Space X Falcon 9 First Stage Landing Prediction

Lab 2: Data wrangling

Estimated time needed: **60** minutes

In this lab, we will perform some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, `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 on a drone ship `False ASDS` means the mission outcome was unsuccessfully landed on a drone ship.

In this lab we will mainly convert those outcomes into Training Labels with `1` means the booster successfully landed `0` means it was unsuccessful.

Falcon 9 first stage will land successfully

Several examples of an unsuccessful landing are shown here:

Objectives

Perform exploratory Data Analysis and determine Training Labels

- Exploratory Data Analysis
- Determine Training Labels

Install the below libraries

In [2]:
```
!pip install pandas
!pip install numpy
```

```
Collecting pandas
  Downloading pandas-2.3.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014
_x86_64.whl.metadata (91 kB)
Collecting numpy>=1.26.0 (from pandas)
  Downloading numpy-2.3.1-cp312-cp312-manylinux_2_28_x86_64.whl.metadata
(62 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/py
thon3.12/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/s
ite-packages (from pandas) (2024.2)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Downloading pandas-2.3.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl (12.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.0/12.0 MB 170.8 MB/s eta 0:
00:00
Downloading numpy-2.3.1-cp312-cp312-manylinux_2_28_x86_64.whl (16.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.6/16.6 MB 186.8 MB/s eta 0:
00:00
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: tzdata, numpy, pandas
Successfully installed numpy-2.3.1 pandas-2.3.0 tzdata-2025.2
Requirement already satisfied: numpy in /opt/conda/lib/python3.12/site-pac
kages (2.3.1)
```

Import Libraries and Define Auxiliary Functions

We will import the following libraries.

In [3]:
```
# Pandas is a software library written for the Python programming languag
import pandas as pd
#NumPy is a library for the Python programming language, adding support f
import numpy as np
```

Data Analysis

Load Space X dataset, from last section.

In [4]:
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdom
df.head(10)
```

Out[4]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocear
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocear
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocear
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None

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

In [5]:

```
df.isnull().sum()/len(df)*100
```

Out[5]:

```
FlightNumber       0.000000
Date               0.000000
BoosterVersion     0.000000
PayloadMass        0.000000
Orbit              0.000000
LaunchSite         0.000000
Outcome            0.000000
Flights            0.000000
GridFins           0.000000
Reused             0.000000
Legs               0.000000
LandingPad        28.888889
Block              0.000000
ReusedCount        0.000000
Serial             0.000000
Longitude          0.000000
Latitude           0.000000
dtype: float64
```

Identify which columns are numerical and categorical:

In [6]:
```
df.dtypes
```

Out[6]:
```
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
```

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

In [7]:
```
# Get count of launches per site
launch_counts = df['LaunchSite'].value_counts()

# Optional: Convert to DataFrame for better formatting
launch_counts_df = launch_counts.reset_index()
launch_counts_df.columns = ['Launch Site', 'Count']

