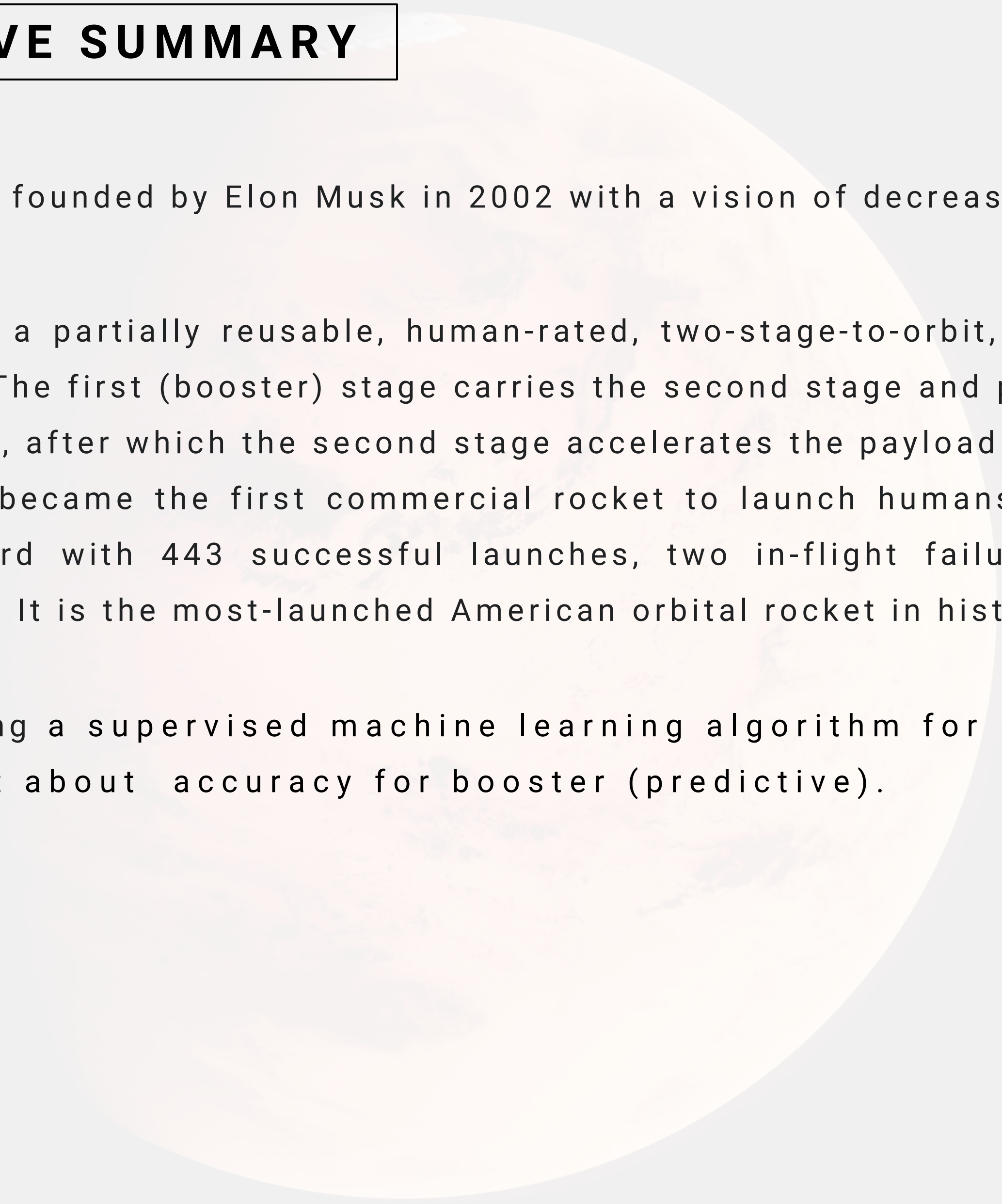
ANALYZING SPACEX LAUNCH SUCCESSES: A DATA-DRIVEN APPROACH

SHUBHAM VIGONIYA

CONTENTS

EXECUTIVE SUMMARY



SpaceX was founded by Elon Musk in 2002 with a vision of decreasing the costs of space launches.

Falcon 9 is a partially reusable, human-rated, two-stage-to-orbit, medium-lift launch vehicle. The rocket has two stage. The first (booster) stage carries the second stage and payload to payload to a predetermined speed and altitude, after which the second stage accelerates the payload to its target orbit
In 2020, it became the first commercial rocket to launch humans to orbit. The Falcon 9 has an exceptional safety record with 443 successful launches, two in-flight failures, one partial failure and one pre-flight destruction. It is the most-launched American orbital rocket in history.

By leveraging a supervised machine learning algorithm for regression and classification model that result about accuracy for booster (predictive).

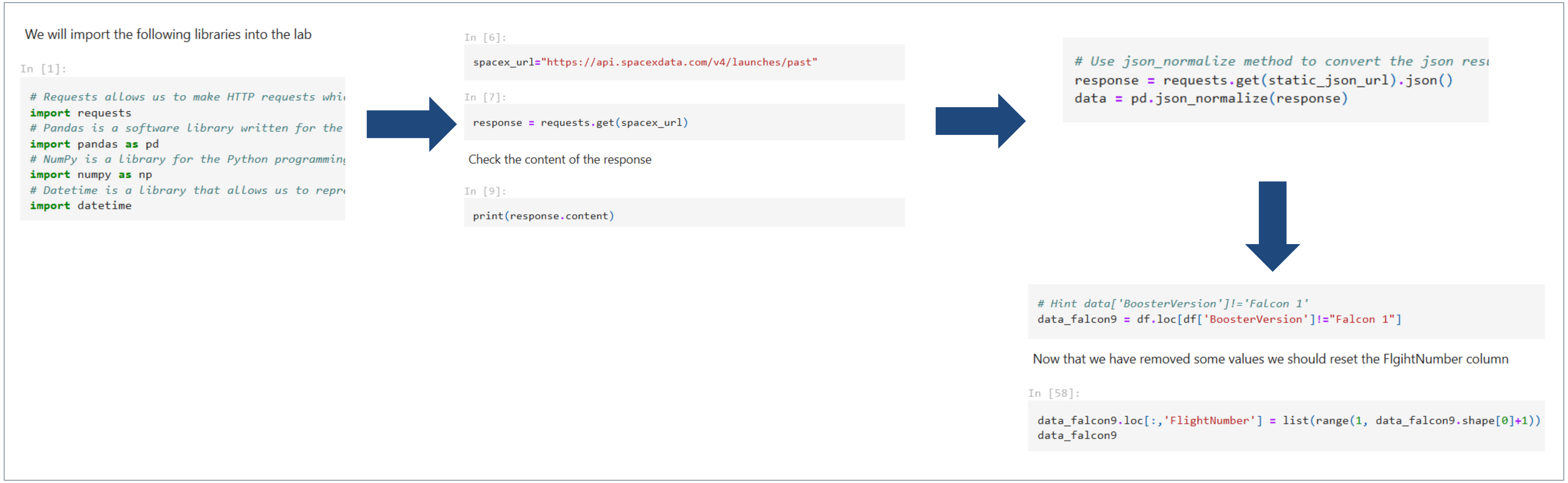
INTRODUCTION

In this project, I 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 I can determine if the first stage will land, I can determine the cost of a launch.

