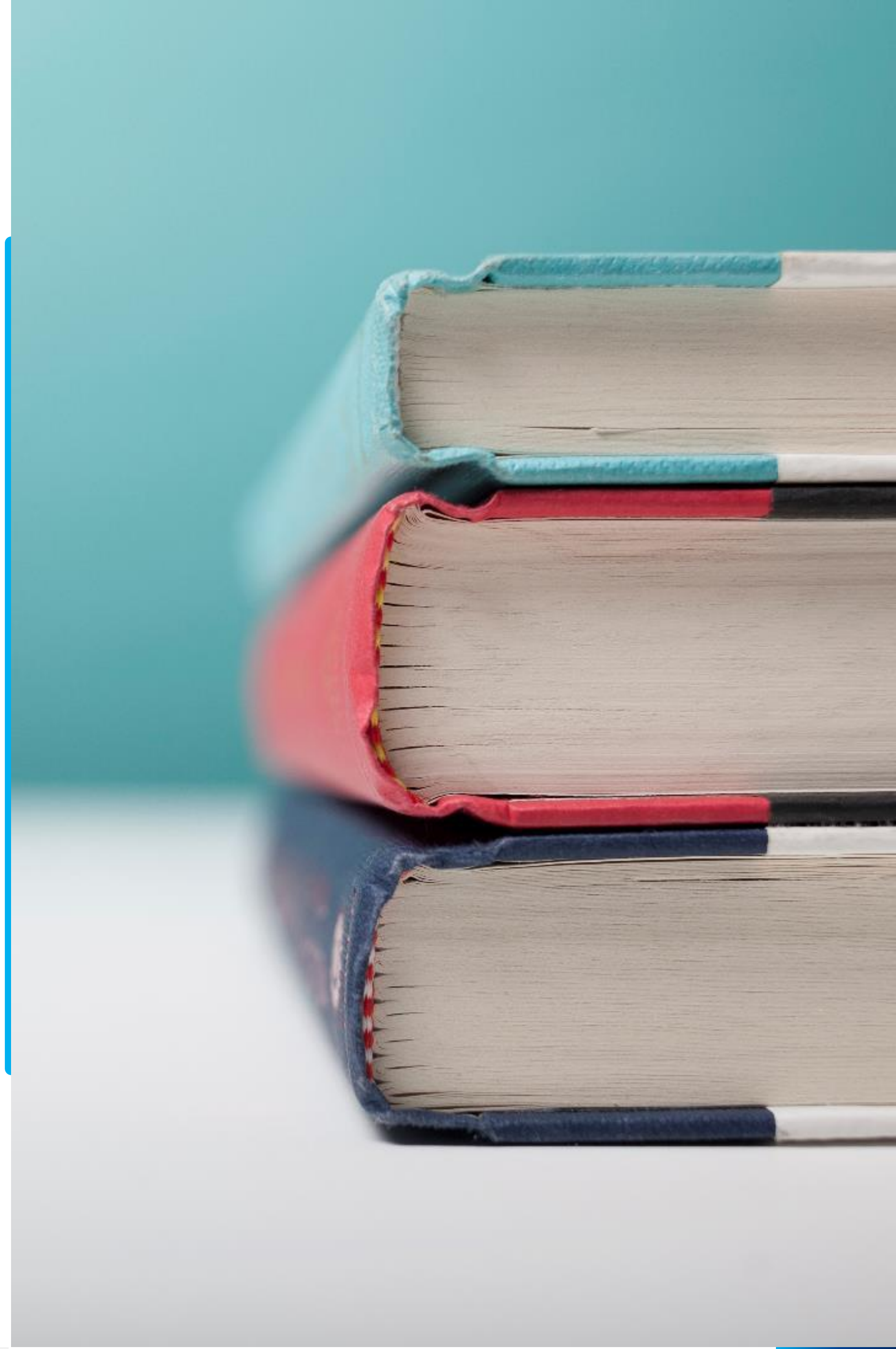
Winning the Space Race with Data Science

Michal Kovac

November 1, 2021

Outline

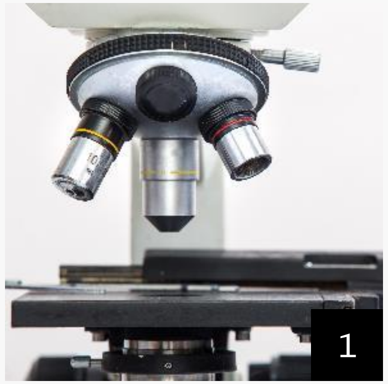
- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

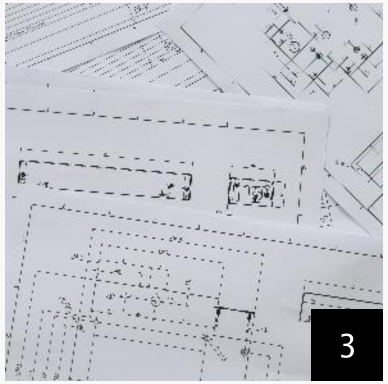


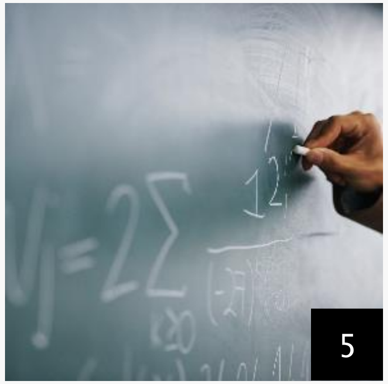
Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Predictive analysis (Classification)

- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results







Introduction

- Project background and context

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.

- Problems that require solution
  - What factors do affect the successful landing
  - What is the relationship between data variables and the success rate
  - How to learn from data to increase the overall success rate

Methodology

Data Collection

- Data collection methodologies:
  - There are 2 alternative approaches of data collection:

A) [SpaceX Rest Api (Link)]

- The data from this API include the information about the launches, payload mass, place from which the rocket launched and more.

B) [Web Scrapping Wikipedia (Link)]

- Using BeautifulSoup, it was possible to obtain the information about launches from Wikipedia.

