

UPASANA DASS

DATA SCIENCE CAPSTONE PROJECT

August 20, 2021

OUTLINE

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





EXECUTIVE SUMMARY

- Summary Of Methodologies
- Summary Of All Results

INTRODUCTION

Founded in 2002, SpaceX has made its name in the space technology with its advanced rockets and spacecraft. Falcon 9 rocket launch has been estimated at a hefty cost of \$62 million dollars, which still seems affordable compared to the cost of \$165 million dollars proposed by other providers.

This capstone project focuses on identifying the cost and predicting the successful launch of SpaceX Falcon 9 first stage.





Data Collection:

Used SpaceX API to extract the SpaceX launch data using ET request.



Data Wrangling to identify:

Missing information

Numerical and categorical information

Total launches per site, number and occurrence of each orbit with respective mission outcomes



Perform EDA using:

Used Db2 in IBM Watson to load dataset and perform SQL based EDA.

Used Numpy, Pandas, Matplotlib and Seaborn library to perform EDA based visualization

METHODOLOGY



Performed Interactive Visual Analytics Using Folium And Plotly Dash

Used Folium to identify launch sites (success/failed), to explore and analyze coordinates of close proximities, plot distance lines to these proximities.
Used Plotly Dash to create interactive dashboard to show the findings.



Performed Predictive Analysis Using Classification Models:

Created ML pipeline, standardized the data, split dataset into training and test data
Calculated best Hyperparameter for SVM, Classification Trees and Logistic Regression models

METHODOLOGY CONTINUED..

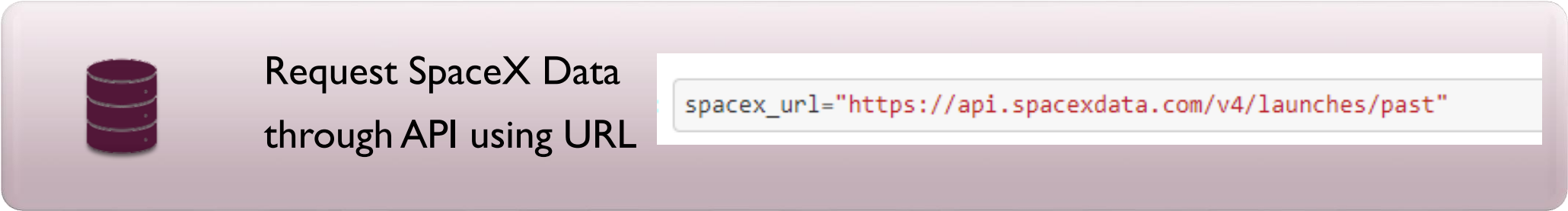



METHODOLOGY EXPLAINED

SUMMARY : SPACEX API DATA COLLECTION

- SpaceX API was used to collect official launch dataset, which included information such as booster name, launchpad, latitude/longitude, orbits, cores, mission outcomes etc.
- With the GET request to the API, we requested the JSON data, decoded the response content and stored in a DataFrame after using .json_normalize method.
- [GitHub URL](#)

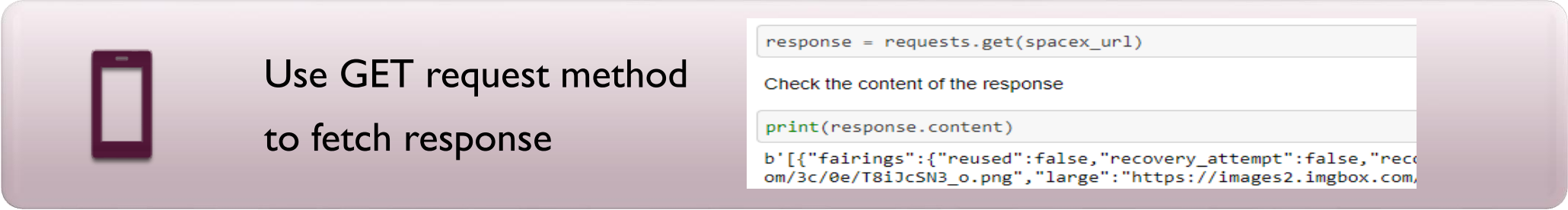
Responsibility	Percentage
Current government	65%
Previous government	25%
Neither	10%




 Request SpaceX Data through API using URL

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

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





Use GET request method to fetch response

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

Check the content of the response

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

b'[{ "fairings": { "reused": false, "recovery_attempt": false, "reco
om/3c/0e/T8iJcSN3_o.png", "large": "https://images2.imgbox.com,

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

Check the content of the response

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

```
b' [{"fairings": {"reused": false, "recovery_attempt": false, "recovery_vehicle": "Orion", "status": "Success"}, "flight": "Orion ES1", "mission": "Orion ES1", "name": "Orion ES1", "orbit": "Orion ES1", "payload": "Orion ES1", "stage": "Orion ES1", "version": "Orion ES1", "large": "https://images2.imgbox.com/3c/0e/T8iJcSN3_o.png"}]
```

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

Check the content of the response

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

```
b' [{"fairings": {"reused": false, "recovery_attempt": false, "recovery_vehicle": "Orion", "status": "Success"}, "flight": "Orion ES1", "mission": "Orion ES1", "name": "Orion ES1", "orbit": "Orion ES1", "payload": "Orion ES1", "stage": "Orion ES1", "version": "Orion ES1", "large": "https://images2.imgbox.com/3c/0e/T8iJcSN3_o.png"}]
```

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

Check the content of the response

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

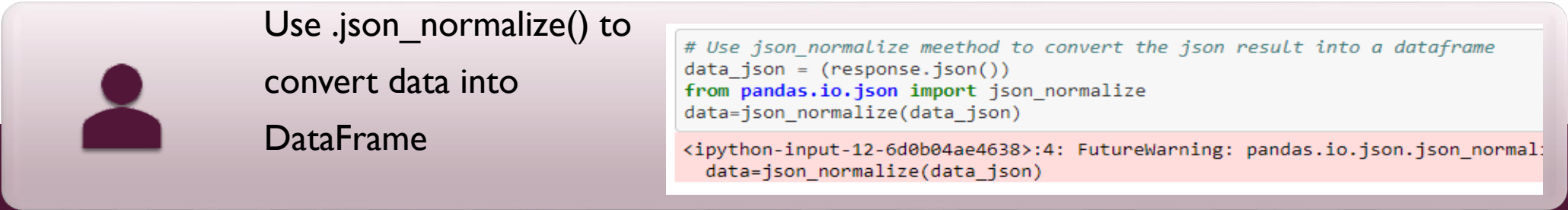
```
b' [{"fairings": {"reused": false, "recovery_attempt": false, "recovery_vehicle": "Orion", "status": "Success"}, "flight": "Orion ES1", "mission": "Orion ES1", "name": "Orion ES1", "orbit": "Orion ES1", "payload": "Orion ES1", "stage": "Orion ES1", "version": "Orion ES1", "large": "https://images2.imgbox.com/3c/0e/T8iJcSN3_o.png"}]
```

