**SpaceX Falcon 9 First Stage Landing Prediction**

**Assignment: Exploring and Preparing Data**

Estimated time needed: **70** minutes

In this assignment, 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 due to the fact that SpaceX can reuse the first stage.

In this lab, you will perform Exploratory Data Analysis and Feature Engineering.

Falcon 9 first stage will land successfully



Several examples of an unsuccessful landing are shown here:

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

**Objectives**

Perform exploratory Data Analysis and Feature Engineering using Pandas and Matplotlib

* Exploratory Data Analysis
* Preparing Data Feature Engineering

**Import Libraries and Define Auxiliary Functions**

We will import the following libraries the lab

In [1]:

*# andas 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 large collection of high-level mathematical functions to operate on these arrays*

**import** numpy **as** np

*# Matplotlib is a plotting library for python and pyplot gives us a MatLab like plotting framework. We will use this in our plotter function to plot data.*

**import** matplotlib.pyplot **as** plt

*#Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics*

**import** seaborn **as** sns

**Exploratory Data Analysis**

First, let's read the SpaceX dataset into a Pandas dataframe and print its summary

In [2]:

df**=**pd**.**read\_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset\_part\_2.csv")

*# If you were unable to complete the previous lab correctly you can uncomment and load this csv*

*# df = pd.read\_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset\_part\_2.csv')*

df**.**head(5)

Out[2]:

|  | **FlightNumber** | **Date** | **BoosterVersion** | **PayloadMass** | **Orbit** | **LaunchSite** | **Outcome** | **Flights** | **GridFins** | **Reused** | **Legs** | **LandingPad** | **Block** | **ReusedCount** | **Serial** | **Longitude** | **Latitude** | **Class** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 2010-06-04 | Falcon 9 | 6104.959412 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B0003 | -80.577366 | 28.561857 | 0 |
| **1** | 2 | 2012-05-22 | Falcon 9 | 525.000000 | LEO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B0005 | -80.577366 | 28.561857 | 0 |
| **2** | 3 | 2013-03-01 | Falcon 9 | 677.000000 | ISS | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B0007 | -80.577366 | 28.561857 | 0 |
| **3** | 4 | 2013-09-29 | Falcon 9 | 500.000000 | PO | VAFB SLC 4E | False Ocean | 1 | False | False | False | NaN | 1.0 | 0 | B1003 | -120.610829 | 34.632093 | 0 |
| **4** | 5 | 2013-12-03 | Falcon 9 | 3170.000000 | GTO | CCAFS SLC 40 | None None | 1 | False | False | False | NaN | 1.0 | 0 | B1004 | -80.577366 | 28.561857 | 0 |

First, let's try to see how the FlightNumber (indicating the continuous launch attempts.) and Payload variables would affect the launch outcome.

We can plot out the FlightNumber vs. PayloadMassand overlay the outcome of the launch. We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

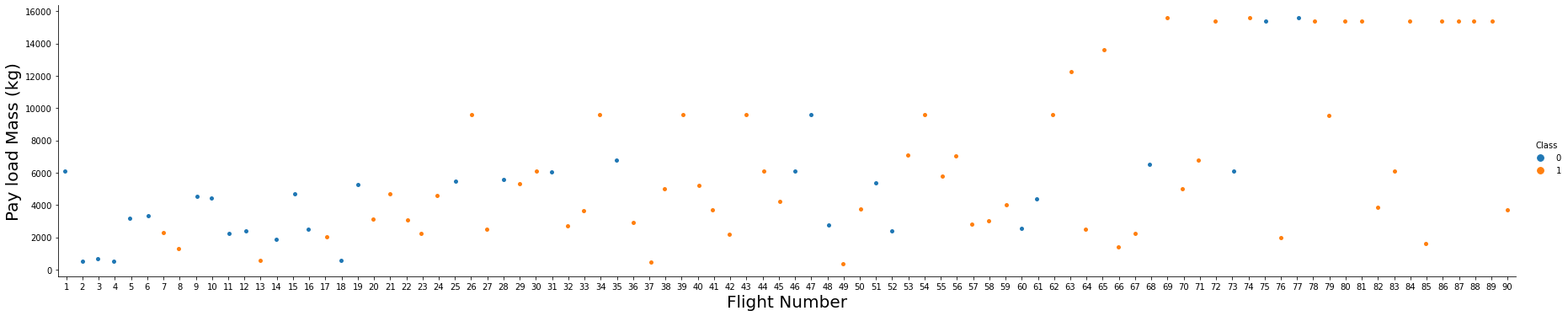
In [3]:

sns**.**catplot(y**=**"PayloadMass", x**=**"FlightNumber", hue**=**"Class", data**=**df, aspect **=** 5)

plt**.**xlabel("Flight Number",fontsize**=**20)

plt**.**ylabel("Pay load Mass (kg)",fontsize**=**20)

plt**.**show()



We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.

