SMART INTERNZ ASSESMENT PROJECT

SpaceX Falcon9 First Stage Landing Success Predictor

Prepared by

AKANKSHA HOTTA	20MIS0137
V UMESH KUMAR RAJU	20MIS0187
TANMAY GHODESWAR	20BCE0895
RISHAB KUMAR TIWARY	20BCE10139

1. INTRODUCTION

1.1 Overview

The SpaceX Falcon 9 is a reusable rocket that SpaceX built to launch payloads into orbit. The Falcon 9 rocket's first stage is intended to re-enter the atmosphere after launch and touch down vertically on a predetermined landing pad or drone ship. Using this technology, SpaceX can reuse the rocket rather than throwing it away after a single use, lowering the cost of space flights. The number of flights, payload mass, orbit, launch site, and other variables all affect how well a Falcon 9 first stage landing launch occurs. We can offer useful information to space agencies, researchers, and other stakeholders by developing a machine learning model that can forecast the success of these launches based on these variables.

Date: Date of launching the flight Falcon 9 and the maiden launch of the rocket version.

Time: Time of launching the flight.

Booster Version: This provides us an idea about which booster version they have specifically used for the capability of landing vertically to facilitate reuse.

Launch Site: It's similar to Starbase where the spaceport, production, and development facility is being provided to Starship Rockets.

Payload: The informations about the payload that has been used to lift maximum kg to geostationary orbit.

Payload Mass (in Kg): The maximum capacity that a specific Falcon rocket can uplift.

Orbit : Specifies the name of the orbit in which an object (electronic equipment), that takes circle in the curved path.

Customer : Signifies the name of the agency in the field of science and technology related to air and space.

Mission Outcome: Enable us to know the success rate of the rocket, launched.

Landing Outcome: Similar to mission outcome, it speaks about the successful landing of Falcon rocket.

1.2 Puropse

The creation of a web application using this machine learning model may offer the space sector a cutting-edge service. It can help increase the effectiveness and success rate of launches for space agencies, for-profit businesses, and other organizations engaged in space missions. Cost savings as well as an improvement in client trust and happiness might come from this.

2. Literature Survey

2.1 Existing Problem

[1] Ferrante, R., 2017. A robust control approach for rocket landing. *Master's thesis*.

[1] intends to provide a vertical rocket landing simulation environment where conventional control and machine learning methods can be developed and assessed. Additionally, it signifies some few aspects like constructing a rocket landing simulator environment , creating and assessing algorithms for vertical rocket landing that combine traditional control and machine learning and concentrating on reusable space systems as a multidisciplinary topic, particularly first stage rockets.

[2] Jenie, Y.I., Suarjaya, W.W.H. and Poetro, R.E., 2019, November. Falcon 9 rocket launch modeling and simulation with thrust vectoring control and scheduling. In *2019 IEEE 6th Asian Conference on Defence Technology (ACDT)* (pp. 25-31). IEEE.

[2] utilizes a mathematical model for simulating the launch of a multistage launch vehicle (LV) from the launch site through orbit insertion is presented. The model was developed in the MATLAB/Simulation environment. Propulsion, aerodynamics, environmental factors, weight, and the engines are all taken into account in the model, which has six degrees of freedom. SpaceX Falcon 9 LV was being used in their operation to deliver cargo to a parking orbit. In order to evaluate the model's capabilities, three simulations were being performed: an open loop point mass simulation, an open loop complete dynamic simulation, and a closed loop with TVC simulation.

[3] Brennan, L. and Vecchi, A., 2020. The orbital circular economy framework: Emblematic evidence from the space industry. *Kindai Management Review*, *8*.

In order to establish a sustainable competitive advantage, [3] suggests a framework known as the Orbital Circular Economy Framework (OCEF), which integrates the concepts of the Circular Economy with a resource-based perspective and dynamic capabilities. Examining the experiences of private operators in the space sector helps to validate the framework. An outline of the Circular Economy (CE) framework, which aims to change society so that it becomes resource-efficient. The need for greater CE to lessen environmental stress, strengthen raw material supply security, boost competitiveness, encourage innovation, accelerate economic growth, and create jobs.

[4] Pidvysotskyi, V., 2021. The Concept of an Inflatable Reusable Launch Vehicle.