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

Check the content of the response

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

```
b' [{"fairings": {"reused": false, "recovery_attempt": false, "recovery_vehicle": "Orion", "status": "Success"}, "flight": "Orion ES1", "mission": "Orion ES1", "name": "Orion ES1", "orbit": "Orion ES1", "payload": "Orion ES1", "stage": "Orion ES1", "version": "Orion ES1", "large": "https://images2.imgbox.com/3c/0e/T8iJcSN3_o.png"}]
```



Use `.json_normalize()` to convert data into DataFrame

```
# Use json_normalize method to convert the json result into a dataframe
data_json = (response.json())
from pandas.io.json import json_normalize
data=json_normalize(data_json)
```

<ipython-input-12-6d0b04ae4638>:4: FutureWarning: pandas.io.json.json_normalize: data=json_normalize(data_json)

```
# Use json_normalize method to convert the json result into a dataframe
data_json = (response.json())
from pandas.io.json import json_normalize
data=json_normalize(data_json)
```

```
# Use json_normalize method to convert the json result into a dataframe
data_json = (response.json())
from pandas.io.json import json_normalize
data=json_normalize(data_json)
```

SUMMARY : SPACEX DATA WEB SCRAPING

- BeautifulSoup4, requests, re, unicodedata and pandas libraries have been used to perform web scraping for spacex data historical launch of Falcon 9 from the Wikipedia page titled List of Falcon 9 and Falcon Heavy Launches.
- The historical launch records are stored in HTML table on the Wikipedia page. The goal of web scraping is to parse the data in html table and convert it into a dataframe for further analysis.
- GitHub URL

SpaceX Web Scrapping Flowchart

Request data using URL

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



Use HTTP GET request method
to fetch response

```
httpresponse = requests.get(static_url)  
httpresponse  
  
<Response [200]>
```



Create BeautifulSoup object
From the HTML Response

```
soup=BeautifulSoup(httpresponse.content, 'html.parser')
```

SUMMARY : EDA WITH DATA VISUALIZATION

- Python's libraries such as Pandas, Numpy, Matplotlib and Seaborn were used for Exploratory Data Analysis.
- Using plotting libraries, i.e. Matplotlib and Seaborn, number of plots such as seaborn's catplot for scatterplot and matplotlib's plot function to plot a bar chart. Through these, relationship between different variables was successfully plotted.
- [GitHub URL](#)

SUMMARY : SPACEX DATA WRANGLING

- SpaceX dataset was explored using fundamental data manipulation and analysis techniques in Python.
- Key libraries used for Exploratory Data Analysis were Pandas and NumPy.
- With EDA, we determined the number of launches on each site, number of occurrence of each orbit, landing outcomes, and created landing outcome labels.
- [GitHub URL](#)

SUMMARY : SQL EDA ON SPACEX DATASET

- IBM Watson's DB2 was used to perform exploratory data analysis using SQL on SpaceX dataset, i.e. the .CSV file.
- After connecting with the DB2 using service credentials, sql queries were executed to retrieve the following information:
 - Unique Launch Sites in space mission
 - Launch Sites starting with certain characters such as 'KSC'
 - Total payload mass carried by boosters launched by customer
 - Average payload mass carried by certain boosters
 - Date of successful landing outcome in drone ship
 - Name of boosters with ground pad success having payload mass between 4000 and 6000
 - Total count of successful and failed mission outcomes
 - Booster versions that have carried maximum payload mass
 - List of successful landing outcomes in ground pad with launch sites, month names and booster versions for months of 2017
- [GitHub URL](#)

EDA USING SQL ON SPACEX DATASET

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

```
%sql select LAUNCH_SITE from SPACEXTBL where LAUNCH_SITE LIKE 'KSC%' fetch first 5 rows only;
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases
Done.
```

launch_site
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A

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

```
%sql select distinct LAUNCH_SITE from DBC93104.SPACEXTBL;
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.
Done.
```

