IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Sarapura Ruben Gabriel>
<27/12/2021>

Outline

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

Executive Summary

- Summary of methodologies

  Trends in programming languages and databases

  Demographics survey

  Technological gap in countries

  Gender gap in jobs

- Summary of all results

  the results were as expected according to the slogan which helped us to conclude the final prediction

Introduction

- ***Background and context of the project***

We predicted if the first stage of the Falcon 9 will land successfully. SpaceX Announces Falcon 9 Rocket Launches on Its Website, Costing $ 62 Million; other providers cost more than $ 165 million each, much of the savings due to SpaceX being able to reuse the first stage. Therefore we will determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternative company wants to bid against SpaceX for a rocket launch.

- ***Common problems that needed solution.***

What influences whether the rocket will land successfully?

The effect that each relationship with certain variables of the rocket will have in determining the success rate of a successful landing.

What conditions does SpaceX have to meet in order to obtain the best results and guarantee the best successful landing rate of the rocket.

Section 1

Methodology

Methodology

Executive Summary

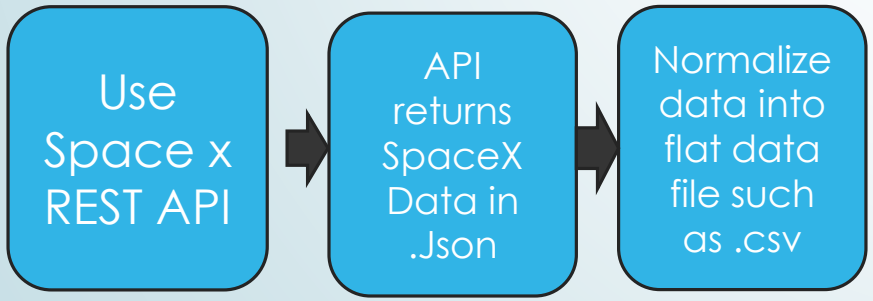
- Data collection methodology:
  - SpaceX Rest API
  - (Web Scrapping) from Wikipedia
- Perform data wrangling
  - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Plotting : Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Methodology

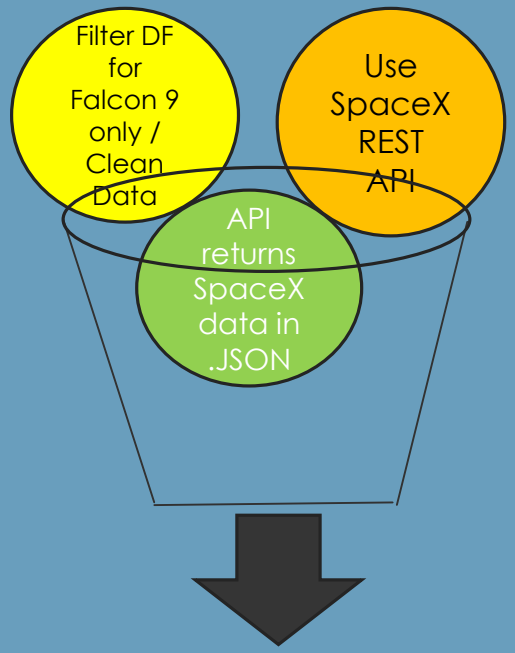
➨ The following data sets was collected

- We work with SpaceX launch data that is collected from the SpaceX REST API.
- This API will give us data about launches, including information about the rocket used, delivered payload, launch specifications, landing specifications and landing result.
- Our goal is to use this data to predict whether SpaceX will land a rocket or not.
- SpaceX REST API endpoints, or URLs, begin with api.spacexdata.com/v4/.
- Another popular data source for Falcon 9 launch data is Wikipedia's web scraping using Beautiful Soup.ñ

SpaceX API



Data collection –SpaceX API



Normalize data into flat data file such as .csv

coursera week1-1

1 .Getting Response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url).json()
```

2. Converting Response to a .json file

```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. 3. Apply custom functions to clean data

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

```
getBoosterVersion(data)
```

4. Assign list to dictionary then dataframe

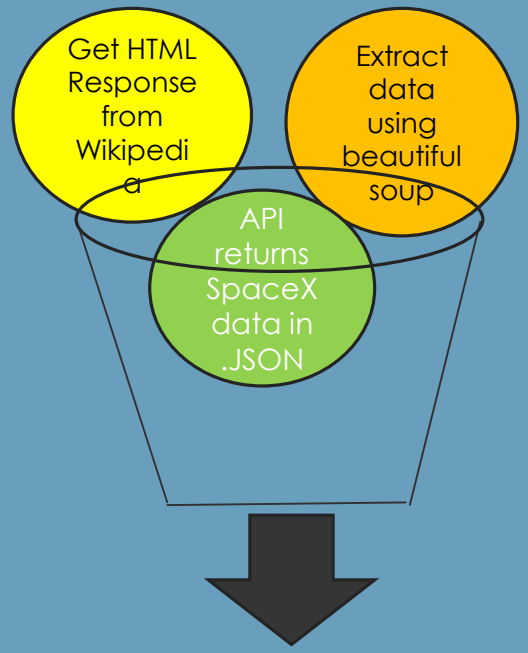
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
df = pd.DataFrame.from_dict(launch_dict)
```

5. Filter dataframe and export to flat file (.csv)

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data collection –Web Scrapping



Normalize data into flat data file such as .csv

Week 1-2

1 .Getting Response from API

```
page = requests.get(static_url)
```

2. Creating BeautifulSoup Object

```
soup = BeautifulSoup(page.text, 'html.parser')
```

3.Finding Tables

```
html_tables = soup.find_all('table')
```

4. Getting column names

```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5.Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']


launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Appending data to keys (refer)

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(
   # get table row
    for rows in table.find_all("tr")
        #check to see if first table
```

7.Converting dictionary a Data Frame