Data Collection (SpaceX Rest API)

- Getting Response

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

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

- Converting to a .json file

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

```
response = requests.get(static_json_url).json()
```

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

- Cleaning Data

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

Data Collection (SpaceX Rest API)

- Assigning list to a dictionary & 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}
```

```
# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

- Filtering df and dealing with missing values

```
# Hint data['BoosterVersion']!='Falcon 1'

data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

Now that we have removed some values we should reset the FlgihtNumber column

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

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(mean)
data_falcon9.isnull().sum()
```

Data Collection (Web Scrapping)

- Getting Response from HTML

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

- BeautifulSoup Object

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

- Finding Tables

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

```
first_launch_table = html_tables[2]
```

- Getting Columns

```
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
```

Data Collection (Web Scrapping)

- 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']=[]
```

- Data to keys

(refer to block 22 in notebook)

- From Dictionary to Dataframe

```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

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

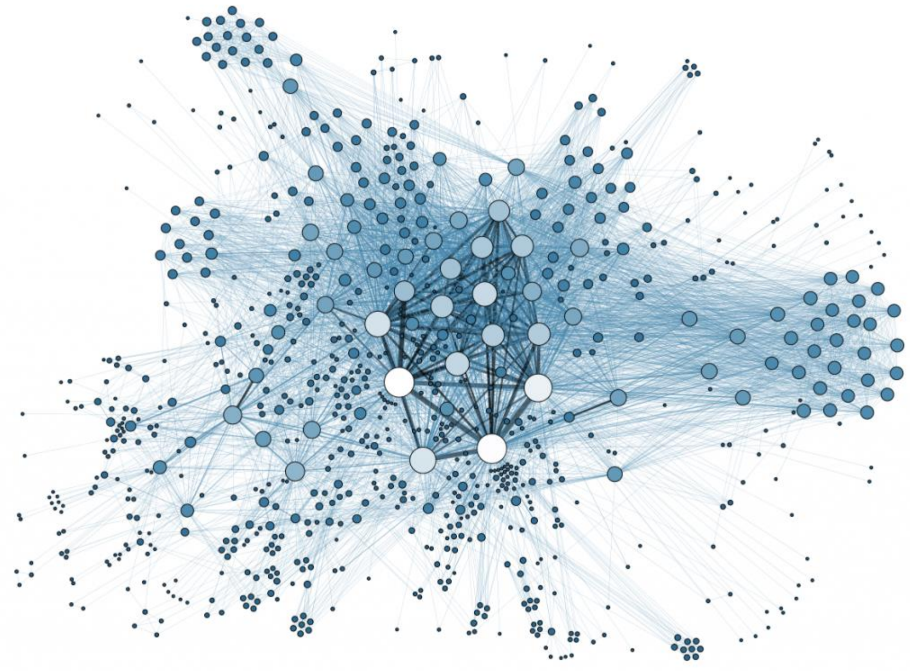
- Dataframe to csv

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

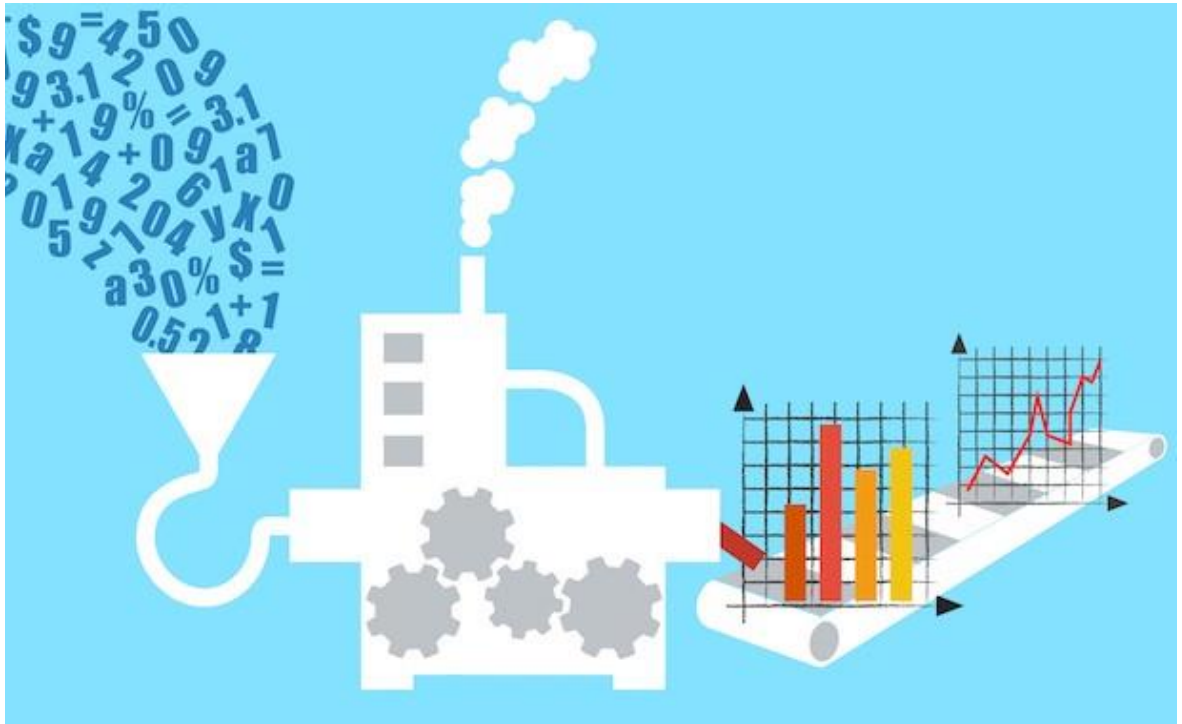
Data Wrangling

- 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.