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

List the total number of successful and failure mission outcomes

```
%sql select distinct (select count(*) from SPACEXTBL where Mission_Outcome like 'Success%') as Successful_Mission,(select count(*) from S
PACEXTBL where Mission_Outcome like 'Failure%') as Failed_Mission from SPACEXTBL ;
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

successful_mission	failed_mission
100	1

SUMMARY : INTERACTIVE MAP USING FOLIUM

- Folium library has been used to add map objects such as circles, markers, marker clusters, lines to identify locations and make them more interactive.
- Launch site locations were effectively analyzed, in addition to identifying other factors such as location of other close proximity buildings etc. Distance between launch site and other proximities is also calculated.
- [GitHub URL](#)

SUMMARY : PLOTLY DASH DASHBOARD

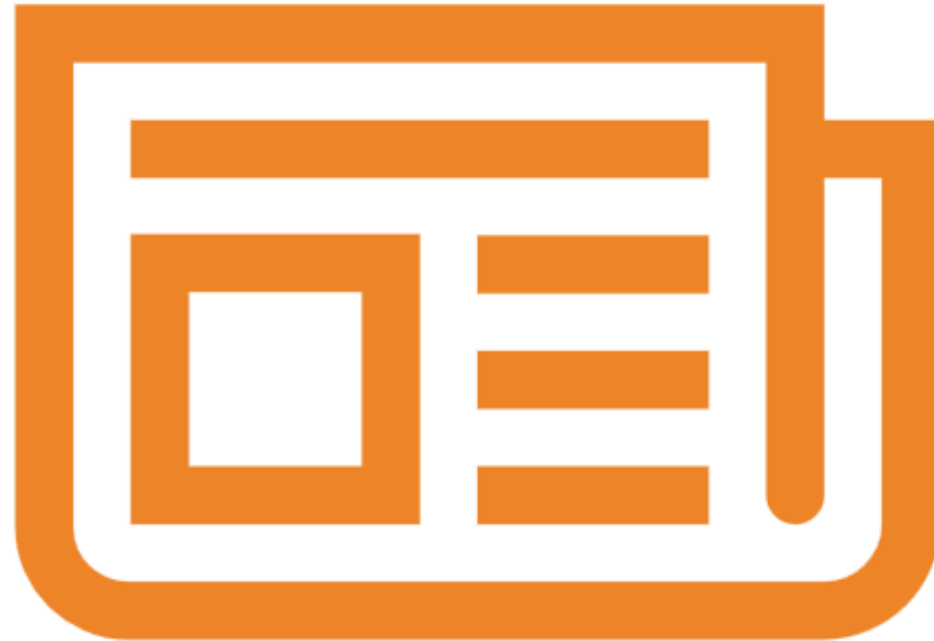
- Plotly Dash has been used to build an interactive dashboard to display key findings such as success rate for different/all launch sites, payload mass and its correlation with different launch sites and booster versions to name a few.
- The dashboard has been made interactive using a number of powerful plotting components such as drop-down menu, range slider, pie chart and scatter plot.
- [GitHub URL](#)

SUMMARY : PREDICTIVE ANALYSIS (CLASSIFICATION)

- Predictive Analysis has been performed by creating a new column for the class (success, i.e. 0 and 1), standardizing the data, splitting the data into training and test data.
- Logistic Regression, SVM, Decision Tree and KNN classification models have been tried and tested to find the best model that fits our problem case.
- **GitHub URL**

RESULTS

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





EDA WITH PYTHON VISUALIZATION

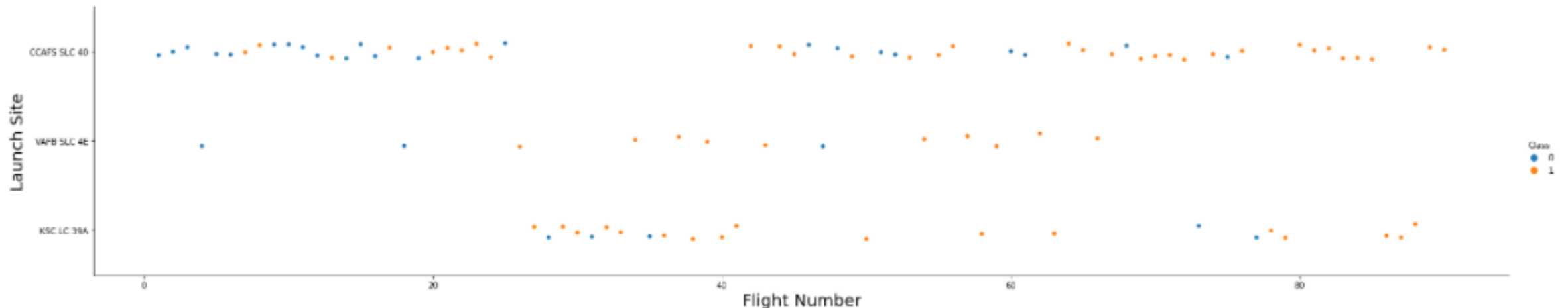
Libraries used: - Pandas, NumPy, Matplotlib

[GitHub URL](#)

FLIGHT NUMBER VS. LAUNCH SITE

The following scatter plot shows the relationship between the various flight numbers and the three key launch sites. The success rate is being denoted by the class, i.e. 0 for failed and 1 for success.

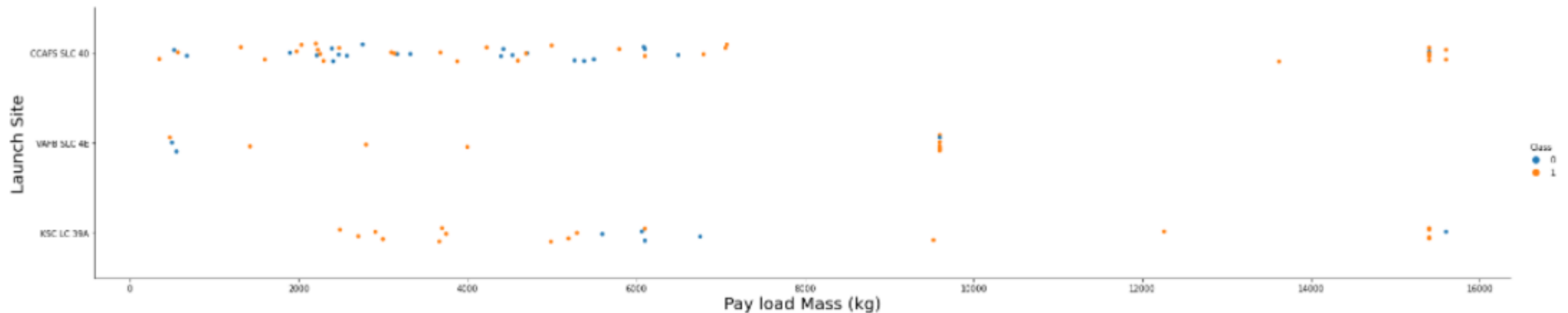
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
plt.savefig('flightnumvslaunchsite.jpg')
```



PAYLOAD VS. LAUNCH SITE

The following scatter plot shows the effect of different payloads on the success rate for various launch sites. The success rate is being denoted by the class, i.e. 0 for failed and 1 for success.

```
# 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(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



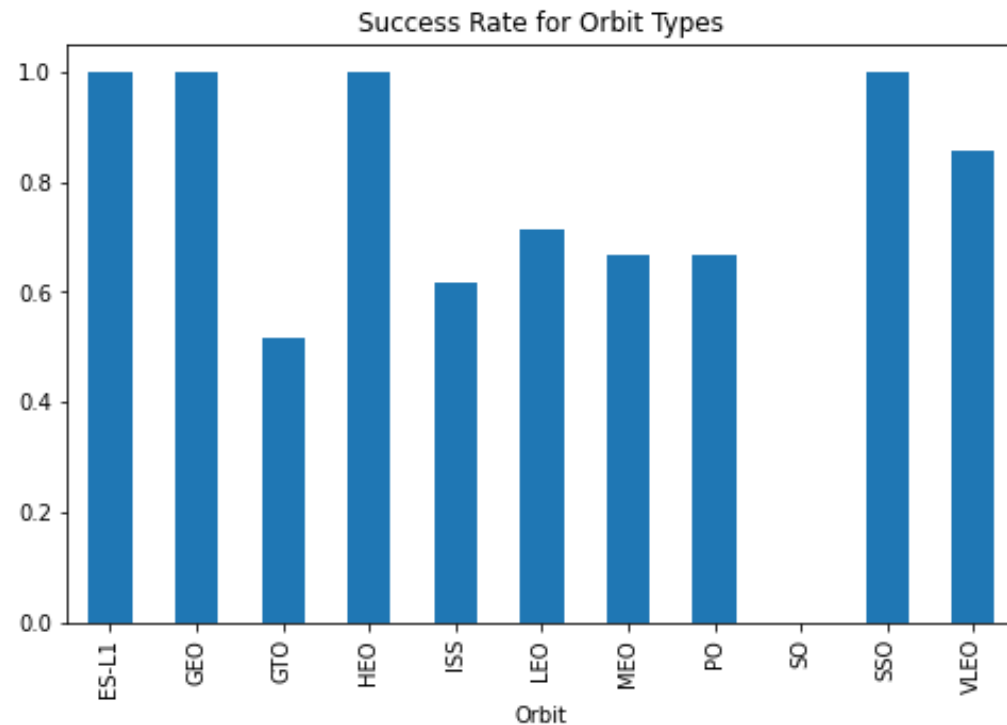
SUCCESS RATE VS. ORBIT TYPE

The following bar chart displays the mean success rate of flights per orbit types.

In the graph, it is evident that 4 distinct Orbit types, namely ES-LI, GEO, HEO and SSO have a success rate of 1.0.

