SpaceX Falcon 9 First Stage Landing Prediction

IBM Data Science Capstone Project

Matteo Vezzelli

June 16, 2023

Contents

Executive Summary

Goal

- Predict SpaceX Falcon 9 first stage landing success rate by using historic launching data.

Data Source

- Data requested from spacexdata.com using API.
- Falcon 9 launching data obtained from Wikipedia using web scraping.

Methodology

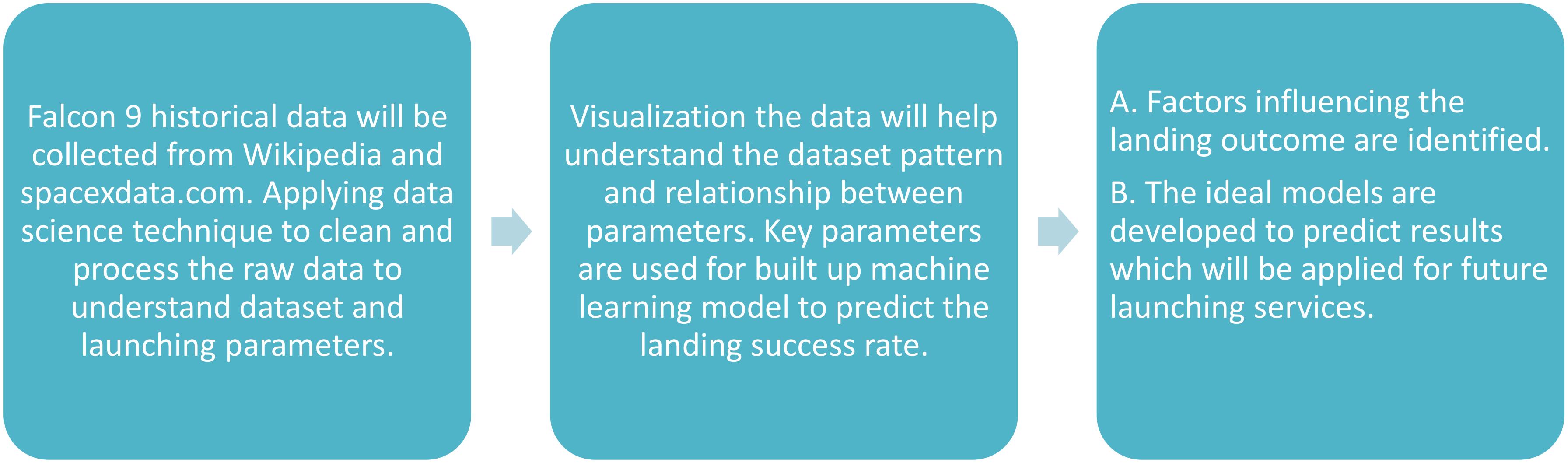
- Data processing: pandas, numpy, sql, BeautifulSoup, requests.
- Visualization and machine learning: matplotlib, seaborn, folium, dash, plotly, scikit-learn.

Results

- The model can predict the landing successful rate with high accuracy using testing data.

Introduction

SpaceY, a new born competitor of SapceX, provide the space launching service to the market. In order to further reduce the cost due to the potential failure of the first stage of landing. SpaceY determined to use SpaceX Falcon 9 historical launching data to predict the success rate of landing.



Falcon 9 historical data will be collected from Wikipedia and spacexdata.com. Applying data science technique to clean and process the raw data to understand dataset and launching parameters.

Visualization the data will help understand the dataset pattern and relationship between parameters. Key parameters are used for built up machine learning model to predict the landing success rate.

A. Factors influencing the landing outcome are identified.

B. The ideal models are developed to predict results which will be applied for future launching services.

Data Collection

And

Exploratory Data Analysis

Collecting is the starting of knowing.

Data Collection and Data Wrangling Methodology

Data Source	spacexdata	https://api.spacexdata.com/v4/rockets/ https://api.spacexdata.com/v4/launchpads/ https://api.spacexdata.com/v4/payloads/ https://api.spacexdata.com/v4/cores/ https://api.spacexdata.com/v4/launches/past
	Wikipedia	https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

	Data Collection	Data Wrangling
Tools	Requests http for humans BeautifulSoup	NumPy pandas

Key methods	requests.get().json()
	pandas.json_normalize(json_file)
	BeautifulSoup(data, ‘html’)
		panda.read_csv()
		df.insnull().sum()
		df[‘feature’].value_counts(); df.mean()

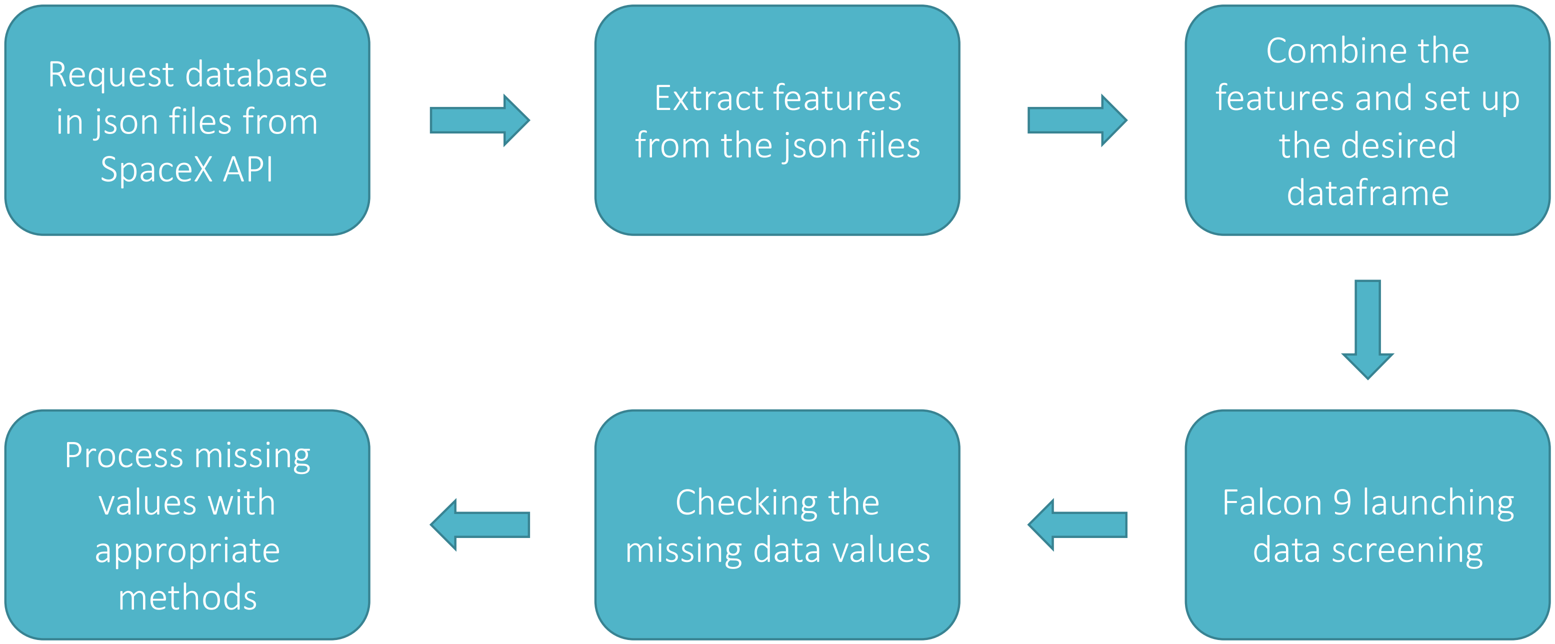
Data Collection

technique	tools	sources	contents
json file decoding	Requests http for humans	spacexdata.com API	Contains the information of Falcon 9, launching pads, payloads, cores, and historical launching records.
Web scarping	BeautifulSoup	Wikipedia Falcon 9 launching	Falcon 9 historical launch records including features.

Data Collection: SpaceX API

Objective	• Request to the SpaceX API.	• Clean the requested data.

Workflow



Request database in json files from SpaceX API

Extract features from the json files

Combine the features and set up the desired dataframe

Falcon 9 launching data screening

Checking the missing data values

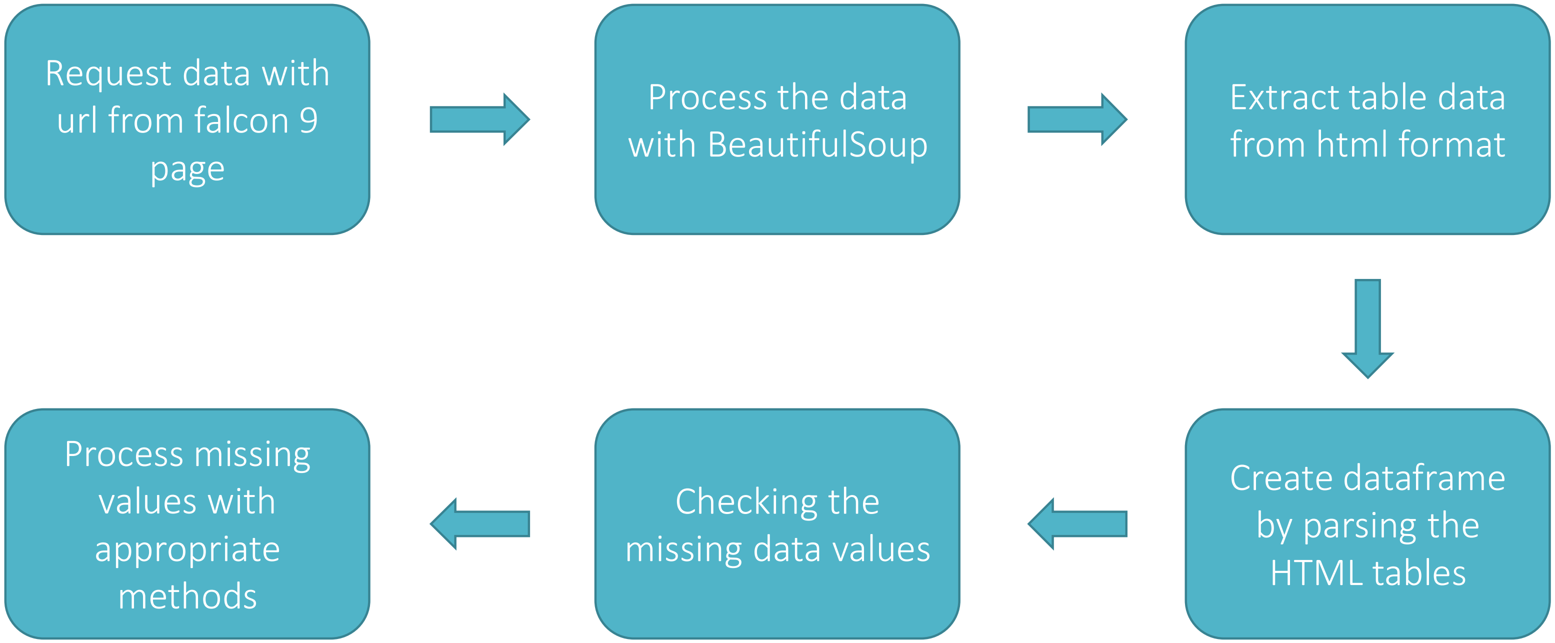
Process missing values with appropriate methods

Data Collection: Web Scrapping Wikipedia Page

Objective

- Extract a Falcon 9 launch records HTML table from Wikipedia.
- Parse the table and convert it into a Pandas data frame.