In this capstone, I will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website.

DATA COLLECTION

Data collected using SpaceX REST API by making a get request to the SpaceX API then parsed the SpaceX launch data using the GET request and decode the response content as a Json using .json() and turn it into a Pandas data frame using .json_normalize(). Filter the data frame only include Falcon 9 lauches.



We will import the following libraries into the lab

In [1]:

```
# Requests allows us to make HTTP requests whi
import requests
# Pandas is a software library written for the
import pandas as pd
# NumPy is a library for the Python programmin
import numpy as np
# Datetime is a library that allows us to repr
import datetime
```

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 [9]:

```
print(response.content)
```

```
# Use json_normalize method to convert the json resu
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

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

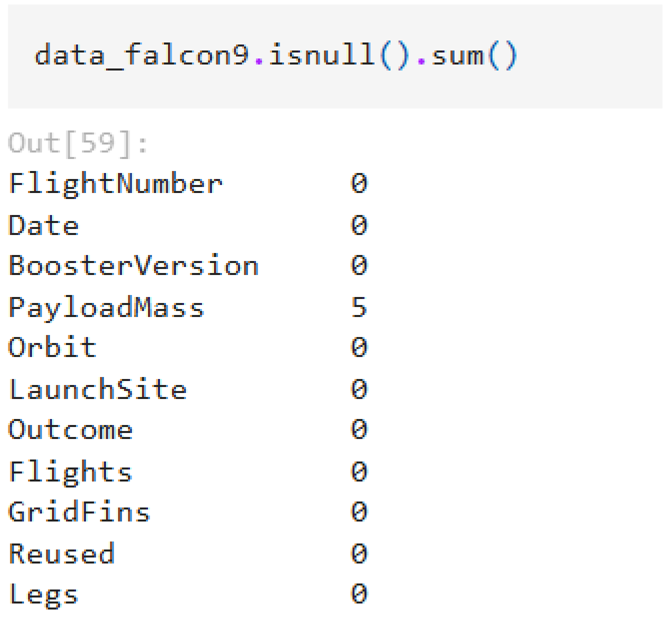
Now that we have removed some values we should reset the FlgihtNumber column

In [58]:

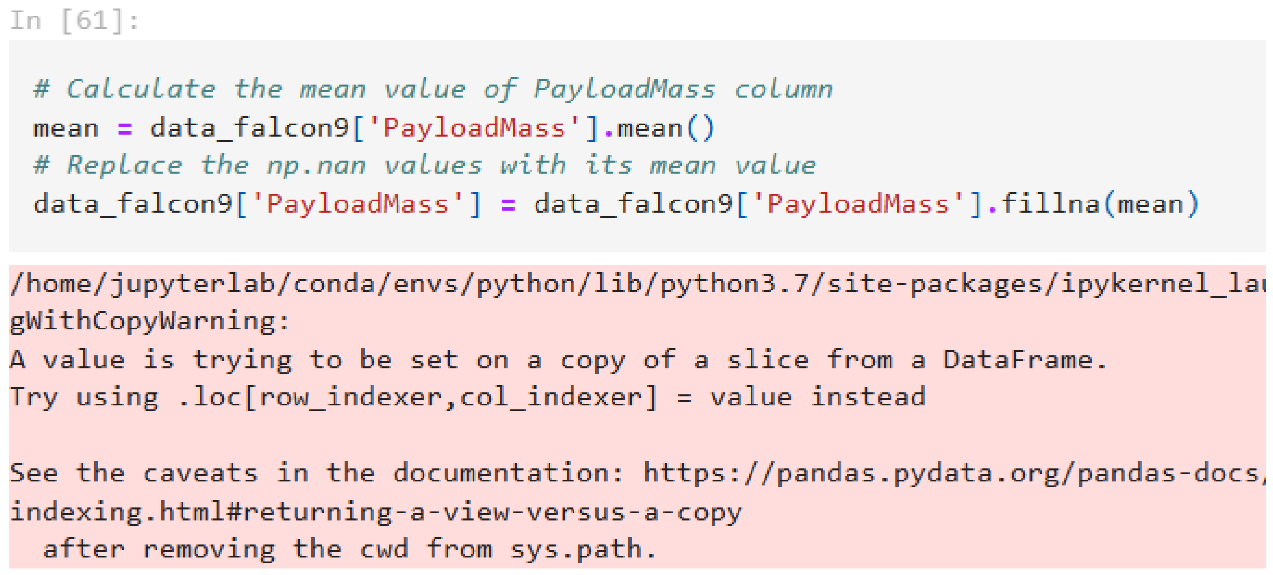
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

DATA WRANGLING

Now I have the dataset to work with, applying some essential checks such as row with missing values in the dataset. Then use *.mean() function to get the mean of* ‘PayloadMass’ column and the .replace() function to replace np.nan values in the data with the mean we have calculated.



```
data_falcon9.isnull().sum()

Out[59]:
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       5
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
```

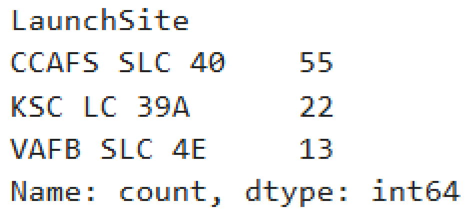


```
In [61]:

# 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)

/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/ipykernel_la
gWithCopyWarning:
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
indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

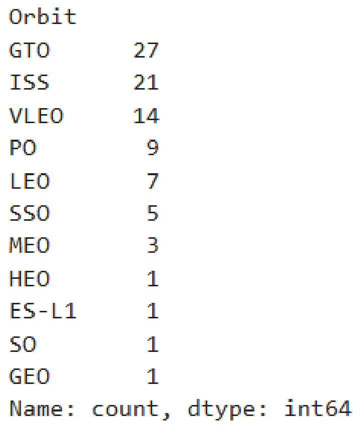
EXPLORATORY DATA ANALYSIS AND VISUALIZATION

I will perform some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models with Feature Engineering to use ‘OneHotEncoder’ to convert those variable that would affect the success rate .