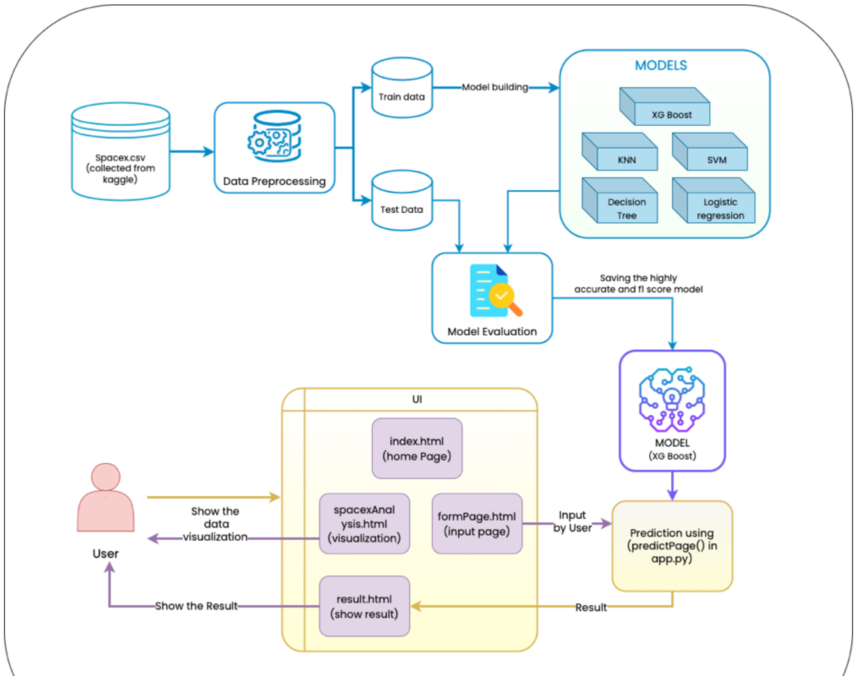
Reusable space transport systems are clearly needed, and research into them is highly sought after. In order to lower the cost of space travel, [4] will address the idea of a launch vehicle with an inflatable, reusable first stage and also emphasizes to analyse the impact of aerodynamic losses on the payload and compute the mass of the inflatable first stage as well as its vertical and horizontal stiffness as a preliminary approximation. The practical ramifications of this new concept for a launch vehicle that features an inflatable, reusable first stage and the ability to greatly boost a launch vehicle's payload capacity. The suggested idea is straightforward, portable, and very technical. It could result in more affordable and effective space transportation technologies if it is successfully developed.

2.2 Proposed Solution

The main goal of this project whether the Falcon9 first stage will land successfully. Apart from research study if we analyze technically there are several methods we can suggest to predict the landing outcome by using ensemble learning techniques that by using Random Forest and XG Boost. As well as to deal with categorical data (like Orbit,Mission_Outcome,Customer) we can use Label Encoder. We can infer a lot of information through several visualization techniques to find the relationship between two variables.

3. Theoretical Analysis

3.1. Block Diagram


Spacex.csv (collected from kaggle)
Data Preprocessing
Train data
Test Data
Model building
MODELS
XG Boost
KNN
SVM
Decision Tree
Logistic regression
Model Evaluation
Saving the highly accurate and f1 score model
MODEL (XG Boost)
UI
index.html (home Page)
spacexAnalysis.html (visualization)
formPage.html (input page)
result.html (show result)
Input by User
Prediction using (predictPage() in app.py)
Result
User
Show the data visualization
Show the Result