Workflow



Request data with url from falcon 9 page

Process the data with BeautifulSoup

Extract table data from html format

Create dataframe by parsing the HTML tables

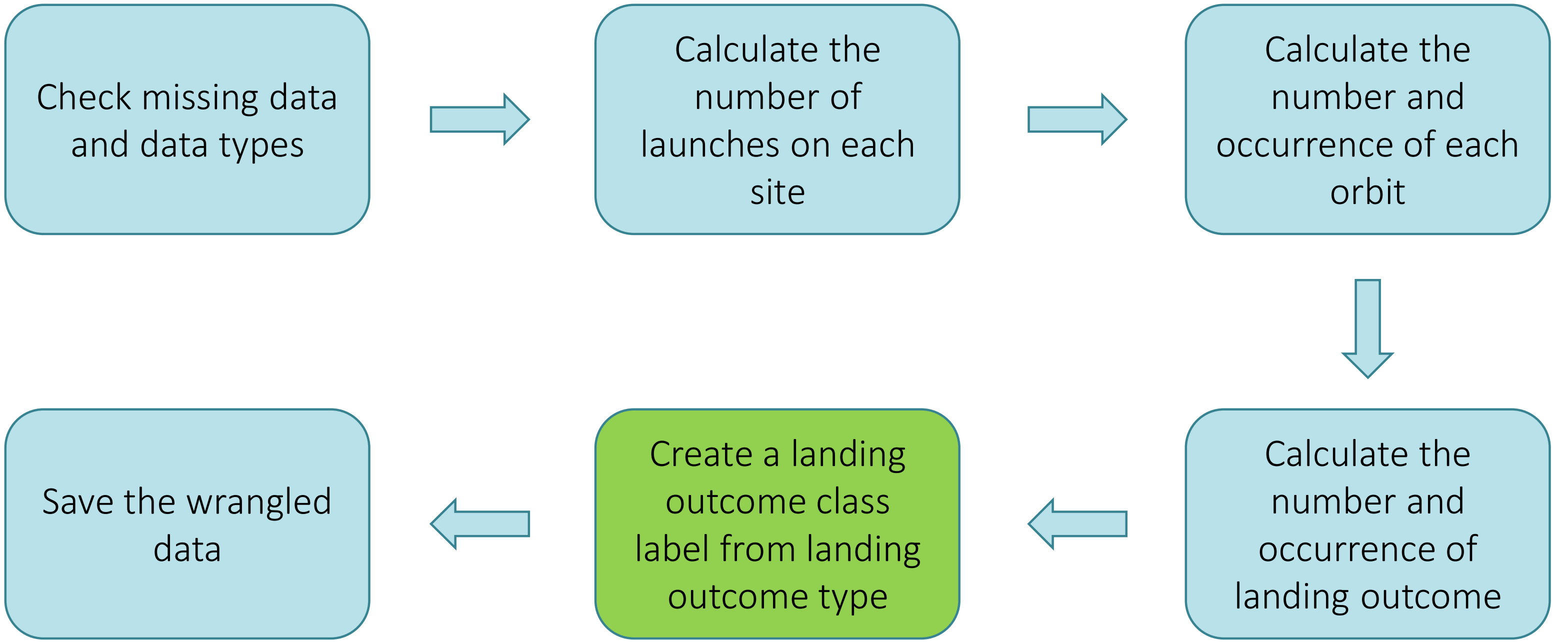
Checking the missing data values

Process missing values with appropriate methods

Data Wrangling

Objective	• Exploratory Data Analysis.	• Determine Training Labels.

Workflow



Check missing data and data types

Calculate the number of launches on each site

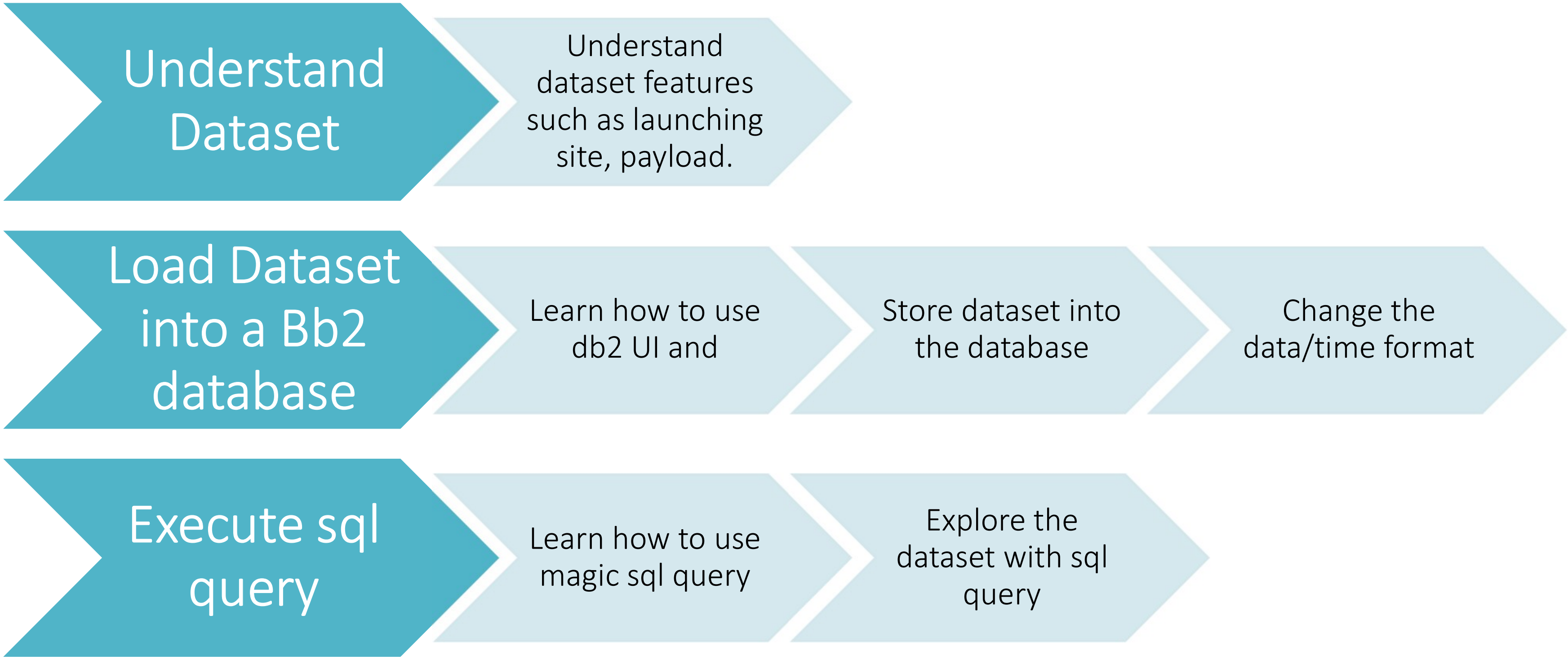
Calculate the number and occurrence of each orbit

Calculate the number and occurrence of landing outcome

Create a landing outcome class label from landing outcome type

Save the wrangled data

SQL Request Objectives



Understand Dataset

Understand dataset features such as launching site, payload.

Load Dataset into a Bb2 database

Learn how to use db2 UI and

Store dataset into the database

Change the data/time format

Execute sql query

Learn how to use magic sql query

Explore the dataset with sql query

SQL Request Result: Display Launch Sites

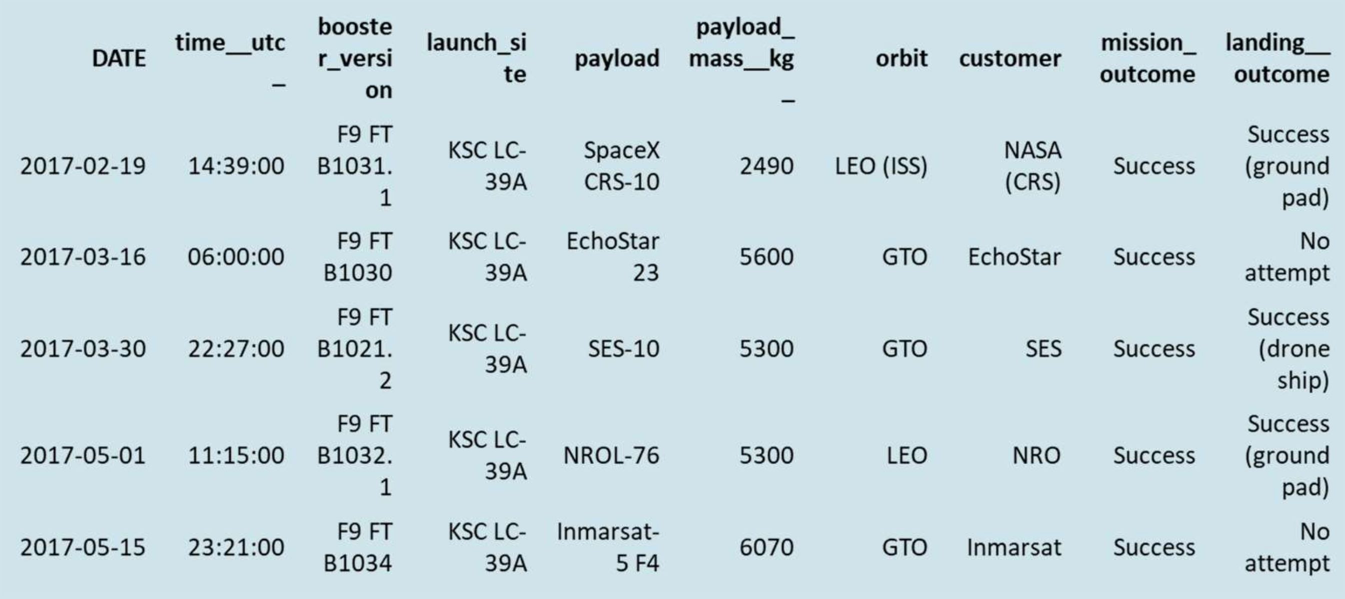
Query	%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
Results	launch_site CCAFS LC-40 CCAFS SLC-40 KSC LC-39A VAFB SLC-4E

SQL Request Result: Display 5 Records of Launch Site of “KSC”

Query

```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'KSC%' LIMIT 5;
```

Results



DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

SQL Request Result: Display the Total Payload Mass from NASA

Query	%sql SELECT SUM(PAYLOAD_MASS__KG_) AS PAYLOADMASS SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)’;
Results	Payloadmass 45596

SQL Request Result: Display Average Payload Carried by F9 v1.1

Query

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS PAYLOADMASS SPACEXTBL WHERE
BOOSTER_VERSION = 'F9 V1.1';
```

Results

```
Averagepayload
2928
```

SQL Request Result: Display Successful Drone Ship Landing Date

Query	%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING__OUTCOME = 'SUCCESS (DRONE SHIP)';
Results	1 2016-04-08

SQL Request Result: Display Boosters with Specific Constraints

Query

```
%sql SELECT  BOOSTER_VERSION FROM SPACEXTBL WHERE  ((LANDING__OUTCOME =
'SUCCESS (GROUND PAD)') & (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000));
```

Results

```
booster_version
_______________
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1
```

SQL Request Result: Display the Total Number of Different Outcomes

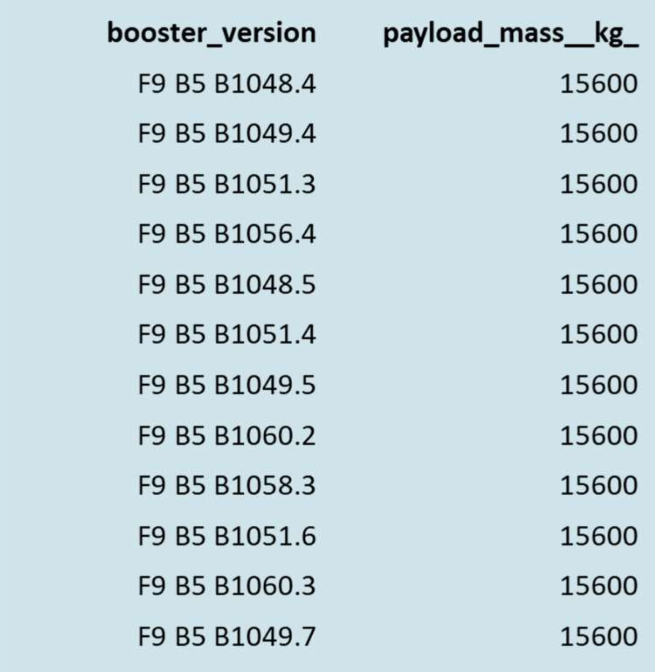
Query	%sql SELECT COUNT(MISSION_OUTCOME) AS OUTCOME FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
Results	Outcome 1 99 1

SQL Request Result: Display the Booster Which Carried The Max. Payload

Query

```
%sql SELECT  BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE
PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

