The New Space Age

Using Data Science to Revolutionise Commercial Rocketry

Thomas Pedersen

24th July 2024

IBM Developer

SKILLS NETWORK

OUTLINE

EXECUTIVE SUMMARY

Context:

- Commercial space industry is thriving with companies like Virgin Galactic, Rocket Lab, Blue Origin, and SpaceX.
- SpaceX's Falcon 9 stands out for its reusability, significantly lowering launch costs compared to competitors.

Objective:

- Determine the launch cost for SpaceY rockets by predicting SpaceX Falcon 9 first stage reusability.

Methodology:

- Collect historical data on SpaceX launches, including mission parameters and first stage landing results.
- Develop dashboards and visualizations to monitor and analyze prediction accuracy and trends over time.
- Use feature engineering to identify predictive factors influencing first stage reusability.
- Train machine learning models (e.g., logistic regression, decision trees) on labeled data to predict successful landings.

Conclusion:

- By leveraging machine learning and comprehensive data analysis, SpaceY can strategically position itself in the market, offering cost-effective alternatives to current industry leaders.

INTRODUCTION

Context

In the rapidly evolving landscape of commercial space travel, the competitive dynamics are driven by technological innovation and cost-efficiency. This project focuses on leveraging data science to address a pivotal challenge faced by emerging players in the industry, exemplified by SpaceY: accurately predicting the reusability of SpaceX's Falcon 9 first stage.

Problem Statement

The central challenge revolves around determining the cost-effectiveness of rocket launches by predicting whether SpaceX will successfully recover and reuse the Falcon 9 first stage. This prediction is critical as it directly influences the pricing strategy for SpaceY's launch services, aiming to compete with established entities like SpaceX.

Key Questions

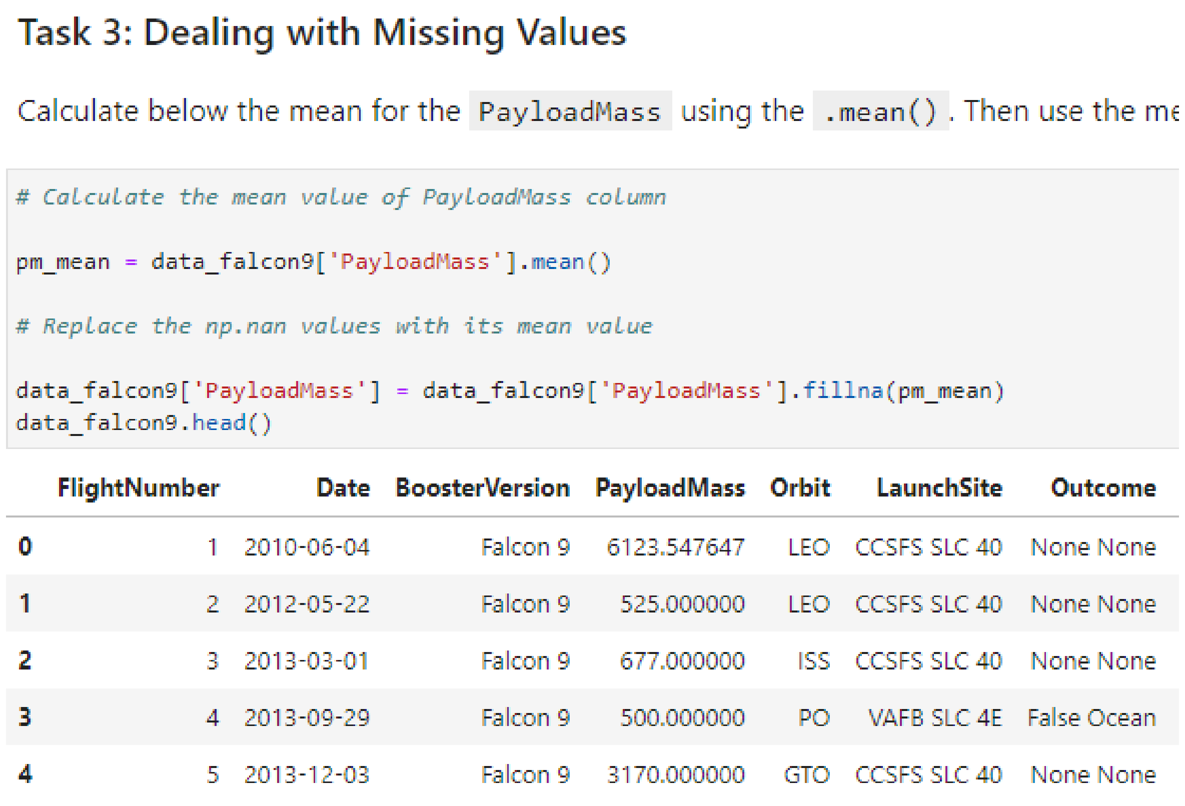
- Can we reliably predict whether SpaceX will reuse the Falcon 9 first stage based on historical mission data?
- How can machine learning models enhance the accuracy and efficiency of predicting first stage reusability compared to traditional methods?
- What insights can be derived from the analysis to inform strategic decisions regarding pricing and market positioning for Space Y?