Block Diagram
SpaceX Falcon9 First stage landing prediction

3.2 Hardware and Software designing

Hardware

x86 64-bit CPU (Intel / AMD architecture)
4 GB RAM
At least 1 GB free disk space

Operating System

Windows 10 or 11
Mac OS X 10.11 or higher, 64-bit

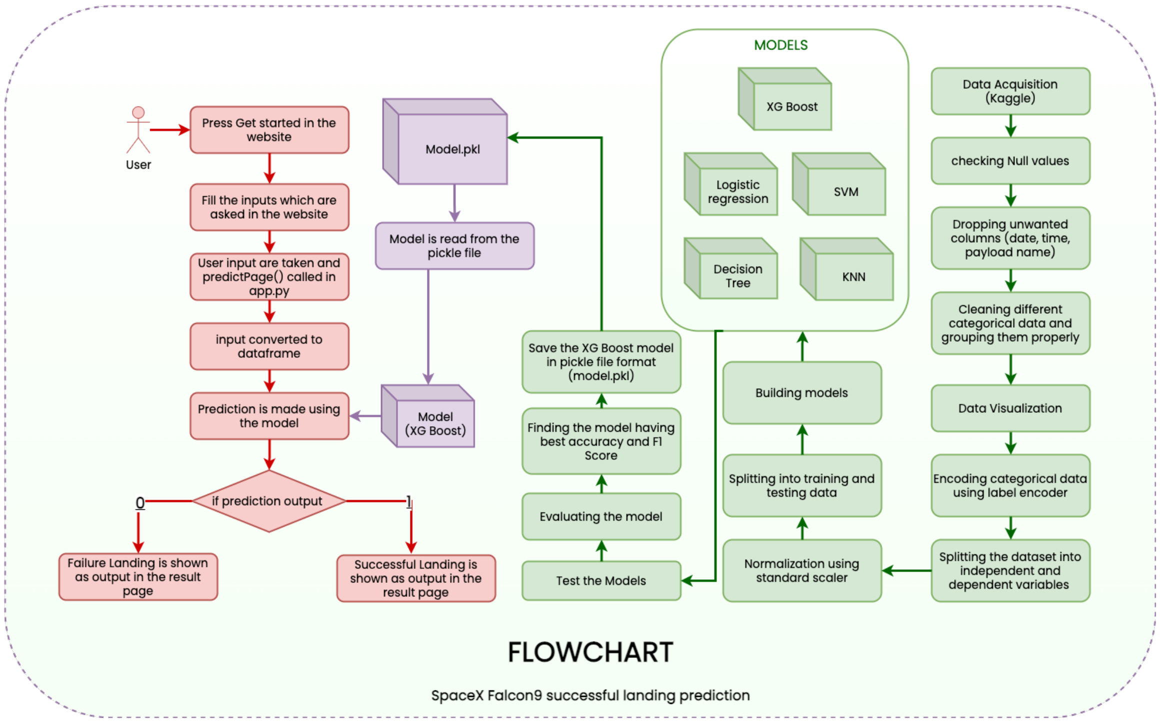
Software

Anaconda Navigator
Jupyter Notebook
Spyder
Visual Studio

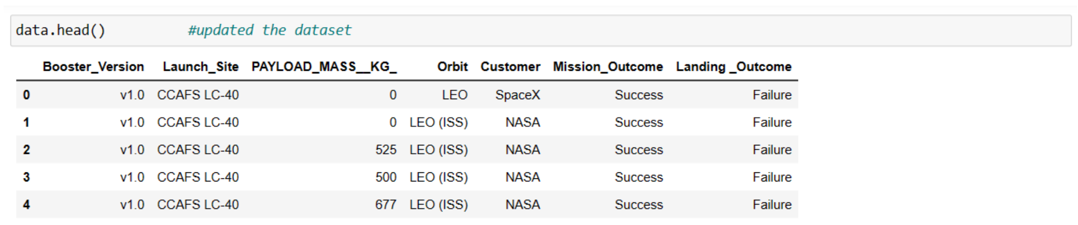
4. Experimental Investigations

While working with the project, we have studied the dataset and techniques to increase the accuracy of the model,we changed categorical values to numerical followed with Standardization.For instance, for column name "Booster_Version" we grouped them into 4 categories (B4,B5,V1.0,V1.1) thereby performed Encoding (Label Encoding). And finally scaling down the values between 0 to 1. As the dataset contains almost categorical features with discrete values so, we have followed same techniques to group down other categorical features. In other words we tried the distribution of classes should almost evenly skewed.