Results



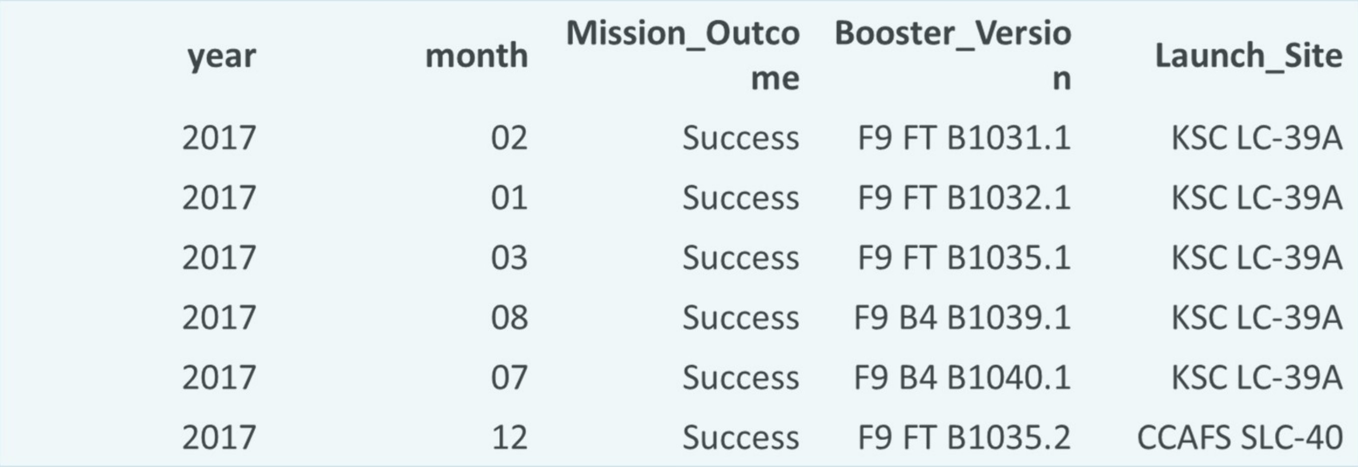
booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

SQL Request Result: Display Record with Multiple Features and Constraints

Query

```
%sql SELECT  SUBSTR(DATE, 1, 4) AS YEAR, SUBSTR(DATE, 6, 2) AS MONTH,
MISSION_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE
((SUBSTR(DATE, 1, 4)='2017') & (LANDING__OUTCOME = 'SUCCESS (GROUND PAD)'));
```

Results



year	month	Mission_Outcome	Booster_Version	Launch_Site
2017	02	Success	F9 FT B1031.1	KSC LC-39A
2017	01	Success	F9 FT B1032.1	KSC LC-39A
2017	03	Success	F9 FT B1035.1	KSC LC-39A
2017	08	Success	F9 B4 B1039.1	KSC LC-39A
2017	07	Success	F9 B4 B1040.1	KSC LC-39A
2017	12	Success	F9 FT B1035.2	CCAFS SLC-40

SQL Request Result: Display Record with Multiple Features and Constraints

Query

```
%sql SELECT  LANDING__OUTCOME, COUNT(*) AS COUNT_LAUNCHES FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY
LANDING__OUTCOME ORDER BY COUNT_LAUNCHES DESC ;
```

Results

```
     landing__outcome    count_launches
           No attempt                10
 Failure (drone ship)                 5
 Success (drone ship)                 5
   Controlled (ocean)                 3
 Success (ground pad)                 3
  Failure (parachute)                 2
 Uncontrolled (ocean)                 2
Precluded (drone ship)                1
```

EDA with Visualization Objectives

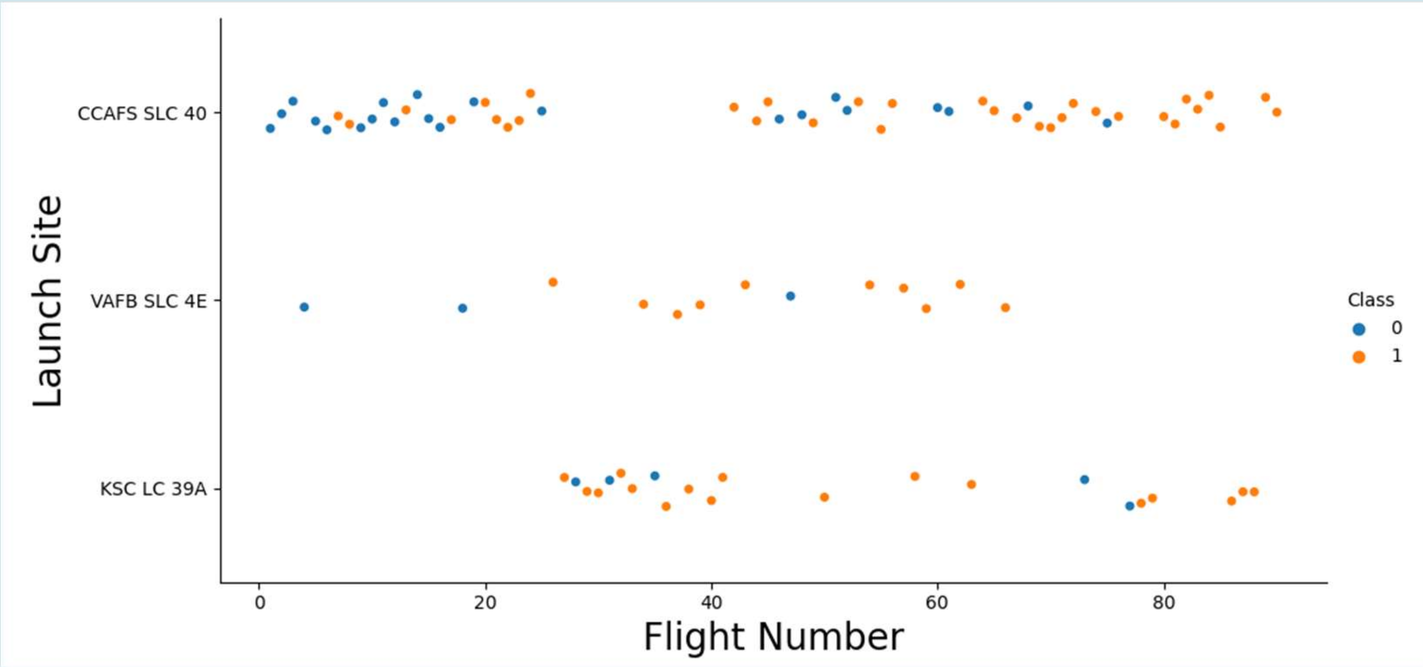
Objectives	Exploratory Data Analysis Visualization	Preparing Data Feature Engineering
Tools	pandas NumPy matplotlib seaborn	Use the pandas function get_dummies and features dataframe to apply OneHotEncoder to the column Orbits LaunchSite LandingPad Serial

Visualization: Flight Number Vs Launch Site

Code

```
sns.catplot(x = 'FlightNumber', y = 'LaunchSite', hue = 'Class', data = df, aspect= 2)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

Results
(Figures)



Comments

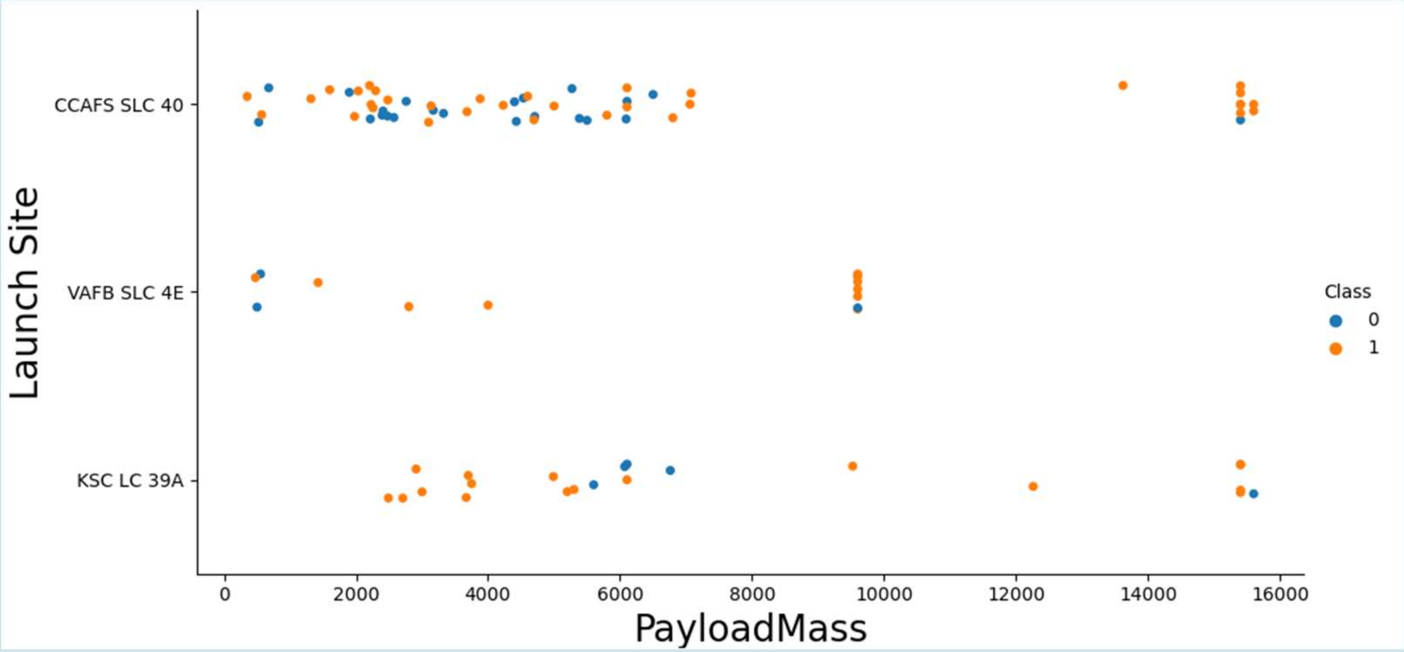
The sites CCAFS SLC40 and VAFB SLC 4E have increased successful rate along with the increasing of flight number. Launch site KSC LC 39 A has higher successful rate overall.

Visualization: Payload and Launch Site

Code

```
sns.catplot(x='PayloadMass', y='LaunchSite', data=df, hue='Class', aspect=2)
plt.xlabel("PayloadMass", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

Results (Figures)



Comments

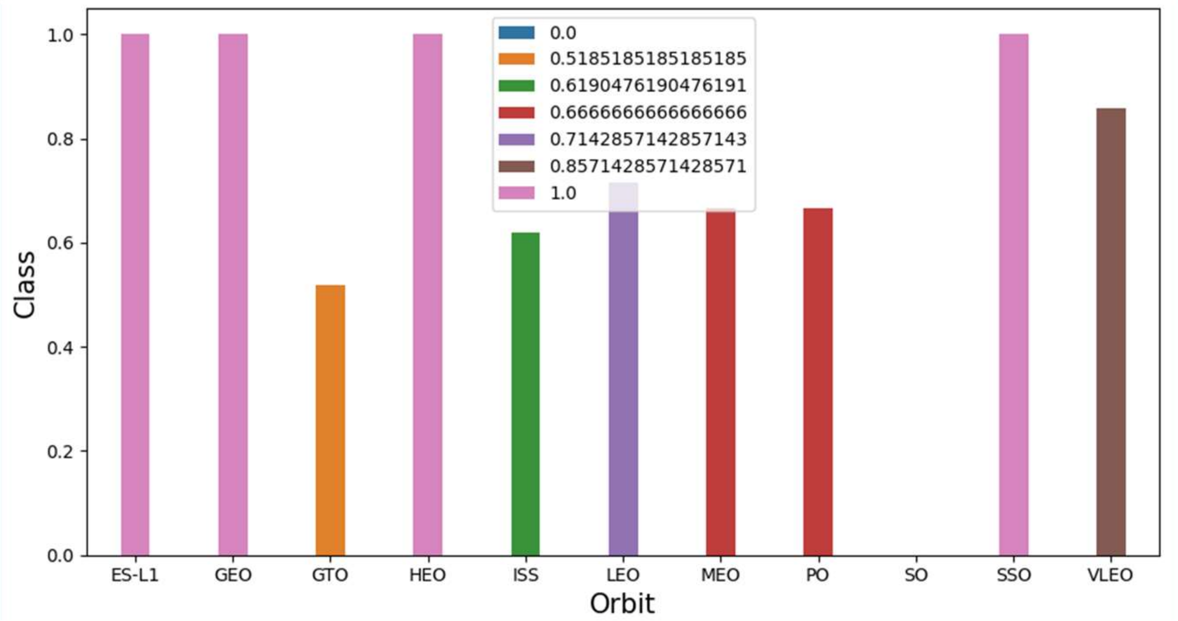
There are no rockets launched for heavy payload mass(greater than 10000) on VAFB SLC 4E.