SECTION 1: METHODOLOGY

1. Data Collection and Wrangling
2. Exploratory Data Analysis
3. Interactive Visual Analytics
4. Predictive Analysis

Data Collection and Wrangling

- Data was collected using the SpaceX REST API
- Data was also collected by web scraping the Falcon 9 launches Wikipedia page
- All data not relevant to the Falcon 9 rocket was removed
- Missing payload mass data was replaced with the mean payload mass value
- Each launch was categorised as having either a success or failure in terms of whether or not the first stage booster landed successfully
- One hot encoding was used on relevant categorical variables to prepare the data for a predictive machine learning model



Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the me

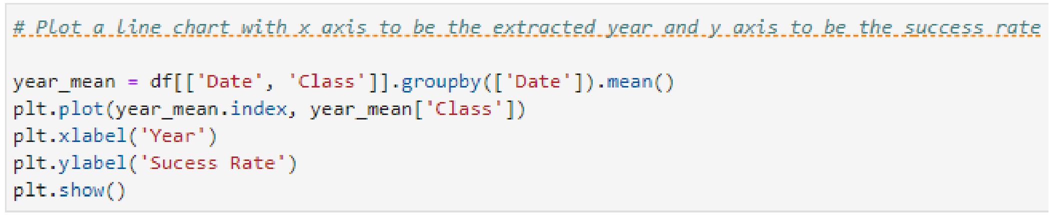
```
# Calculate the mean value of PayloadMass column

pm_mean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value

data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(pm_mean)
data_falcon9.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome
0	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None

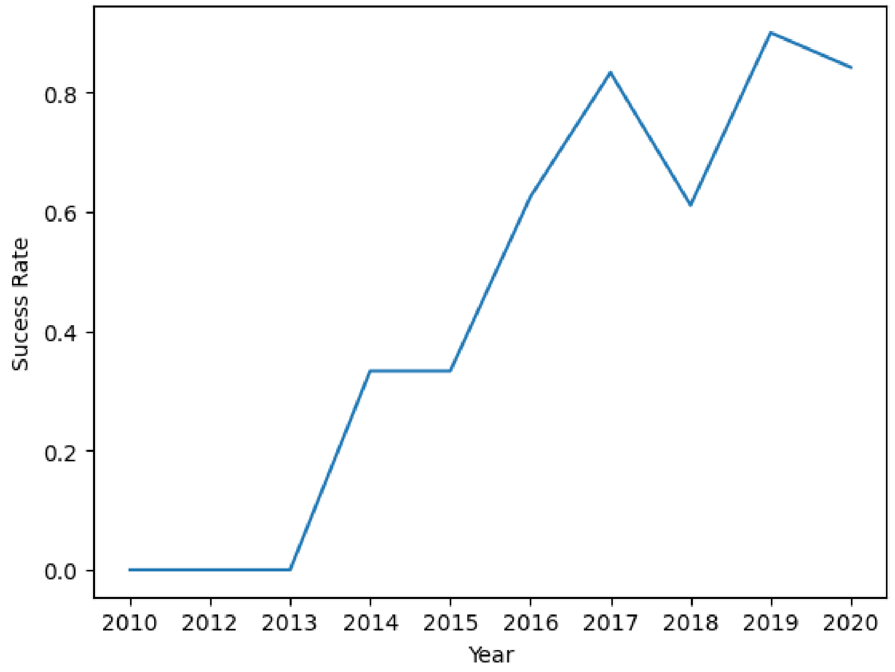
Exploratory Data Analysis

- SQL was used to develop a greater understanding of the data, including:
    - The names of the distinct launch sites
    - The average payload mass
    - The date of the first successful outcome on a ground pad
    - Which booster versions have carried the maximum payload

- Matplotlib and Pandas were used to determine the nature of the relationships between variables, including:
    - Flight number vs launch site vs class
    - Payload vs launch site vs class
    - Payload vs orbit type vs class
    - Year vs success rate