```
In [8]: # HINT use groupby method on Orbit column and get the mean of Class column
dfg = df.groupby(['Orbit'])['Class'].mean()
dfg.plot(kind='bar', title='Success Rate for Orbit Types',figsize=(8, 5))
```

```
Out[8]: <AxesSubplot:title={'center':'Success Rate for Orbit Types'}, xlabel='Orbit'>
```



FLIGHT NUMBER VS. ORBIT TYPE

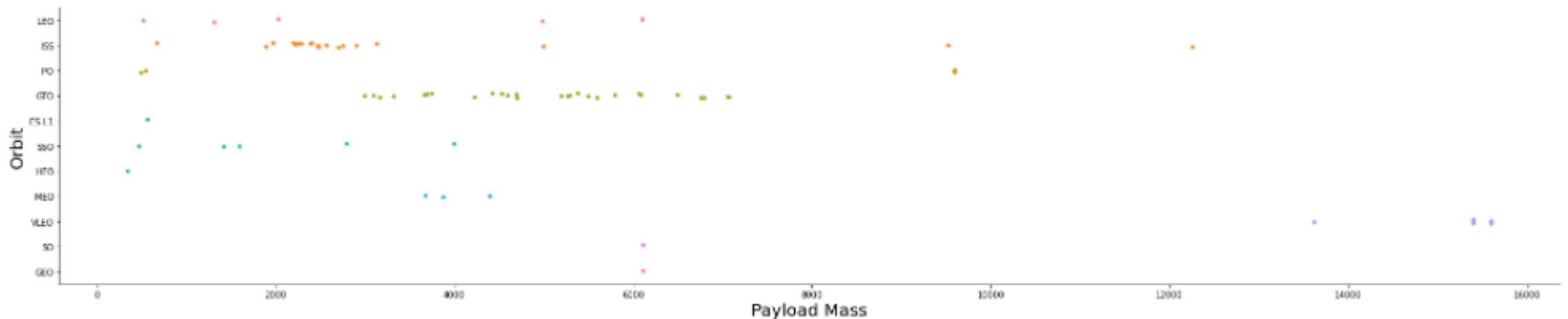
The following scatter plot shows the success rate of various flights for different orbit types. The success rate is being denoted by the class, i.e. 0 for failed and 1 for success.

```
# 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(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```


PAYLOAD VS. ORBIT TYPE

The following scatter plot shows the relationship between the payload mass and orbit types. The plot clearly shows that the heavy payloads have negative influence on GTO orbits and Polar LEO orbits. Whereas the lighter payload have more influence on most of the orbits.

```
# 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',data=df,aspect=5)
plt.xlabel('Payload Mass',fontsize=20)
plt.ylabel('Orbit',fontsize=20)
plt.show()
```



LAUNCH SUCCESS YEARLY TREND

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations



EDA USING SQL THROUGH IBM WATSON

Libraries used: - IBM_DB, IBM_DB_SA, SQLALCHEMY

[GitHub URL](#)

ALL LAUNCH SITE NAMES

The following SQL query has been used to retrieve unique launch sites in the SPACEX space mission dataset. Below the query is the output.

Dataset specifications in DB2: -

Tablename – SPACEXTBL

Tablespace – DBC93104

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

```
%sql select distinct LAUNCH_SITE from DBC93104.SPACEXTBL;
```

```
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.  
Done.
```

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

LAUNCH SITE NAMES BEGIN WITH 'CCA'

The following SQL query has been used to fetch all launch sites from the SPACEXTBL which begin with 'CCA'. A wild card operator has been used in the query to filter out the pattern and get the appropriate result. This query has been modified to only fetch first 5 rows only, which can be left out if one wants to get a complete list.

```
In [26]: %sql select LAUNCH_SITE from SPACEXTBL where LAUNCH_SITE LIKE 'CCA%' fetch first 5 rows only;  
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.database  
Done.
```

```
Out[26]:
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

TOTAL PAYLOAD MASS

The following SQL statement has been used to get a sum of total Payload Mass (KG), that was carried by boosters from NASA which is 45,596kgs. Since NASA is listed as a customer in SPACEXTBL, we filtered the query using the WHERE clause. This SQL query can be modified to get sum/min/max/average of payload mass carried by boosters from any of the listed customers in the database/dataset.

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

```
: %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER='NASA (CRS)';
```

```
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb  
Done.
```

```
27]: 1  
45596
```

AVERAGE PAYLOAD MASS BY F9 V1.1

The following SQL statement has been used to get an average of Payload Mass (KG), carried by booster version named F9 v1.1, which is 2,928kg. An alias (optional) has been used to display the average result as 'AVG_PAYLOAD_MASS'. The query has been filtered using the WHERE clause to specify the booster version. This SQL query can be modified to get sum/min/max/average of payload for other booster versions.

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

```
[ ]: %sql select AVG(PAYLOAD_MASS__KG_) as AVG_PAYLOAD_MASS from SPACEXTBL where BOOSTER_VERSION='F9 v1.1';  
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdo  
Done.
```

```
[28]: avg_payload_mass  
      2928
```

SUCCESSFUL GROUND PAD LANDING WITH PAYLOAD BETWEEN 4000 AND 6000

The following SQL statement has been used to display only the booster versions with payload mass between 4,000kg and 6,000kg that have a successful ground pad landing. Multiple WHERE clause with AND operator has been used to get the required result. This SQL query can be modified to display booster versions between different payload mass with different landing outcomes.

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

```
In [30]: %sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_>=4000 and PAYLOAD_MASS__KG_<6000 and LandingOutcome='Success (ground pad)';
```

```
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

```
Out[30]:
```

booster_version

F9 FT B1032.1

F9 B4 B1040.1

F9 B4 B1043.1

TOTAL SUCCESSFUL AND FAILURE MISSION OUTCOMES

The following SQL statement has been used to get total count of successful and failed missions. The Distinct statement has been used to avoid redundancy in results. Count has been used to count the occurrence of each mission outcomes, i.e. Success and Fail using wildcard operator.

```
%sql select distinct (select count(*) from SPACEXTBL where Mission_Outcome like 'Success%') as Successful_Mission,(select count(*) from SPACEXTBL where Mission_Outcome like 'Failure%') as Failed_Mission from SPACEXTBL ;
```

List the total number of successful and failure mission outcomes

```
%sql select distinct (select count(*) from SPACEXTBL where Mission_Outcome like 'Success%') as Successful_Mission,(select count(*) from SPACEXTBL where
```

```
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb  
Done.
```

```
7]: successful_mission failed_mission  
100 1
```

BOOSTERS CARRYING MAXIMUM PAYLOAD

The following SQL statement has been used to retrieve a list of only booster versions which have carried the maximum payload mass. This statement comprises of a main query, which takes input from the subquery using the max function for payload mass.

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

```
%sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS_KG_ in (select max(PAYLOAD_MASS_KG_) from SPACEXTBL);
```

```
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:313:
Done.
```

```
32]: booster_version
      F9 B5 B1048.4
      F9 B5 B1049.4
      F9 B5 B1051.3
      F9 B5 B1056.4
      F9 B5 B1048.5
      F9 B5 B1051.4
      F9 B5 B1049.5
      F9 B5 B1060.2
      F9 B5 B1058.3
      F9 B5 B1051.6
      F9 B5 B1060.3
      F9 B5 B1049.7
```

2015 LAUNCH RECORDS

The following SQL statement has been used to fetch the booster versions, launch site and the name of months in which the landing outcomes for drone ship was a failure for the year 2015. MONTHNAME function has been used to convert the month from the date column and display its name. YEAR function has been used to extract the year from the date column. This statement used WHERE clause with AND operator for filtering the results.

```
In [38]: %sql select MONTHNAME(Date) as Month, Booster_Version, LandingOutcome, LAUNCH_SITE from SPACEXTBL where LandingOutcome='Failure (drone ship)' and YEAR(Date)=2015;

* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