Visualization: Success Rate of Each Orbit

Code

```
df_orbit=df.groupby(['Orbit'])['Class'].mean().reset_index()
sns.barplot(x='Orbit', y='Class', data=df_orbit, hue='Class',dodge=False, width = 0.3)
plt.legend(loc='upper center')
plt.xlabel("Orbit", fontsize=15)
plt.ylabel("Class", fontsize=15)
plt.show()
```

Results
(Figures)



Comments

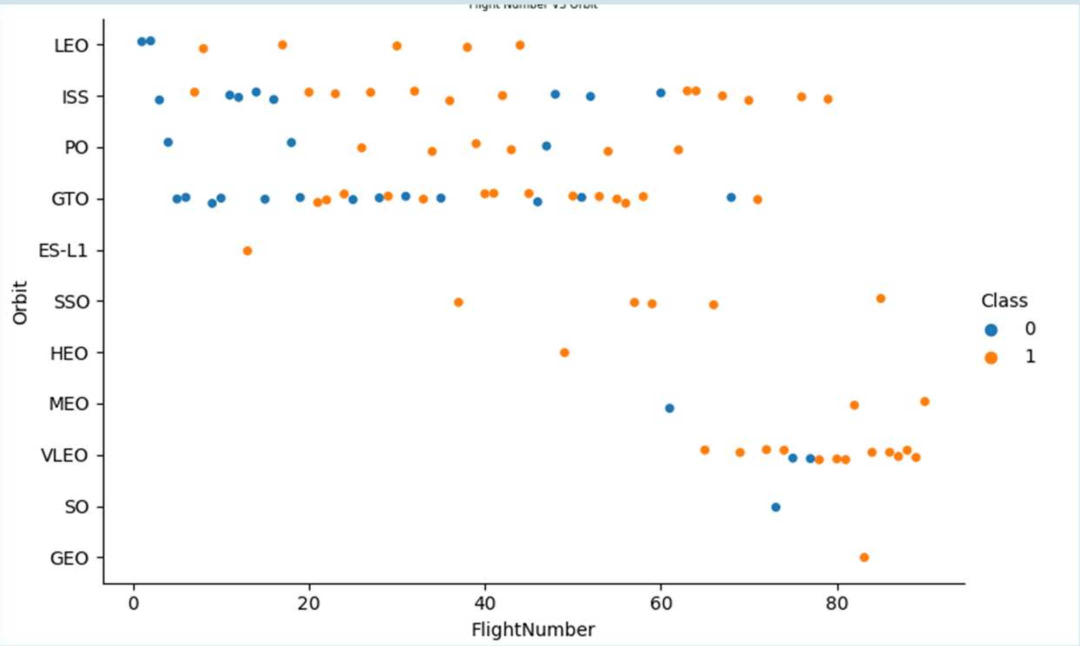
Orbits ES-L1, GEO, HEO and SSO have higher success rate.

Visualization: FlightNumber VS Orbit type

Code

```
sns.catplot(x='FlightNumber', y='Orbit', data =df, hue='Class', height = 5,  aspect=1.5)
plt.title("Flight Number VS Orbit", fontsize=6)
plt.show(block=True)
```

Results (Figures)



Comments

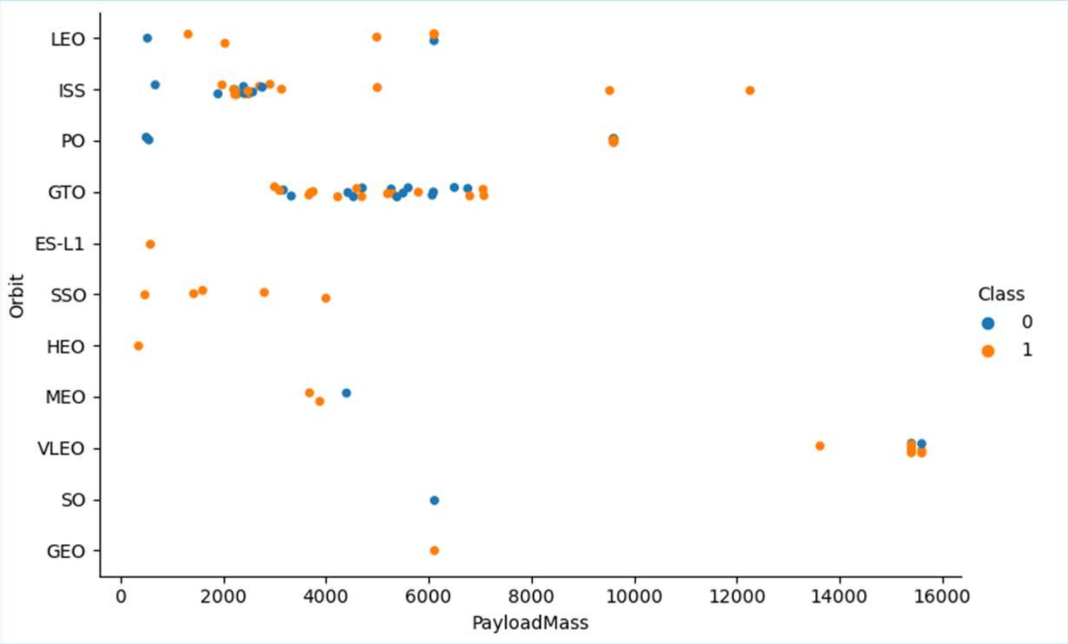
In the LEO orbit the success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Visualization: Payload VS Orbit type

Code

```
sns.catplot(x='PayloadMass', y='Orbit', data=df, hue='Class',height = 8, aspect =1)
plt.show()
```

Results
(Figures)

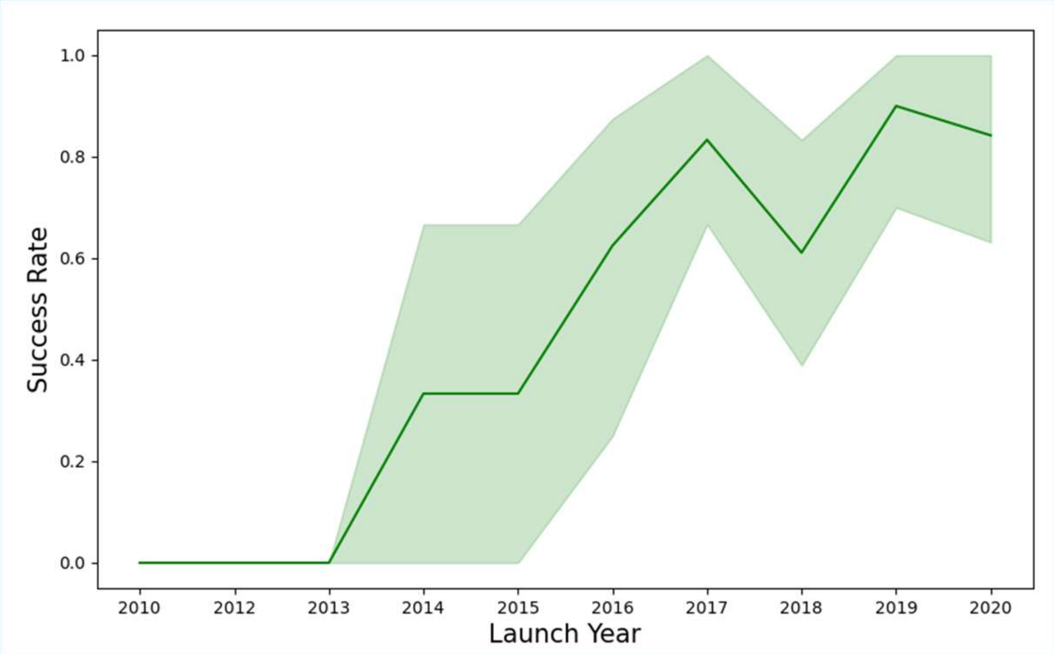


Comments

SSO, LEO and ISS have higher successful landings. GTO has mixed landing outcomes.

Visualization: Launch Success Yearly Trend

Code	sns.lineplot(x='Date', y='Class', data=df, color='green')
	plt.xlabel("Launch Year", fontsize=15)
	plt.ylabel("Success Rate", fontsize=15)
	plt.show()

Results (Figures)	
Comments	The success rate since 2013 kept increasing till 2020.

Features Engineering: Create dummy variables to categorical columns

Task	Code

Create dummy variables to categorical columns	features_one_hot=pd.get_dummies(features, columns=['Orbit', 'LaunchSite', 'LandingPad', 'Serial'])
	features_one_hot.head()

Cast all numeric columns to float64	features_one_hot.astype('float64')

Interactive Data Analysis
And
Machine Learning

Diving into the data; discovering the knowledge.

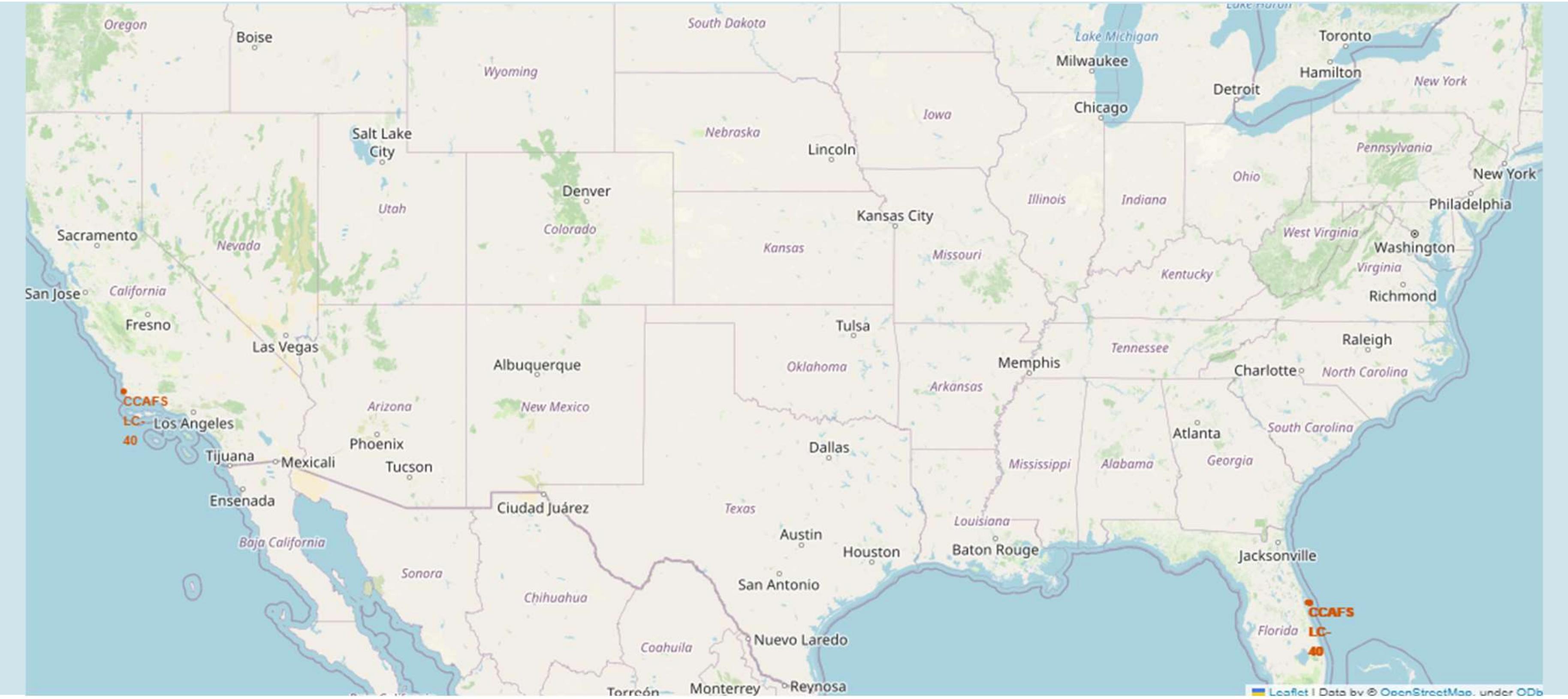
Interactive Visual Analytics Objectives and Methodology