5. Flowchart


User
Press Get started in the website
Fill the inputs which are asked in the website
User input are taken and predictPage() called in app.py
input converted to dataframe
Prediction is made using the model
if prediction output
0
1
Failure Landing is shown as output in the result page
Successful Landing is shown as output in the result page
Model.pkl
Model is read from the pickle file
Model (XG Boost)
MODELS
XG Boost
Logistic regression
SVM
Decision Tree
KNN
Save the XG Boost model in pickle file format (model.pkl)
Finding the model having best accuracy and F1 Score
Evaluating the model
Test the Models
Building models
Splitting into training and testing data
Normalization using standard scaler
Data Acquisition (Kaggle)
checking Null values
Dropping unwanted columns (date, time, payload name)
Cleaning different categorical data and grouping them properly
Data Visualization
Encoding categorical data using label encoder
Splitting the dataset into independent and dependent variables
FLOWCHART
SpaceX Falcon9 successful landing prediction

6. Result

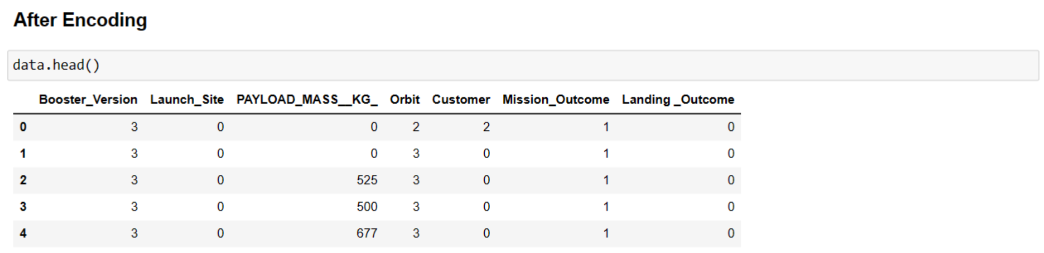


```
data.head()         #updated the dataset
```

	Booster_Version	Launch_Site	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	v1.0	CCAFS LC-40	0	LEO	SpaceX	Success	Failure
1	v1.0	CCAFS LC-40	0	LEO (ISS)	NASA	Success	Failure
2	v1.0	CCAFS LC-40	525	LEO (ISS)	NASA	Success	Failure
3	v1.0	CCAFS LC-40	500	LEO (ISS)	NASA	Success	Failure
4	v1.0	CCAFS LC-40	677	LEO (ISS)	NASA	Success	Failure

Fig 1 : After Pre-processing

After grouping categorical features (Booster_Version,Launch_Site,Customer,Landing_Outcome)



After Encoding

```
data.head()
```

	Booster_Version	Launch_Site	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	3	0	0	2	2	1	0
1	3	0	0	3	0	1	0
2	3	0	525	3	0	1	0
3	3	0	500	3	0	1	0
4	3	0	677	3	0	1	0

Fig 2 : After Encoding

Performing Label Encoding for overall pre-processed data. And we formulated some inferences like

For Booster_Version

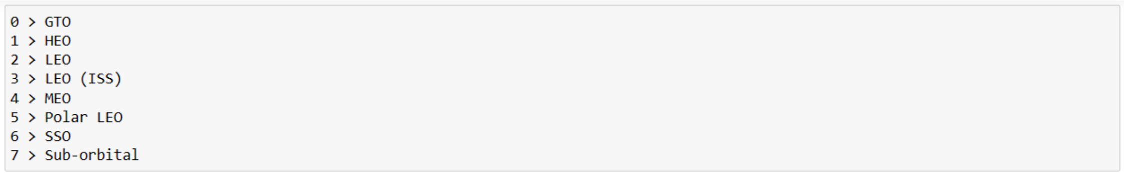
```
0 > B4
1 > B5
2 > FT
3 > v1.0
4 > v1.1
```

Similarly for other categorical features

Launch_Site

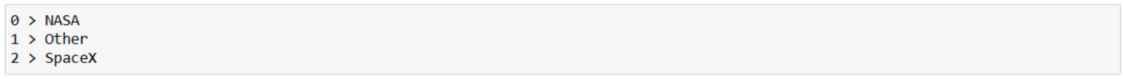
```
0  > CCAFS LC-40
1  > CCAFS SLC-40
2  > KSC LC-39A
3  > VAFB SLC-4E
```

Orbit



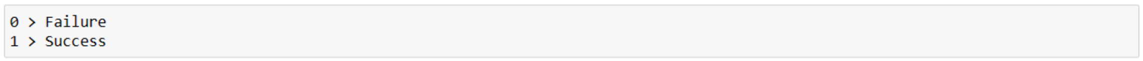
```
0 > GTO
1 > HEO
2 > LEO
3 > LEO (ISS)
4 > MEO
5 > Polar LEO
6 > SSO
7 > Sub-orbital
```

