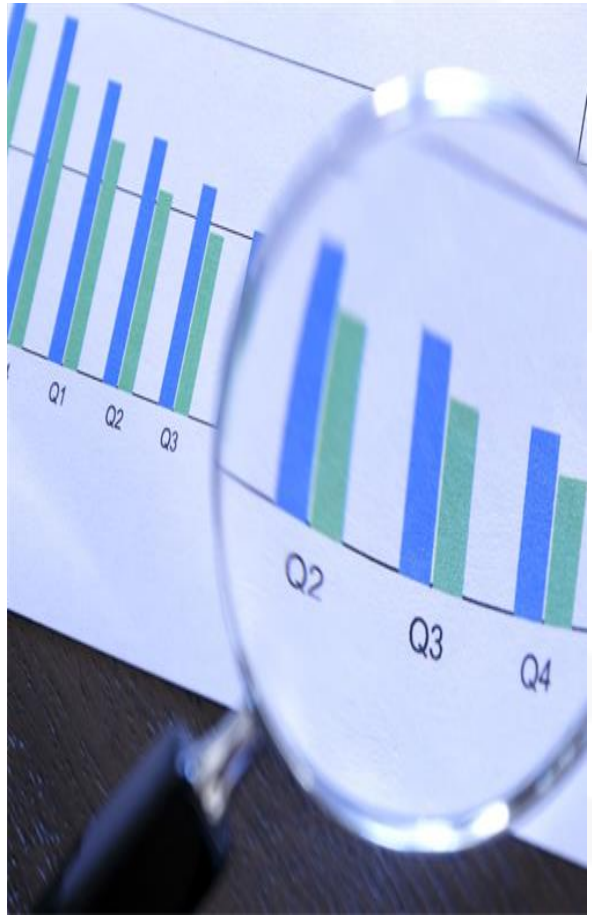


Analyze spacex operation.

JOSE ANDREILDO JANUARIO FERNANDES

23/07/2022

OUTLINE

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

EXECUTIVE SUMMARY

- Summary of methodologies
- -Data Collection through API
- -Data Collection with Web Scraping
- -Data Wrangling
- -Exploratory Data Analysis with SQL
- -Exploratory Data Analysis with Data Visualization
- -Interactive Visual Analytics with Folium
- -Machine Learning Prediction

INTRODUCTION

- The company Space X set out to create a new way of launching rockets, making the engine stage reusable. This significantly reduced the cost. For this, several launches for tests were carried out. This study aims to show how the launches were and how we can use big data and artificial intelligence tools to understand and improve the efficiency of the process.

METHODOLOGY

- Data was collected using API (data SpaceX) and web scraping (wikipedia).
- Perform exploratory data analysis (EDA) using SQL
- Data visualisation using Folium.
- Predictive analysis

Data was collected using API (data SpaceX) and web scraping (wikipedia)

- We use pandas and numpy library

```
# Requests allows us to make HTTP requests which we will use to get data from an API
import requests
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a
import numpy as np
# Datetime is a library that allows us to represent dates
import datetime

# Setting this option will print all collumns of a dataframe
pd.set_option('display.max_columns', None)
# Setting this option will print all of the data in a feature
pd.set_option('display.max_colwidth', None)
```

- We use the API (spacex) and Wikipedia.

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

```
response = requests.get(spacex_url)
```

Check the content of the response

```
print(response.content)
```

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

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

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

Perform exploratory data analysis (EDA) using SQL

- We download the analysis database and deal with sql.

Let us first load the SQL extension and establish a connection with the database

```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

```
import csv, sqlite3

con = sqlite3.connect("my_data1.db")
cur = con.cursor()
```

```
!pip install -q pandas==1.1.5
```

```
%sql sqlite:///my_data1.db
```

```
'Connected: @my_data1.db'
```

```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/data/Spac
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

- With SQL, we studied about the release dates and locations, the load carried, and the results.