Objective	Tools
• Mark all launch sites on a map. • Mark the success/failed launches for each site on the map. • Calculate the distances between a launch site to its proximities.	Folium
• Build an Interactive Dashboard with Ploty Dash	plotly | Dash

Launch Sites Locations Analysis with Folium

Task 1

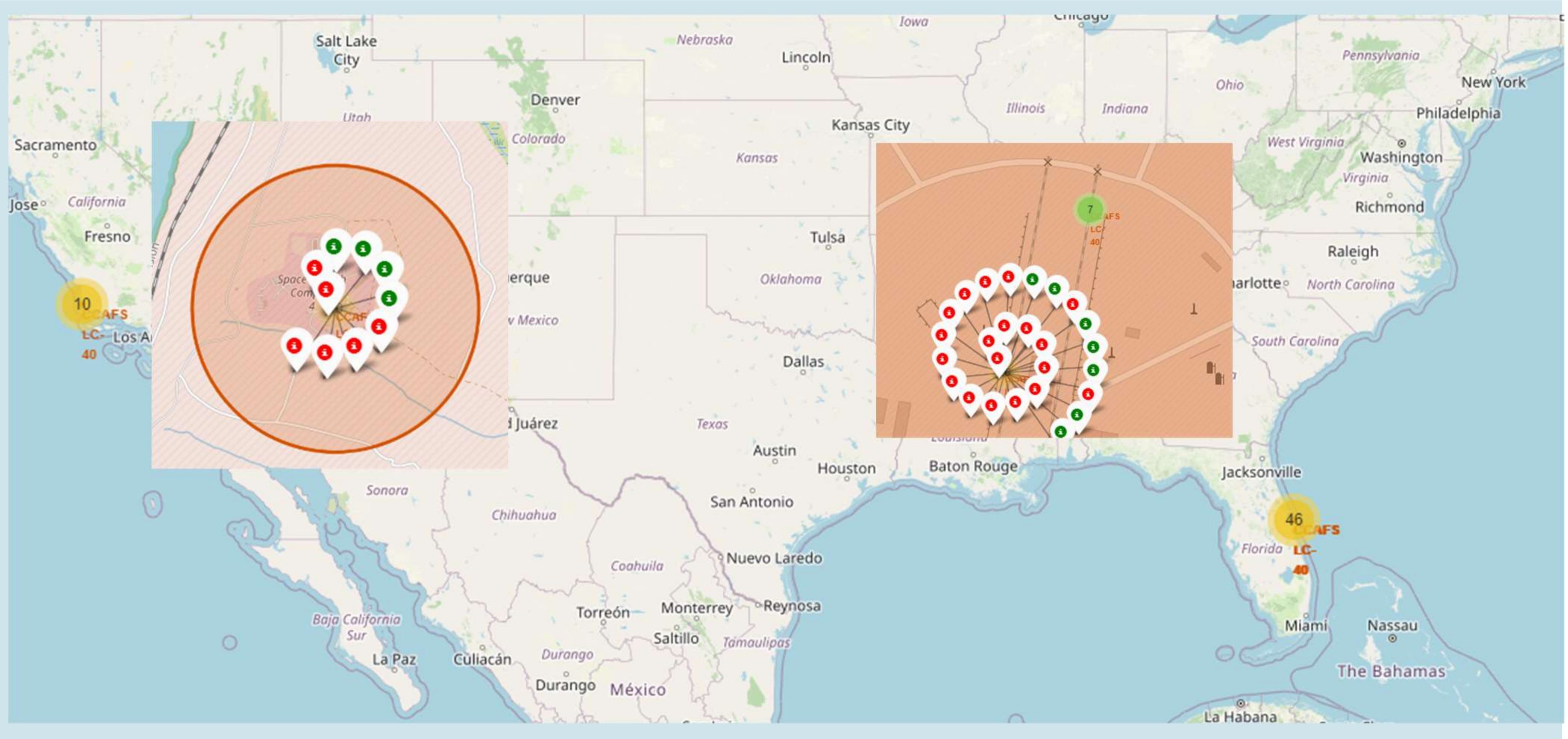
Mark all launch sites on a map.



Launch Sites Locations Analysis with Folium

Task 2

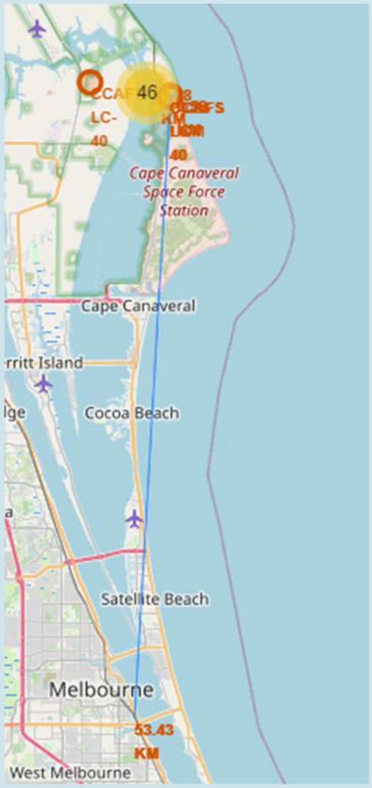
Mark the success/failed launches for each site on the map.

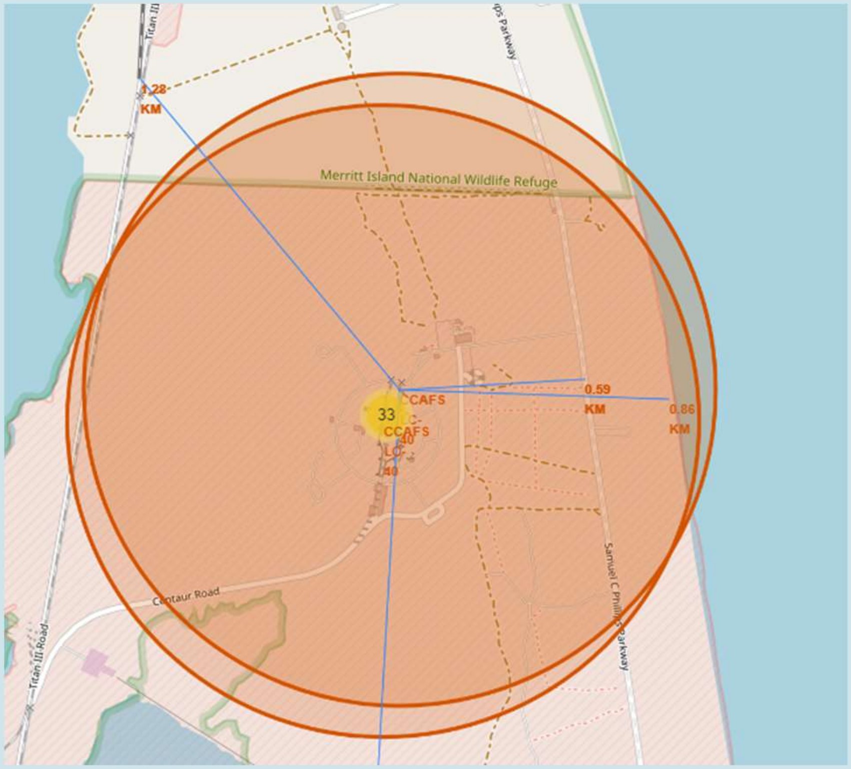


Launch Sites Locations Analysis with Folium

Task 3

Calculate the distances between a launch site to its proximities.

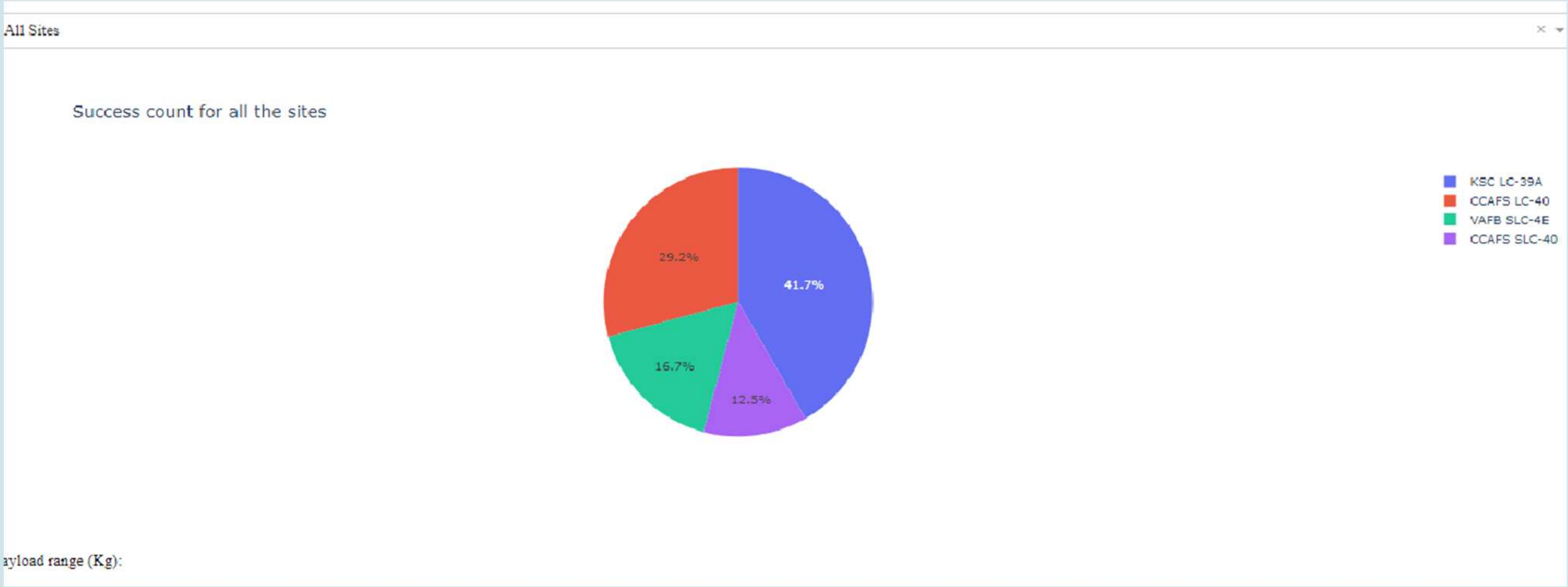



1.28 KM
Merritt Island National Wildlife Refuge
0.59 KM
0.86 KM
33
CCAFS
CCAFS
LC-40
Centaur Road
Titan III Road
Samuel C Phillips Parkway

Build an Interactive Dashboard with Ploty Dash

Task 1

Display the success count achieved by each launch site.

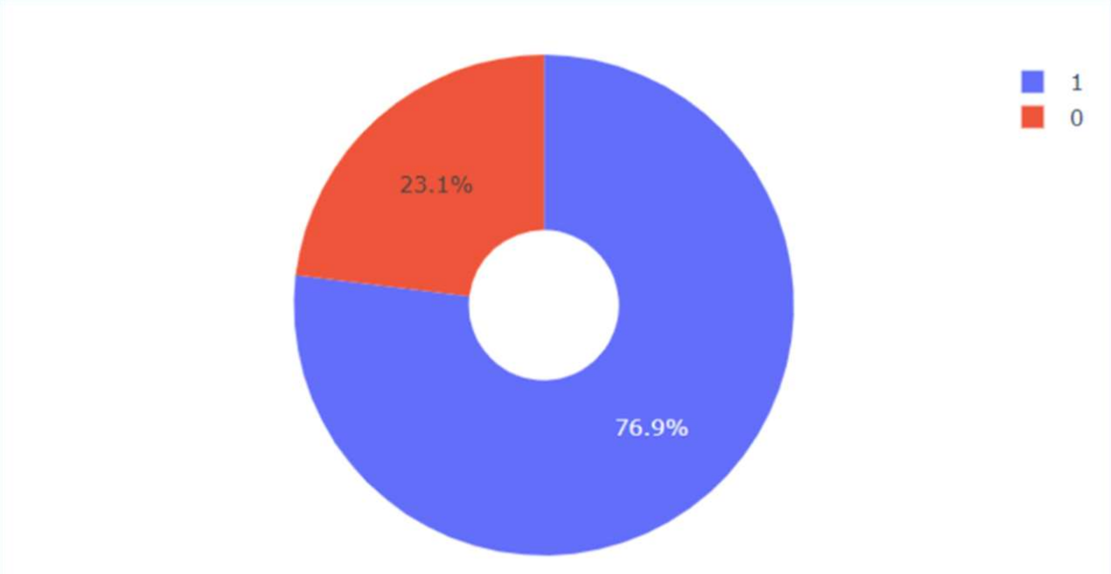


KSC LC-39A has the highest percentage of success count.