Customer

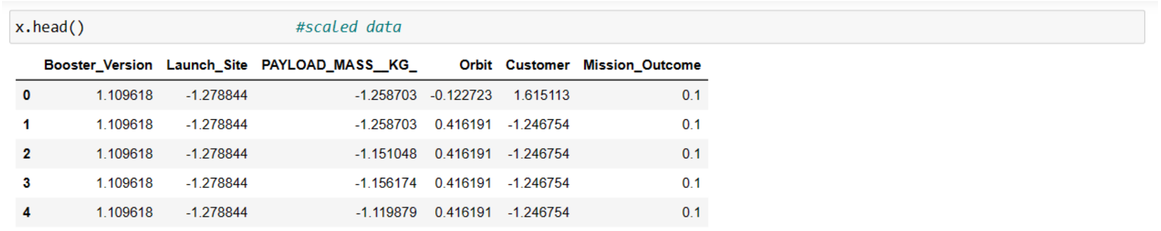


```
0 > NASA
1 > Other
2 > SpaceX
```

Landing_Outcome



```
0 > Failure
1 > Success
```

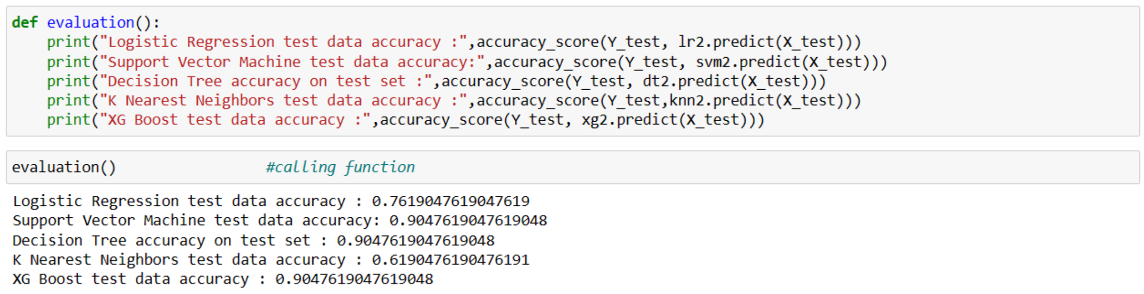


```
x.head()                        #scaled data
```

	Booster_Version	Launch_Site	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome
0	1.109618	-1.278844	-1.258703	-0.122723	1.615113	0.1
1	1.109618	-1.278844	-1.258703	0.416191	-1.246754	0.1
2	1.109618	-1.278844	-1.151048	0.416191	-1.246754	0.1
3	1.109618	-1.278844	-1.156174	0.416191	-1.246754	0.1
4	1.109618	-1.278844	-1.119879	0.416191	-1.246754	0.1

Fig 3 : After Normalization : Standard Scaler

After performing Label Encoding we splitted the data into x(independent variables) and y(dependent variable) and followed with we performed Standardization with the data stored in “x”.



```
def evaluation():
    print("Logistic Regression test data accuracy :",accuracy_score(Y_test, lr2.predict(X_test)))
    print("Support Vector Machine test data accuracy:",accuracy_score(Y_test, svm2.predict(X_test)))
    print("Decision Tree accuracy on test set :",accuracy_score(Y_test, dt2.predict(X_test)))
    print("K Nearest Neighbors test data accuracy :",accuracy_score(Y_test,knn2.predict(X_test)))
    print("XG Boost test data accuracy :",accuracy_score(Y_test, xg2.predict(X_test)))
```

```
evaluation()                    #calling function
```

```
Logistic Regression test data accuracy : 0.7619047619047619
Support Vector Machine test data accuracy: 0.9047619047619048
Decision Tree accuracy on test set : 0.9047619047619048
K Nearest Neighbors test data accuracy : 0.6190476190476191
XG Boost test data accuracy : 0.9047619047619048
```

Fig 4 : Accuracy of different classification models

We can infer that SVM, KNN, and XG Boost having accuracy around 90%. But we can use one of the ensemble learning technique i.e. XG Boost for the deployment of model.