- [Git Hub Link to the notebook]



EDA with Visualization



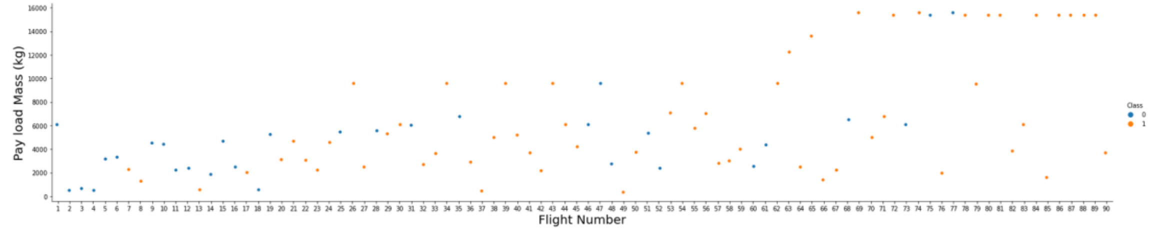
- In this assignment, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is due to the fact that SpaceX can reuse the first stage. Using Pandas and Matplotlib, we were able to prepare:
  - Exploratory Data Analysis
  - Data Feature Engineering

- Git Hub Link to the notebook

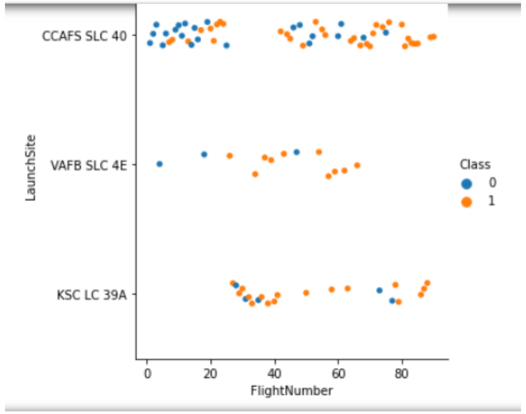
EDA with Visualization

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

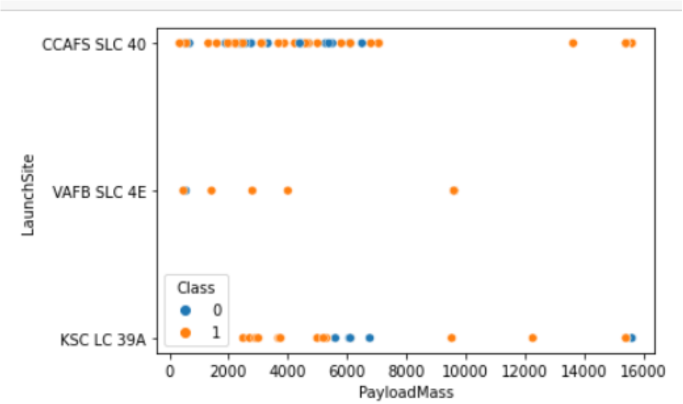
```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



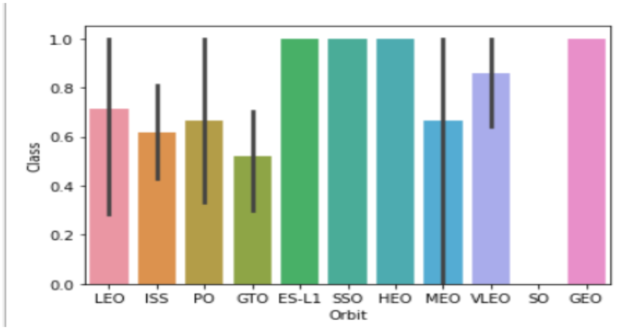
- Visualizing the relationship between Flight Number and Launch Site



- Visualizing the relationship between Payload and Launch Site



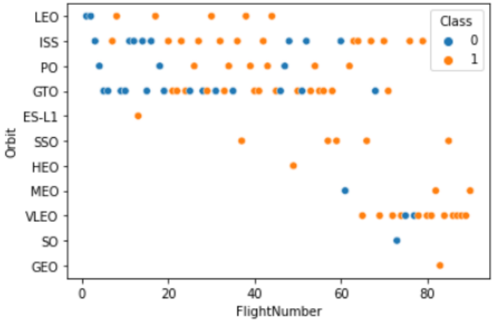
- Visualizing the relationship between success rate of each orbit type¶



EDA with Visualization

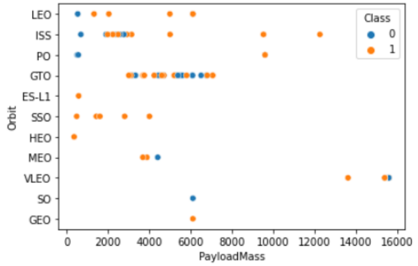
- Visualizing the relationship between FlightNumber and Orbit type

```
sns.scatterplot(x="FlightNumber", y="Orbit", hue="Class", data=df)
plt.show()
```



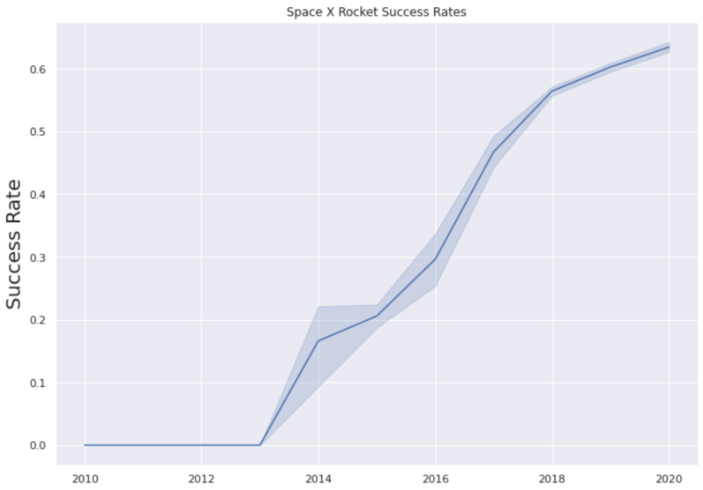
- Visualizing the relationship between Payload and Orbit type