print("Launch Counts by Site:")
print(launch_counts_df)
```

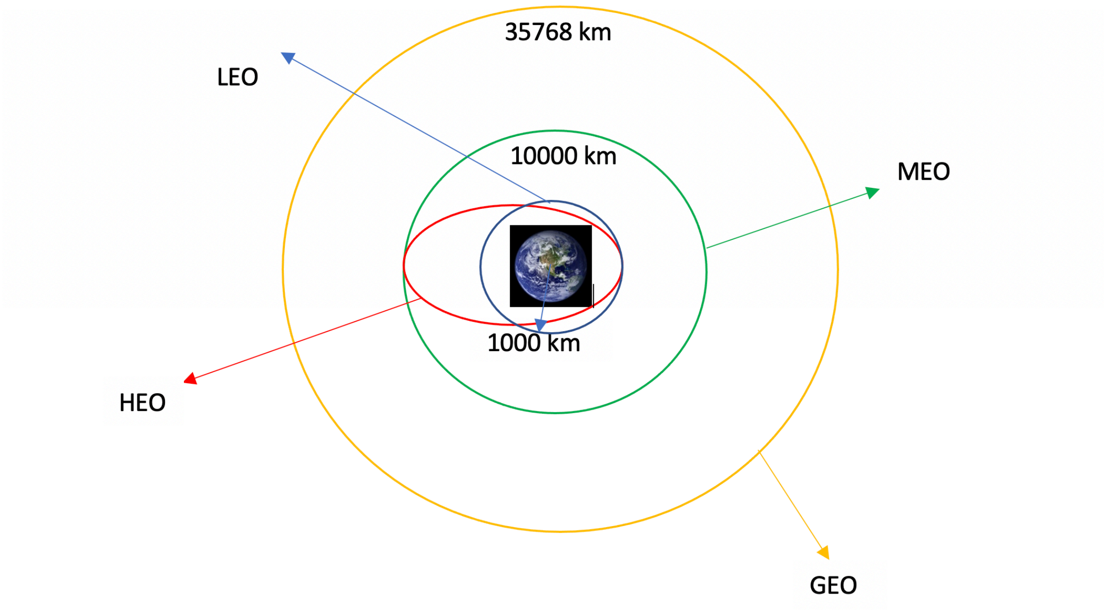
```
Launch Counts by Site:
    Launch Site  Count
0  CCAFS SLC 40     55
1    KSC LC 39A     22
2    VAFB SLC 4E     13
```

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

- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO [1].

- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation[2].
- **GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website [3] .
- **SSO (or SO)**: It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .
- **ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth [5] .
- **HEO** A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth [6].
- **ISS** A modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies: NASA (United States), Roscosmos (Russia), JAXA (Japan), ESA (Europe), and CSA (Canada) [7]
- **MEO** Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit. These are "most commonly at 20,200 kilometers (12,600 mi), or 20,650 kilometers (12,830 mi), with an orbital period of 12 hours [8]
- **HEO** Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi) [9]
- **GEO** It is a circular geosynchronous orbit 35,786 kilometres (22,236 miles) above Earth's equator and following the direction of Earth's rotation [10]
- **PO** It is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited (usually a planet such as the Earth [11]

some are shown in the following plot:



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 [8]:
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

Out[8]:
```
Orbit
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
HEO       1
ES-L1     1
SO        1
GEO       1
Name: count, dtype: int64
```

TASK 3: Calculate the number and occurence of mission outcome of the orbits

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 [9]:
```
# landing_outcomes = values on Outcome column
# Count occurrences of each landing outcome
landing_outcomes = df['Outcome'].value_counts()

# Optional: Convert to DataFrame
```

```
landing_outcomes_df = landing_outcomes.reset_index()
landing_outcomes_df.columns = ['Outcome', 'Count']

print("Landing Outcomes:")
print(landing_outcomes_df)
```

```
Landing Outcomes:
        Outcome  Count
0     True ASDS     41
1     None None     19
2     True RTLS     14
3    False ASDS      6
4    True Ocean      5
5   False Ocean      2
6     None ASDS      2
7    False RTLS      1
```

`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 [10]:
```
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 [19]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Out[19]:
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

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 [23]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
```

```
# Create landing_class list using list comprehension
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['O

# Optional: Add it back to DataFrame as a new column
df['landing_class'] = landing_class

# Print first few values to verify
print("First 5 landing_class values:", landing_class[:5])
```

```
First 5 landing_class values: [0, 0, 0, 0, 0]
```

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 [24]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

Out[24]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

In [25]:
```
df.head(5)
```

Out[25]:

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

We can use the following line of code to determine the success rate:

In [26]:
```
df["Class"].mean()
```

Out[26]:
```
np.float64(0.6666666666666666)
```

We can now export it to a CSV for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

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

Authors

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Nayef Abou Tayoun is a Data Scientist at IBM and pursuing a Master of Management in Artificial intelligence degree at Queen's University.