Data Science For Space Race

Ronald Yu
2023.4.15

Outline

- Introduction
- Methodology
- Results
- Conclusion

Introduction

In this capstone, we will take the role of a data scientist working for a new rocket company. Space Y that would like to compete with SpaceX founded by Billionaire industrialist Allon Musk. My job is to determine the price of each launch. I will do this by gathering information about Space X and creating dashboards for my team. I will also determine if SpaceX will reuse the first stage. Instead of using rocket science to determine if the first stage will land successfully, I will train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

Methodology

- Data Collection API
- Data Wrangling
- Exploratory Analysis Using SQL
- Exploratory Analysis Using Pandas and Matplotlib
- Interactive Visual Analytics with Folium lab
- Build an Interactive Dashboard with Ploty Dash
- Machine Learning Prediction

Data Collection

- I will use the Python BeautifulSoup package to web scrape some HTML tables that contain valuable Falcon 9 launch records. Then I need to parse the data from those tables and convert them into a Pandas data frame for further visualization and analysis.

requesting rocket launch data from SpaceX API with URL

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

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

Check the content of the response

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

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/wat

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

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

use the API again to get information about the launches using the IDs given for each launch

```
# Lets take a subset of our dataframe keeping only the features we want and the flight_number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multip
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also 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])

# We also want to 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

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Data Wrangling

- We would like landing outcomes to be converted to Classes y. y. (either 0 or 1). 0 is a bad outcome, that is, the booster did not land. 1 is a good outcome, that is, the booster did land. The variable Y will represent the classification variable that represents the outcome of each launch.

Calculate the number of launches on each site

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()

CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Each launch aims to an dedicated orbit, and here are some common orbit types:

Calculate the number and occurrence of each orbit

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

GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

Calculate the number and occurence of mission outcome per orbit type

```
# 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
```

Create a landing outcome label from Outcome column

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class= []
for x in df['Outcome']:
    if x 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

```
df['Class']=landing_class
df[['Class']].head(8)
```

Exploratory Analysis Using SQL

List the total number of successful and failure mission outcomes

```
[13]: %sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;

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

[13]:

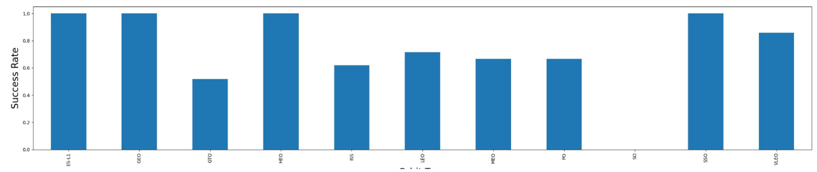
missionoutcomes
1
98
1
1

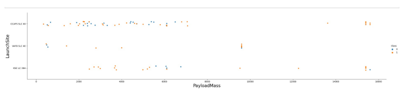
List the names of the booster_versions which have carried the maximum payload mass

