# SpaceX Falcon 9 first stage Landing Prediction

## Lab 1: Collecting the data

Estimated time needed: **45** minutes

In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of 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 information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this lab, you will collect and make sure the data is in the correct format from an API. The following is an example of a successful and launch.


Several examples of an unsuccessful landing are shown here:


Most unsuccessful landings are planned. Space X performs a controlled landing in the oceans.

## Objectives

In this lab, you will make a get request to the SpaceX API. You will also do some basic data wrangling and formating.

- Request to the SpaceX API
- Clean the requested data

---

## Import Libraries and Define Auxiliary Functions

We will import the following libraries into the lab

In [1]:
```
# Requests allows us to make HTTP requests which we will use to get data fro
import requests
# Pandas is a software library written for the Python programming language f
import pandas as pd
# NumPy is a library for the Python programming language, adding support for
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)
```

Below we will define a series of helper functions that will help us use the API to extract information using identification numbers in the launch data.

From the `rocket` column we would like to learn the booster name.

In [2]:
```
# Takes the dataset and uses the rocket column to call the API and append th
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [3]:
```
# Takes the dataset and uses the launchpad column to call the API and append
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [4]:
```
# Takes the dataset and uses the payloads column to call the API and append
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+lo
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

In [5]:
```
# Takes the dataset and uses the cores column to call the API and append the
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landir
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
```

```
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

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

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

Check the content of the response

In [8]:
```
type(response.content)
```

Out[8]:
```
bytes
```

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:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomai
```

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

In [10]:
```
response.status_code
```

Out[10]:
```
200
```

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

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
json_data = response.json()
data = pd.json_normalize(json_data)
```