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

payloadmass	1
619967	2010-06-04

1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
2	Success	F9 v1.1 B1013	CCAFS LC-40
3	Success	F9 v1.1 B1014	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40
4	Success	F9 v1.1 B1016	CCAFS LC-40
6	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

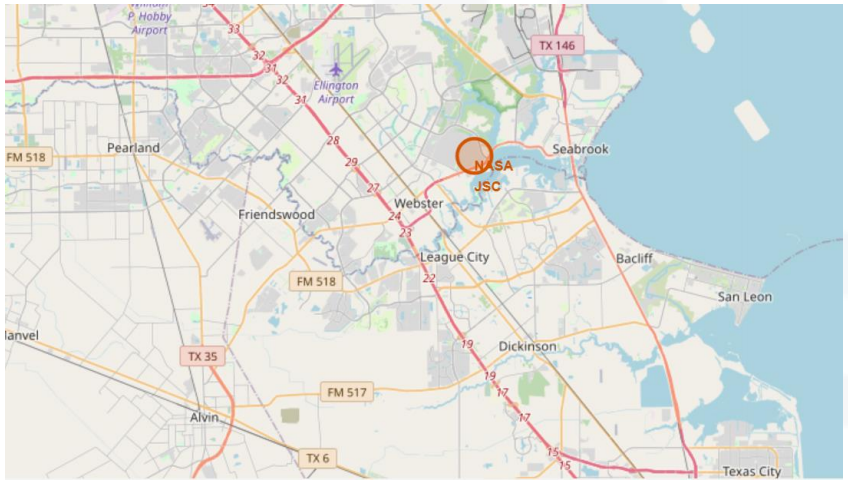
Data visualisation using Folium.