```
In [ ]:  df=pd.DataFrame(launch_dict)
```

8.Data frame to CSV

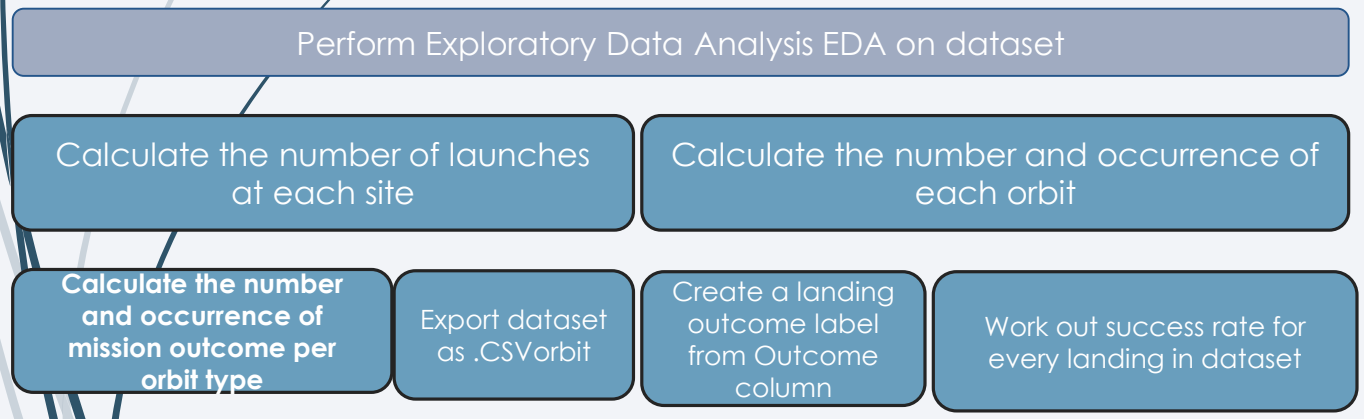
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

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.

We mainly convert those outcomes into Training Labels with1means the booster successfully landed 0 means it was unsuccessful.

Process



Github URL for Notebook

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

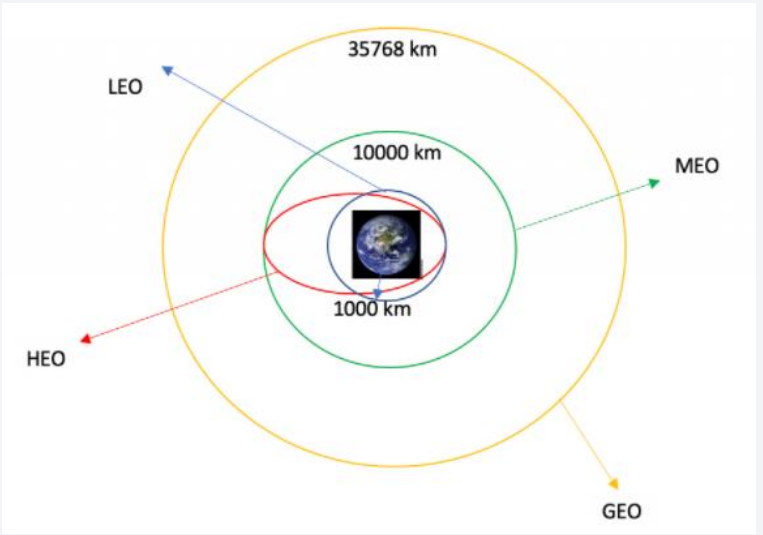


Diagram showing common orbit types SpaceX uses

EDA with Data Visualization

Scatter Graphs being drawn:

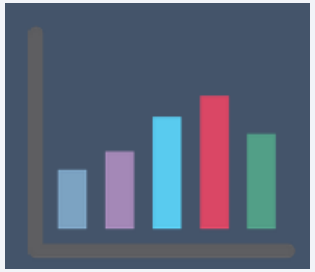
- Flight Number VS. Payload Mass
- Flight Number VS. Launch Site
- Payload VS. Launch Site
- Orbit VS. Flight Number
- Payload VS. Orbit Type
- Orbit VS. Payload Mass

Scatter plots showhow much one variable is affected by another. The relationship between two variables is called their correlation .Scatter plotsusually consist of a large body ofdata.

Github URL to Notebook

Bar Graph being drawn:

Mean VS. Orbit



A bar diagram makes it easy to compare sets of data between different groups at a glance. Thegraph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes.Bar charts can also show big changes in data over time.

Line Graph being drawn:

Success Rate VS. Year

Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

EDA with SQL

- Performed SQL queries to gather information about the dataset.

For example of some questions we were asked about the data we needed information about. Which we are using SQL queries to get the answers in the dataset :

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_sitefor the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Github URL to Notebook

Build an Interactive Map with Folium

To visualize the Launch Data into an interactive map. We took the Latitude and Longitude Coordinates at each launch site and added a *Circle Marker around each launch site with a label of the name of the launch site.*

We assigned the dataframe launch_outcomes(failures, successes) to *classes 0 and 1* with Greenand Redmarkers on the map in a MarkerCluster()

Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. **Lines** are drawn on the map to measure distance to landmarks

Example of some trends in which the Launch Site is situated in.

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Github URL to Notebook

Build a Dashboard with Plotly Dash

Used Python Anywhere to host the website live 24/7 so your can play around with the data and view the data

- **The dashboard is built with Flask and Dash web framework.**

Graphs

-Pie Chart showing the total launches by a certain site/all sites

-display relative proportions of multiple classes of data.

-size of the circle can be made proportional to the total quantity it represents.

Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions

-It shows the relationship between two variables.

-It is the best method to show you a non-linear pattern.

-The range of data flow, i.e. maximum and minimum value, can be determined.

-Observation and reading are straightforward.

Predictive Analysis (Classification)

BUILDING MODEL

- Load our dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to Grid Search CV