Build an Interactive Dashboard with Ploty Dash

Task 2

Display the success percentage achieved by a single site.

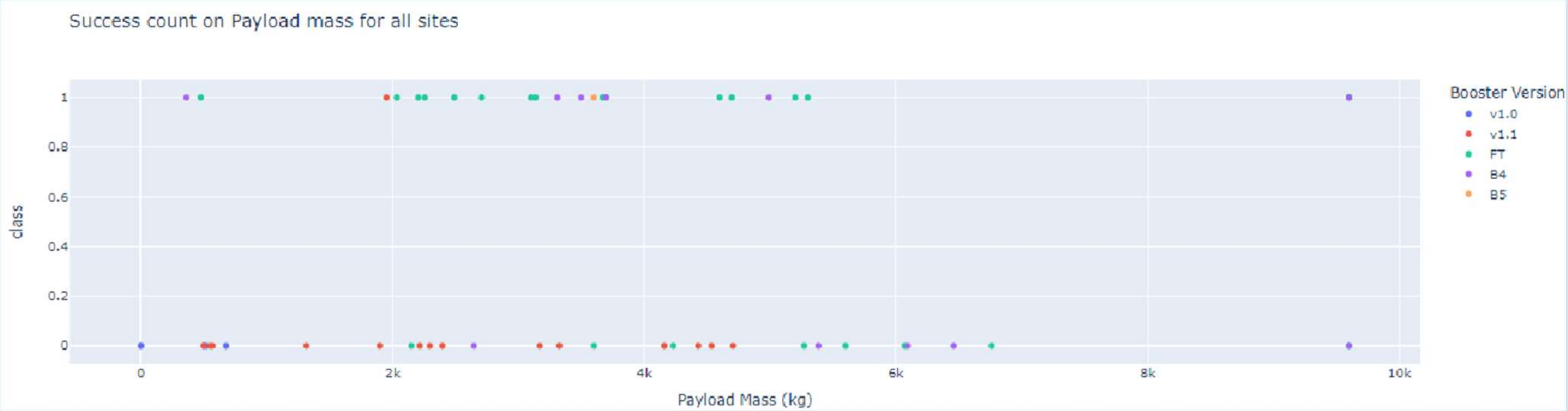


KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Build an Interactive Dashboard with Ploty Dash

Task 3

Display success count on payload mass for all sites.



Predictive Analysis Objectives and Methodology

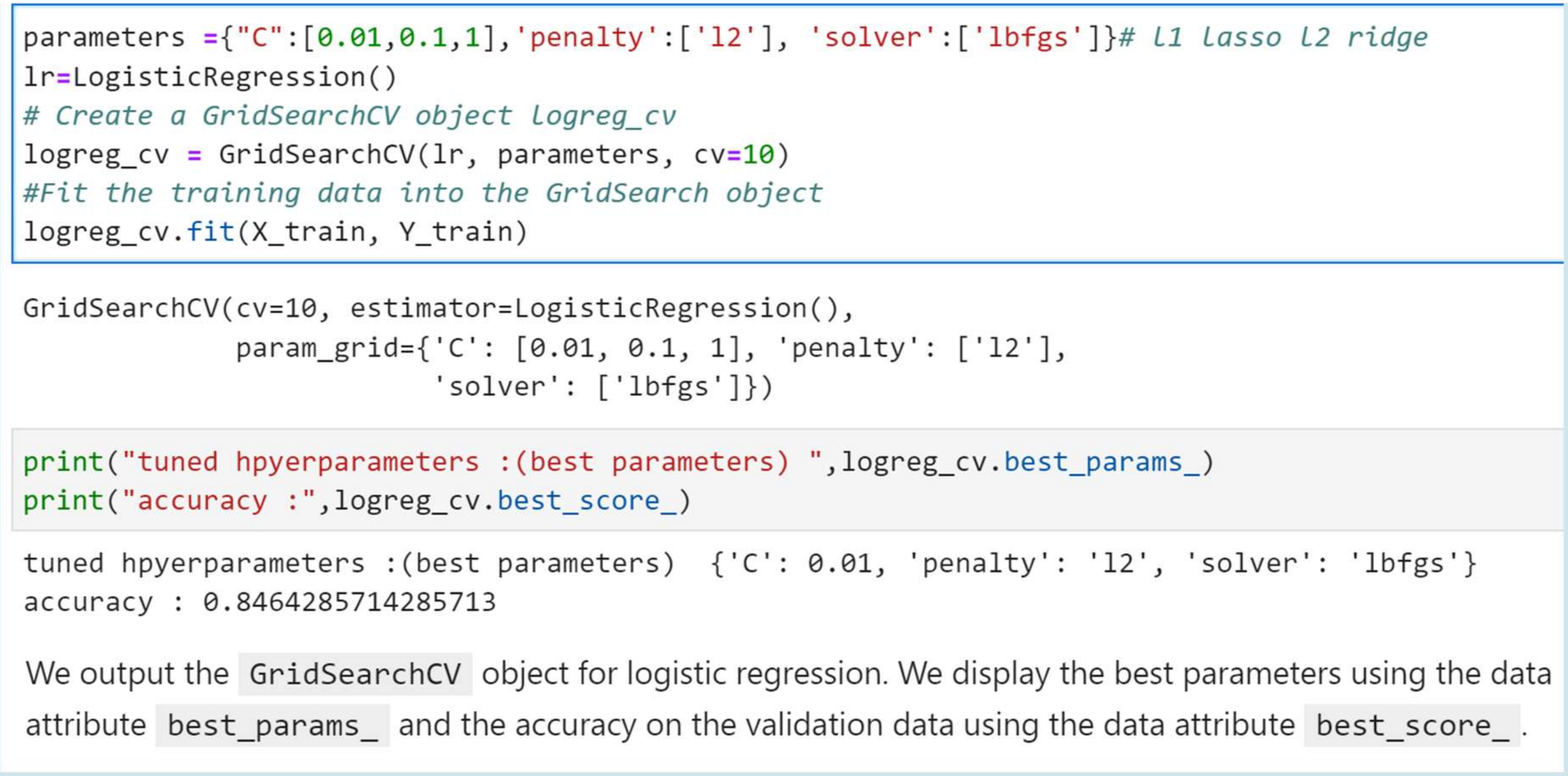
Objective	Tools
• Perform exploratory data analysis and determine training labels • Create a column for the class • Standardize the data • Split into training data and test data	pandas matplotlib seaborn
• Find best Hyperparameter for SVM, Classification Trees and Logistic Regression • Find the method performs best using test data	scikit learn

Predictive Analysis:

Task 1

Create a logistic regression object then create a GridSearchCV object logreg_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.

Results



```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
# Create a GridSearchCV object logreg_cv
logreg_cv = GridSearchCV(lr, parameters, cv=10)
#Fit the training data into the GridSearch object
logreg_cv.fit(X_train, Y_train)
```

```
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
```

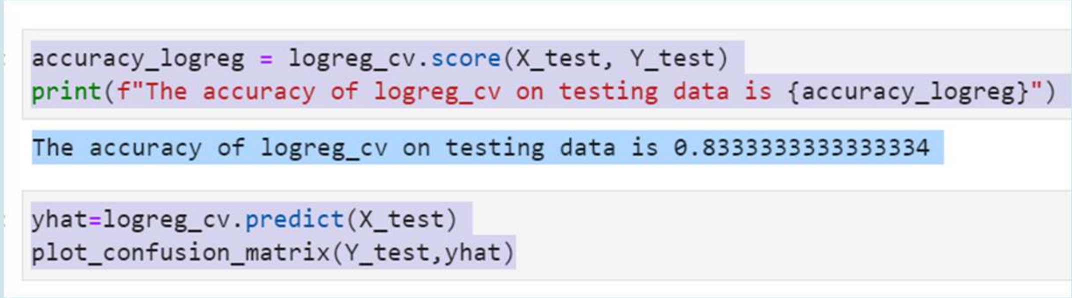
We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

Predictive Analysis:

Task 2

Calculate the accuracy on the test data using the method score.

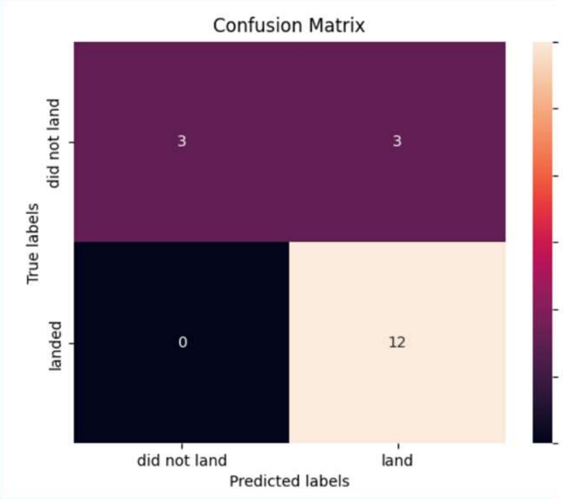
Results



```
accuracy_logreg = logreg_cv.score(X_test, Y_test)
print(f"The accuracy of logreg_cv on testing data is {accuracy_logreg}")
```

```
The accuracy of logreg_cv on testing data is 0.8333333333333334
```

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

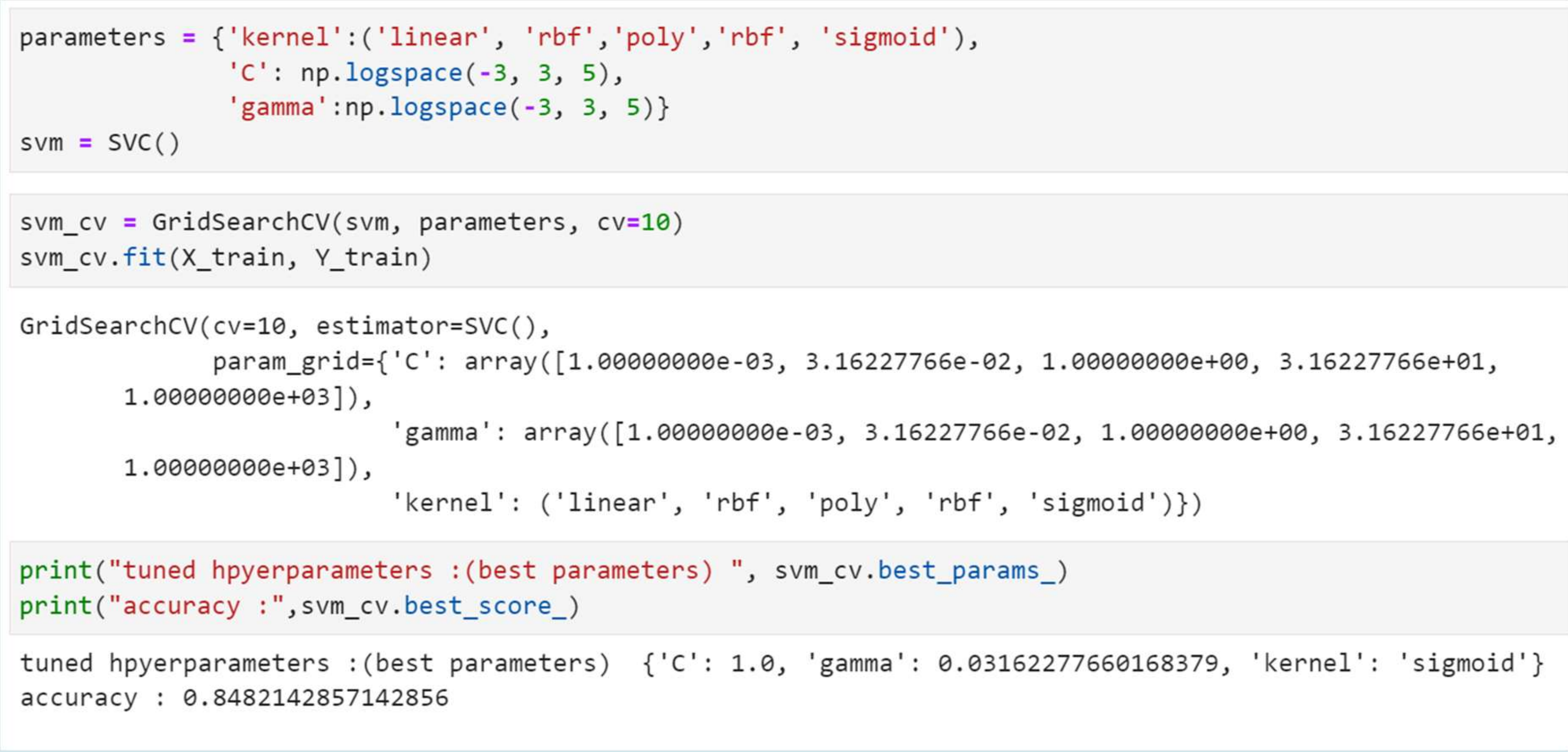


Predictive Analysis:

Task 3

Create a support vector machine object then create a GridSearchCV object svm_cv with cv - 10. Fit the object to find the best parameters from the dictionary parameters.