```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate

year_mean = df[['Date', 'Class']].groupby(['Date']).mean()
plt.plot(year_mean.index, year_mean['Class'])
plt.xlabel('Year')
plt.ylabel('Sucess Rate')
plt.show()
```

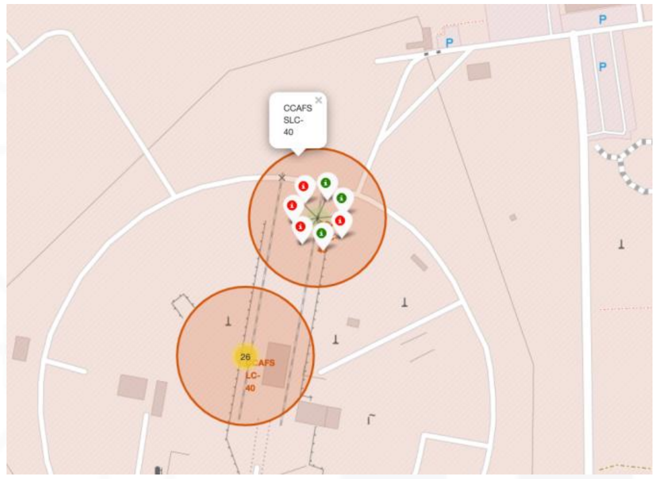


IBM Developer

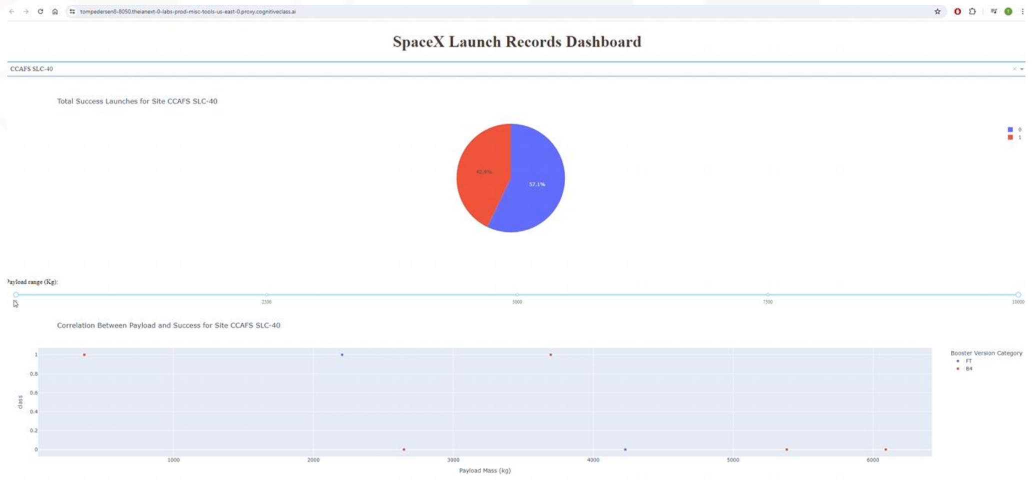
SKILLS NETWORK

Interactive Visual Analytics

- Folium was used to analyse geographical launch site data
  - Launch sites were marked
  - Successes and failures were displayed with marker clusters
  - Distances between launch sites and points of interest were calculated and displayed



- Plotly Dash was used to create an interactive dash board displaying visual insights including:
  - Pie chart showing proportion of success by launch site
  - Pie charts showing success rate for each launch site
  - Scatter charts showing payload mass vs booster version vs class for each launch site