- Fit our datasets into the Grid Search CV objects and train our dataset.

EVALUATING MODEL

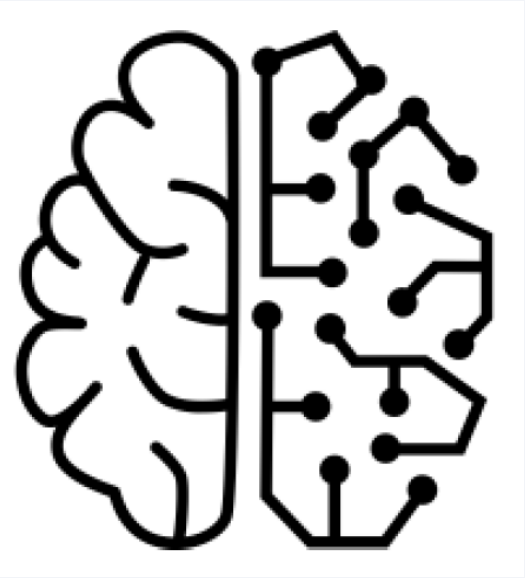
- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

IMPROVING MODEL

- Feature Engineering
- Algorithm Tuning

FINDING THE BEST PERFORMING CLASSIFICATION MODEL

- The model with the best accuracy score wins the best performing model
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook.



GitHub link to source code

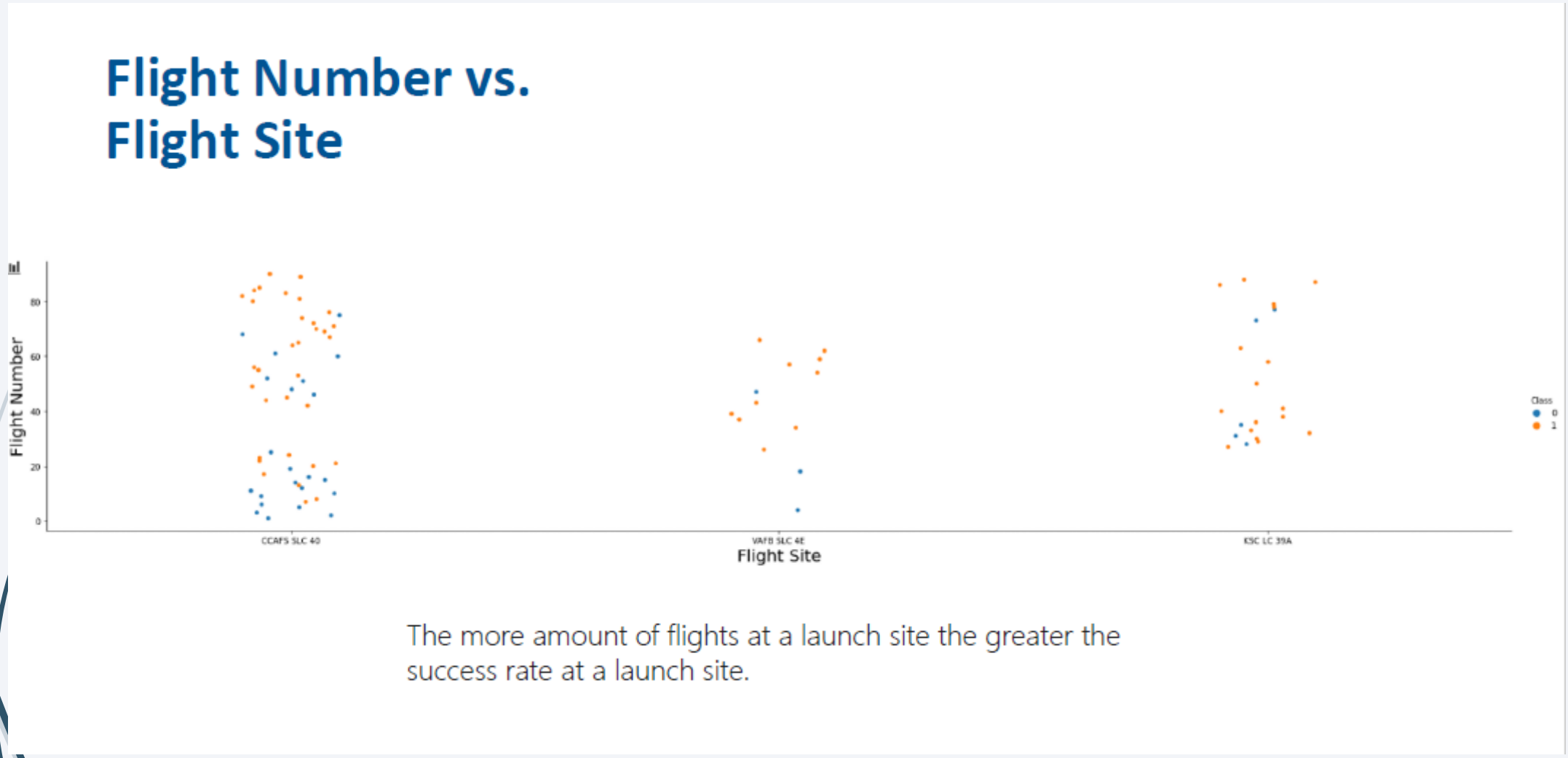
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site



The more amount of flights at a launch site the greater the success rate at a launch site.

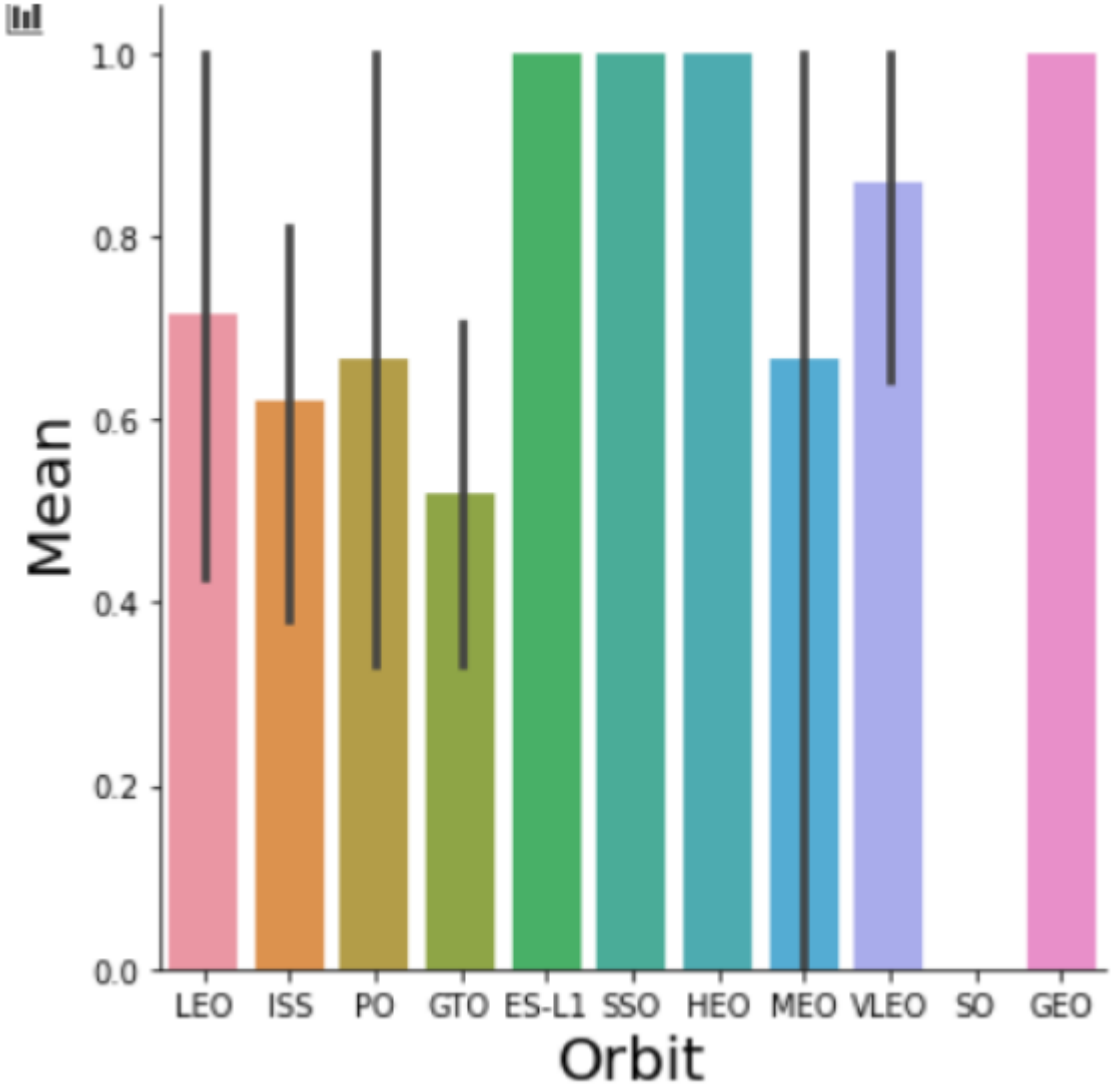
Payload vs. Launch Site

The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket.
There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch.

Success Rate vs. Orbit Type

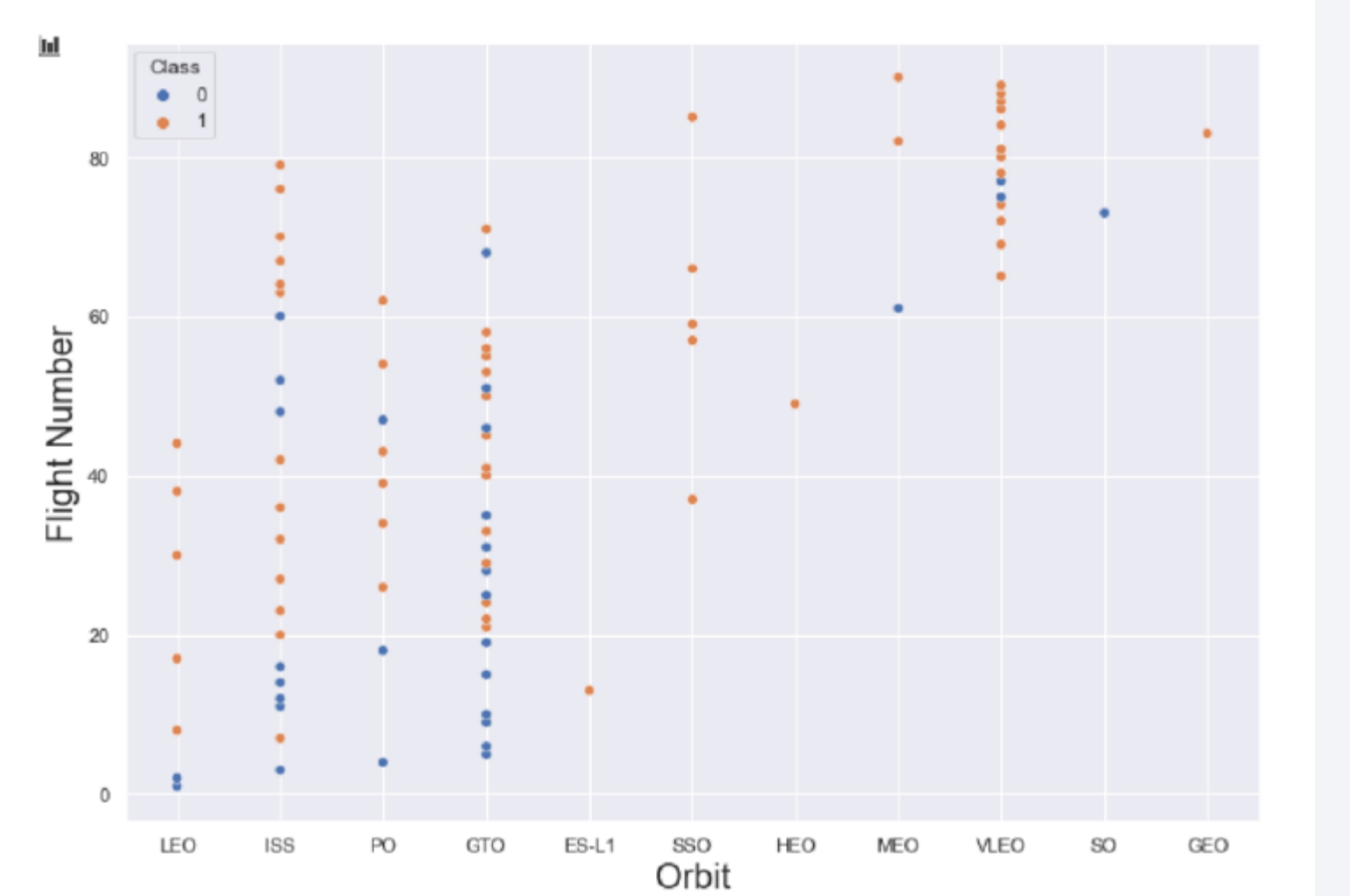
Success rate vs. Orbit type

- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate20



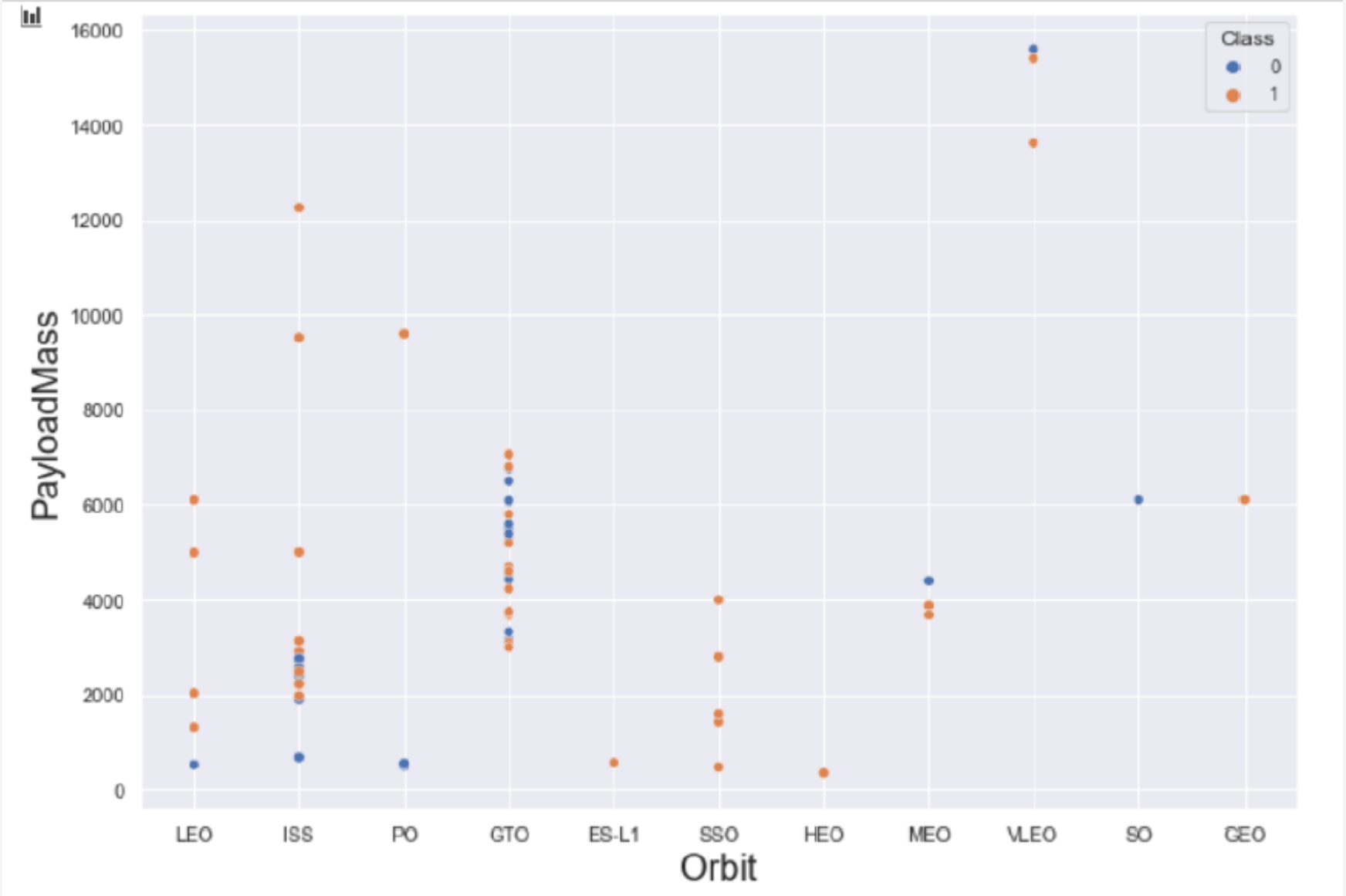
Flight Number vs. Orbit Type

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.21



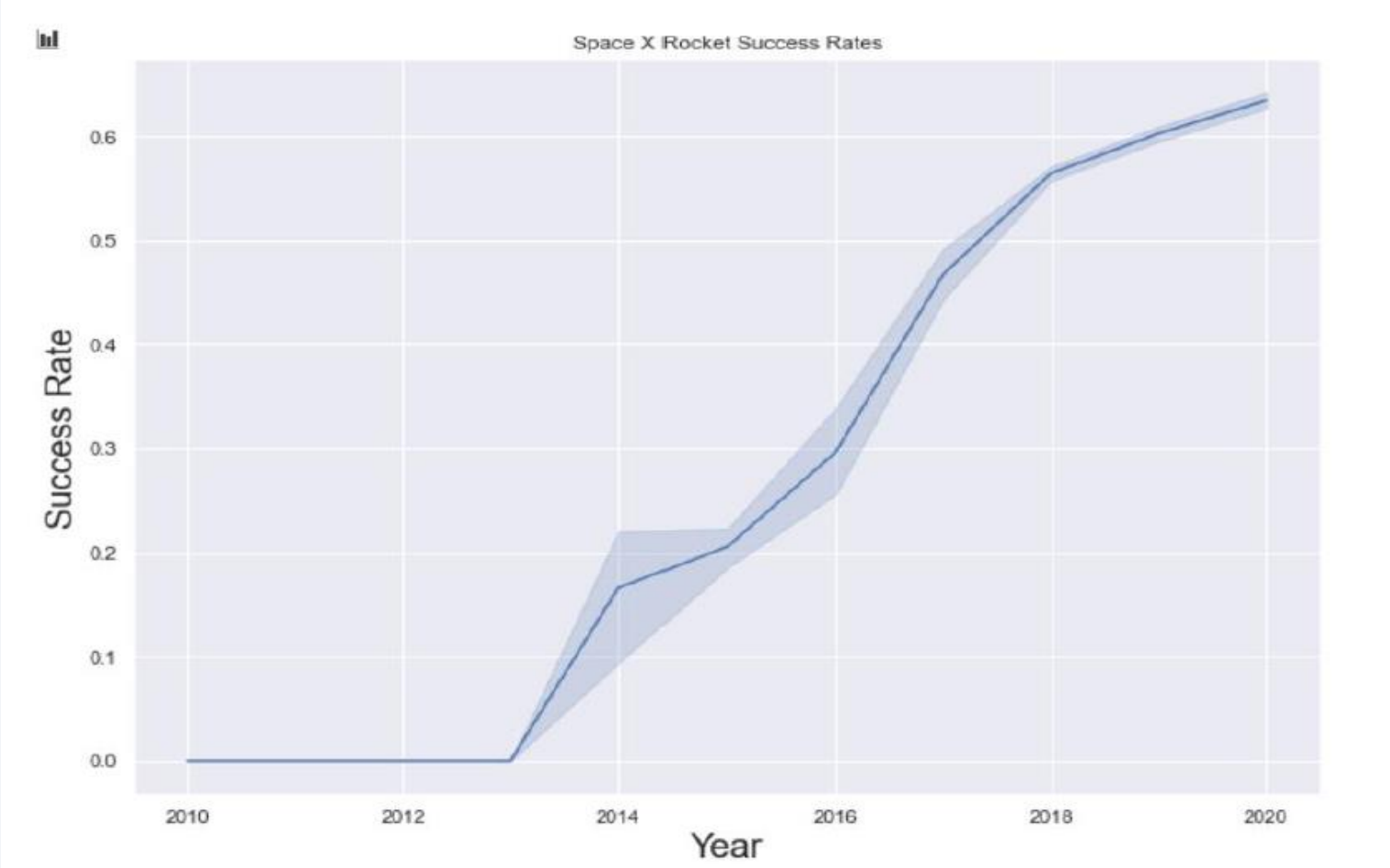
Payload vs. Orbit Type

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



Launch Success Yearly Trend

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



All Launch Site Names

SQL QUERY

26 Launch site names begin with `CCA`selectTOP5*fromtblSpaceXWHERELaunch_SiteLIKE'KSC%'

QUERY EXPLAINATION

Using the word ***TOP 5*** in the query means that it will only show 5 records from ***tblSpaceX*** and ***LIKE*** keyword has a wild card
with the words ***'KSC%'***the percentage in the end suggests that the Launch_Site name must start with KSC.

	Date	Time_UTC	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	19-02-2017	2021-07-02 14:39:00.0000000	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
1	16-03-2017	2021-07-02 06:00:00.0000000	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2	30-03-2017	2021-07-02 22:27:00.0000000	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
3	01-05-2017	2021-07-02 11:15:00.0000000	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
4	15-05-2017	2021-07-02 23:21:00.0000000	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

All Launch Site Names

SQL QUERY

26 Launch site names begin with `CCA`selectTOP5*fromtblSpaceXWHERELaunch_SiteLIKE'KSC%'

QUERY EXPLAINATION

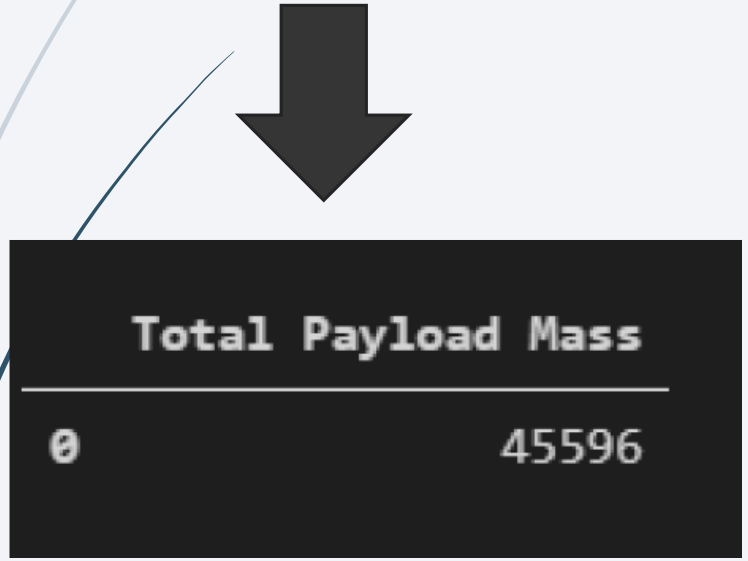
Using the word ***TOP 5*** in the query means that it will only show 5 records from ***tblSpaceX*** and ***LIKE*** keyword has a wild card
with the words ***'KSC%'***the percentage in the end suggests that the Launch_Site name must start with KSC.

	Date	Time_UTC	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	19-02-2017	2021-07-02 14:39:00.0000000	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
1	16-03-2017	2021-07-02 06:00:00.0000000	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2	30-03-2017	2021-07-02 22:27:00.0000000	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
3	01-05-2017	2021-07-02 11:15:00.0000000	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
4	15-05-2017	2021-07-02 23:21:00.0000000	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass by Customer NASA (CRS)

SQL QUERY