```
sns.scatterplot(x="PayloadMass", y="Orbit", hue="Class", data=df)
plt.show()
```



- Visualize the launch success yearly trend

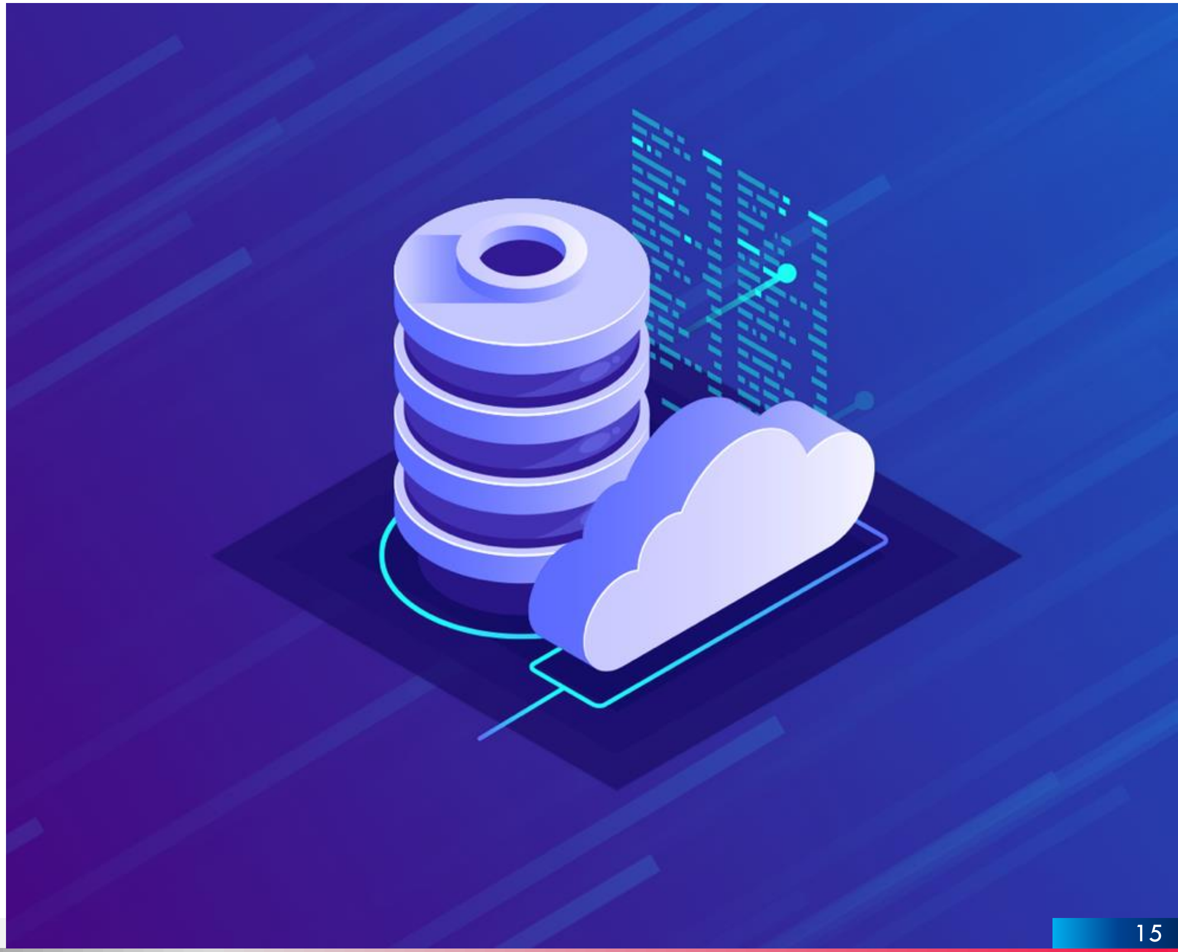
```
year = pd.DatetimeIndex(df['Date']).year
year = np.array(list(year))
successratelist = []
successrate = 0.00
records = 1
data = 0
for x in df['Class']:
    data = x + data
    successrate = data/records
    successratelist.append(successrate)
    records= records +1

successratelist = np.array(successratelist)
d = {'successrate':successratelist,'year':year}
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.lineplot(data=d, x="year", y="successrate" )

plt.xlabel("Year",fontsize=20)
plt.title('Space X Rocket Success Rates')
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



EDA with SQL

- In upcoming slides, we're going to create a database with SpaceX data and then query using python.

- [Git Hub Link to the notebook]



EDA with SQL

Task 1

Display the names of the unique launch sites in the space mission

```
%sql select LAUNCH_SITE from SPACEXTBL3 group by LAUNCH_SITE
```

launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Task 2

Display 5 records where launch sites begin with the string 'CCA'

```
%sql select * from SPACEXTBL3 where LAUNCH_SITE like 'CCA%' limit 5
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	06:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	03:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	12:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	03:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

EDA with SQL

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql select launch_site, SUM(PAYLOAD_MASS__KG_) as total_mass \
from SPACEXTBL3 \
GROUP BY launch_site \
ORDER BY TOTAL_MASS DESC LIMIT 10
```

launch_site	total_mass
CCAFS SLC-40	254037
KSC LC-39A	208837
VAFB SLC-4E	89730
CCAFS LC-40	67363

Task 4 ¶

Display average payload mass carried by booster version F9 v1.1

```
%sql select booster_version, AVG(PAYLOAD_MASS__KG_) as average_mass \
from SPACEXTBL3 where booster_version = 'F9 v1.1'\
GROUP BY booster_version \
ORDER BY AVERAGE_MASS DESC LIMIT 100
```

booster_version	average_mass
F9 v1.1	2928

Task 5

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql select * from SPACEXTBL3 where landing__outcome like 'Success (ground pad%' \
order by DATE ASC LIMIT 10
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2015-12-22	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)
2016-07-18	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-02-19	02:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-06-03	09:07:00	F9 FT B1035.1	KSC LC-39A	SpaceX CRS-11	2708	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-08-14	04:31:00	F9 B4 B1039.1	KSC LC-39A	SpaceX CRS-12	3310	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-09-07	02:00:00	F9 B4 B1040.1	KSC LC-39A	Boeing X-37B OTV-5	4990	LEO	U.S. Air Force	Success	Success (ground pad)
2017-12-15	03:36:00	F9 FT B1035.2	CCAFS SLC-40	SpaceX CRS-13	2205	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2018-01-08	01:00:00	F9 B4 B1043.1	CCAFS SLC-40	Zuma	5000	LEO	Northrop Grumman	Success (payload status unclear)	Success (ground pad)