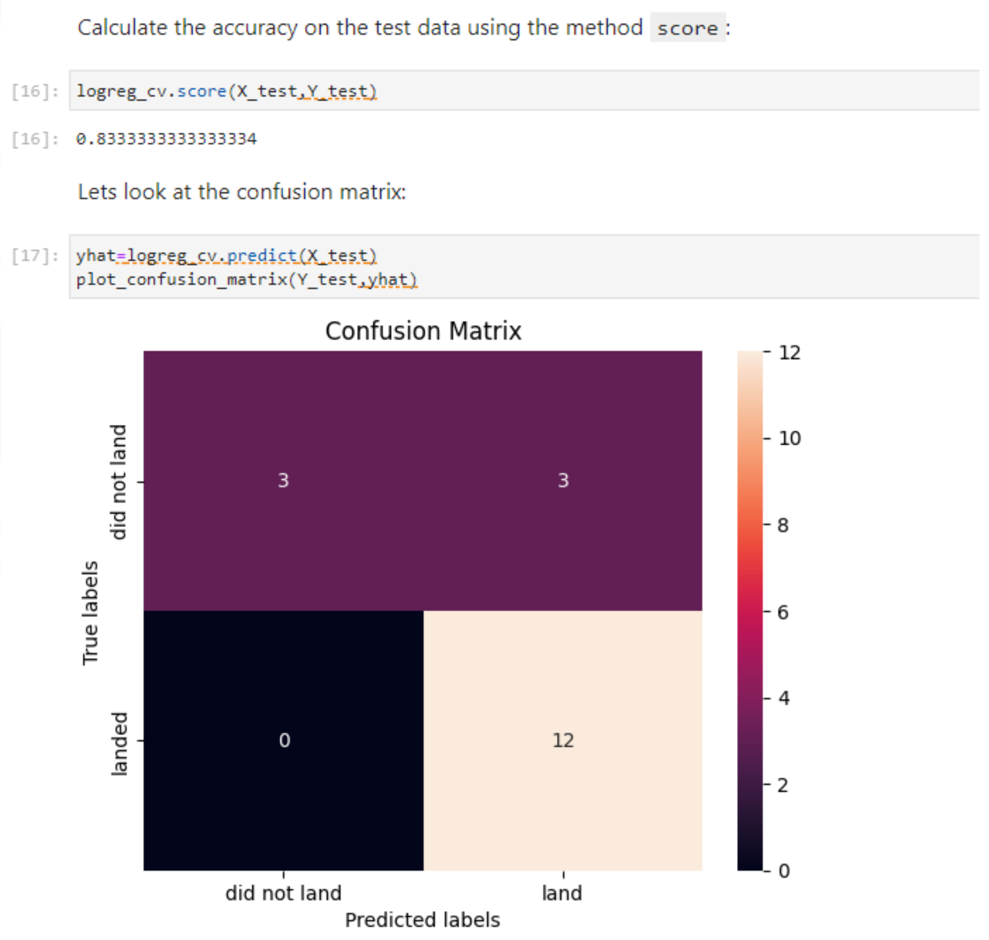


tompedersen8-8050.theianext-0-labs-prod-misc-tools-us-east-0.proxy.cognitiveclass.ai
SpaceX Launch Records Dashboard
CCAFS SLC-40
Total Success Launches for Site CCAFS SLC-40
0
1
42.9%
57.1%
Payload range (Kg):
2500
5000
7500
10000
Correlation Between Payload and Success for Site CCAFS SLC-40
class
Payload Mass (kg)
Booster Version Category
FT
B4

Predictive Analysis

- The data underwent preprocessing and was split into train and test sets

- Confusion matrices and accuracy scores were used to compare the following algorithms under their optimal parameters to determine the most accurate predictive machine learning model:
  - Logistic regression
  - Support vector machine
  - Decision tree classifier
  - k nearest neighbours



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