Using the dataframe `data` print the first 5 rows

In [12]:
```
# Get the head of the dataframe
print(data.head())
```

```
       static_fire_date_utc  static_fire_date_unix    net  window  \
0  2006-03-17T00:00:00.000Z           1.142554e+09  False     0.0
1                      None                    NaN  False     0.0
2                      None                    NaN  False     0.0
3  2008-09-20T00:00:00.000Z           1.221869e+09  False     0.0
4                      None                    NaN  False     0.0

                     rocket success  \
0  5e9d0d95eda69955f709d1eb   False
1  5e9d0d95eda69955f709d1eb   False
2  5e9d0d95eda69955f709d1eb   False
3  5e9d0d95eda69955f709d1eb    True
4  5e9d0d95eda69955f709d1eb    True


failures  \
0                                                 [{'time': 33, 'altitude': No
ne, 'reason': 'merlin engine failure'}]
1            [{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation
leading to premature engine shutdown'}]
2  [{'time': 140, 'altitude': 35, 'reason': 'residual stage-1 thrust led to c
ollision between stage 1 and stage 2'}]
3
[]
4
[]


details  \
0
Engine failure at 33 seconds and loss of vehicle
1  Successful first stage burn and transition to second stage, maximum altitu
de 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit,
Failed to recover first stage
2
Residual stage 1 thrust led to collision between stage 1 and stage 2
3                        Ratsat was carried to orbit on the first successful
orbital launch of any privately funded and developed, liquid-propelled carrie
r rocket, the SpaceX Falcon 1
4
None

  crew ships capsules                                                payloads
\
0   []    []       []                             [5eb0e4b5b6c3bb0006eeb1e1]
1   []    []       []                             [5eb0e4b6b6c3bb0006eeb1e2]
2   []    []       []  [5eb0e4b6b6c3bb0006eeb1e3, 5eb0e4b6b6c3bb0006eeb1e4]
3   []    []       []                             [5eb0e4b7b6c3bb0006eeb1e5]
4   []    []       []                             [5eb0e4b7b6c3bb0006eeb1e6]

                  launchpad  flight_number         name  \
0  5e9e4502f5090995de566f86              1    FalconSat
1  5e9e4502f5090995de566f86              2      DemoSat
2  5e9e4502f5090995de566f86              3  Trailblazer
3  5e9e4502f5090995de566f86              4       RatSat
4  5e9e4502f5090995de566f86              5     RazakSat
```

```
                   date_utc   date_unix                 date_local  \
0  2006-03-24T22:30:00.000Z  1143239400  2006-03-25T10:30:00+12:00
1  2007-03-21T01:10:00.000Z  1174439400  2007-03-21T13:10:00+12:00
2  2008-08-03T03:34:00.000Z  1217734440  2008-08-03T15:34:00+12:00
3  2008-09-28T23:15:00.000Z  1222643700  2008-09-28T11:15:00+12:00
4  2009-07-13T03:35:00.000Z  1247456100  2009-07-13T15:35:00+12:00

  date_precision  upcoming  \
0           hour     False
1           hour     False
2           hour     False
3           hour     False
4           hour     False


cores  \
0  [{'core': '5e9e289df35918033d3b2623', 'flight': 1, 'gridfins': False, 'leg
s': False, 'reused': False, 'landing_attempt': False, 'landing_success': Non
e, 'landing_type': None, 'landpad': None}]
1  [{'core': '5e9e289ef35918416a3b2624', 'flight': 1, 'gridfins': False, 'leg
s': False, 'reused': False, 'landing_attempt': False, 'landing_success': Non
e, 'landing_type': None, 'landpad': None}]
2  [{'core': '5e9e289ef3591814873b2625', 'flight': 1, 'gridfins': False, 'leg
s': False, 'reused': False, 'landing_attempt': False, 'landing_success': Non
e, 'landing_type': None, 'landpad': None}]
3  [{'core': '5e9e289ef3591855dc3b2626', 'flight': 1, 'gridfins': False, 'leg
s': False, 'reused': False, 'landing_attempt': False, 'landing_success': Non
e, 'landing_type': None, 'landpad': None}]
4  [{'core': '5e9e289ef359184f103b2627', 'flight': 1, 'gridfins': False, 'leg
s': False, 'reused': False, 'landing_attempt': False, 'landing_success': Non
e, 'landing_type': None, 'landpad': None}]

   auto_update    tbd launch_library_id                        id  \
0         True  False              None  5eb87cd9ffd86e000604b32a
1         True  False              None  5eb87cdaffd86e000604b32b
2         True  False              None  5eb87cdbffd86e000604b32c
3         True  False              None  5eb87cdbffd86e000604b32d
4         True  False              None  5eb87cdcffd86e000604b32e

  fairings.reused fairings.recovery_attempt fairings.recovered fairings.ships
\
0           False                     False              False             []
1           False                     False              False             []
2           False                     False              False             []
3           False                     False              False             []
4           False                     False              False             []

                                links.patch.small  \
0  https://images2.imgbox.com/94/f2/NN6Ph45r_o.png
1  https://images2.imgbox.com/f9/4a/ZboXReNb_o.png
2  https://images2.imgbox.com/6c/cb/na1tzhHs_o.png
3  https://images2.imgbox.com/95/39/sRqN7rsv_o.png
4  https://images2.imgbox.com/ab/5a/Pequxd5d_o.png

                                links.patch.large links.reddit.campaign  \
```

```
0  https://images2.imgbox.com/5b/02/QcxHUb5V_o.png                  None
1  https://images2.imgbox.com/80/a2/bkWotCIS_o.png                  None
2  https://images2.imgbox.com/4a/80/k1oAkY0k_o.png                  None
3  https://images2.imgbox.com/a3/99/qswRYzE8_o.png                  None
4  https://images2.imgbox.com/92/e4/7Cf6MLY0_o.png                  None

  links.reddit.launch links.reddit.media links.reddit.recovery  \
0                None               None                  None
1                None               None                  None
2                None               None                  None
3                None               None                  None
4                None               None                  None

  links.flickr.small links.flickr.original  \
0                 []                    []
1                 []                    []
2                 []                    []
3                 []                    []
4                 []                    []


links.presskit  \
0
None
1
None
2
None
3
None
4  http://www.spacex.com/press/2012/12/19/spacexs-falcon-1-successfully-deliv
ers-razaksat-satellite-orbit

                                links.webcast links.youtube_id  \
0  https://www.youtube.com/watch?v=0a_00nJ_Y88      0a_00nJ_Y88
1  https://www.youtube.com/watch?v=Lk4zQ2wP-Nc      Lk4zQ2wP-Nc
2  https://www.youtube.com/watch?v=v0w9p3U8860      v0w9p3U8860
3  https://www.youtube.com/watch?v=dLQ2tZEH6G0      dLQ2tZEH6G0
4  https://www.youtube.com/watch?v=yTaIDooc8Og      yTaIDooc8Og

                                                                links.artic
le  \
0  https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-launch.ht
ml
1      https://www.space.com/3590-spacex-falcon-1-rocket-fails-reach-orbit.ht
ml
2       http://www.spacex.com/news/2013/02/11/falcon-1-flight-3-mission-summa
ry
3                                        https://en.wikipedia.org/wiki/Rats
at
4                       http://www.spacex.com/news/2013/02/12/falcon-1-flight
-5

                                     links.wikipedia  fairings
0                  https://en.wikipedia.org/wiki/DemoSat       NaN
1                  https://en.wikipedia.org/wiki/DemoSat       NaN
```

```
2  https://en.wikipedia.org/wiki/Trailblazer_(satellite)       NaN
3                  https://en.wikipedia.org/wiki/Ratsat       NaN
4                https://en.wikipedia.org/wiki/RazakSAT       NaN
```

You will notice that a lot of the data are IDs. For example the rocket column has no information about the rocket just an identification number.