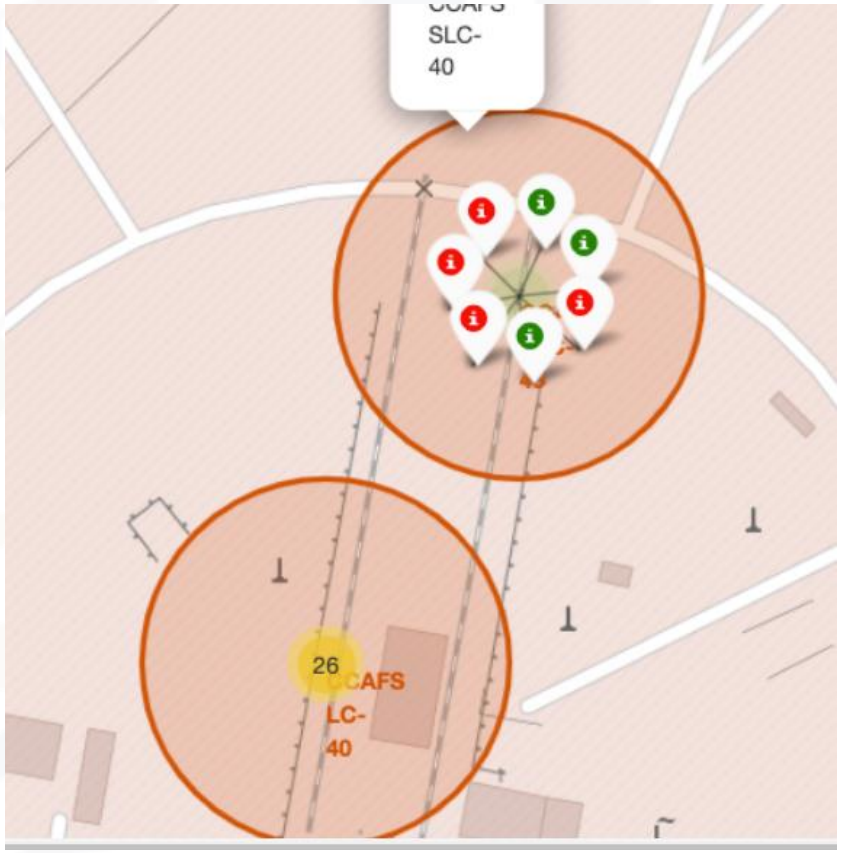
- Defined circle and map coordinates.

```
# Start location is NASA Johnson Space Center
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

We could use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate. For example,

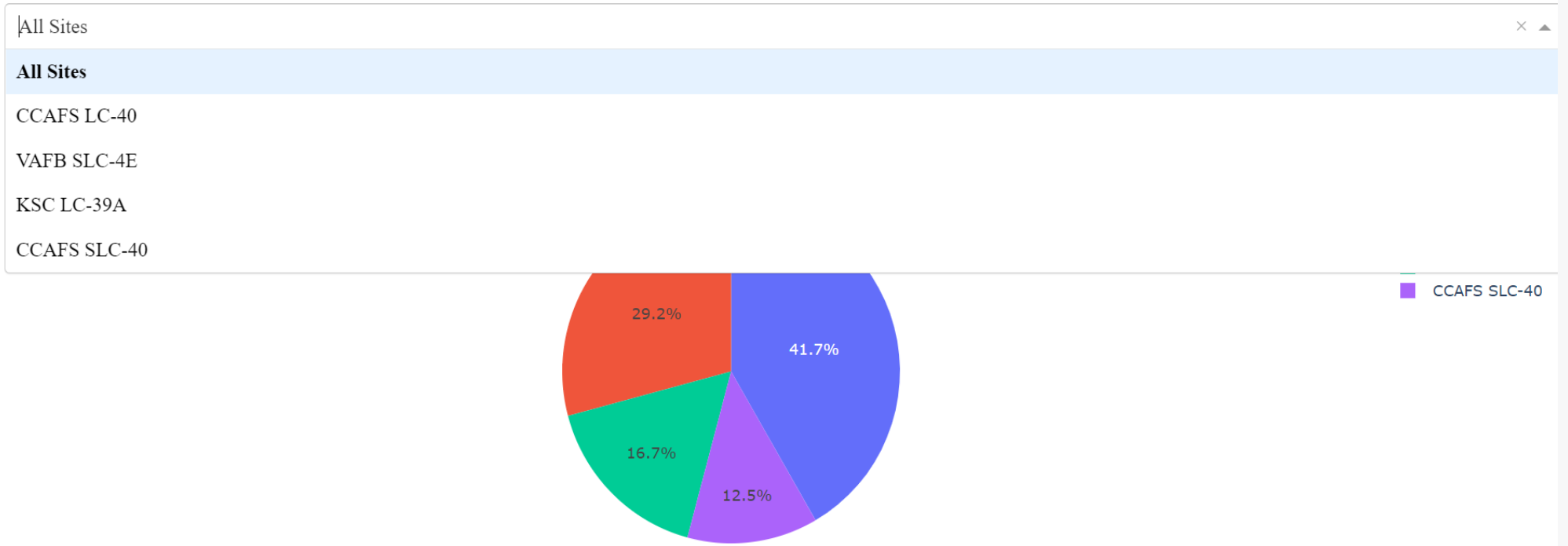
```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```





Data visualisation using Folium.

- Dashboard.



Predictive analysis

-

- The model's accuracy was verified and the confusion matrix was plotted.

```
In [18]: 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
```