```
[16]: logreg_cv.score(X_test,Y_test)
[16]: 0.8333333333333334
```

Lets look at the confusion matrix:

```
[17]: yhat=logreg_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```

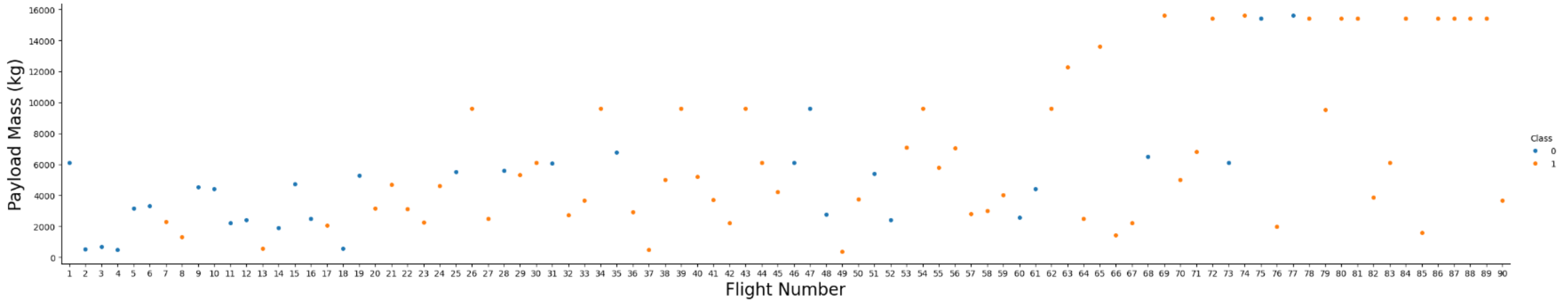
SECTION 2: RESULTS

1. Exploratory Data Analysis With Visualisation
2. Exploratory Data Analysis With SQL
3. Interactive Map With Folium
4. Plotly Dash Dashboard
5. Predictive Analysis (Classification)

EDA With Visualisation 1

Scatter chart: Flight Number vs Payload Mass vs Class

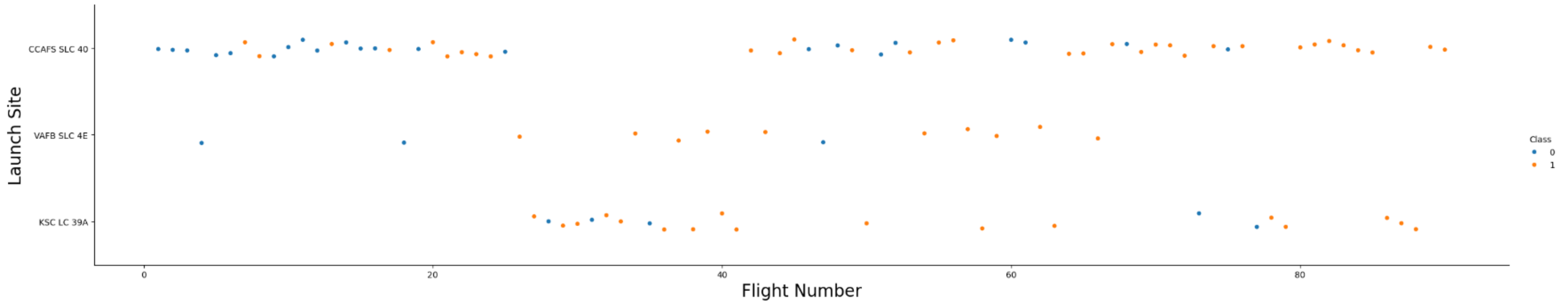
- As the flight number increases, the first stage is more likely to land successfully
- The larger the payload, the less likely the first stage will return