```
Select UM(PAYLOAD_MASS_KG_)TotalPayloadMass from tblspaceX
where Customer='NASA(CRS)'',TotalPayloadMass
```



	Total Payload Mass
0	45596

QUERY EXPLAINATION

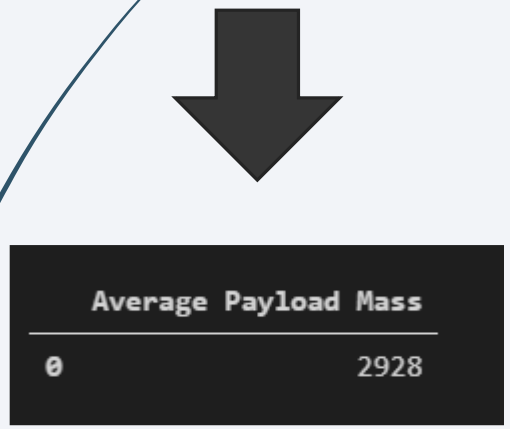
Using the function ***SUM*** summates the total in the column ***PAYLOAD_MASS_KG_***

The ***WHERE***clause filters the dataset to only perform calculations on ***Customer NASA (CRS)***

Average Payload Mass by F9 v1.1

SQL QUERY

```
Select AVG(PAYLOAD_MASS_KG_)Average Payload Mass from tblSpaceX where
Booster_Version='F9v1.1'
```



Average Payload Mass
0 2928

QUERY EXPLAINATION

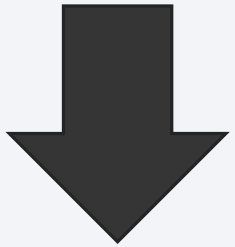
Using the function ***AVG*** works out the average in the column ***PAYLOAD_MASS_KG_***

The ***WHERE***clause filters the dataset to only perform calculations on ***Booster_version F9 v1.1***

The date where the successful landing outcome in drone ship was achieved

SQL QUERY

Select MIN(Date) SLO from tblSpaceX where Landing_Outcome="Success(droneship)"



Date which first Successful landing outcome in drone ship was acheived.
06-05-2016

QUERY EXPLAINATION

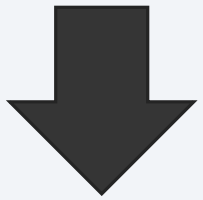
Using the function ***MIN*** works out the minimum date in the column ***Date***

The ***WHERE***clause filters the dataset to only perform calculations on ***Landing_Outcome Success (drone ship)***

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL QUERY

