,)
```

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
In [12]: parameters ={'C':[0.01,0.1,1],
                      'penalty':['l2'],
                      'solver':['lbfgs']}
```

```
In [14]: 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)
Out[14]: GridSearchCV(cv=10, estimator=LogisticRegression(),
                      param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                                  'solver': ['lbfgs']})
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
In [15]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
         print("accuracy :",logreg_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
         accuracy : 0.8464285714285713
```

GitHub link (click and follow)

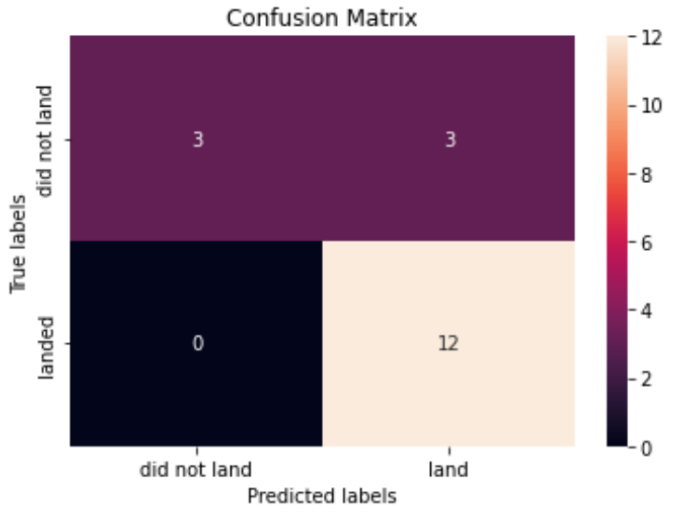
https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/8.%20%20Predictive%20Analysis%20(Classification).ipynb

Predictive Analysis (Classification)

Confusion matrix

Lets look at the confusion matrix:

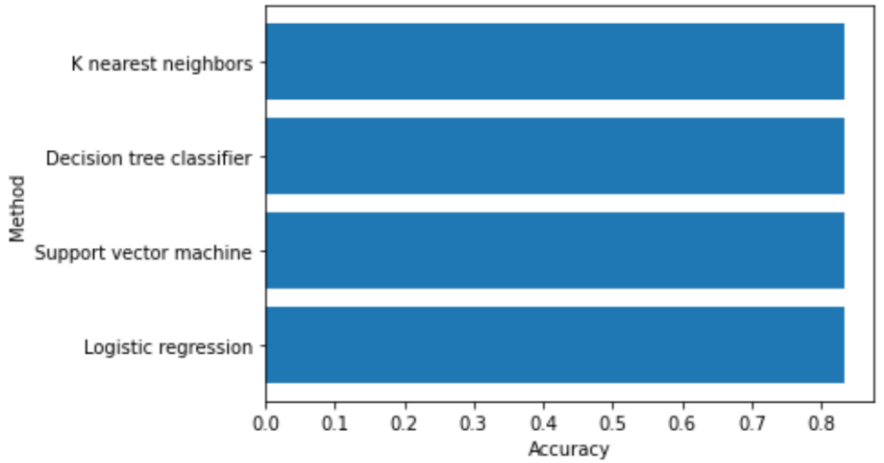
In [17]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Model selection

In [34]:
```
import numpy as np
import matplotlib.pyplot as plt

plt.barh(methods, accuracy)
plt.xlabel('Accuracy')
plt.ylabel('Method')
plt.show()
```



GitHub link (click and follow)

https://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project/blob/main/8.%20%20Predictive%20Analysis%20(Classification).ipynb

Discussion

- As the number of flights increased, the success rate increased, and recently it has exceeded 80%
- Orbital types SSO, HEO, GEO, and ES-L1 have the highest success rate (100%)
- The launch site is close to railways, highways, and coastline, but far from cities
- KSLC-39A has the highest number of launch successes and the highest success rate among all sites
- The launch success rate of low weighted payloads is higher than that of heavy weighted payloads
- Models used in this project predicted the landing of first launch with predicted 83.33%

Special Insight

- Models used in this project predicted the landing of first launch with 83.33% accuracy . However, if we take a closer look only 50% of successful landings are accurately predicted. This is reflected in all models.
- The reason behind this could come from the fact that the data is biased toward unsuccessful landing. Y_train contains 24 successful and 48 unsuccessful data counts.
- The bias can be improved by oversampling (SMOTE)
- Doing SMOTE may also increase the model accuracy

CONCLUSION

Models used in this project predicted the landing of first launch with predicted 83.33%. Therefore; there is plenty of room for success if another company decides to with SpaceX

APPENDIX

GitHub link (click and follow)

hthttps://github.com/BayeTechis/Data-Science-and-Machine-Learning-Capstone-Project