Next, let's drill down to each site visualize its detailed launch records.

**TASK 1: Visualize the relationship between Flight Number and Launch Site**

Use the function catplot to plot FlightNumber vs LaunchSite, set the parameter x parameter to FlightNumber,set the y to Launch Site and set the parameter hue to 'class'

In [4]:

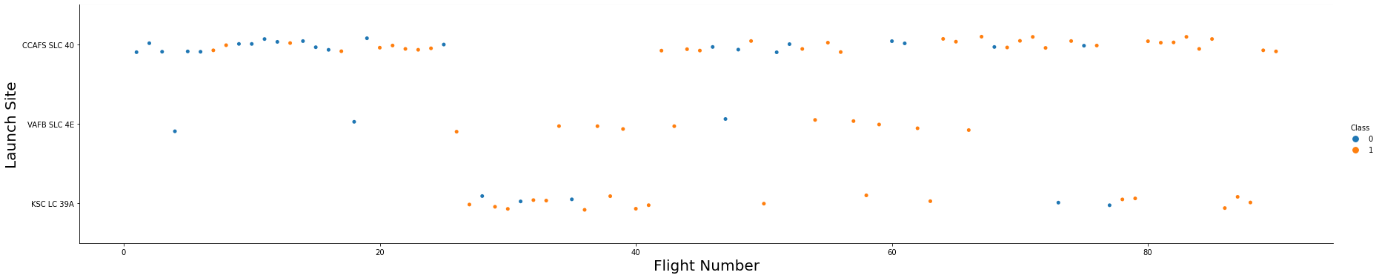
*# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value*

sns**.**catplot(x**=**'FlightNumber', y**=**'LaunchSite', hue**=**'Class', data**=**df, aspect**=**5)

plt**.**xlabel('Flight Number', fontsize**=**20)

plt**.**ylabel('Launch Site', fontsize**=**20)

plt**.**show()



Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

**Explanation:**

* The earliest flights all failed while the latest flights all succeeded.
* The CCAFS SLC 40 launch site has about a half of all launches.
* VAFB SLC 4E and KSC LC 39A have higher success rates.
* It can be assumed that each new launch has a higher rate of success.

**TASK 2: Visualize the relationship between Payload and Launch Site**

We also want to observe if there is any relationship between launch sites and their payload mass.

In [5]:

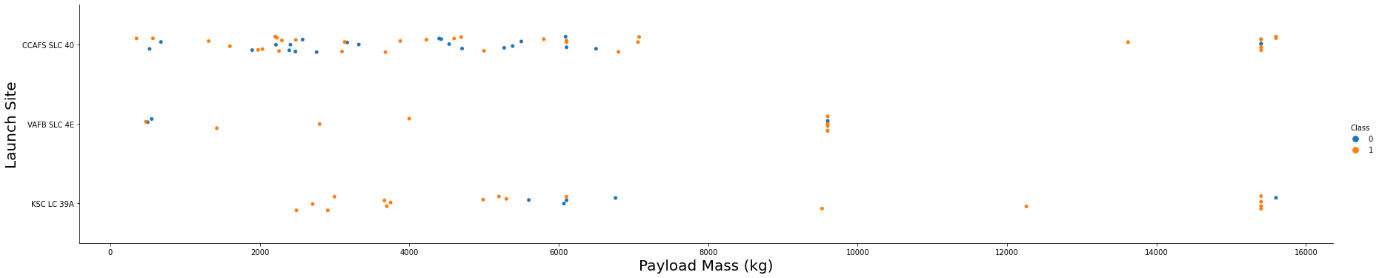
*# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value*

sns**.**catplot(x**=**'PayloadMass', y**=**'LaunchSite', hue**=**'Class', data**=**df, aspect **=** 5)

plt**.**xlabel('Payload Mass (kg)',fontsize**=**20)

plt**.**ylabel('Launch Site',fontsize**=**20)

plt**.**show()



Now try to explain any patterns you found in the Payload Vs. Launch Site scatter point chart.

**Explanation:**

* For every launch site the higher the payload mass, the higher the success rate.
* Most of the launches with payload mass over 7000 kg were successfull.
* KSC LC 39A has a 100% success rate for payload mass under 5500 kg too.