```
Select Booster_Version from tblSpaceX where Landing_Outcome =
'Success(groundpad)'       AND Payload_MASS_KG_> 4000
ANDPayload_MASS_KG_<6000
```



	Date which first Successful landing outcome in drone ship was acheived.
0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

QUERY EXPLAINATION

Selecting only ***Booster_Version***

The ***WHERE*** clause filters the dataset to ***Landing_Outcome = Success (drone ship)***

The ***AND*** clause specifies additional filter conditions ***Payload_MASS_KG_***>4000AND***Payload_MASS_KG_<6000***

Total Number of Successful and Failure Mission Outcomes

SQL QUERY

```
SELECT (SELECTCount(Mission_Outcome) from tblSpaceX where Mission_Outcome LIKE
'%Success%') as Successful_Mission_Outcomes,

(SELECTCount(Mission_Outcome) from tblSpaceX where Mission_Outcome

LIKE  '%Failure%') as Failure_Mission_Coutcomes
```

	Successful_Mission_Outcomes	Failure_Mission_Outcomes
0	100	1

QUERY EXPLAINATION

a much harder query I must say, we used subqueries here to produce the results. The ***LIKE '%foo%'*** wildcard shows that in the record the ***foo*** phrase is in any part of the string in the records for example.

PHRASE "(Drone Ship was a Success)"
LIKE '%Success%'
Word 'Success' is in the phrase the filter will include it in the dataset

Boosters Carried Maximum Payload

SQL QUERY

```
SELECT DISTINCT Booster_Version,
MAX(PAYLOAD_MASS_KG_) AS [Maximum Payload Mass]
FROM tbl SpaceX GROUP BY Booster_Version
ORDER BY [Maximum Payload Mass]DESC
```

QUERY EXPLAINATION

Using the word *DISTINCT* in the query means that it will only show Unique values in the *Booster_Version* column from *tblSpaceX*

GROUP BY puts the list in order set to a certain condition.

DESC means its arranging the dataset into descending order

	Booster_Version	Maximum Payload Mass
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
...	...	...
92	F9 v1.1 B1003	500
93	F9 FT B1038.1	475
94	F9 B4 B1045.1	362
95	F9 v1.0 B0003	0
96	F9 v1.0 B0004	0

97 rows × 2 columns

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

SQL QUERY