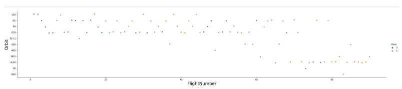
```
%sql SELECT BOOSTER_VERSION,PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FRO

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

Booster_Version	PAYLOAD_MASS__KG_
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

Exploratory Analysis Using Pandas and Matplotlib

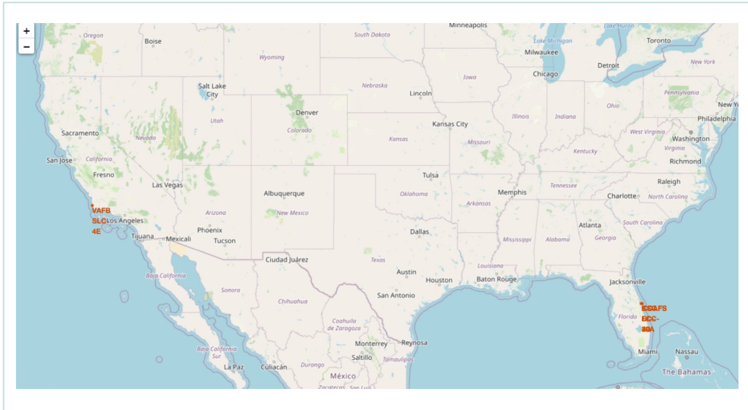






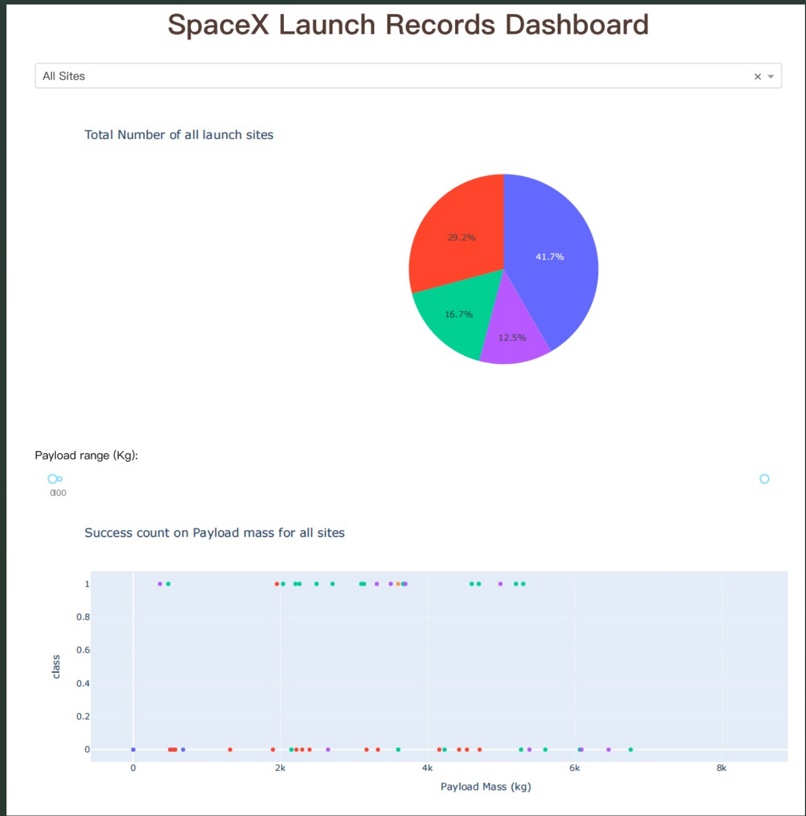
- Most Launches are Launched from CCAFS-SLC-40
- CCAFS SLC 40 has more higher payload launches
- GEO,HEO & ES-L1,SS) have high success rate

Interactive Visual Analytics with Folium lab



- Launch sites are in close proximity to coastline
- Launch sites are in close proximity to highways, which allows for easily transport required people and property.
- Launch sites are in close proximity to railways, which allows transport for heavy cargo.
- Launch sites are not in close proximity to cities, which minimizes danger to population dense areas.

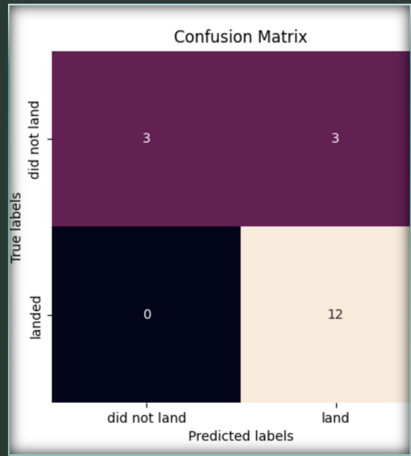
Interactive Dashboard with Ploty Dash

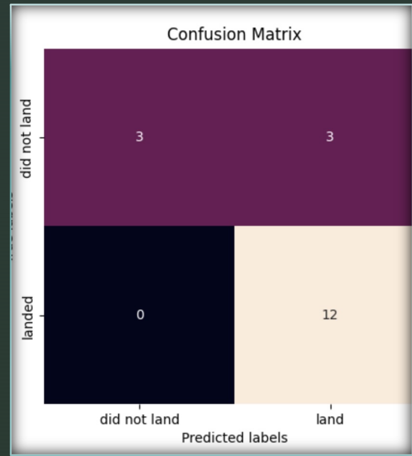


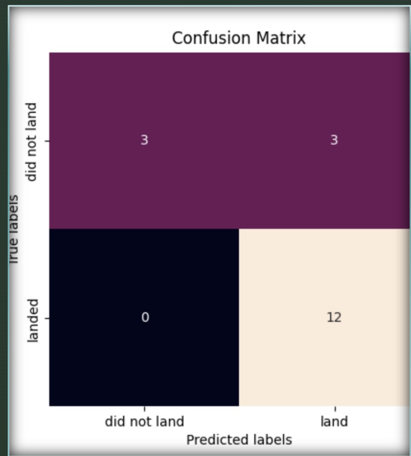
Machine Learning Prediction

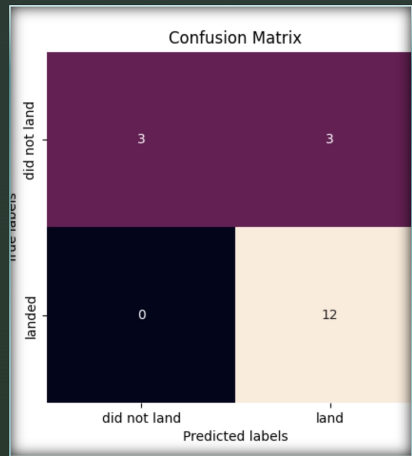
```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.8333333333333334
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```









Conclusion

- The success of a mission can be decidedd by factors like launch site, the orbit and especially the number of previous launches.
- The orbits GEO, HEO, SSO, ES-L1 have relatively high success rate.
- Depending on the orbits, the payload mass influence the success of a mission. In general, low weighted payloads perform better than the heavy weighted payloads.
- The Decision Tree has a better train accuracy in this dataset.