```
Out[38]:
```

MONTH	booster_version	landingoutcome	launch_site
January	F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
April	F9 v1.1 B1015	Failure (drone ship)	CCAFS LC-40

2017 LAUNCH RECORDS

The following SQL statement has been used to fetch the booster versions, launch site and the name of months in which the landing outcomes for ground pad was a success for the year 2017. MONTHNAME function has been used to convert the month from the date column and display its name. YEAR function has been used to extract the year from the date column. This statement used WHERE clause with AND operator for filtering the results.

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

```
: %sql select MONTHNAME(Date) as Month, Booster_Version, LandingOutcome, LAUNCH_SITE from SPACEXTBL where LandingOutcome='Success (ground pad)' and YEAR(Date)=2017;
```

```
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

33]:

MONTH	booster_version	landingoutcome	launch_site
February	F9 FT B1031.1	Success (ground pad)	KSC LC-39A
May	F9 FT B1032.1	Success (ground pad)	KSC LC-39A
June	F9 FT B1035.1	Success (ground pad)	KSC LC-39A
August	F9 B4 B1039.1	Success (ground pad)	KSC LC-39A
September	F9 B4 B1040.1	Success (ground pad)	KSC LC-39A
December	F9 FT B1035.2	Success (ground pad)	CCAFS SLC-40

RANK SUCCESS COUNT BETWEEN DATE RANGE

The following SQL statement has been used to get a count of successful landing outcomes and to rank them in descending order between the dates 2010-06-04 and 2017-03-20. COUNT function, WHERE clause, wildcard operator, groupby and order by have been used to construct this SQL statement to produce the required result.

Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

```
%sql select count(LandingOutcome),LandingOutcome from SPACEXTBL where Date between '2010-06-04' and '2017-03-20' and LandingOutcome like 'Success%' group by LandingOutcome
```

```
* ibm_db_sa://dbc93104:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