```
SELECT COUNT(Landing_Outcome)
FROM     tblSpaceX
WHERE  (Landing_Outcome LIKE '%Success%')
AND (Date > '04-06-2010')
AND (Date < '20-03-2017')
```

QUERY EXPLAINATION

Function ***COUNT*** counts records in column

WHERE filters data

- ***LIKE (wildcard)***
- ***AND (conditions)***
- ***AND (conditions)***

	Successful Landing Outcomes Between 2010-06-04 and 2017-03-20
0	34

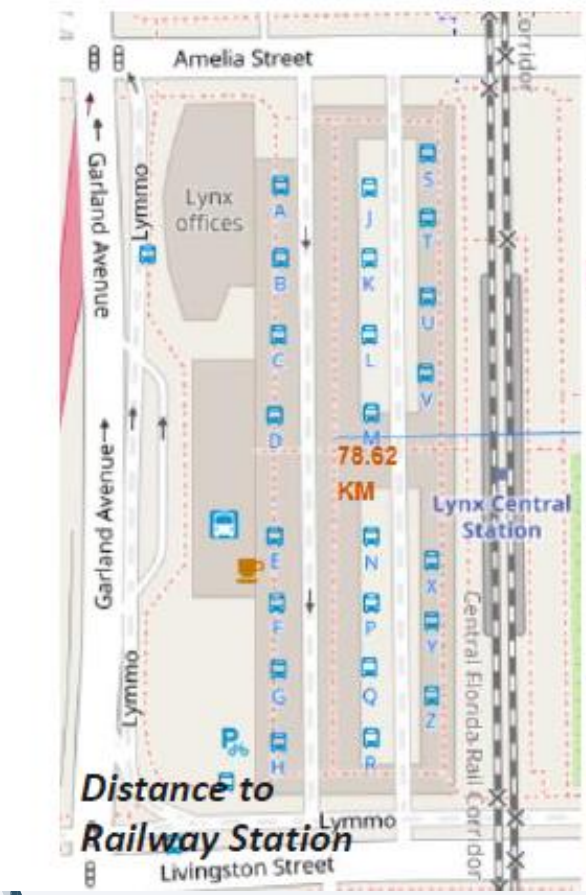
Section 4

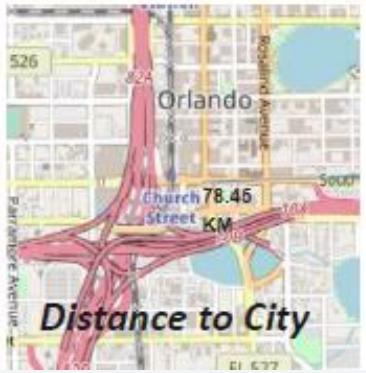
Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

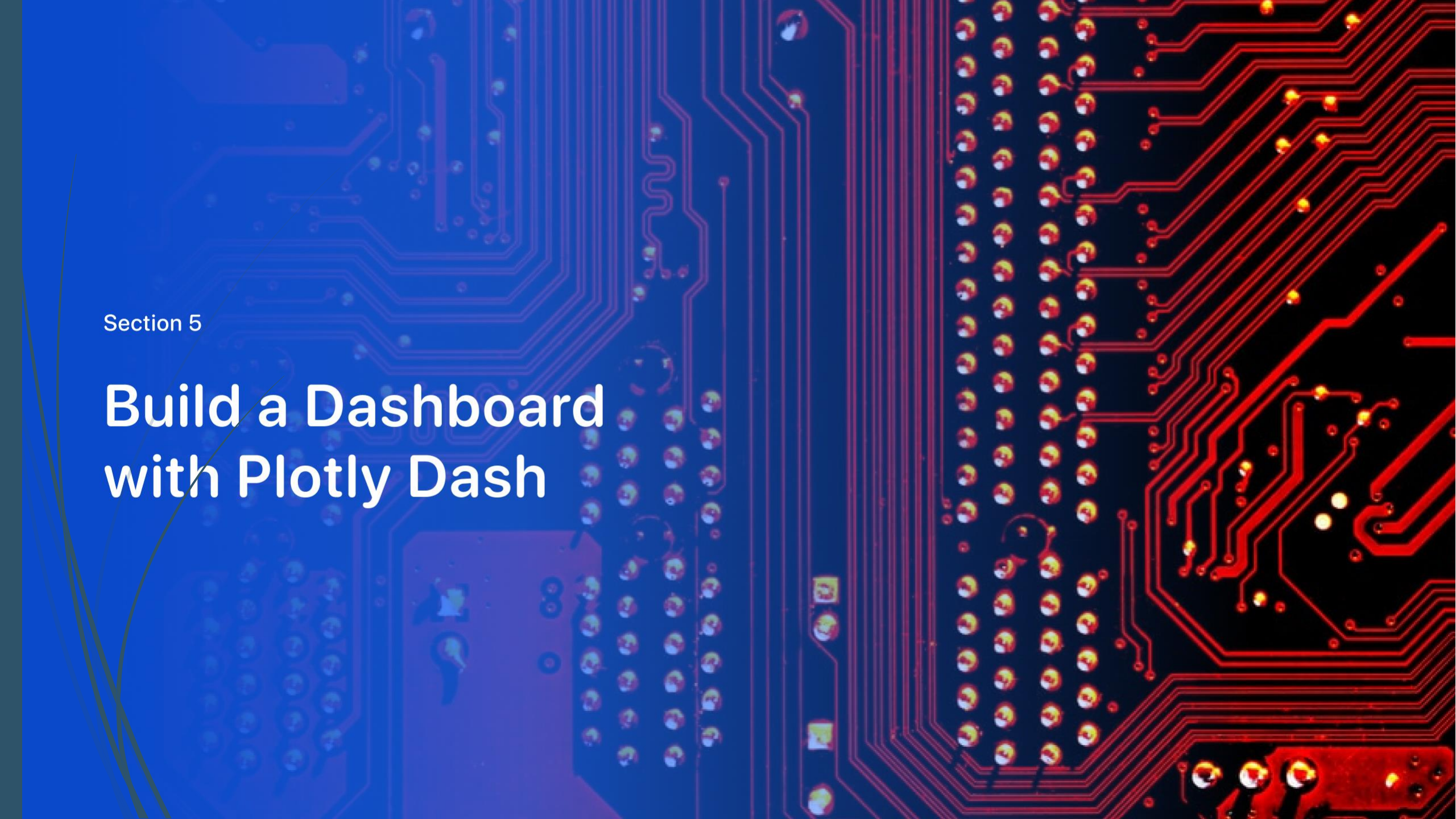
<Folium Map Screenshot 2>

<Folium Map Screenshot 3>

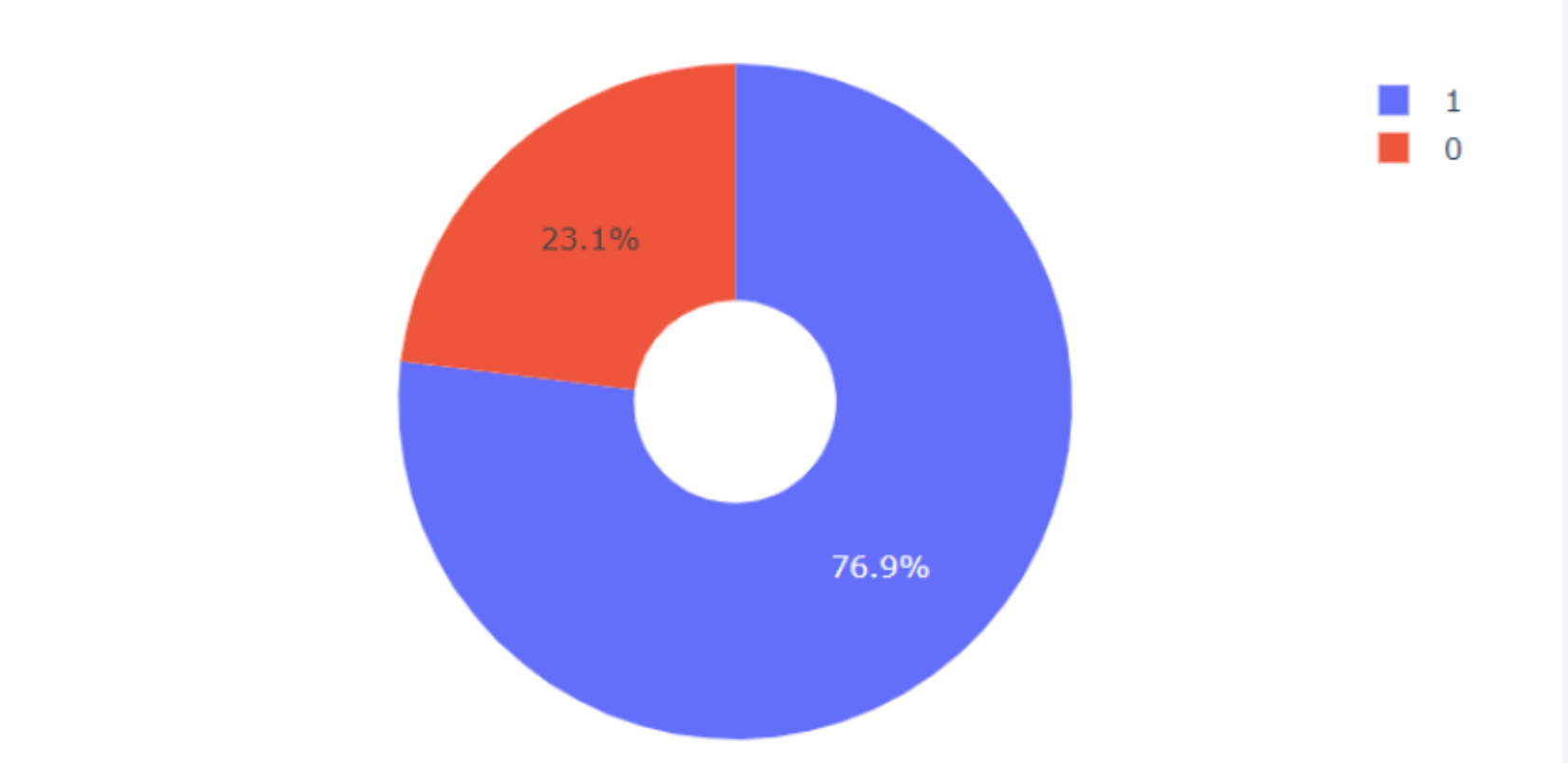




- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

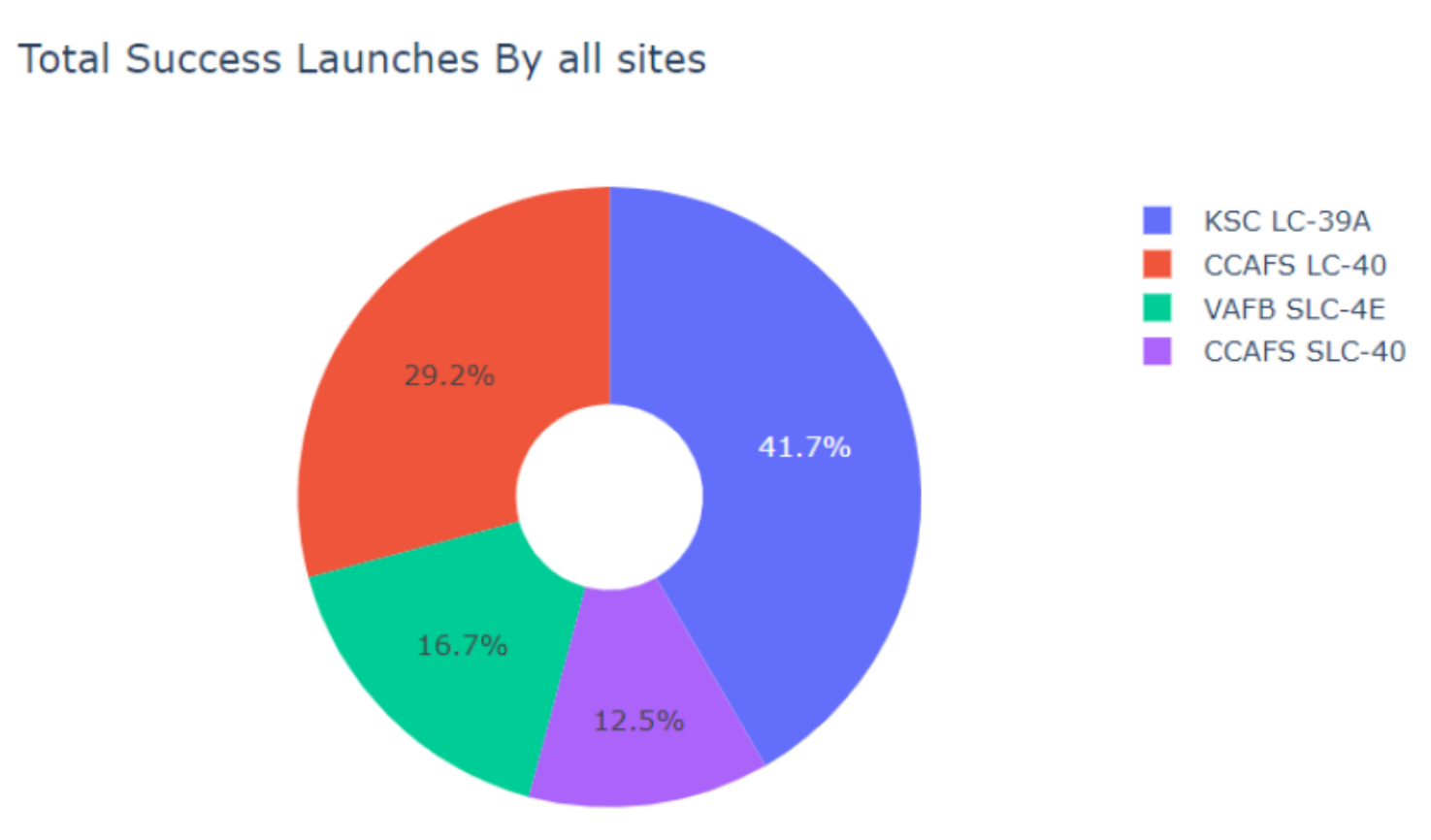

Section 5
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



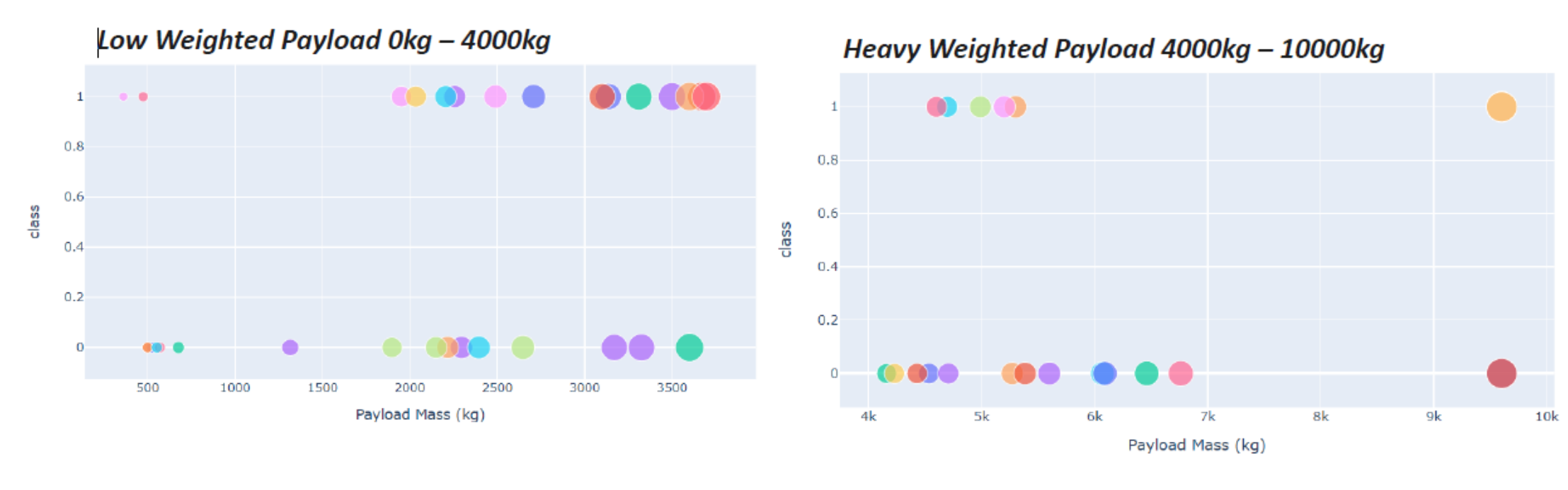
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

<Dashboard Screenshot 2>



We can see that KSC LC-39A had the most successful launches from all the sites

<Dashboard Screenshot 3>



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

Predictive Analysis (Classification)

Classification Accuracy

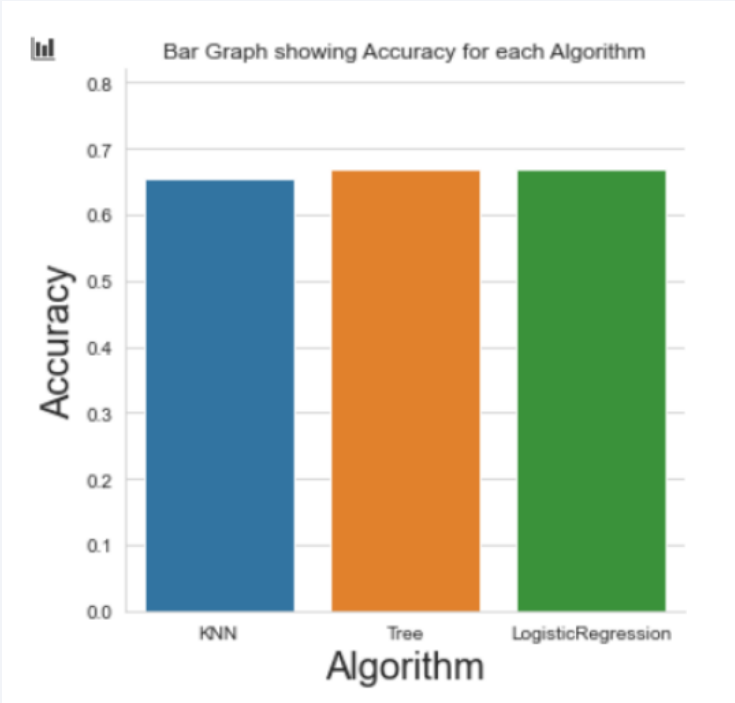
Classification Accuracyusing training data

As you can see our accuracy is extremely close but we do have a winner its down to decimal places! using this function