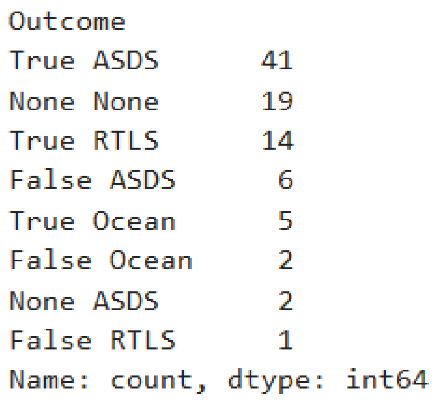
1. Calculate the number of launches on each sites: use the method .value_counts() to determine the number of launches on each sites.
2. Calculate the number and occurrence of each orbit: Same as 1.
3. Calculate the number and occurrence of mission outcomes of the orbit: Same as 1.
4. Create a landing outcome label from outcome column where ‘0’ indicates that the first stage did not land successfully; one means the first stage landed successfully. By taking the mean of the new ‘Class’ column, the success rate is 0.66.



```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```



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



```
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: count, dtype: int64
```

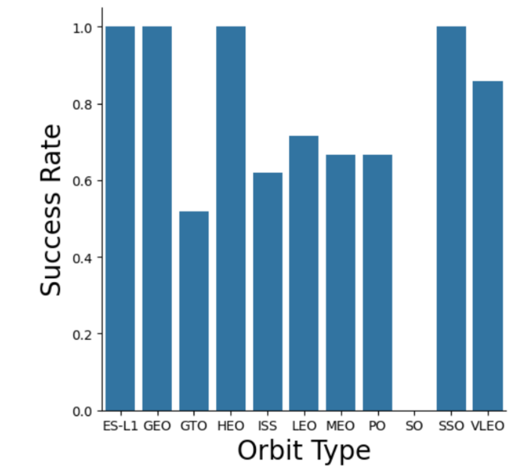
VISUALIZATION

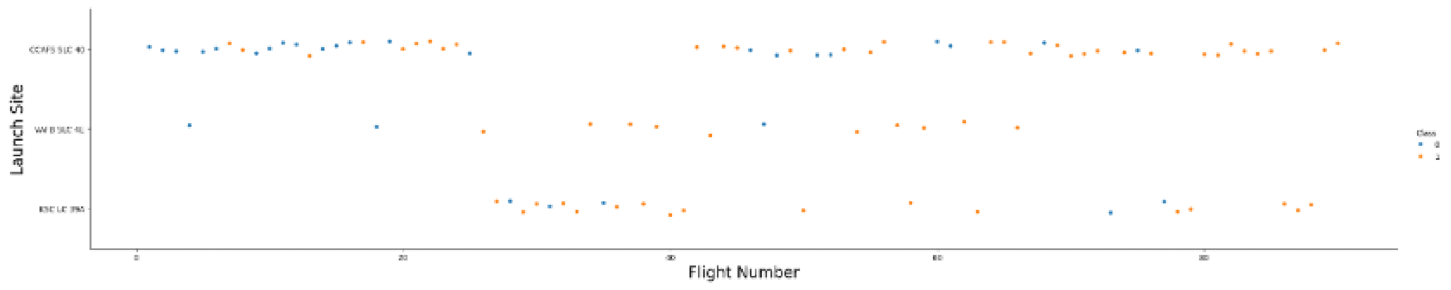
Why it is important?

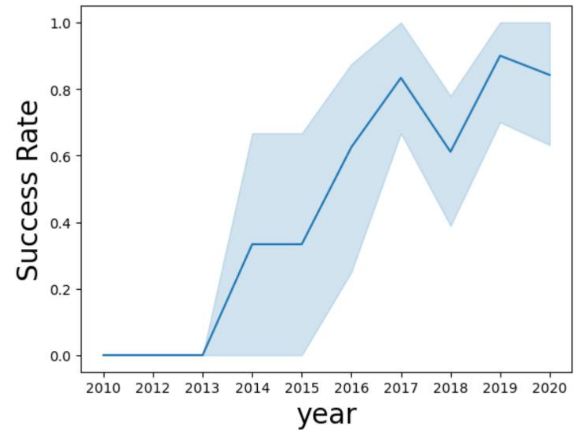
Visualization is a crucial part of Exploratory Data Analysis (EDA) because it allow us to visually identify patterns, trends, outlier and relationship between variable in the dataset.

- Visualize the relationship between 'Flight Number' and 'Launchsite'.
- Visualize the relationship between 'PayloadMass' and 'Launch Site'.
- Visualize the relationship between success rate of each orbit type.
- Visualize the relationship between 'FlightNumber' and Orbit type.
- Visualize the relationship between 'Payload Mass' and Orbit Type.
- Visualize the launch success yearly trend.

Libraries used: Matplotlib, Seaborn, Pandas, Numpy







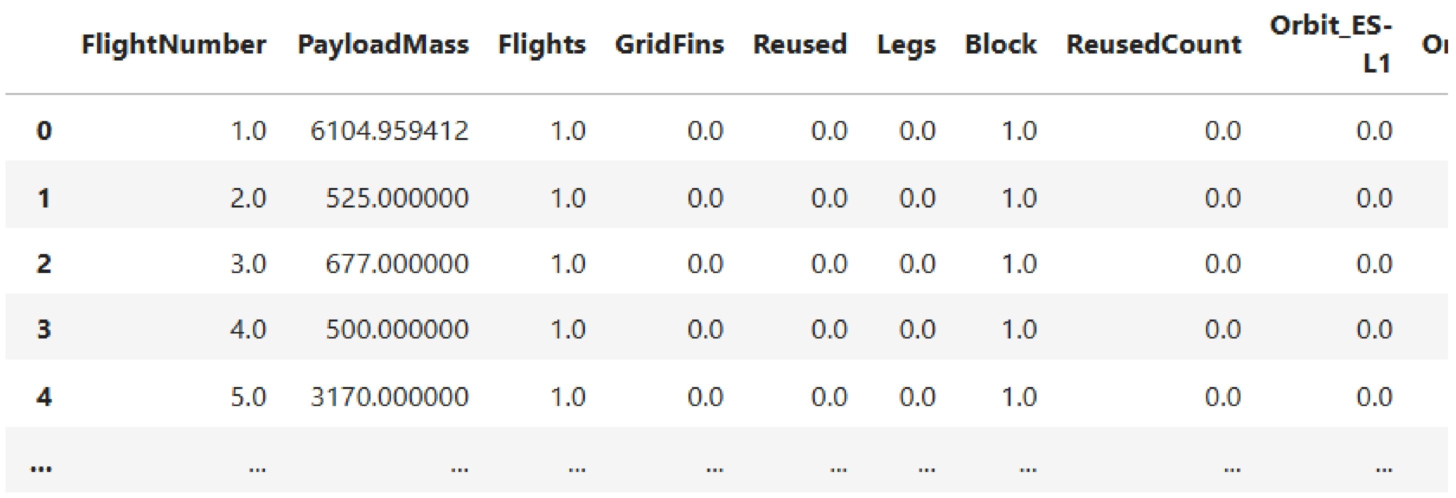
FEATURE ENGINEERING

Why it is important?

The idea of feature engineering is we need to obtain insight of how each important variable would affect the success rate, so selecting those features that will be used in success prediction.

Methods:

1. Create dummy variable and apply ‘OneHotEncoder’ to the column ‘Orbit’, ‘LaunchSite’, ‘LandingPad’ and ‘Serial’.
2. Cast all these column (OneHotEncoder) to float data type (float64).



	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Or
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	0.0	0.0	
1	2.0	525.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	
2	3.0	677.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	
3	4.0	500.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	
4	5.0	3170.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	

OneHotEncoder
Orbit
LaunchSite
LandingPad
Serial

EXPLORATORY DATA ANALYSIS AND SQL

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper.

Establish the connection with database and load the tabular data with ‘prettytable’ then convert into pandas data frame. The data frame is ready to fetch results based on SQL queries. The following task executed by leveraging SQL is listed.

QUERY 1-5

1. Display the name of the unique launch sites in the space mission.

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
```