7. Advantages and Disadvantages

Advantages

1. **Enhanced Data Collection:** An increased volume of data from various missions can be gathered because to the higher launch frequency. Large datasets are essential for ML models, and having access to more varied and comprehensive data can enhance model training and performance.

2. **Cost-Effective Training:** The need of attaining training data produced from satellite imagery or other space-based sensors can benefit from this cost decrease. For researchers, companies, and organizations, using satellite data to train ML models becomes more economically viable with more accessible space.

3. **Rapid iteration and experimentation:** Researchers and data scientists can construct strong and precise ML algorithms more quickly by quickly iterating on their ML models, adjusting parameters, and testing ideas.

Disadvantages

1. **Availability of datasets:** Due to the landing and recovery processes taking up some volume and weight, this lowers the rocket's overall payload capacity. The amount of equipment or sensors that may be launched may be constrained in some circumstances due to this reduced payload capacity, which could have an impact on the quantity and calibre of data available for ML applications.

2. **Dependency on SpaceX's Launch Services:** Reliable availability to launch services is essential for the success of ML projects using space-based data. The main supplier of reusable rocket stages with successful landing capability at this time is SpaceX. If SpaceX's launch schedule or availability change, or if alternative launch providers are favoured for certain mission needs, ML programmes that largely rely on Falcon 9 stages for data collecting may be constrained. Projects involving machine learning (ML) may be at risk of disruption due to this reliance on a single source.

3. **Other Challenges:** ML initiatives that rely on precise launch schedules or time-sensitive data collecting may encounter difficulties. For ML applications, uncertainties and time restrictions can be introduced by the need to balance the requirements of several missions with the schedule for renovation.

8. Applications

1. **Tracking space debris and preventing collisions:** As there are more satellites in orbit, there are more worries about space debris and collision risks. To track space debris, forecast future collisions, and optimize satellite manoeuvres for collision avoidance, ML models can analyse data gathered by satellites. The deployment of specialized satellites for tracking space debris is made possible by the reusable Falcon 9 stages, and continual data collecting is made possible to enhance collision avoidance algorithms.

2. **Autonomous Satellite Operations:** The use of ML algorithms for autonomous satellite operations, such as payload management, attitude control, and navigation, is possible. ML models can be trained using real-time sensor inputs and previous mission data to improve mission success rates, increase efficiency, and optimize satellite operations.

9. Conclusion

There are several chances and benefits associated with working on ML projects under the SpaceX Falcon 9 success landing stage. The Falcon 9 stages' ability to be reused enables more frequent launches at lower costs, increasing the availability and variety of satellite data for ML applications. This results in enhanced model performance, accuracy, and training.

For purposes including prediction the successful Landing_Outcome machine learning (ML) techniques can make use of satellite data gathered by Falcon 9 flights. Reusable stages' ability to facilitate a continuous flow of data allows for real-time monitoring and analysis, which helps to improve predictions and decision-making.

10. Future Scope

We can make various recommendations for future research, including timetable optimization and determining the most effective ignition schedule. The failure phase of Falcon 9 rockets stage 1, which has not been covered yet, can also be used to complete the model. These next projects can aid in further dynamic and control assessments and boost the launch procedure' effectiveness.

11. Bibliography

Appendix

A. Source Code

Importing the libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the dataset

In [2]:

```
data = pd.read_csv('Spacex.csv')
data
```

```
data.head()              #first 5 rows

data.tail()              #last 5 rows

data.shape               #dimension of datasets

data.info()              #information about the columns
```

Data Preprocessing

Checking the missing values

In [7]:

```
data.isnull().any()
```

Dropping the columns

In [8]:

```
data = data.drop(columns = ['Date', 'Time (UTC)','Payload'])
```

In [9]:

```
data.head()            #update in dataset
```

In [10]:

```
data['Launch_Site'].value_counts()
```

In [11]:

```
data['Orbit'].value_counts()
```

For Booster Version

In [12]:

```
data.loc[data['Booster_Version'].str.contains('B5'), 'Booster_Version'] = 'B
5'
data.loc[data['Booster_Version'].str.contains('B4'), 'Booster_Version'] = 'B
4'
data.loc[data['Booster_Version'].str.contains('v1.0'), 'Booster_Version'] =
'v1.0'
data.loc[data['Booster_Version'].str.contains('v1.1'), 'Booster_Version'] =
'v1.1'
data.loc[data['Booster_Version'].str.contains('FT'), 'Booster_Version'] = 'F
T'
```