EDA with SQL

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql select * from SPACEXTBL3 where landing__outcome like 'Success (drone ship%' and payload_mass__kg_ > 4000 and payload_mass__kg_ < 6000 \
order by DATE ASC LIMIT 10
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2016-05-06	05:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-08-14	05:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2017-03-30	10:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-10-11	10:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200	GTO	SES EchoStar	Success	Success (drone ship)

Task 7

List the total number of successful and failure mission outcomes

```
%sql select COUNT(mission_outcome) as outcome from SPACEXTBL3 where mission_outcome = 'Success'
```

outcome
99

EDA with SQL

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql select MAX(payload_mass__kg_) as max_mass from SPACEXTBL3
```

max_mass
15600

Task 9

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql select * from SPACEXTBL3 where landing__outcome like 'Failure (drone ship%' and DATE like '2015%' \
order by DATE ASC LIMIT 10
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2015-01-10	09:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-04-14	08:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)

EDA with SQL

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
%sql select DATE, landing__outcome from SPACEXTBL3 where DATE between '2010-06-04' and '2017-03-20' \
order by DATE ASC LIMIT 100
```

DATE	landing__outcome
2010-06-04	Failure (parachute)
2010-12-08	Failure (parachute)
2012-05-22	No attempt
2012-10-08	No attempt
2013-03-01	No attempt
2013-09-29	Uncontrolled (ocean)
2013-12-03	No attempt
2014-01-06	No attempt
2014-04-18	Controlled (ocean)