2. Display 5 records where launch sites begin with the string ‘CCA’.

```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

3. Display the total payload mass carries by booster launched by NASA (CRS).

```
%sql SELECT sum(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer == 'NASA (CRS)';
```

4. Display average payload mass caried by booster version F9 v1.1.

```
%sql SELECT avg(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1%';
```

5. List of date when the first successful landing outcome in ground pad was achieve.

```
%sql SELECT min(Date) FROM SPACEXTBL WHERE Landing_Outcome == 'Success (ground pad)';
```

https://www.sqlalchemy.org/

https://pypi.org/project/prettytable/

QUERY 6-10

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

```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 40000 AND 6000;
```

7. List the total number of successful and failure mission outcomes.

```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) FROM SPACEXTBL GROUP BY Mission_Outcome;
```

8. List the names of the ‘Booster_Version’ which have carried the maximum payload mass. Use a subquery.

```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ == (SELECT max(PAYLOAD_MASS__KG_)FROM SPACEXTBL);
```

9. List the records which will display the month names, failure ‘Landing_Outcomes’ in drone ship, booster versions, ‘Launch_Sites’ for the month in year 2015.

```
%sql SELECT substr(Date,6,2) as month, Date,Booster_Version, Launch_site, [Landing_Outcome] FROM SPACEXTBL
    WHERE [Landing_Outcome] = 'Failure (drone ship)' AND substr(Date,0,5)='2015';
```

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 Landing_Outcome , COUNT(*) as Count_Outcome FROM SPACEXTBL   WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
    GROUP BY Landing_Outcome ORDER BY Count_Outcome DESC
```

INTERACTIVE VISUALIZATION WITH FOLIUM

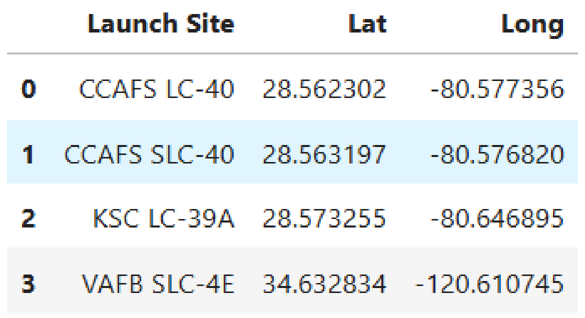
Leveraging folium library in python to visualize each site’s location on map using site’s latitude and longitude coordinates. Building a launch site certainly involves many factor(essential to check) which is listed.

1. Proximity to railways.
2. Proximity to highways
3. Proximity to coastline.
4. Do launch sites keep certain distance away from cities.

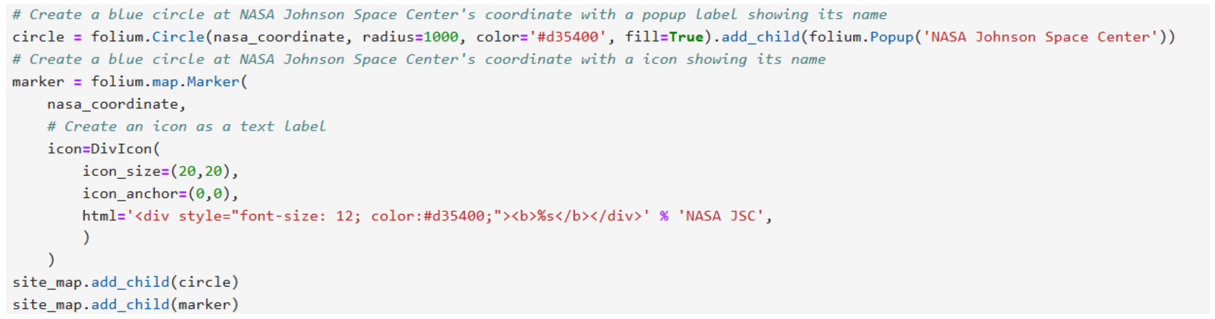
The work is divided into three task.

Task 1: Mark all launch sites on a maps

Embedding coordinate for each sites to visualize those location by pinning them on map. Add markers to a Folium map by specifying their latitude and longitude coordinates.