For Customer

In [14]:

```
data.loc[data['Customer'].str.contains('NASA'), 'Customer'] = 'NASA'
data.loc[data['Customer'].str.contains('SpaceX'), 'Customer'] = 'SpaceX'
data.loc[(data['Customer'] != 'NASA' ) & (data['Customer'] != 'SpaceX'),'Customer'] = 'Other'
```

For Mission Outcome

In [16]:

```
data.loc[data['Mission_Outcome'].str.contains('Success'), 'Mission_Outcome'] = 'Success'
data.loc[data['Mission_Outcome'].str.contains('Failure'), 'Mission_Outcome'] = 'Failure'
```

For Landing Outcome

In [18]:

```
data.loc[data['Landing _Outcome'].str.contains('Success'), 'Landing _Outcome'] = 'Success'
data.loc[data['Landing _Outcome'].str.contains('Controlled'), 'Landing _Outcome'] = 'Success'
data.loc[data['Landing _Outcome'].str.contains('Failure'), 'Landing _Outcome'] = 'Failure'
data.loc[data['Landing _Outcome'].str.contains('No'), 'Landing _Outcome'] = 'Failure'
data.loc[data['Landing _Outcome'].str.contains('Uncontrolled'), 'Landing _Outcome'] = 'Failure'
data.loc[data['Landing _Outcome'].str.contains('Precluded'), 'Landing _Outcome'] = 'Failure'
```

In [20]:

```
data.head()          #updated the dataset
```

Vizualization Techniques

Univariate Analysis

In [21]:

```
plt.scatter(x=data.index,y=data['PAYLOAD_MASS__KG_'])          #Scatter Plot
```

Out[21]:

```
<matplotlib.collections.PathCollection at 0x1e238447520>
```

In [22]:

```
plt.hist(data['Orbit'])                                #Histogram
sns.distplot(data['PAYLOAD_MASS__KG_'])                #Distribution Plot
```

Bivariate Analysis

```
sns.barplot(x=data.Customer,y=data.PAYLOAD_MASS__KG_) #Bar Plot : Relationsh
ip between Payload Mass provided by the Customers
sns.barplot(x=data.Mission_Outcome, y=data.PAYLOAD_MASS__KG_)    #Bar Plot :
 Relationship between mission success rate provide the amount of Payload
```

Encoding Techniques

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Booster_Version'] = le.fit_transform(data['Booster_Version'])
data['Launch_Site'] = le.fit_transform(data['Launch_Site'])
data['Orbit'] = le.fit_transform(data['Orbit'])
data['Customer'] = le.fit_transform(data['Customer'])
data['Mission_Outcome'] = le.fit_transform(data['Mission_Outcome'])
data['Landing _Outcome'] = le.fit_transform(data['Landing _Outcome'])
```

After Encoding

```
data.head()
```

Infering necessary information after Encoding

For Booster Version

```
data['Booster_Version'].value_counts()
data['Launch_Site'].value_counts()
```

For Orbit

```
data['Orbit'].value_counts()
data['Customer'].value_counts()
data['Mission_Outcome'].value_counts()
data['Landing _Outcome'].value_counts()
x = data.drop(columns = ['Landing _Outcome'], axis = 1)
y = data[['Landing _Outcome']]
```

Normalization

```
#Standardization

from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
x_scale = ss.fit_transform(x)     #the output will be 1D array
label = x.columns
x = pd.DataFrame(x_scale, columns = label)
x.head()                            #scaled data
```

Train Test Split

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size = 0.20, rando
m_state = 10)
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

Building Model - Using Hyperparameters

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

Logistic Regression

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

logreg_cv.best_score_                                                #best scor
e using after hypertuning

logreg_cv.best_params_                                               #best para

lr2 = LogisticRegression(C = 1, penalty = 'l2', solver = 'lbfgs')     #buildin
g the model using best_params_combination
lr2
lr2.fit(X_train,Y_train)                                              #training
 the model using best_params_combination
lr2.predict(X_test)                                                  #testing t
he model using best_params_combination
```