EDA with SQL

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
%sql select DATE, landing__outcome from SPACEXTBL3 where DATE between '2010-06-04' and '2017-03-20' \
order by DATE ASC LIMIT 100
```

DATE	landing__outcome
2010-06-04	Failure (parachute)
2010-12-08	Failure (parachute)
2012-05-22	No attempt
2012-10-08	No attempt
2013-03-01	No attempt
2013-09-29	Uncontrolled (ocean)
2013-12-03	No attempt
2014-01-06	No attempt
2014-04-18	Controlled (ocean)

Visual Analytics - Interactive Map with Folium

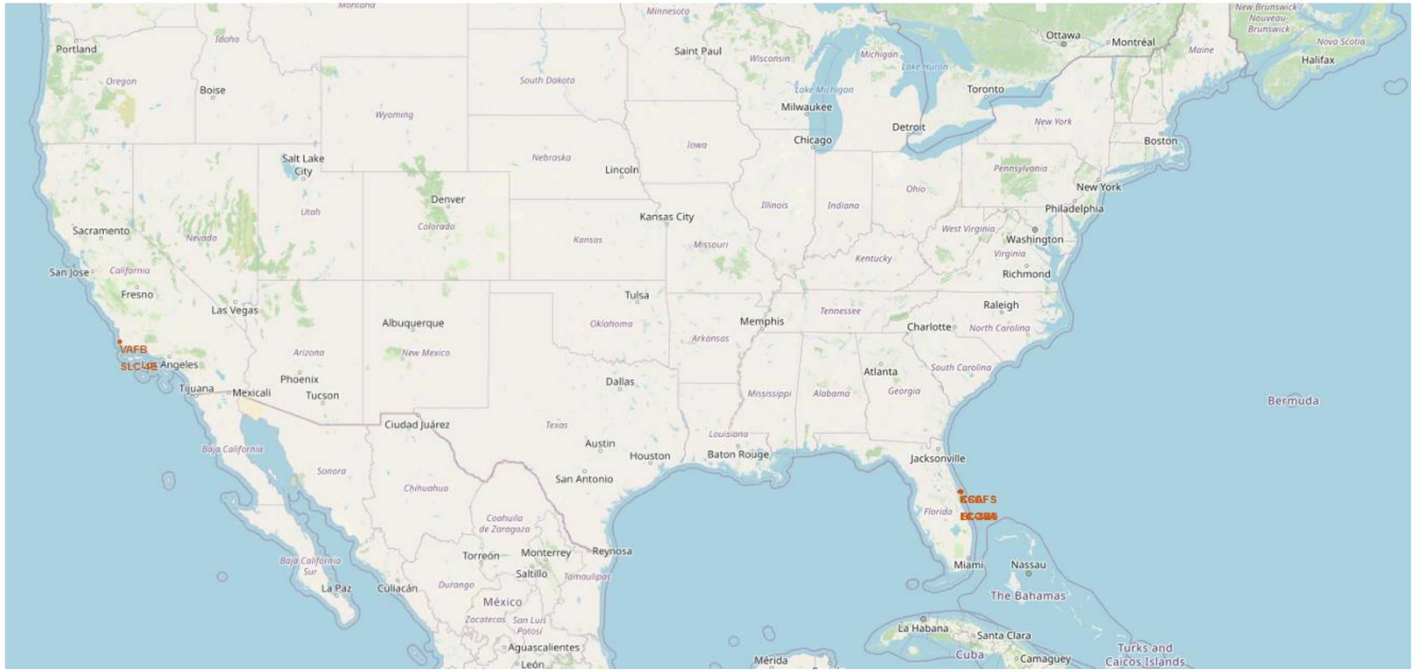
- In the previous exploratory data analysis section, we have visualized the SpaceX launch dataset using matplotlib and seaborn and discovered some preliminary correlations between the launch site and success rates. In this section, we will be performing more interactive visual analytics using Folium.

- [Git Hub Link to the notebook]

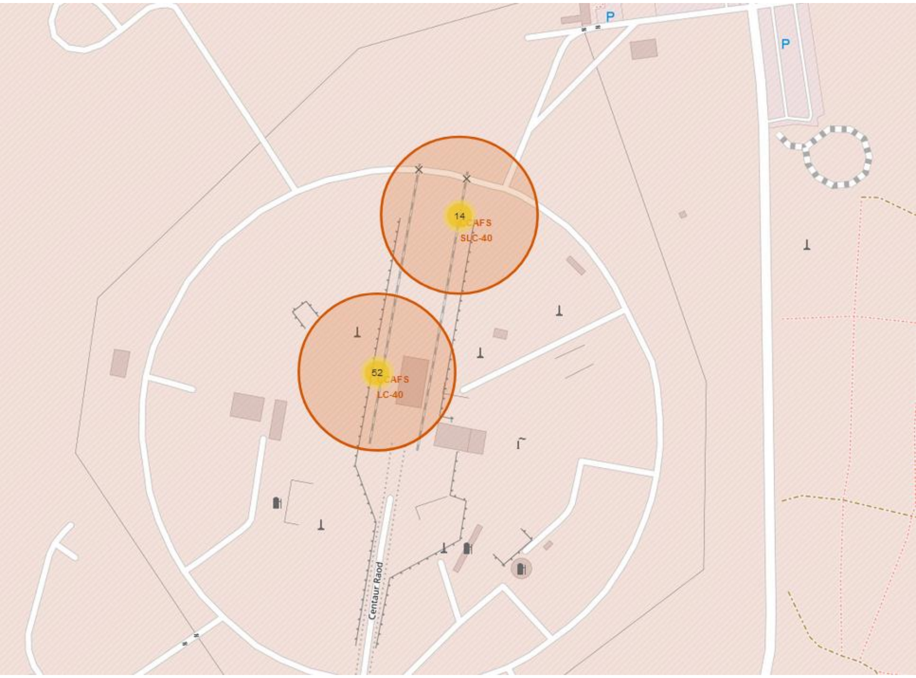
Visual Analytics - Interactive Map with Folium

Task 1: Mark all launch sites on a map

First, let's try to add each site's location on a map using site's latitude and longitude coordinates

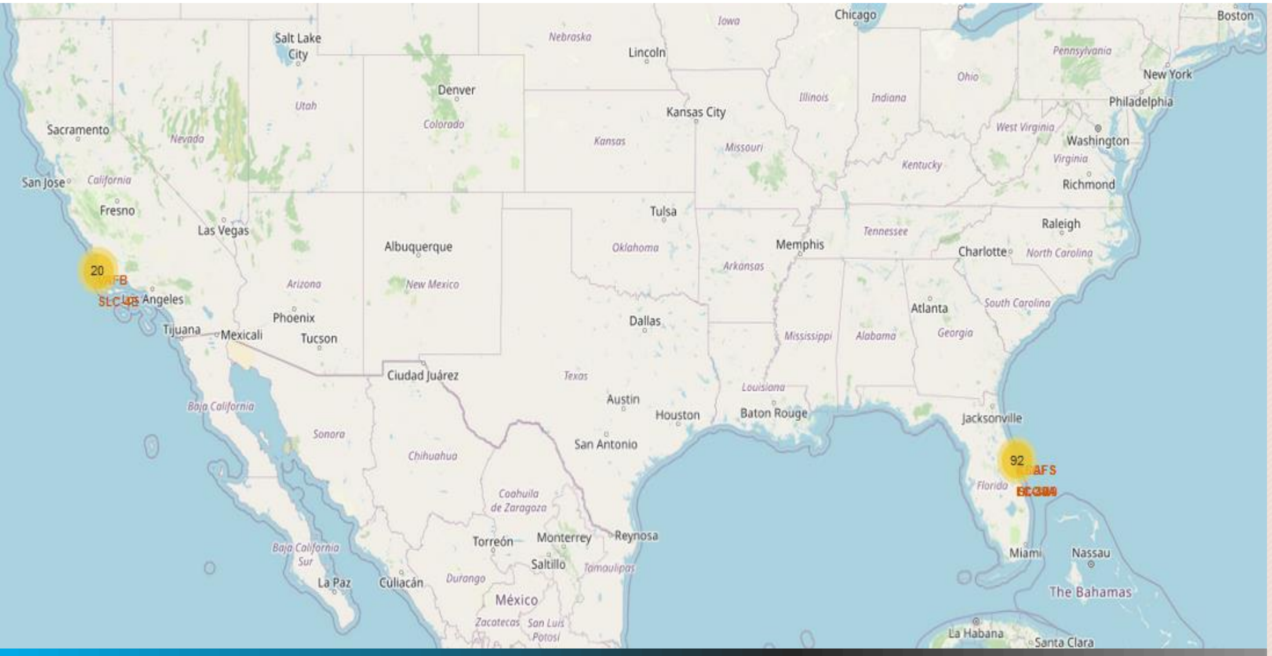


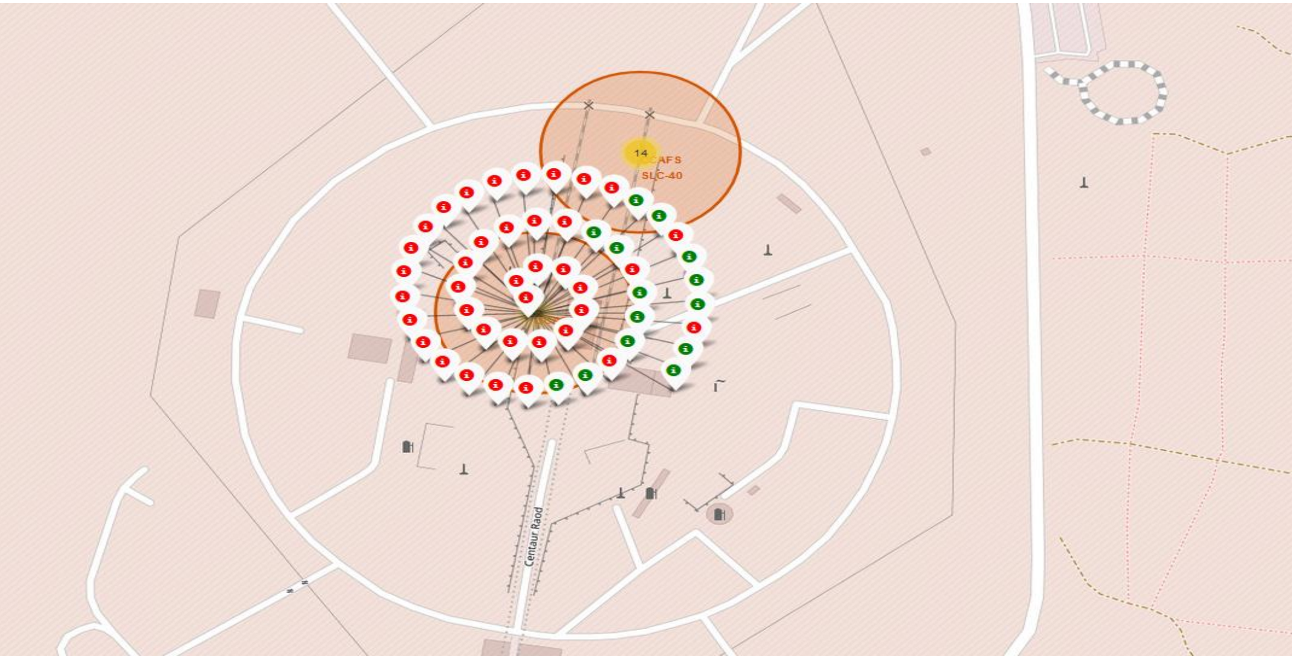