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

In [13]:
```
# Lets take a subset of our dataframe keeping only the features we want and
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'c

# We will remove rows with multiple cores because those are falcon rockets w
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 sing
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 extra
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)]
```

- From the `rocket` we would like to learn the booster name
- From the `payload` we would like to learn the mass of the payload and the orbit that it is going to
- From the `launchpad` we would like to know the name of the launch site being used, the longitude, and the latitude.
- From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, whether the core is reused, whether legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

The data from these requests will be stored in lists and will be used to create a new dataframe.

In [14]:
```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
```

```
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

These functions will apply the outputs globally to the above variables. Let's take a looks at `BoosterVersion` variable. Before we apply `getBoosterVersion` the list is empty:

In [15]:
```
BoosterVersion
```

Out[15]:
```
[]
```

Now, let's apply `getBoosterVersion` function method to get the booster version

In [16]:
```
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

In [17]:
```
BoosterVersion[0:5]
```

Out[17]:
```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

In [18]:
```
# Call getLaunchSite
getLaunchSite(data)
```

In [19]:
```
# Call getPayloadData
getPayloadData(data)
```

In [20]:
```
# Call getCoreData
getCoreData(data)
```

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

In [21]:
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
```

```
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

In [22]:
```
# Create a data from launch_dict
launch_df = pd.DataFrame(launch_dict)
```

Show the summary of the dataframe

In [23]:
```
# Show the head of the dataframe
launch_df.head()
```

Out[23]:

| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | I |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2006-03-24 | Falcon 1 | 20.0 | LEO | Kwajalein Atoll | None None | |
| **1** | 2 | 2007-03-21 | Falcon 1 | NaN | LEO | Kwajalein Atoll | None None | |
| **2** | 4 | 2008-09-28 | Falcon 1 | 165.0 | LEO | Kwajalein Atoll | None None | |
| **3** | 5 | 2009-07-13 | Falcon 1 | 200.0 | LEO | Kwajalein Atoll | None None | |
| **4** | 6 | 2010-06-04 | Falcon 9 | NaN | LEO | CCSFS SLC 40 | None None | |

## 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 [24]:
```
# 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 [25]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/indexing.py:1773: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

Out[25]:

| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome |
|---|---|---|---|---|---|---|---|
| **4** | 1 | 2010-06-04 | Falcon 9 | NaN | LEO | CCSFS SLC 40 | None None |
| **5** | 2 | 2012-05-22 | Falcon 9 | 525.0 | LEO | CCSFS SLC 40 | None None |
| **6** | 3 | 2013-03-01 | Falcon 9 | 677.0 | ISS | CCSFS SLC 40 | None None |
| **7** | 4 | 2013-09-29 | Falcon 9 | 500.0 | PO | VAFB SLC 4E | False Ocean |
| **8** | 5 | 2013-12-03 | Falcon 9 | 3170.0 | GTO | CCSFS SLC 40 | None None |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **89** | 86 | 2020-09-03 | Falcon 9 | 15600.0 | VLEO | KSC LC 39A | True ASDS |
| **90** | 87 | 2020-10-06 | Falcon 9 | 15600.0 | VLEO | KSC LC 39A | True ASDS |
| **91** | 88 | 2020-10-18 | Falcon 9 | 15600.0 | VLEO | KSC LC 39A | True ASDS |
| **92** | 89 | 2020-10-24 | Falcon 9 | 15600.0 | VLEO | CCSFS SLC 40 | True ASDS |
| **93** | 90 | 2020-11-05 | Falcon 9 | 3681.0 | MEO | CCSFS SLC 40 | True ASDS |

90 rows × 17 columns

# Data Wrangling

We can see below that some of the rows are missing values in our dataset.

In [26]:
```
data_falcon9.isnull().sum()
```

Out[26]:
```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        5
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

Before we can continue we must deal with these missing values. The `LandingPad` column will retain None values to represent when landing pads were not used.

## Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [27]:
```
# Calculate the mean value of PayloadMass column
data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan,data_falcon9['PayloadMass'].mean(
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/ge
neric.py:6619: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
```

You should see the number of missing values of the `PayLoadMass` change to zero.

In [28]:
```
data_falcon9.isnull().sum()
```

Out[28]:
```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

Now we should have no missing values in our dataset except for in `LandingPad`.

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.

`data_falcon9.to_csv('dataset_part_1.csv', index=False)`

In [29]:
```
data_falcon9.to_csv('dataset_part_1.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.

# Change Log

| Date (YYYY-MM-DD) | Version | Changed By | Change Description |
|---|---|---|---|
| 2020-09-20 | 1.1 | Joseph | get result each time you run |
| 2020-09-20 | 1.1 | Azim | Created Part 1 Lab using SpaceX API |
| 2020-09-20 | 1.0 | Joseph | Modified Multiple Areas |