**TASK 3: Visualize the relationship between success rate of each orbit type**

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a bar chart for the sucess rate of each orbit

In [6]:

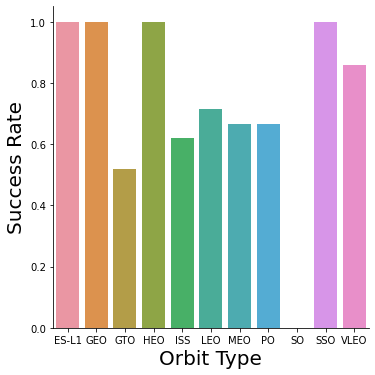
*# HINT use groupby method on Orbit column and get the mean of Class column*

sns**.**catplot(x**=** 'Orbit', y **=** 'Class', data **=** df**.**groupby('Orbit')['Class']**.**mean()**.**reset\_index(), kind **=** 'bar')

plt**.**xlabel('Orbit Type',fontsize**=**20)

plt**.**ylabel('Success Rate',fontsize**=**20)

plt**.**show()



Analyze the ploted bar chart try to find which orbits have high sucess rate.

**Explanation:**

* Orbits with 100% success rate are:
  + ES-L1
  + GEO
  + HEO
  + SSO
* Orbits with 0% success rate are:
  + SO
* Orbits with success rate between 50% and 85%:
  + GTO
  + ISS
  + LEO
  + MEO
  + PO

**TASK 4: Visualize the relationship between FlightNumber and Orbit type**

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [7]:

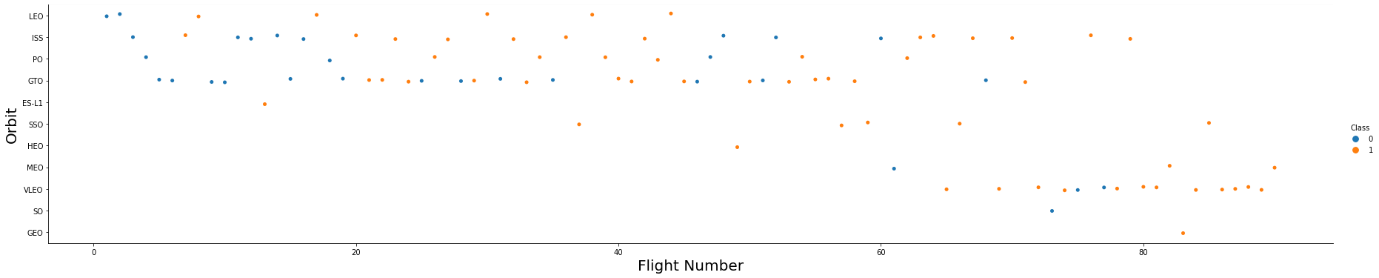
*# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value*

sns**.**catplot(x **=** 'FlightNumber', y **=** 'Orbit', hue **=** 'Class', data **=** df, aspect **=** 5)

plt**.**xlabel('Flight Number', fontsize **=** 20)

plt**.**ylabel('Orbit', fontsize **=** 20)

plt**.**show()



You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

**TASK 5: Visualize the relationship between Payload and Orbit type**

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

In [8]:

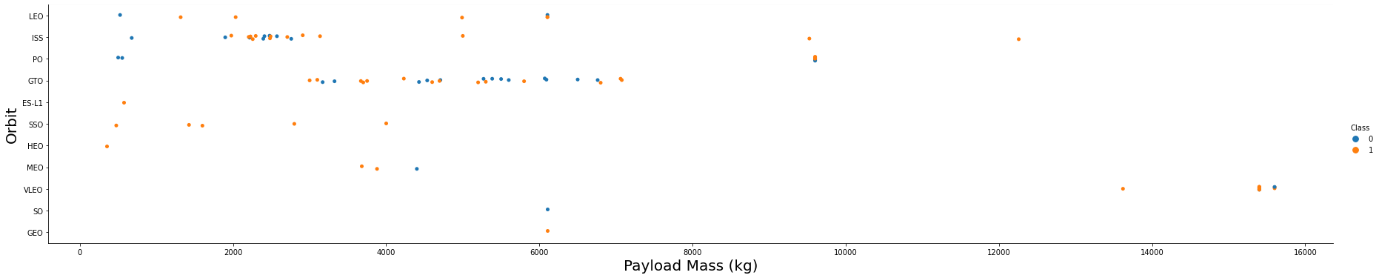
*# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value*

sns**.**catplot(x **=** 'PayloadMass', y **=** 'Orbit', hue **=** 'Class', data **=** df, aspect **=** 5)

plt**.**xlabel('Payload Mass (kg)', fontsize **=** 20)

plt**.**ylabel('Orbit', fontsize **=** 20)

plt**.**show()



You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

**TASK 6: Visualize the launch success yearly trend**

You can plot a line chart with x axis to be Year and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

In [9]:

*# A function to Extract years from the date*

year**=**[]

**def** Extract\_year(date):

**for** i **in** df["Date"]:

year**.**append(i**.**split("-")[0])

**return** year

In [10]:

*# Plot a line chart with x axis to be the extracted year and y axis to be the success rate*