Visual Analytics - Interactive Map with Folium



Task 2: Mark the success/failed launches for each site on the map

Next, let's enhance the map by adding the launch outcomes for each site, and see which sites have high success rates. Recall that data frame spacex_df has detailed launch records, and the class column indicates if this launch was successful or not

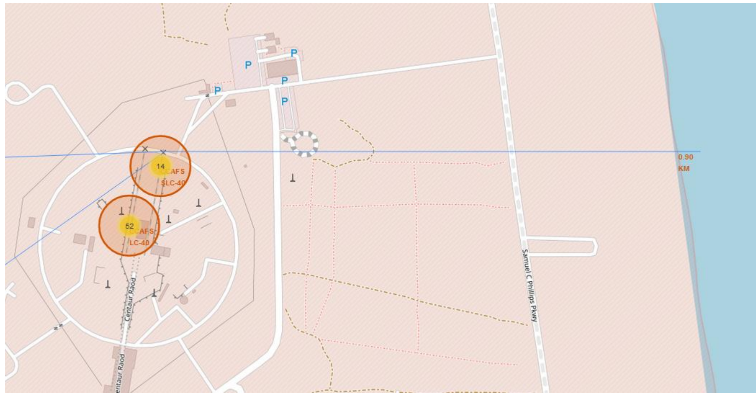




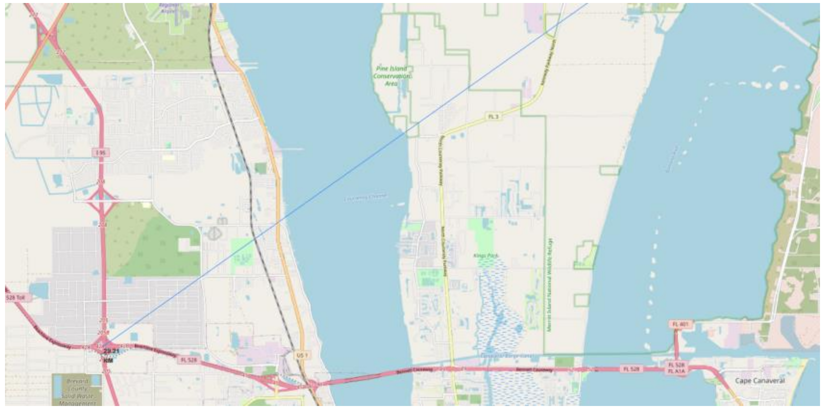
Visual Analytics - Interactive Map with Folium

TASK 3: Calculating the distances between a launch site to its proximities

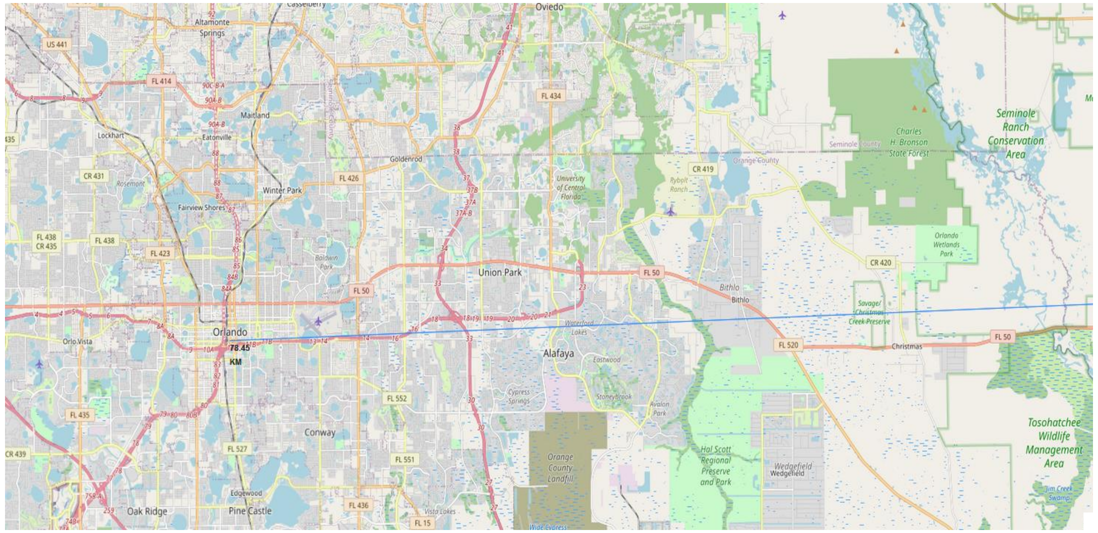
a) distance between the coastline point and the launch site.



b) distance to highway



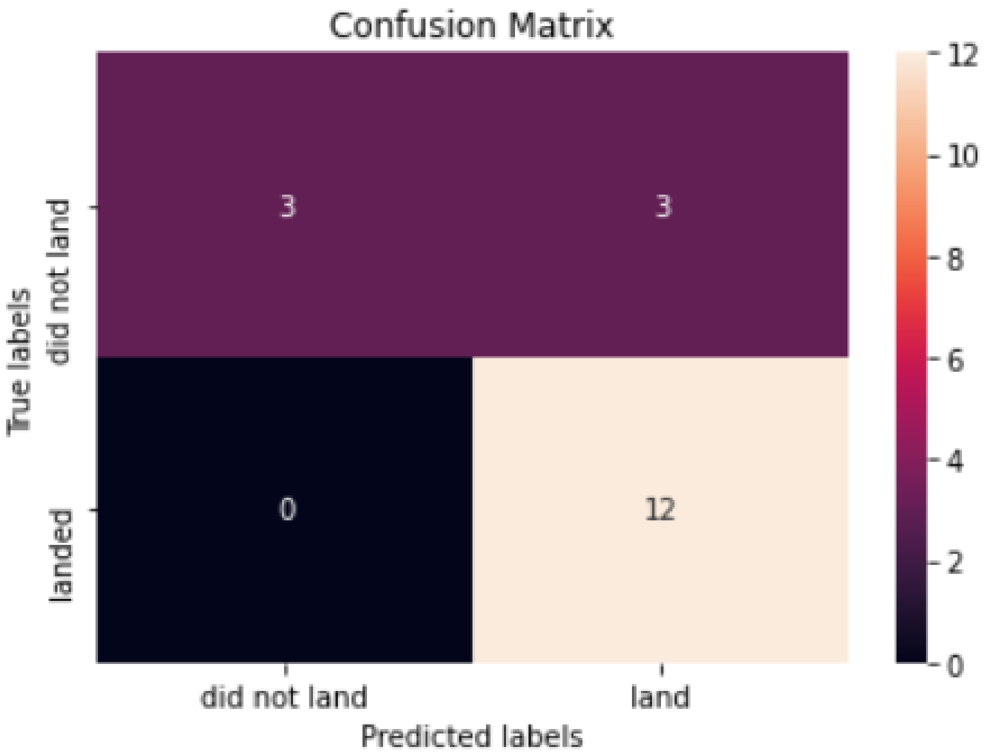
c) distance to Orlando



Predictive Analysis (classification)

- In this section, we will perform Data Analysis and determine Training Labels. Also, we will find the best hyperparameter for SVM, Classification Trees and Logistic Regression.

- [Git Hub Link to the notebook]()



Predictive Analysis (classification)

Finding the best hyperparameter for SVM

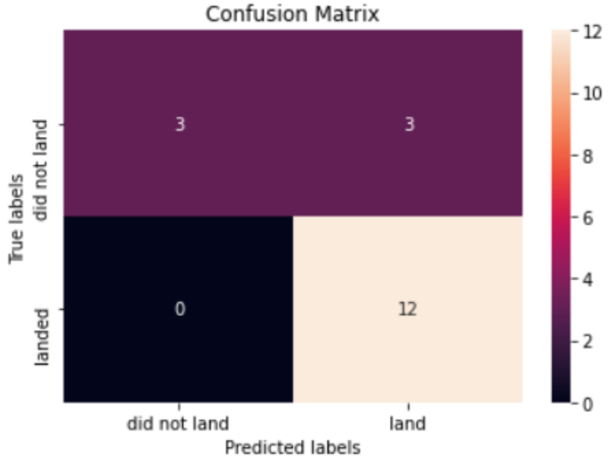
Calculate the accuracy on the test data using the method `score` :