EDA With Visualisation 2

Scatter chart: Flight Number vs Launch Site vs Class

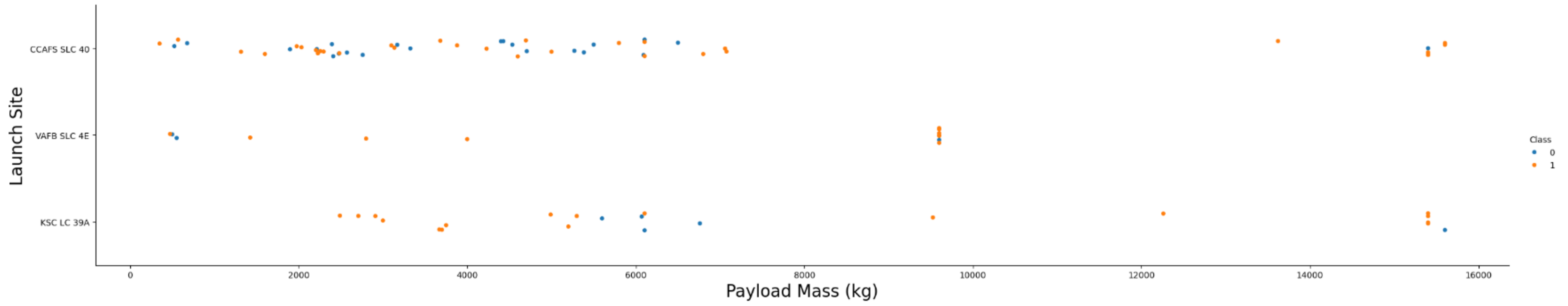
- Most early flights were launched from CCAFS SLC 40. This might explain the lower landing success rate at this site
- The most flights have been launched from CCAFS SLC 40, so this site has the largest sample size



EDA With Visualisation 3

Scatter chart: Payload Mass vs Launch Site vs Class

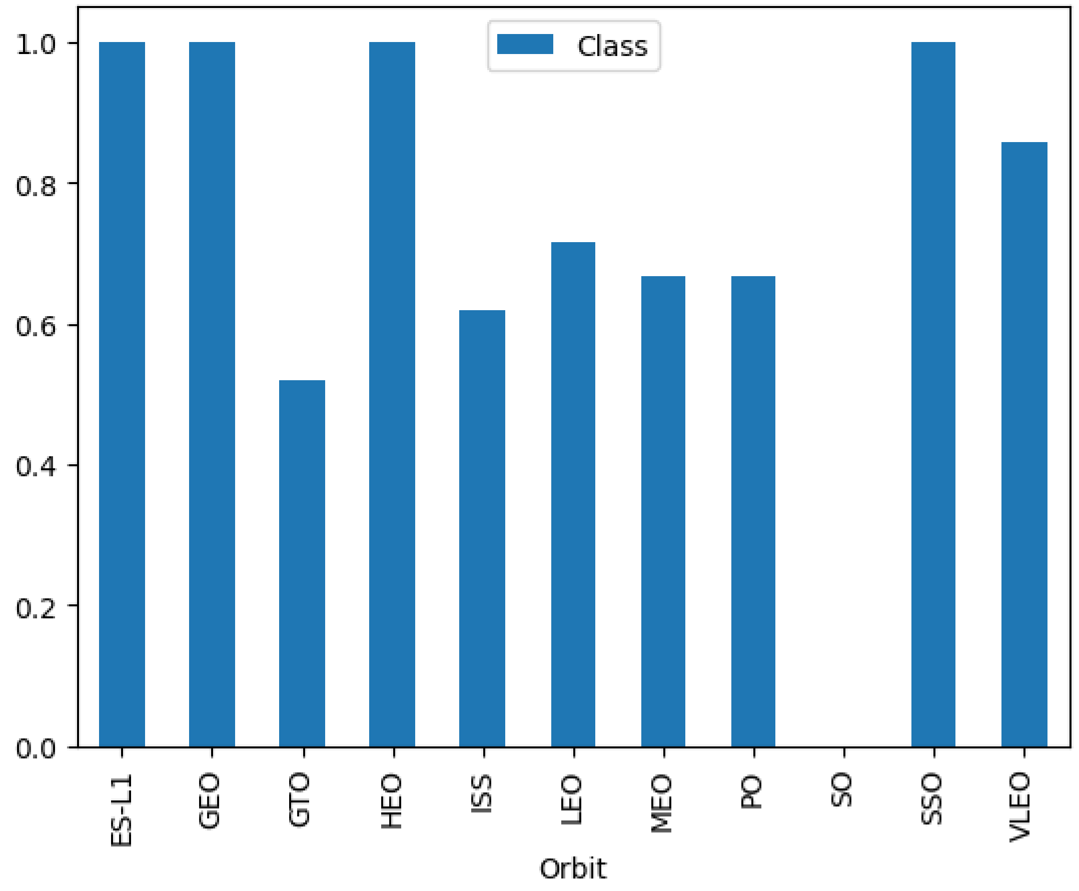
- No flight with a payload mass greater than 10000 kg has been launched from the site VAFB-SLC
- KSC LC 39A has a complete landing success rate for flights with a payload mass less than 5500 kg