```
bestalgorithm = max(algorithms, key=algorithms.get)
```

	Accuracy	Algorithm
0	0.653571	KNN
1	0.667857	Tree
2	0.667857	LogisticRegression

The tree algorithm wins!!



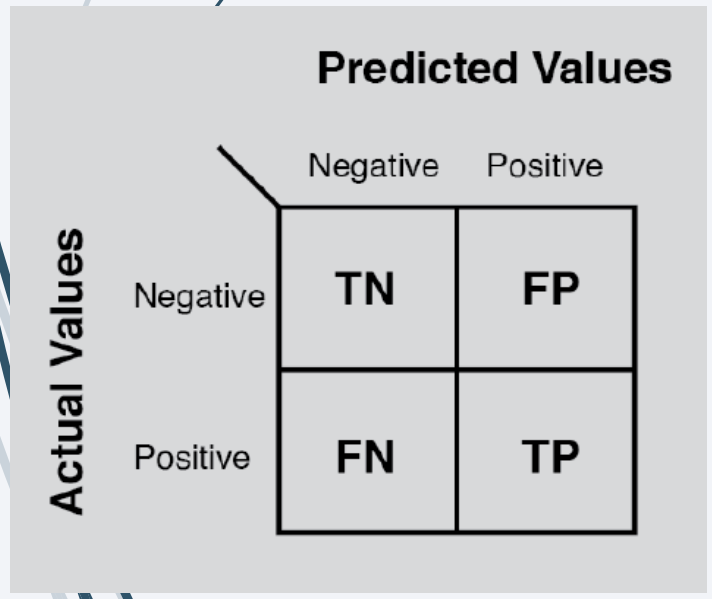

```
Best Algorithm is Tree with a score of 0.6678571428571429
Best Params is : {'criterion': 'gini', 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