```
print("accuracy: ",svm_cv.score(X_test,Y_test))
```

```
accuracy:  0.8333333333333334
```

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Finding the best hyperparameter for classification trees

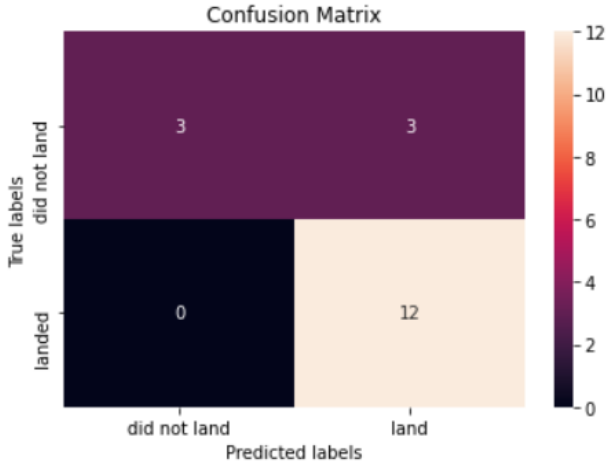
Calculate the accuracy of tree_cv on the test data using the method `score` :

```
print("accuracy: ",tree_cv.score(X_test,Y_test))
```

```
accuracy:  0.7777777777777778
```

We can plot the confusion matrix

```
yhat = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Predictive Analysis (classification)

Finding the best hyperparameter for KNN

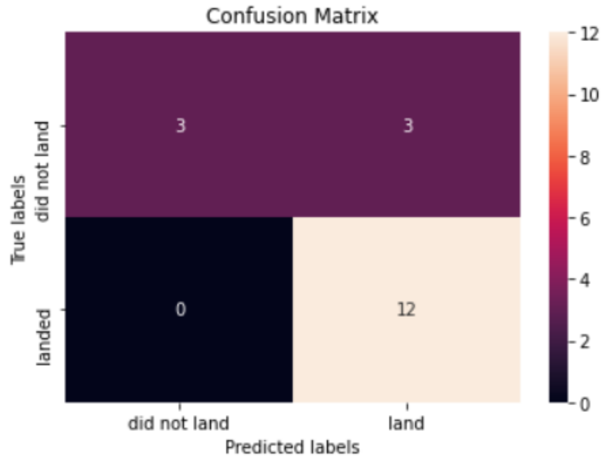
Calculate the accuracy of tree_cv on the test data using the method `score` :

```
print("accuracy: ",knn_cv.score(X_test,Y_test))
```

```
accuracy:  0.8333333333333334
```

We can plot the confusion matrix

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Finding the best method

Find the method performs best:

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8875
Best Params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
```

Conclusion

1. The best success rate has Orbit GEO, HEO, SSO, ES-L1

2. As the years go by, the launches are more and more successful which means the SpaceX has ben able to learn from mistakes and continuously improve

3. The most success launches are from KSC LC-39A location

4. Lower the payload, the better the outcome of the launch

5. The best machine learning algorithm is Tree Classification

Thank You

Michal Kovac