```
4]: 1  landingoutcome
    5  Success (drone ship)
    3  Success (ground pad)
```



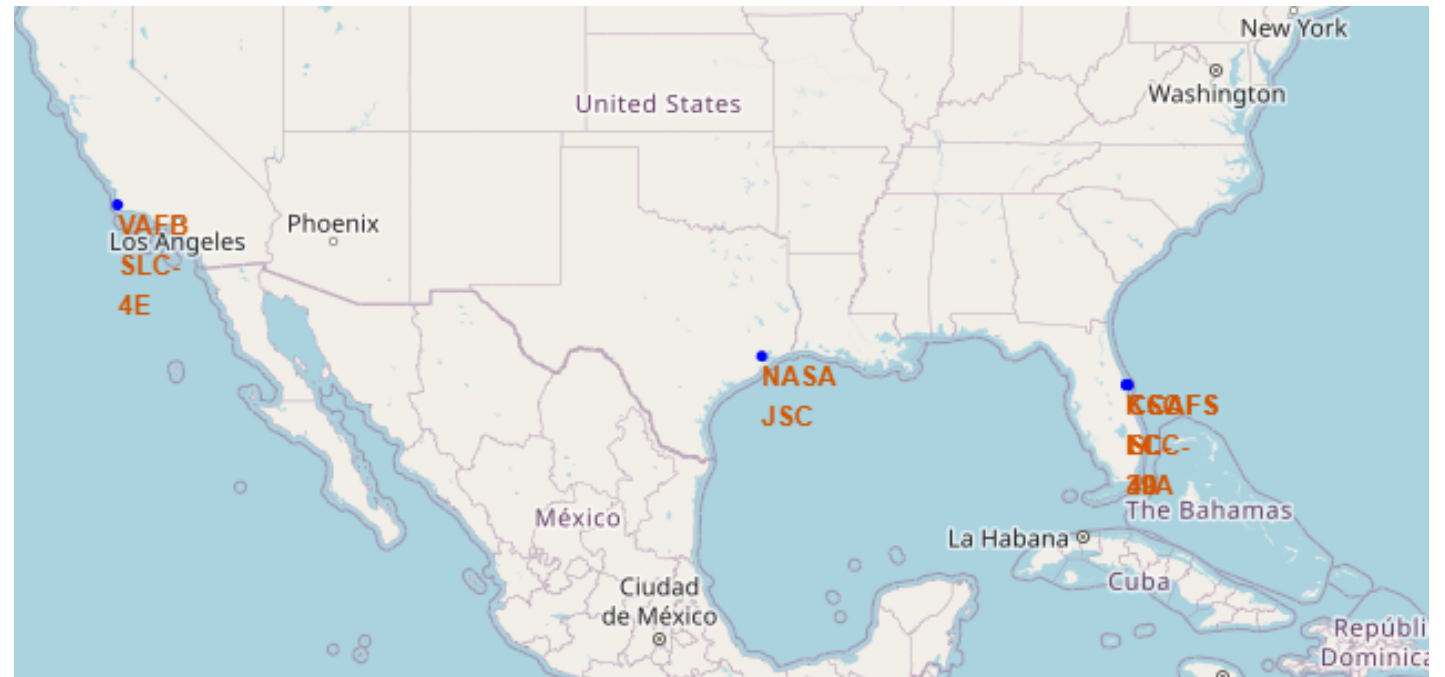
INTERACTIVE MAP WITH FOLIUM

Libraries used: - Folium, WGET, Pandas

[GitHub URL](#)

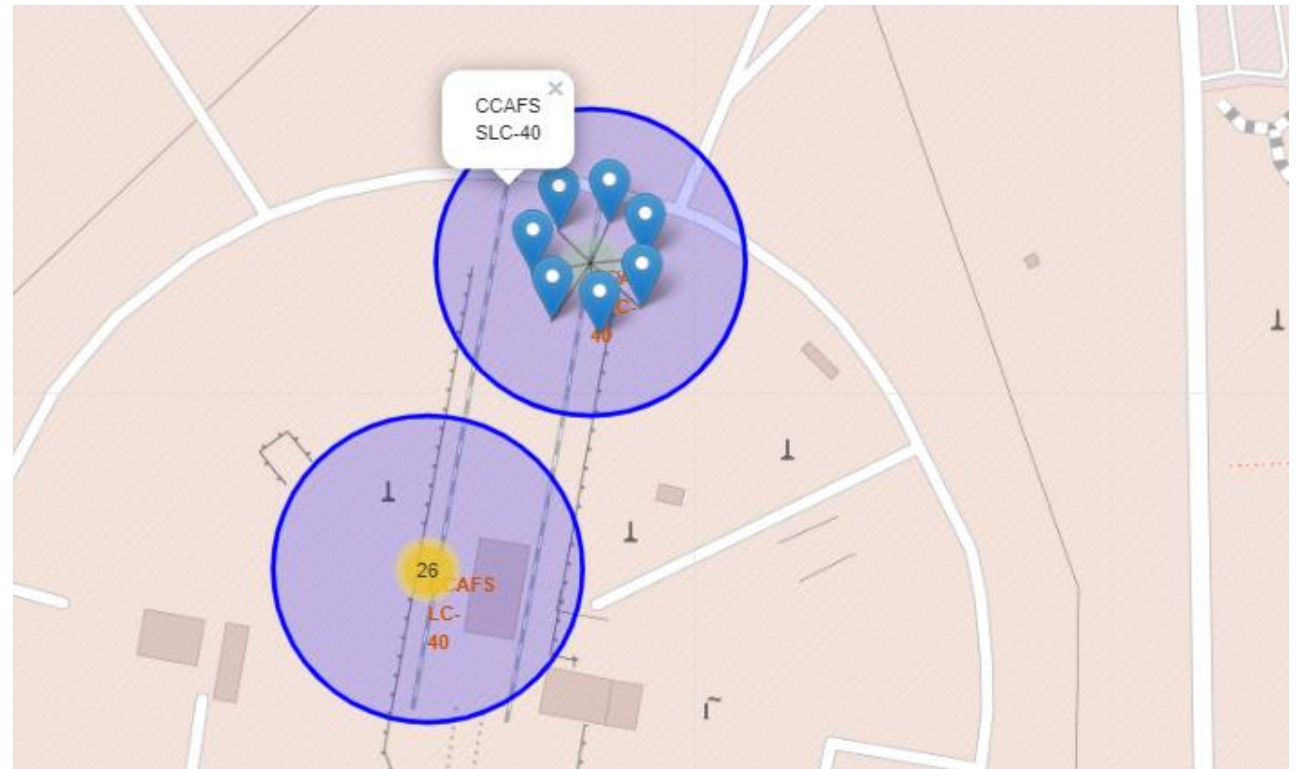
US SPACE LAUNCH SITES

In the given screenshot we can see that with the help of Folium library, we can identify and plot the coordinates of different locations,. The screenshot identifies different US Space launch sites on a global map.



COLOR LABELED RECORDS - LAUNCH SITES

The following screenshot displays the different launch records of the launch site over which the mouse has hovered. The icons are green for successful launch, whereas red for failed ones. This has been achieved using the Marker Cluster and its various properties in Folium. Using Marker Cluster, we can group together locations and put them together as a cluster on a Folium map.





BUILD A DASHBOARD WITH PLOTLY DASH

Libraries used: - pandas, dash, dash_html_components,
dash_core_components, jupyter_dash, plotly.express, plotly.graph_objects

[GitHub URL](#)

SPACEX LAUNCH RECORDS DASHBOARD

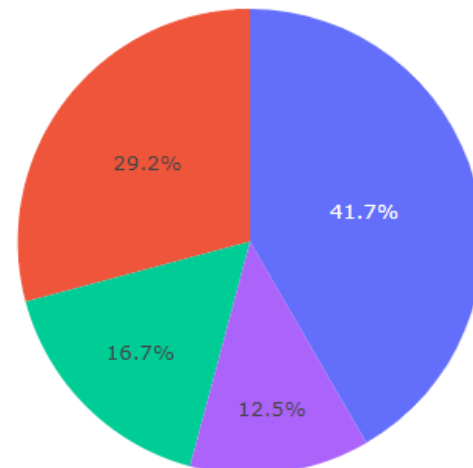
The SpaceX Launch Records Dashboard has been designed using Plotly and Dash, and in this screenshot it shows a drop down menu and a graph for the selected options. The user can choose to view records pertaining to either 'All Sites' or individual launch sites, which will show a success rate bar chart accordingly.

SpaceX Launch Records Dashboard

All Sites



Total Success Launches by all Sites

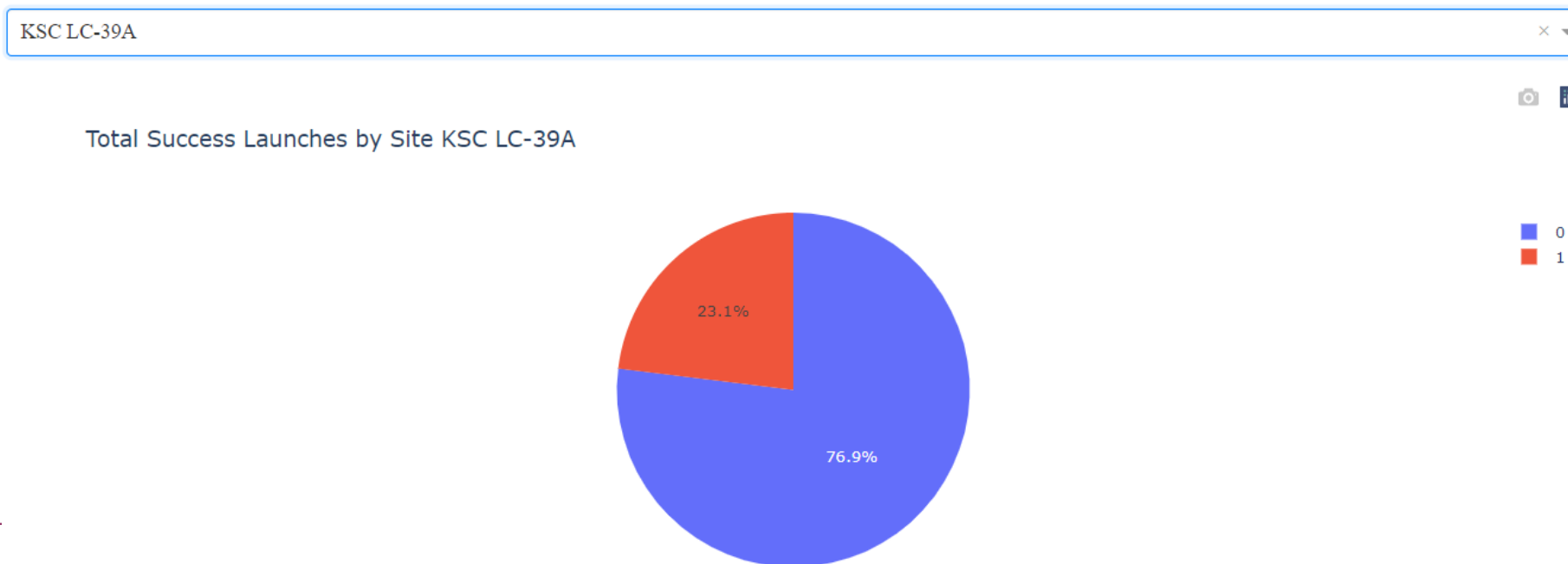


- KSC LC-39A
- CCAFS LC-40
- VAFB SLC-4E
- CCAFS SLC-40

SPACEX LAUNCH RECORDS DASHBOARD

The following screenshot displays the launch site with highest success rate of all other launch sites. This site is 'KSC LC-39A', with a success rate of 76.9% and a failure rate of 23.1% identified by blue and red color respectively. The user can similarly choose a different launch site from the dropdown menu to see the success rate. A title for the bar chart has also been added to help user identify the information.

SpaceX Launch Records Dashboard



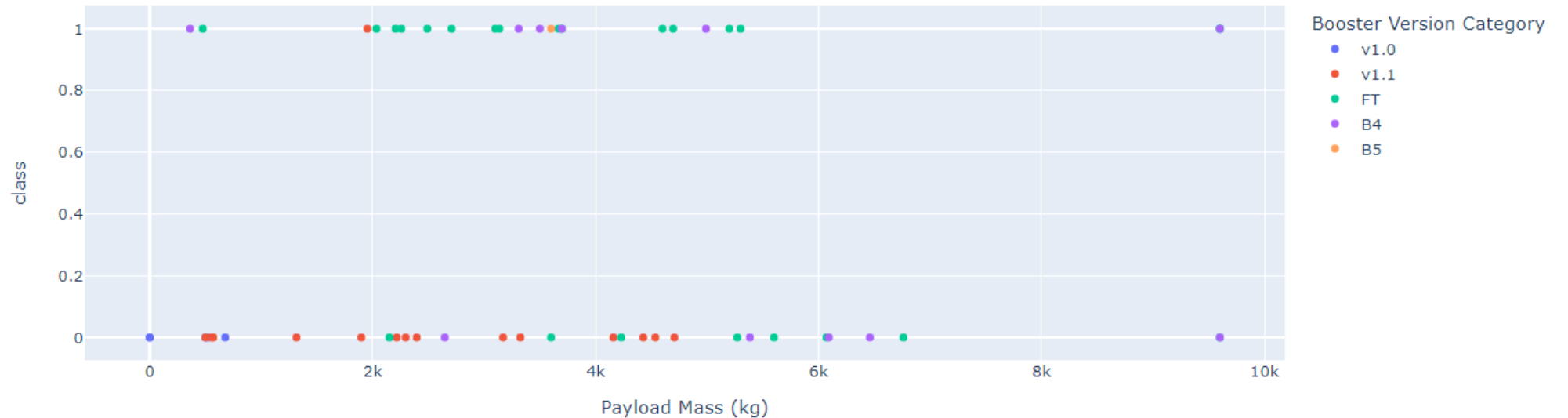
SPACEX LAUNCH RECORDS DASHBOARD

The following screenshot of the dashboard displays the correlation between Payload Mass (kg) and Success for all sites as selected by the user. On the right hand, we can also see the colored dots for different booster version categories. On the top is a range slider that helps the user input a payload range.