EDA With Visualisation 4

Bar chart: Orbit Type vs Success Rate

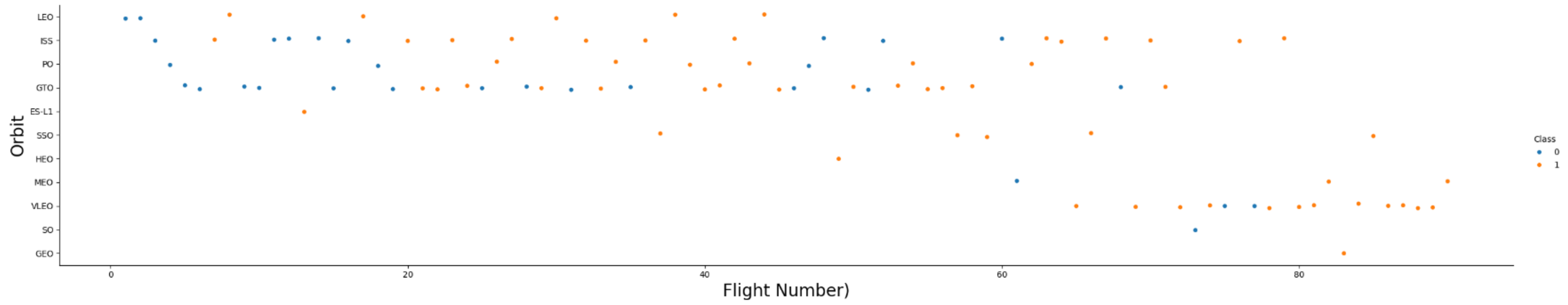
- The highest landing success rate (100%) has been observed for orbit types ES-L1, GEO, HEO and SSO
- The orbit type SO has the lowest landing success rate at 0%



EDA With Visualisation 5

Scatter chart: Flight Number vs Orbit Type vs Class

- Landing success seems to be related to the number of flights for the orbit type LEO
- There seems to be no relationship between number of flights and landing success when in GTO orbit