	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

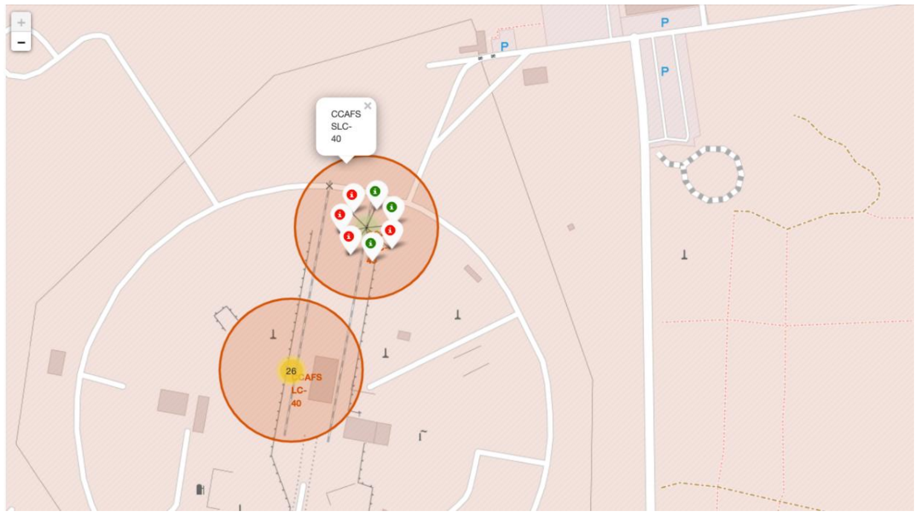


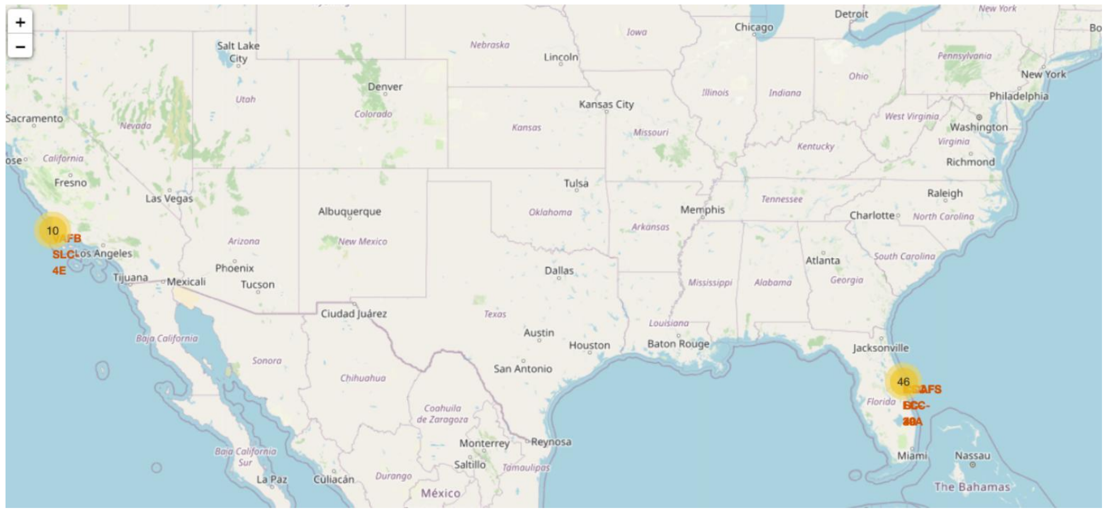
```python
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

Task 2

Mark the success/failed launches for each site on the map.

First create the marker_cluster object and add new column ‘class’ which takes Boolean value such as 0 and 1. If a launch was successful (class=1), then we use a green marker and if a launch was failed, we use a red marker (class=0). By identifying the color-labeled we can clearly observe which sites have relatively high success rates.



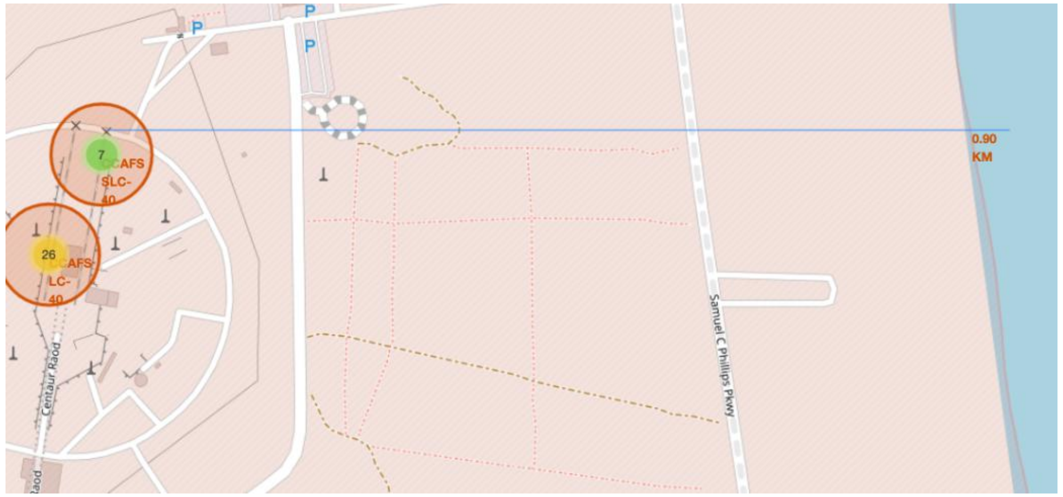


https://python-visualization.github.io/folium/latest/

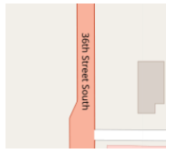
Task 3

Calculate the distance between a launch site to its proximities.

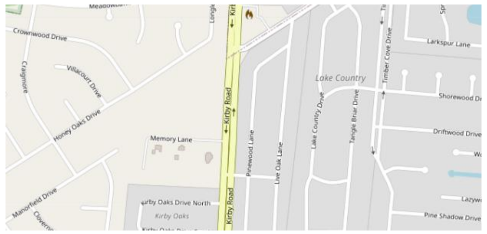
By adding *MousePosition* we can zoom in to a launch site and explore its proximity to see if you can easily find any railways, highway, coastline, etc. Get those coordinates to calculate the distance to the launch site. In the project I calculate the distance from coastline to the launch site and create a polyline to represent the distance on the map. The distance is 0.90 km.



Railway



Highway

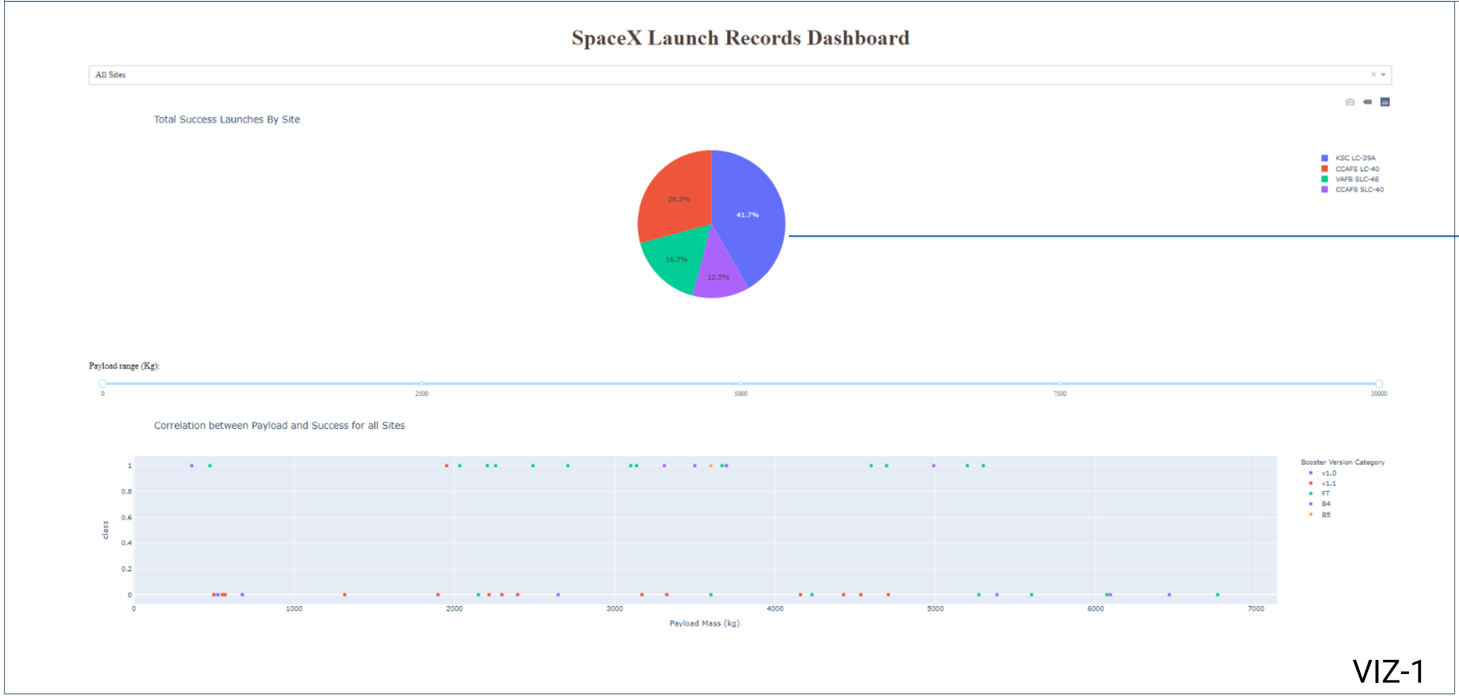

City map

PLOTLY DASH APPLICATION

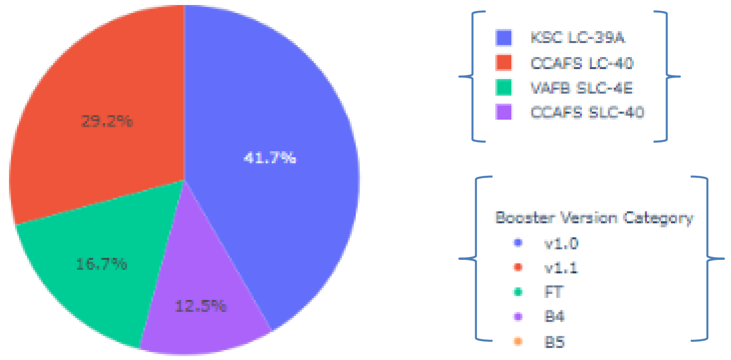
A Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time.

This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart. You will be guided to build this dashboard application via the following tasks:

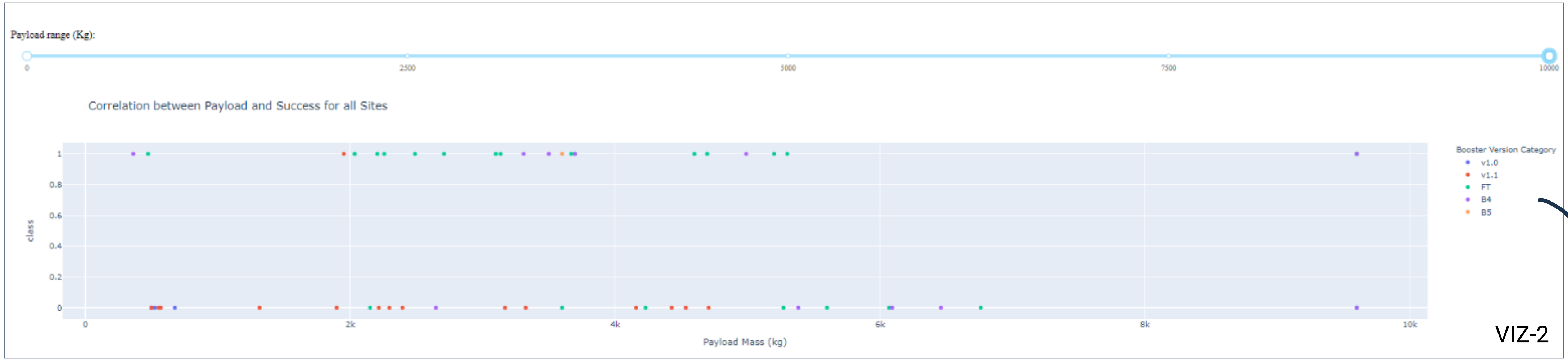
- **TASK 1**: Add a Launch Site Drop-down Input Component.
- **TASK 2**: Add a callback function to render success-pie-chart based on selected site dropdown.
- **TASK 3**: Add a Range Slider to Select Payload.
- **TASK 4**: Add a callback function to render the success-payload-scatter-chart scatter plot.



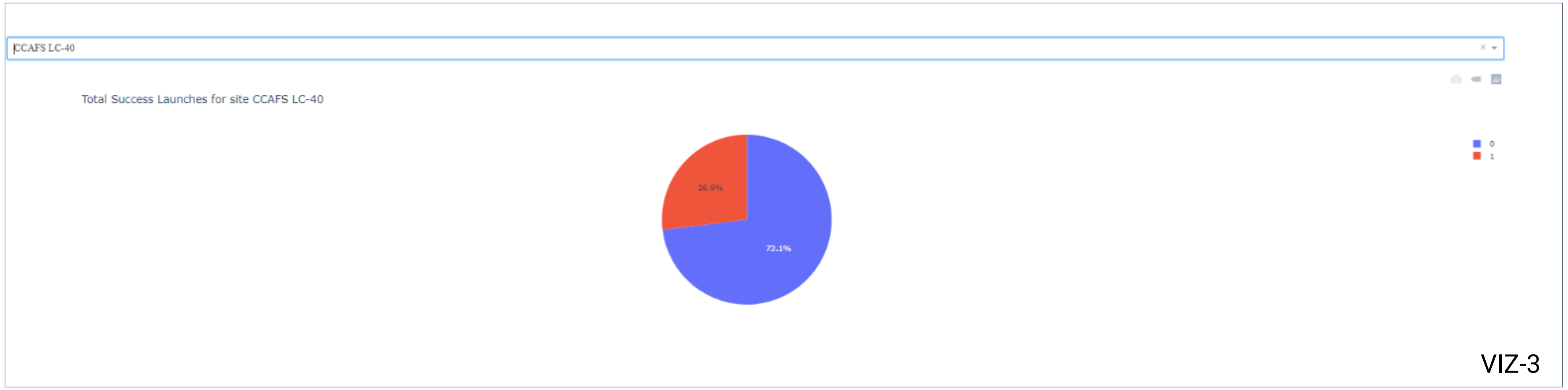
VIZ-1



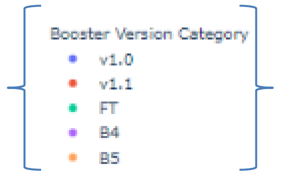
VISUALIZATION 2-3



VIZ-2



VIZ-3



https://plotly.com/

PREDICTIVE ANALYSIS (CLASSIFICATION)

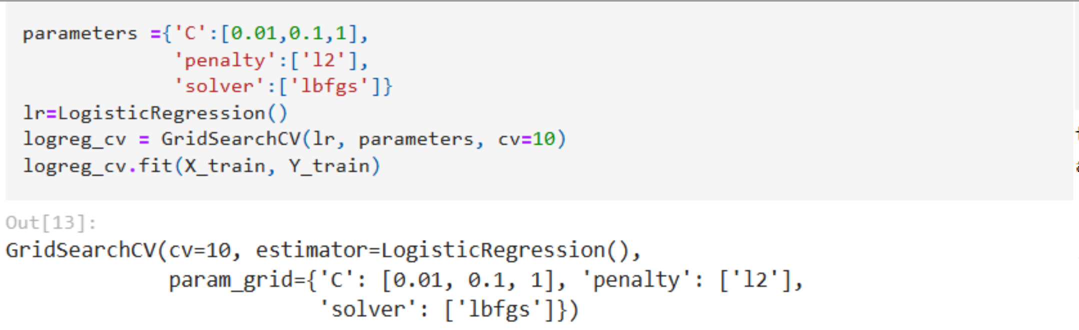
The Classification supervised machine learning model which predict if the first stage will land based on the given data (features). Therefore, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch, As other providers cost upward of 165 million dollars each, much of saving is because SpaceX can reuse the first stage.

The Falcon 9 rocket launches with a cost of 62 million dollars;

Library: Numpy, Pandas, Seaborn, Matplotlib, Sklearn.

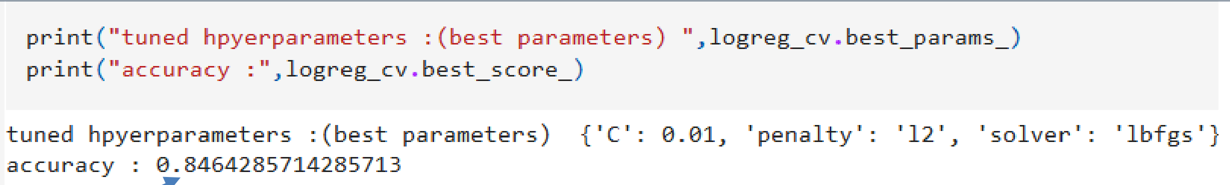
TASK 1-4

- Loading the data as a pandas data frame by using *.read_csv()*.
- Splitting the data frame into X and Y, where X represent the features and Y represent the target variable "Class". Convert Y to numpy ndarray utilizing *.to_numpy()* method.
- Standardize the data X using *.StandardScaler()*.
- Splitting the data in train and test set such that *X_train, Y_train, X_test, Y_test*.
- Create a logistic regression object then create GridSeachCV object with cv = 10 and parameters and fit the object as *.fit()*.