Results



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

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

```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

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

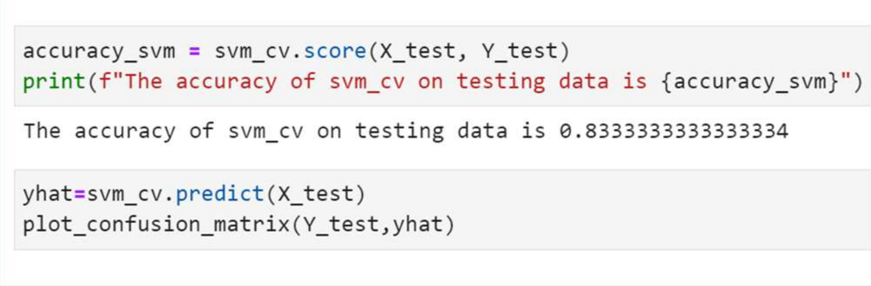
```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

Predictive Analysis:

Task 4

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

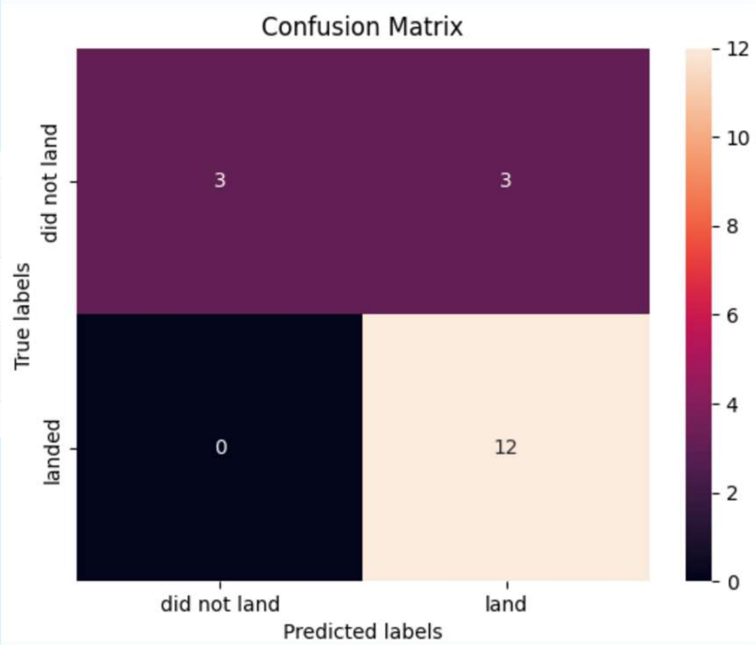
Results



```
accuracy_svm = svm_cv.score(X_test, Y_test)
print(f"The accuracy of svm_cv on testing data is {accuracy_svm}")
```

```
The accuracy of svm_cv on testing data is 0.8333333333333334
```

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

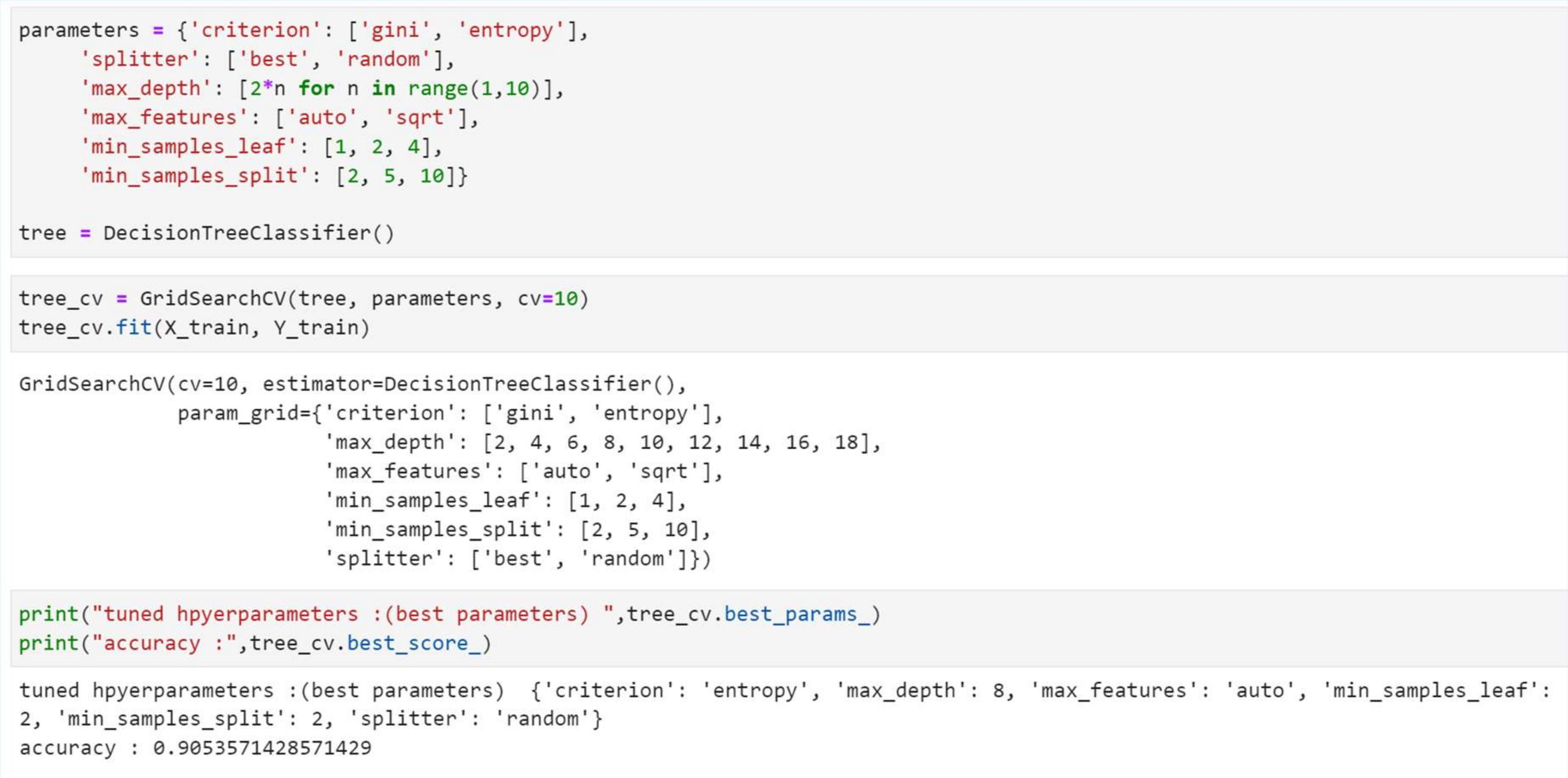


Predictive Analysis:

Task 5

Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.

Results



```
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]}

tree = DecisionTreeClassifier()
```

```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

```
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']})
```

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

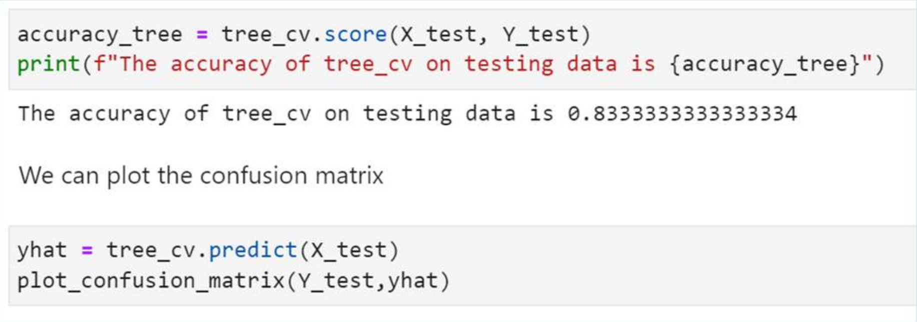
```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf':
2, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.9053571428571429
```

Predictive Analysis:

Task 6

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

Results

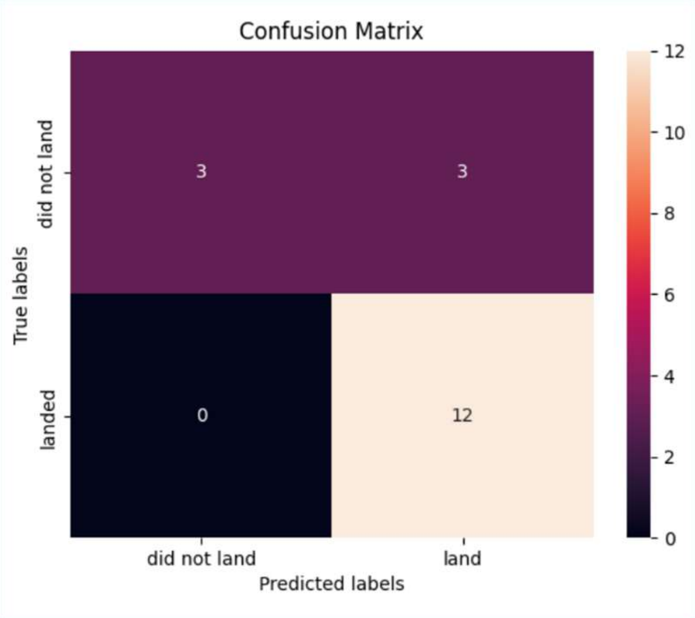


```
accuracy_tree = tree_cv.score(X_test, Y_test)
print(f"The accuracy of tree_cv on testing data is {accuracy_tree}")
```