SVM

```
from sklearn.svm import SVC
svm = SVC()
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),'C': np.log
space(-3, 3, 5),'gamma':np.logspace(-3, 3, 5)}
svm_cv = RandomizedSearchCV(svm, parameters, cv= 10)
svm_cv.fit(X_train,Y_train)
svm_cv.best_score
svm_cv.best_params_
svm2 = SVC(C = 31.622776601683793, gamma = 1, kernel = 'linear')
svm2.fit(X_train,Y_train)
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
```

```
parameters = {'criterion': ['gini', 'entropy'],'splitter': ['best', 'random
'],'max_depth': [2*n for n in range(1,10)],'max_features': ['auto', 'sqrt'],
'min_samples_leaf': [1, 2, 4],'min_samples_split': [2, 5, 10]}
dt_cv = GridSearchCV(dt, parameters, cv= 10)
dt_cv.fit(X_train, Y_train)
dt_cv.best_score
dt_cv.best_params_
dt2 = DecisionTreeClassifier(criterion = 'entropy', max_depth = 4, max_featu
res = 'sqrt', min_samples_leaf = 2, min_samples_split = 5, splitter = 'best
')
dt2.fit(X_train,Y_train)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],'algorithm': ['
auto', 'ball_tree', 'kd_tree', 'brute'],'p': [1,2]}
knn_cv = GridSearchCV(knn, parameters, cv= 10)
knn_cv.fit(X_train, Y_train)
knn_cv.best_score_
knn_cv.best_params
knn2 = KNeighborsClassifier(algorithm = 'auto', n_neighbors = 4, p = 1)
knn2.fit(X_train,Y_train)
```

XG Boost

```
import xgboost as xgb
xg = xgb.XGBClassifier()
params = {
        'min_child_weight': [1, 5, 10],
        'gamma': [0.5, 1, 1.5, 2, 5],
        'subsample': [0.6, 0.8, 1.0],
        'colsample_bytree': [0.6, 0.8, 1.0],
        'max_depth': [3, 4, 5]
        }
xg_cv = RandomizedSearchCV(xg, param_distributions=params, n_iter=5, scoring
='roc_auc', n_jobs=4, cv=10, verbose=3)
xg_cv.fit(X_train,Y_train)
xg_cv.best_score_                                                      #best
xg_cv.best_params_                                                   #best p
xg2 = xgb.XGBClassifier(subsample = 1, min_child_weight = 1, max_depth = 3,
gamma = 1.5, colsample_bytree = 0.8)
xg2.fit(X_train,Y_train)
```

Accuracy Score

```
from sklearn.metrics import accuracy_score
def evaluation():
    print("Logistic Regression test data accuracy :",accuracy_score(Y_test, l
r2.predict(X_test)))
    print("Support Vector Machine test data accuracy:",accuracy_score(Y_test,
 svm2.predict(X_test)))
    print("Decision Tree accuracy on test set :",accuracy_score(Y_test, dt2.p
redict(X_test)))
    print("K Nearest Neighbors test data accuracy :",accuracy_score(Y_test,kn
n2.predict(X_test)))
    print("XG Boost test data accuracy :",accuracy_score(Y_test, xg2.predict
(X_test)))
evaluation()                  #calling function
```

```
Logistic Regression test data accuracy : 0.7619047619047619
Support Vector Machine test data accuracy: 0.9047619047619048
Decision Tree accuracy on test set : 0.9047619047619048
K Nearest Neighbors test data accuracy : 0.6190476190476191
XG Boost test data accuracy : 0.9047619047619048
```

Other Evaluation Metrics - XG Boost

```
from sklearn.metrics import confusion_matrix,roc_auc_score,precision_recall_
fscore_support, f1_score,roc_curve,classification_report
cm = confusion_matrix(Y_test,xg2.predict(X_test))
ax= plt.subplot()
sns.heatmap(cm, annot=True, ax = ax)
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['Failure', 'Success']); ax.yaxis.set_ticklabels(['Fa
ilure', 'Success'])
roc_auc_score(Y_test,xg2.predict(X_test))
roc_curve(Y_test,xg2.predict(X_test))
precision_recall_fscore_support(Y_test,xg2.predict(X_test))
f1_score(Y_test,xg2.predict(X_test))
print(classification_report(Y_test,xg2.predict(X_test)))
```

Model Deployment¶

```
import pickle
pickle.dump(xg2, open('model.pkl','wb'))
```