```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

```
Out[13]:
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```



```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

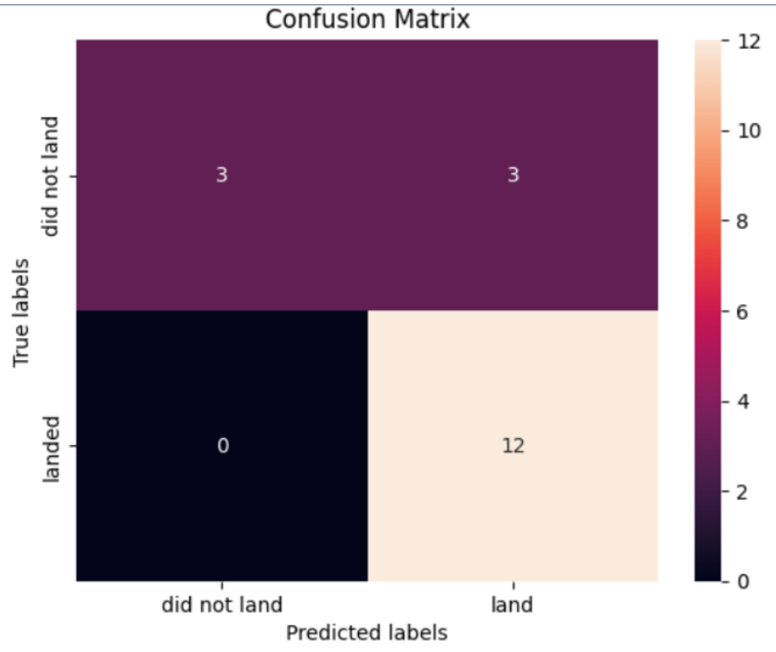
```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

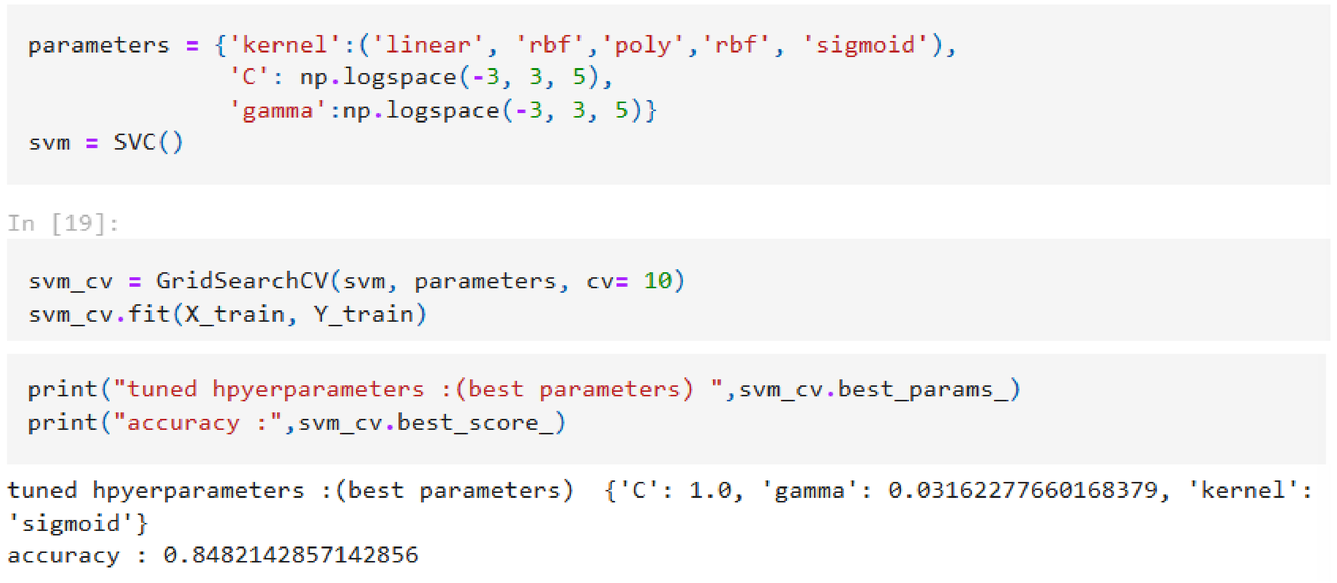
Accuracy on train set

PREDICTIVE ANALYSIS (CLASSIFICATION)

TASK 5 – 7: Logistic Regression and Support Vector Machine

- Calculate the accuracy on the test data using method score (inbuilt function in sklearn library).
- Plot the confusion matrix.
- Create a logistic regression object and GridSearchCV with cv =10, , then fit the object on given parameters. Train and test accuracy estimated as **0.8464**, **0.8333**.
- Create support vector machine object the create GridSearchCV object on train set with estimator = *SVC()*, cv = 10, then fit the object on given parameters. Train and test accuracy estimated as **0.848214**, **0.848214**.