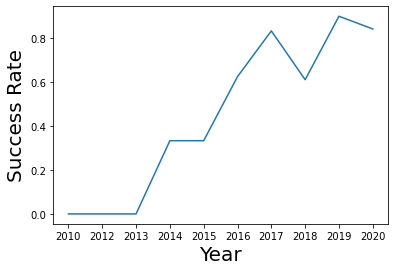
years **=** df**.**groupby(Extract\_year(df['Date']))**.**mean()['Class']

sns**.**lineplot(x **=** years**.**index, y **=** years)

plt**.**xlabel('Year', fontsize **=** 20)

plt**.**ylabel('Success Rate', fontsize **=** 20)

plt**.**show()



you can observe that the sucess rate since 2013 kept increasing till 2020

**Features Engineering**

By now, you should obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

In [11]:

features **=** df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block', 'ReusedCount', 'Serial']]

features**.**head()

Out[11]:

|  | **FlightNumber** | **PayloadMass** | **Orbit** | **LaunchSite** | **Flights** | **GridFins** | **Reused** | **Legs** | **LandingPad** | **Block** | **ReusedCount** | **Serial** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | 6104.959412 | LEO | CCAFS SLC 40 | 1 | False | False | False | NaN | 1.0 | 0 | B0003 |
| **1** | 2 | 525.000000 | LEO | CCAFS SLC 40 | 1 | False | False | False | NaN | 1.0 | 0 | B0005 |
| **2** | 3 | 677.000000 | ISS | CCAFS SLC 40 | 1 | False | False | False | NaN | 1.0 | 0 | B0007 |
| **3** | 4 | 500.000000 | PO | VAFB SLC 4E | 1 | False | False | False | NaN | 1.0 | 0 | B1003 |
| **4** | 5 | 3170.000000 | GTO | CCAFS SLC 40 | 1 | False | False | False | NaN | 1.0 | 0 | B1004 |

**TASK 7: Create dummy variables to categorical columns**

Use the function get\_dummies and features dataframe to apply OneHotEncoder to the column Orbits, LaunchSite, LandingPad, and Serial. Assign the value to the variable features\_one\_hot, display the results using the method head. Your result dataframe must include all features including the encoded ones.

In [12]:

*# HINT: Use get\_dummies() function on the categorical columns*

features\_one\_hot **=** pd**.**get\_dummies(features[['Orbit', 'LaunchSite', 'LandingPad', 'Serial']])

features\_one\_hot**.**head()

Out[12]:

|  | **Orbit\_ES-L1** | **Orbit\_GEO** | **Orbit\_GTO** | **Orbit\_HEO** | **Orbit\_ISS** | **Orbit\_LEO** | **Orbit\_MEO** | **Orbit\_PO** | **Orbit\_SO** | **Orbit\_SSO** | **...** | **Serial\_B1048** | **Serial\_B1049** | **Serial\_B1050** | **Serial\_B1051** | **Serial\_B1054** | **Serial\_B1056** | **Serial\_B1058** | **Serial\_B1059** | **Serial\_B1060** | **Serial\_B1062** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 72 columns

**TASK 8: Cast all numeric columns to float64**

Now that our features\_one\_hot dataframe only contains numbers cast the entire dataframe to variable type float64

In [13]:

*# HINT: use astype function*

features\_one\_hot**.**astype('float64')

Out[13]:

|  | **Orbit\_ES-L1** | **Orbit\_GEO** | **Orbit\_GTO** | **Orbit\_HEO** | **Orbit\_ISS** | **Orbit\_LEO** | **Orbit\_MEO** | **Orbit\_PO** | **Orbit\_SO** | **Orbit\_SSO** | **...** | **Serial\_B1048** | **Serial\_B1049** | **Serial\_B1050** | **Serial\_B1051** | **Serial\_B1054** | **Serial\_B1056** | **Serial\_B1058** | **Serial\_B1059** | **Serial\_B1060** | **Serial\_B1062** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **85** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **86** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| **87** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **88** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **89** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |

90 rows × 72 columns

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.

features\_one\_hot.to\_csv('dataset\_part\_3.csv', index=False)

In [14]:

features\_one\_hot**.**to\_csv('dataset\_part\_3.csv', index**=False**)

**Authors**

[Joseph Santarcangelo](https://www.linkedin.com/in/joseph-s-50398b136/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork26802033-2021-01-01) 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](https://www.linkedin.com/in/nayefaboutayoun/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork26802033-2021-01-01) is a Data Scientist at IBM and pursuing a Master of Management in Artificial intelligence degree at Queen's University.

**Change Log**

| **Date (YYYY-MM-DD)** | **Version** | **Changed By** | **Change Description** |
| --- | --- | --- | --- |
| 2020-09-20 | 1.0 | Joseph | Modified Multiple Areas |
| 2020-11-10 | 1.1 | Nayef | updating the input data |

Copyright © 2020 IBM Corporation. All rights reserved.