```
The accuracy of tree_cv on testing data is 0.8333333333333334
```

We can plot the confusion matrix

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

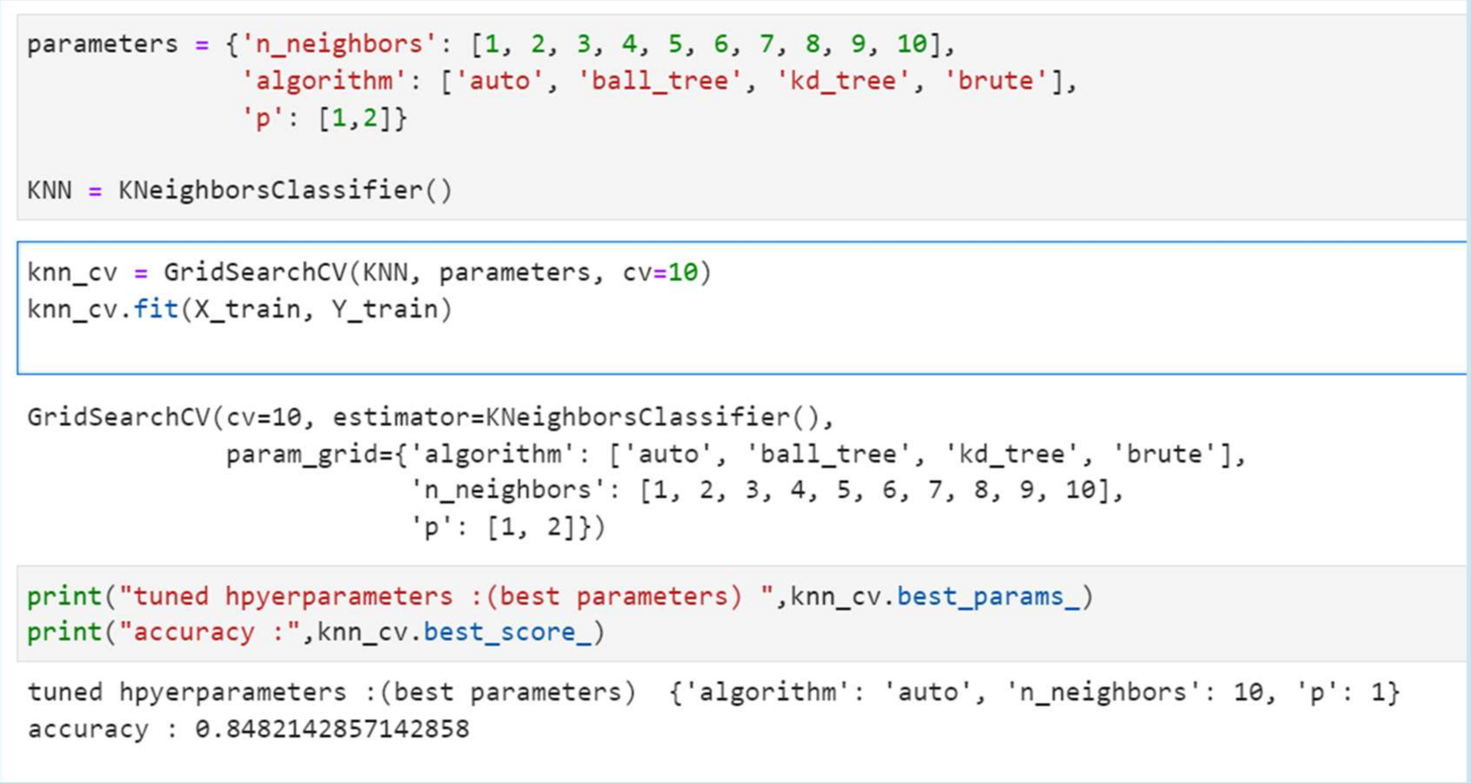


Predictive Analysis:

Task 7

Create a k nearest neighbors object then create a GridSearchCV object knn_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.

Results



```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

```
knn_cv = GridSearchCV(KNN, parameters, cv=10)
knn_cv.fit(X_train, Y_train)
```

```
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]})
```

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

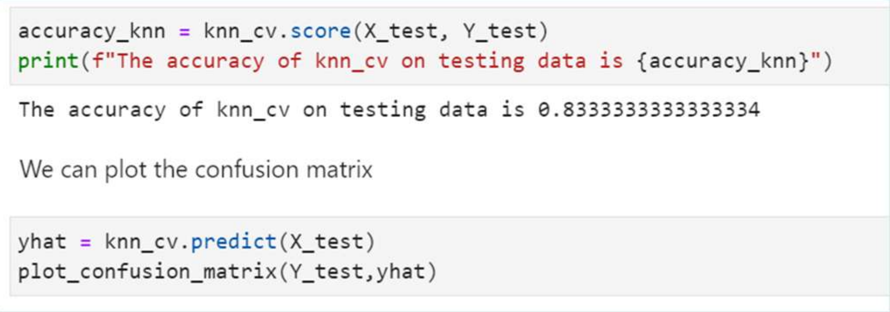
```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

Predictive Analysis:

Task 8

Calculate the accuracy of knn_cv on the test data using the method score:

Results

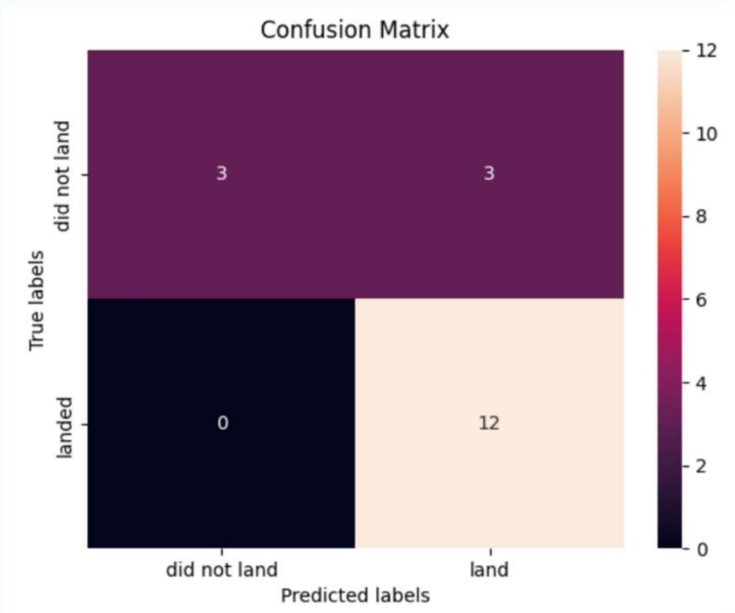


```
accuracy_knn = knn_cv.score(X_test, Y_test)
print(f"The accuracy of knn_cv on testing data is {accuracy_knn}")
```

```
The accuracy of knn_cv on testing data is 0.8333333333333334
```

We can plot the confusion matrix

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

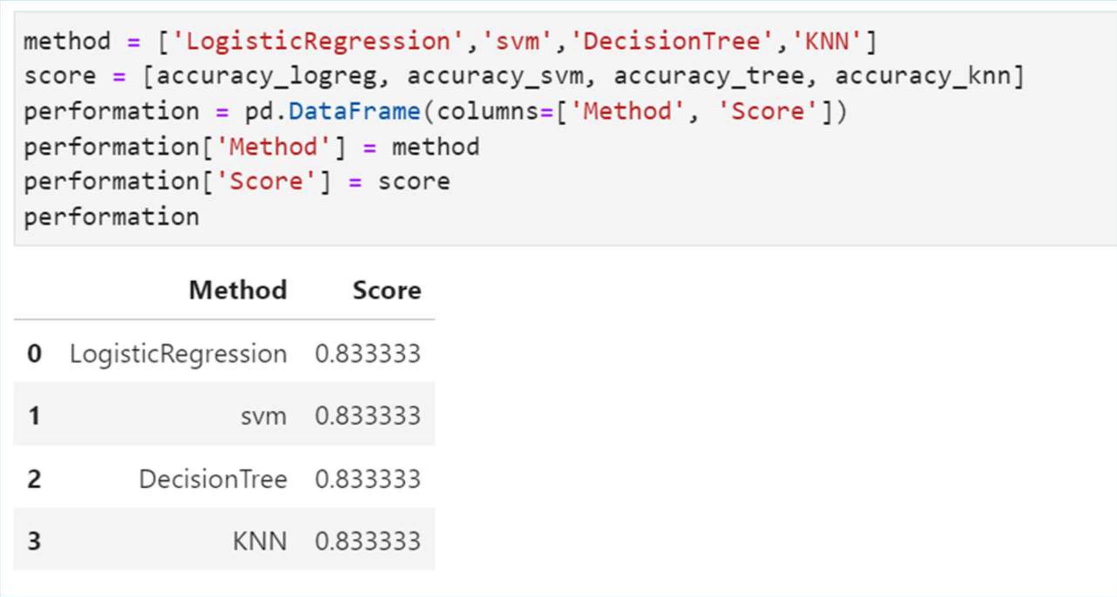


Predictive Analysis:

Task 9

Find the method performs best:

Results



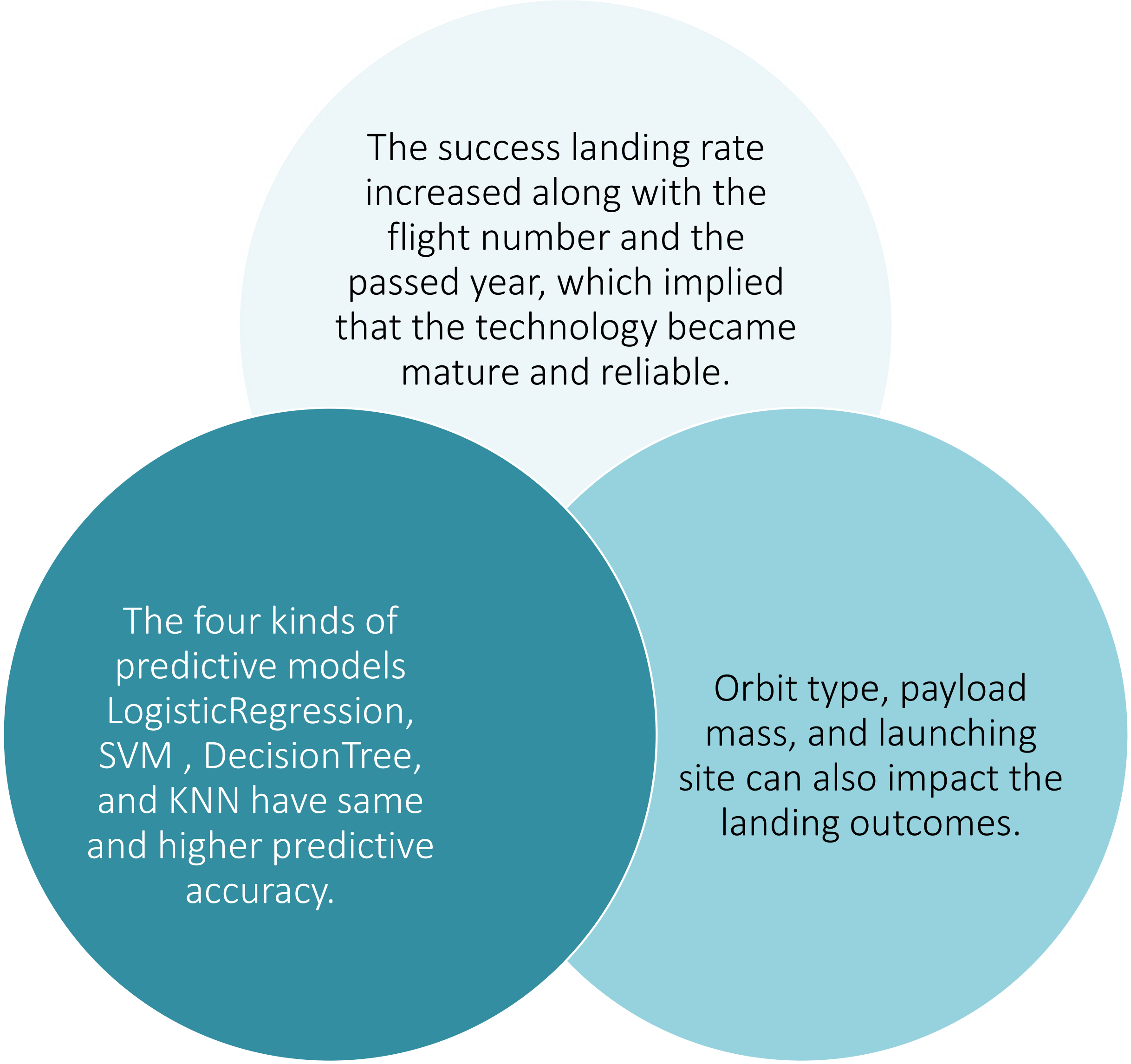
```python
method = ['LogisticRegression','svm','DecisionTree','KNN']
score = [accuracy_logreg, accuracy_svm, accuracy_tree, accuracy_knn]
performation = pd.DataFrame(columns=['Method', 'Score'])
performation['Method'] = method
performation['Score'] = score
performation
```

	Method	Score
0	LogisticRegression	0.833333
1	svm	0.833333
2	DecisionTree	0.833333
3	KNN	0.833333

Results
And
Takeaway

The information from data will empower a person.

Conclusion



The success landing rate increased along with the flight number and the passed year, which implied that the technology became mature and reliable.

The four kinds of predictive models LogisticRegression, SVM , DecisionTree, and KNN have same and higher predictive accuracy.

Orbit type, payload mass, and launching site can also impact the landing outcomes.

Appendix

Notebook GitHub links:

1_1_Data_collection_API.ipynb
1_2_Data_collection_web-scraping.ipynb
1_3_Data_collection_data-wrangling.ipynb
2_1_EDA_with_SQL.ipynb
2_2_EDA_with_visualizations.ipynb
3_1_Interactive_visual_analytics_with_Folium.ipynb
3_2_Interactive_dashboard_with_Ploty_Dash.py
4_1_Machine_Learning_prediction.ipynb