```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
In [19]:
```

```
svm_cv = GridSearchCV(svm, parameters, cv= 10)
svm_cv.fit(X_train, Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel':
'sigmoid'}
accuracy : 0.8482142857142856
```

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

PREDICTIVE ANALYSIS (CLASSIFICATION)

TASK 8 – 9: Decision Tree Classifier

- Create a decision tree classifier object then create GridSearchCV object with cv = 10 , fit the model based on given parameter. In the criterion used as 'gini' and 'entropy' as impurity measuring metrics.
- Accuracy on train and test set is **0.873214** and **0.873214**.

TASK 10 – 11 : K-Nearest Neighbors

- Similarly using GridSearchCV object, K nearest Neighbors classification fit the model and find accuracy for train and test data.
- Accuracy of test and train using k nearest neighbors is **0.848214**, **0.848214**.

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

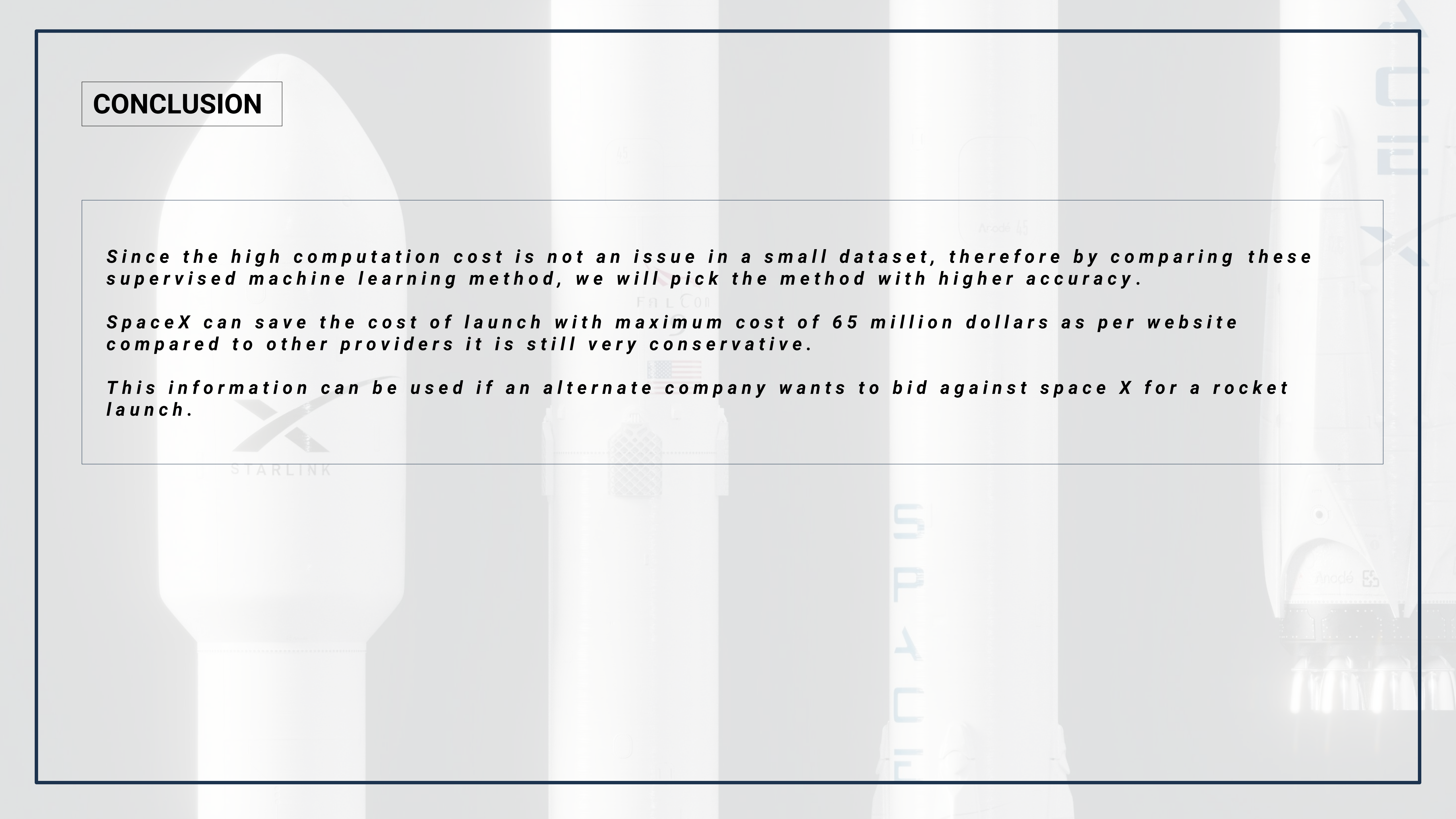
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

PREDICTIVE ANALYSIS (Result)

TASK 12: Best Supervised Machine Learning method.

Idea: Choosing the method based on accuracy of the model

Supervised machine learning methods	Accuracy
Logistic Regression	0.833
Support Vector Machine	0.848214
Decision Tree Classifier	**0.873214** 👍
K- Nearest Neighbors	0.848214



CONCLUSION

Since the high computation cost is not an issue in a small dataset, therefore by comparing these supervised machine learning method, we will pick the method with higher accuracy.

SpaceX can save the cost of launch with maximum cost of 65 million dollars as per website compared to other providers it is still very conservative.

This information can be used if an alternate company wants to bid against space X for a rocket launch.

THANK YOU