Payload range (Kg):



Correlation between Payload and Success for all Sites

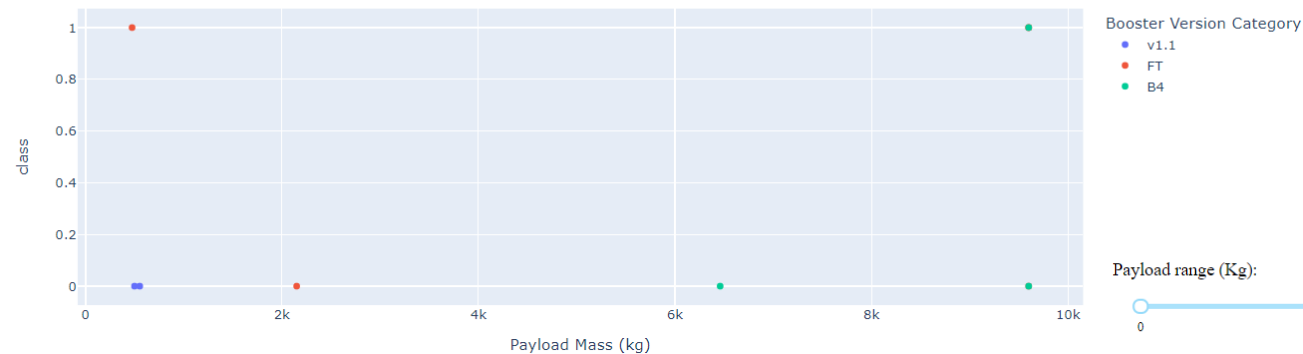


SPACEX LAUNCH RECORDS DASHBOARD

Payload range (Kg):

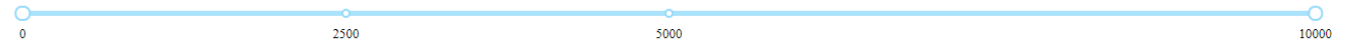


Correlation between Payload and Success for Site VAFB SLC-4E

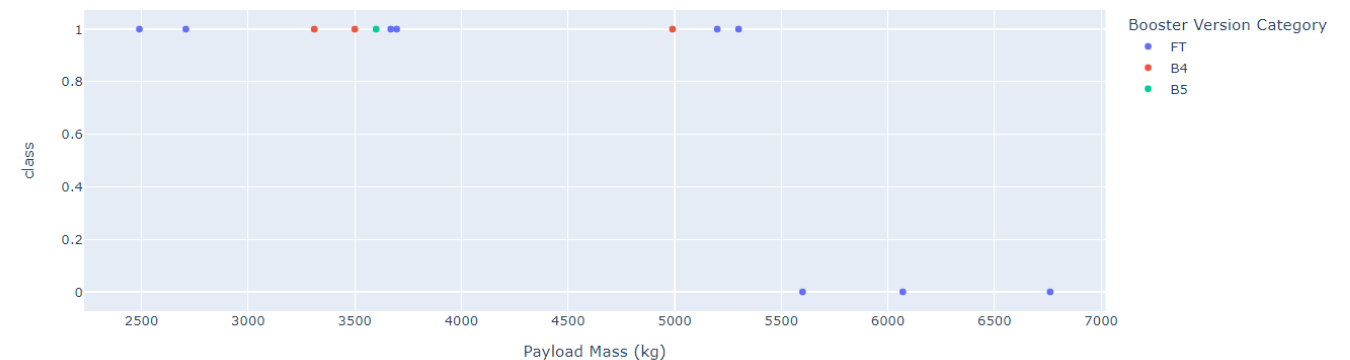


The following screenshots displays the correlation between Payload Mass (kg) and Success for 2 sites as selected by the user. On the right hand, we can also see the colored dots for different booster version categories. On the top is a range slider that helps the user input a payload range.

Payload range (Kg):



Correlation between Payload and Success for Site KSC LC-39A





PREDICTIVE ANALYSIS (CLASSIFICATION)

Libraries used: - pandas, numpy, matplotlib, seaborn, sklearn

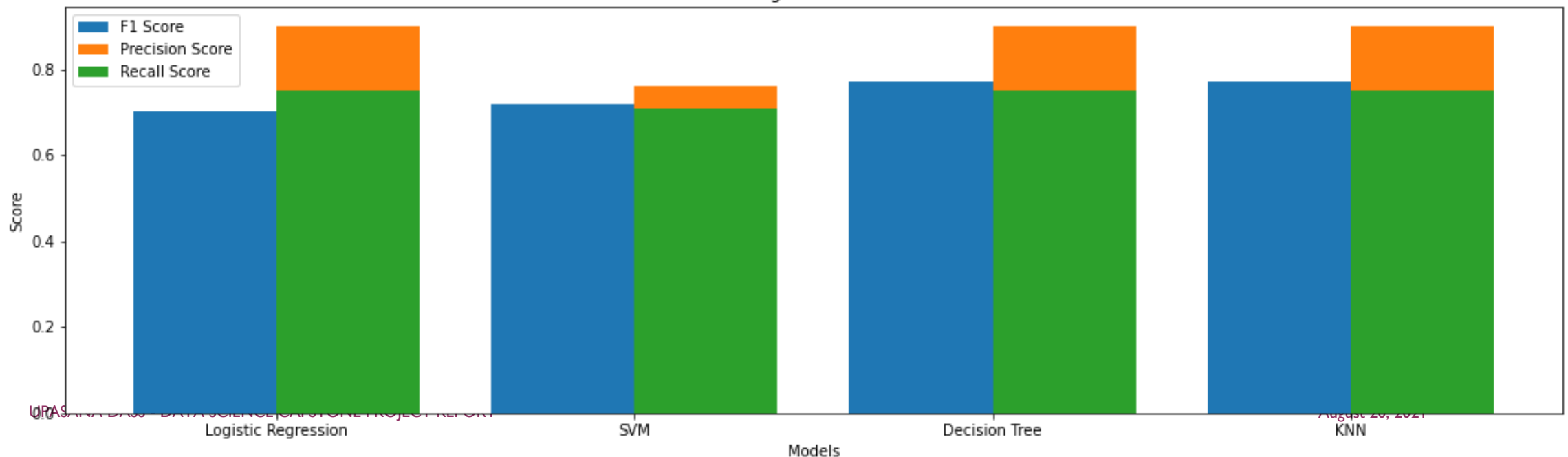
[GitHub URL](#)

CLASSIFICATION ACCURACY

Model Evaluation:

Precision Score of LR,DT and KNN are all high, i.e. 0.9. Recall for LR, DT and KNN are all high, i.e. 0.75. Also, F1 score for LR, DT and KNN is close to 1, with a value of 0.7. All these three models have the highest classification accuracy, making them the best fit. While SVM has the low values as compared to the rest of the models, making it the least preferable option.

Model Evaluation using F1, Precision and Recall Score



CONFUSION MATRIX

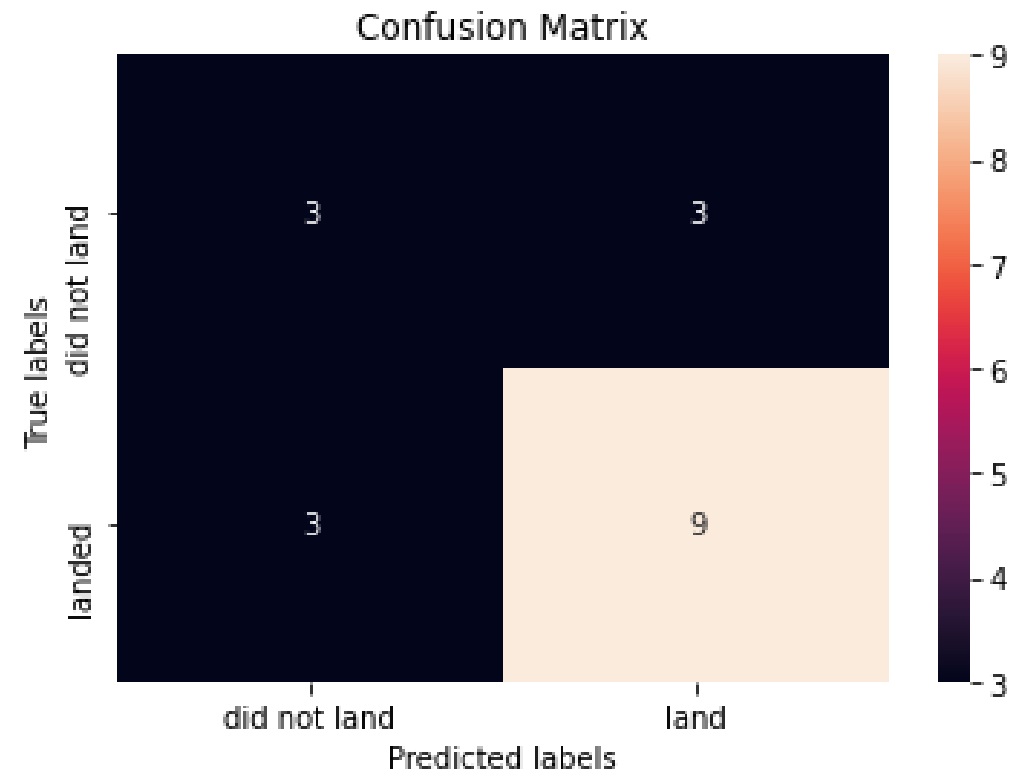
Following is the screenshot of the best performing model's confusion matrix, i.e. Decision Tree. The model's predicted labels were as follows:

Predicted Labels		True Labels	
Did not land	3	Did not land	3
Land	9	Land	3

The confusion matrix is true for the 'did not land' labels. Whereas, it's prediction for the 'land' label is not completely true, which is off by a difference of 6.

Also, the classification accuracy of Decision Tree model in the above slide prove that it fares well as compared to the other models.

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



CONCLUSION



- Data Science using Python and other secondary options played crucial role in understanding whether or not Falcon 9 can successfully.
- SQL queries were executed on the historical data using DB2
- Folium was used to effectively plot the launch sites and cluster launch status to understand how other factors played role in success or failure of the launches.