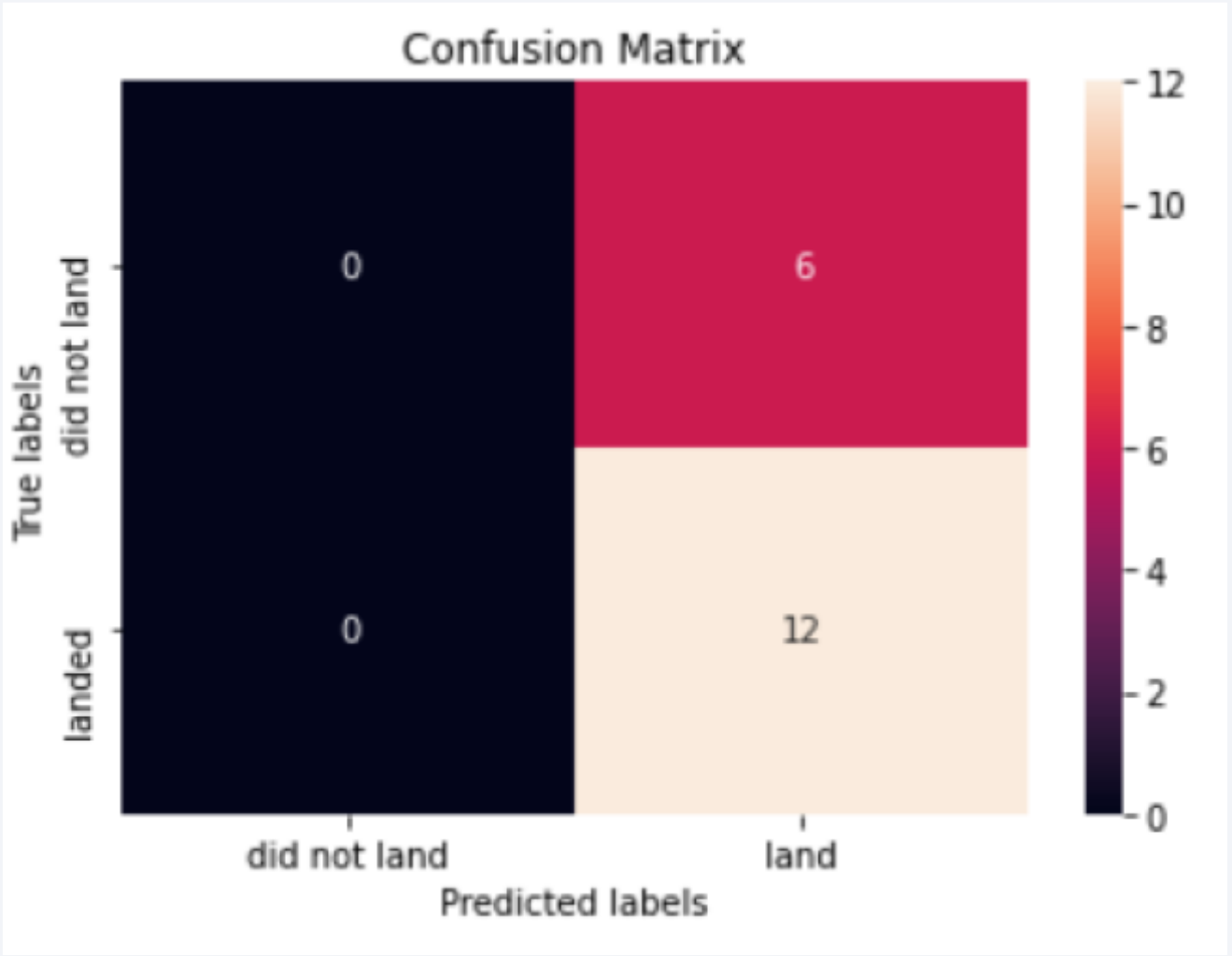
After selecting the best hyperparameters for the decision tree classifier using the validation data, we achieved 83.33% accuracy on the test data.

Confusion Matrix

Confusion Matrixfor the Tree

Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the major problem is false positives.





Conclusions

- The Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

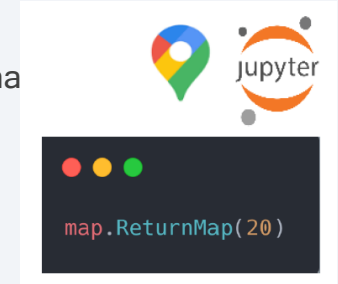
Appendix

- Haversine formula
- ADGGoogleMaps Module (not used but created)
- Module sqlserver (ADGSQLSERVER)
- PythonAnywhere 24/7 dashboard

ADG Google Maps Module

Introduction

This is a python class I designed to make my life easier and others when getting coordina from addresses and producing a Folium Python Map in a Jypyter Notebook. This class processing is powered by the Google Geocoding API.



Returning Folium Python Mapin Jypyter Notebook assuming you declared mapclass In 1.
.ReturnMap(size)
this function returns this Folium map in the Jypyter Notebook

Prerequisites

Sign up for a Google Geocoding API Set up a Google API Key Head to library under APIs & Services and add Geocoding API.

Requirements

Google API Secret (API Key)

Python Usage Documentation

Getting Cords

Simple arguments to follow by the object __init__(self,api_key,address) - *Look at following usage for guidance*

```
import ADGGoogleMaps
map = ADGGoogleMaps.ADGGoogleMaps("google_api_secret_key","your_address")
cords = map.GetCordsFromAddress()
cords
```

This returns a list with your coordinates of that address example ['longitude','latitude']

Haversine formula

Introduction

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.

Usage

Why did I use this formula? First of all, I believe the Earth is round/elliptical. I am not a Flat Earth Believer! Jokes aside when doing Google research for integrating my ADGGoogleMaps API with a Python function to calculate the distance using two distinct sets of {longitudinal, latitudinal} list sets. Haversine was the trigonometric solution to solve my requirements above.

Formula

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{(1-a)}\right)$$

$$d = R \cdot c$$

```
#Variables
d is the distance between the two points along a great circle of the sphere (see spherical distance),
r is the radius of the sphere.
φ1, φ2 are the latitude of point 1 and latitude of point 2 (in radians),
λ1, λ2 are the longitude of point 1 and longitude of point 2 (in radians).
```

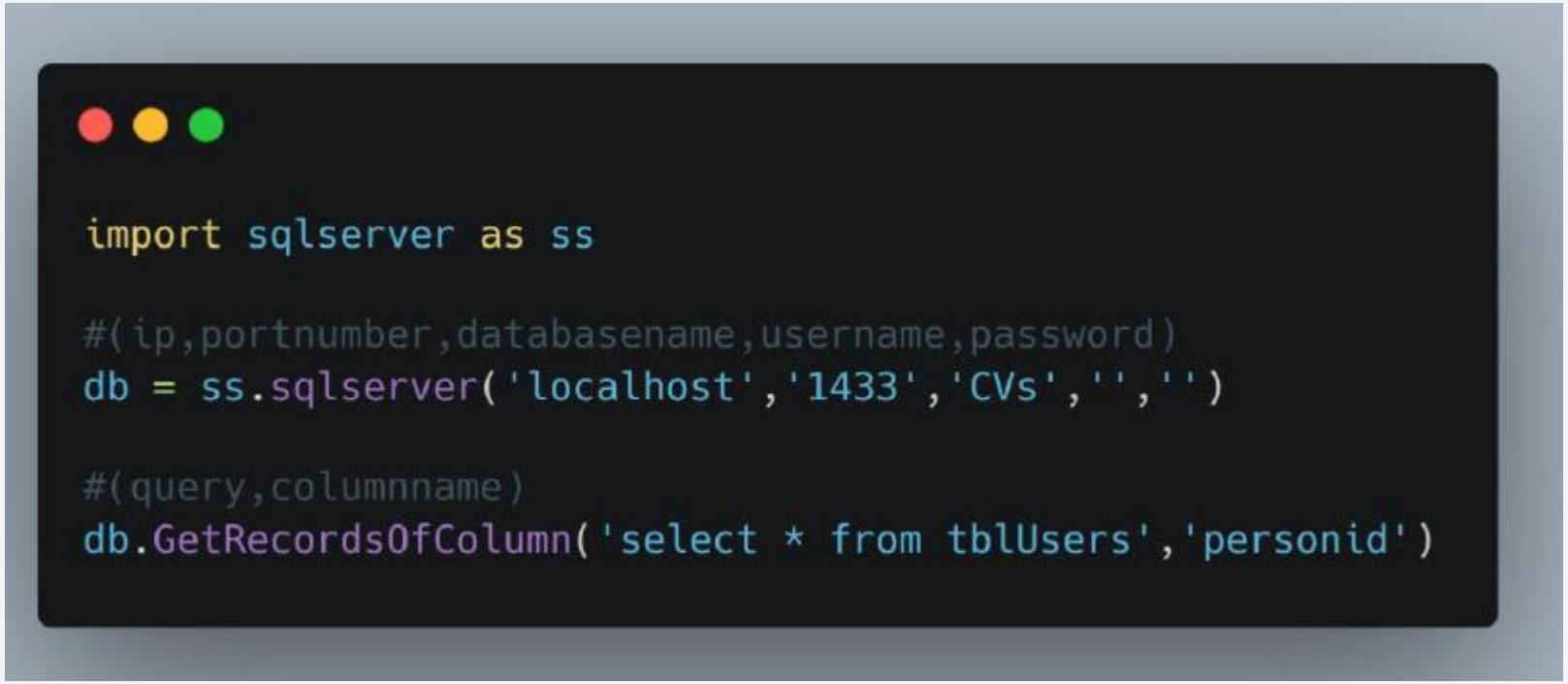
ADGSQL server

Introduction

basically I just wanted an easier way to get my data into my python programming by coding my own module powered by ODBC to simplify my code

Implementation

Pull data from columns
Extract Records
Run Stored Procedures …



```
import sqlserver as ss

#(ip,portnumber,databasename,username,password)
db = ss.sqlserver('localhost','1433','CVs','','')

#(query,columnname)
db.GetRecordsOfColumn('select * from tblUsers','personid')
```

Python Anywhere

Introduction

I wanted to put my python website running 24/7 on the cloud so anyone can view it then I came across PythonAnywhere.

Implementation

Pull data from columns
Extract Records
Run Stored Procedures …

```
import sqlserver as ss

#(ip,portnumber,databasename,username,password)
db = ss.sqlserver('localhost','1433','CVs','','')

#(query,columnname)
db.GetRecordsOfColumn('select * from tblUsers','personid')
```

Thank you!