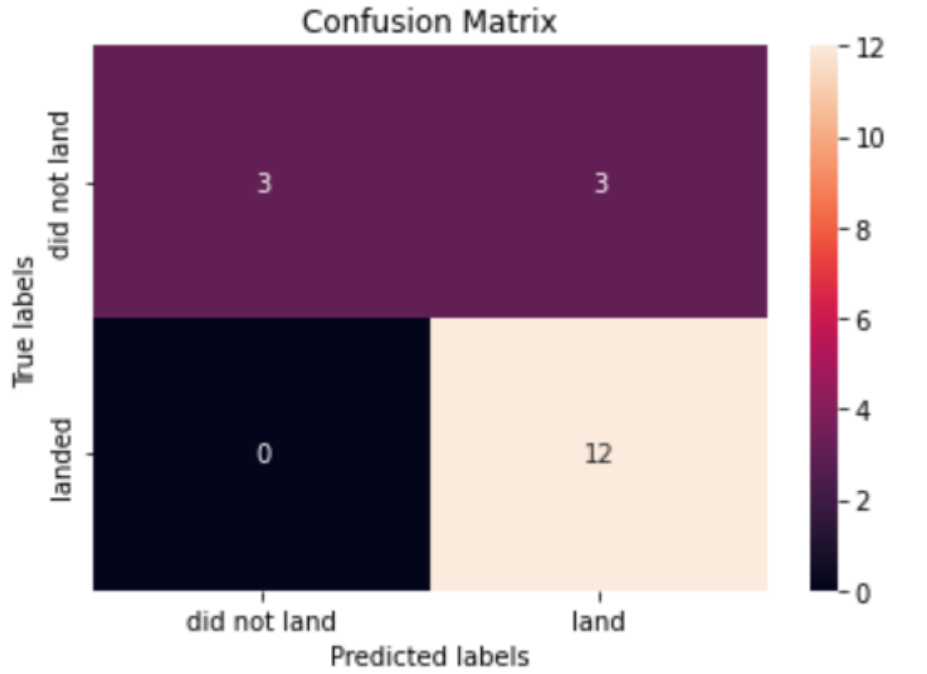
TASK 5

Calculate the accuracy on the test data using the method `score`:

```
In [19]: logreg_cv.score(X_test, Y_test)
```

```
Out[19]: 0.8333333333333334
```

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



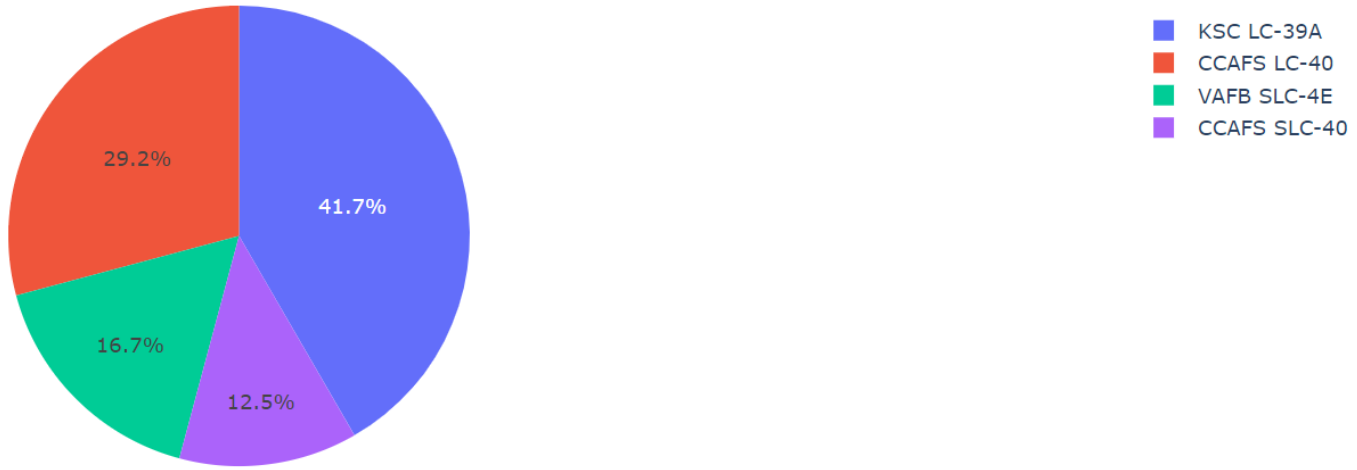
RESULTS

- EDA
- Dashboard production.
- Predictive analytics with machine learning

DASHBOARD

All Sites

Success Count for all launch sites



DASHBOARD


Payload range (Kg):
0
1k
2k
3k
4k
5k
6k
7k
8k
9k
10k
Success count on Payload mass for all sites
class
1
0.8
0.6
0.4
0.2
0
0
2k
4k
6k
8k
10k
Payload Mass (kg)
Booster Version Category
v1.0
v1.1
FT
B4
B5