- Plotly Dash was used to create an interactive dashboard to display the key findings.
- Machine Learning was used to classify and evaluate the different models that best fit our data need.

APPENDIX



- SQL Queries
- Plotly Dashboard Code

SQL QUERIES

- Display names of unique launch sites in space mission:
 - `%sql select distinct LAUNCH_SITE from DBC93104.SPACEXTBL;`
- Display 5 records where launch sites begin with 'KSC':
 - `%sql select LAUNCH_SITE from SPACEXTBL where LAUNCH_SITE LIKE 'KSC%' fetch first 5 rows only;`
- Display total payload mass carried by boosters launched by NASA(CRS):
 - `%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER='NASA (CRS)';`
- Display average payload mass carried by booster version F9 v1.1:
 - `%sql select AVG(PAYLOAD_MASS__KG_) as AVG_PAYLOAD_MASS from SPACEXTBL where BOOSTER_VERSION='F9 v1.1';`
- List date of successful landing outcome in droneshop:
 - `%sql select min(Date) as date from SPACEXTBL where LandingOutcome='Success (drone ship)';`

SQL QUERIES

- List names of boosters with successful ground pad landing with payload ground mass between 4000 and 6000:
 - %sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_>=4000 and PAYLOAD_MASS__KG_<6000 and LandingOutcome='Success (ground pad)';
- List total number of successful and failure mission outcomes:
 - %sql select distinct (select count(*) from SPACEXTBL where Mission_Outcome like 'Success%') as Successful_Mission,(select count(*) from SPACEXTBL where Mission_Outcome like 'Failure%') as Failed_Mission from SPACEXTBL ;
- List names of booster versions which have carried maximum payload mass using subquery:
 - %sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ in (select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
- List records which display month names, successful landing outcomes in ground pad, booster versions, launch sites for months in year 2017
 - %sql select MONTHNAME(Date) as Month, Booster_Version, LandingOutcome, LAUNCH_SITE from SPACEXTBL where LandingOutcome='Success (ground pad)' and YEAR(Date)=2017;
- Rank the count of successful landing outcomes between 2010-06-04 and 2017-03-20 in descending order
 - %sql select count(LandingOutcome),LandingOutcome from SPACEXTBL where Date between '2010-06-04' and '2017-03-20' and LandingOutcome like 'Success%' group by LandingOutcome order by LandingOutcome ASC;

PLOTLY DASHBOARD CODE



Adobe Acrobat Document