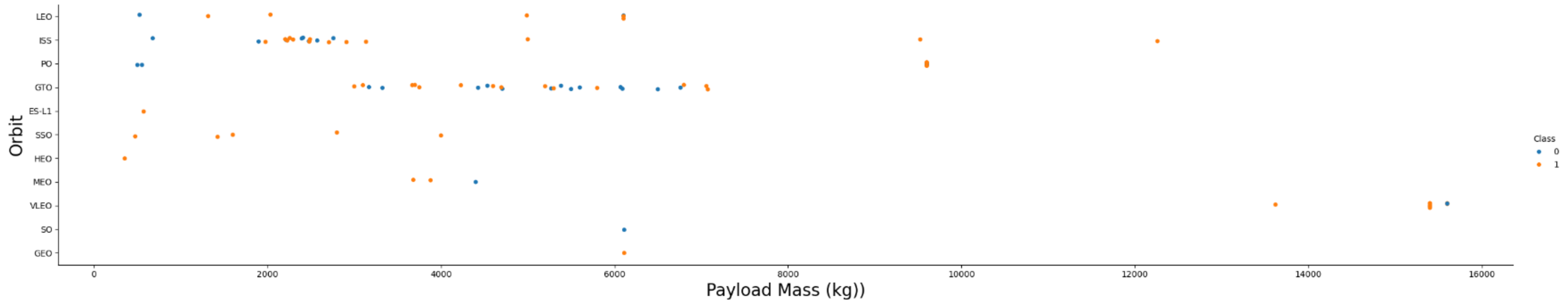
IBM Developer

SKILLS NETWORK

EDA With Visualisation 6

Scatter chart: Payload Mass vs Orbit Type vs Class

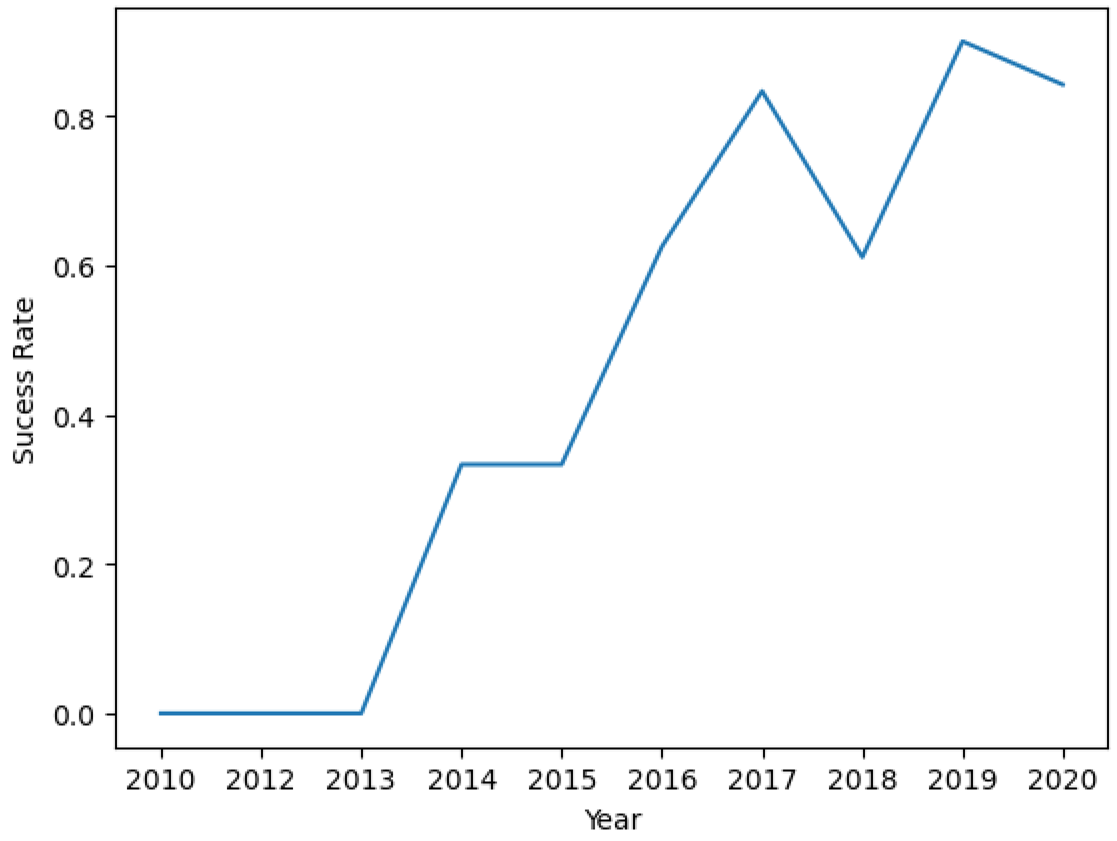
- Flights with heavier payloads have more success with landing when the orbit type is Polar, LEO and ISS
- For the GTO orbit type there does not seem to be a strong correlation between landing success and payload mass



EDA With Visualisation 7

Line graph: Year vs Landing Success Rate

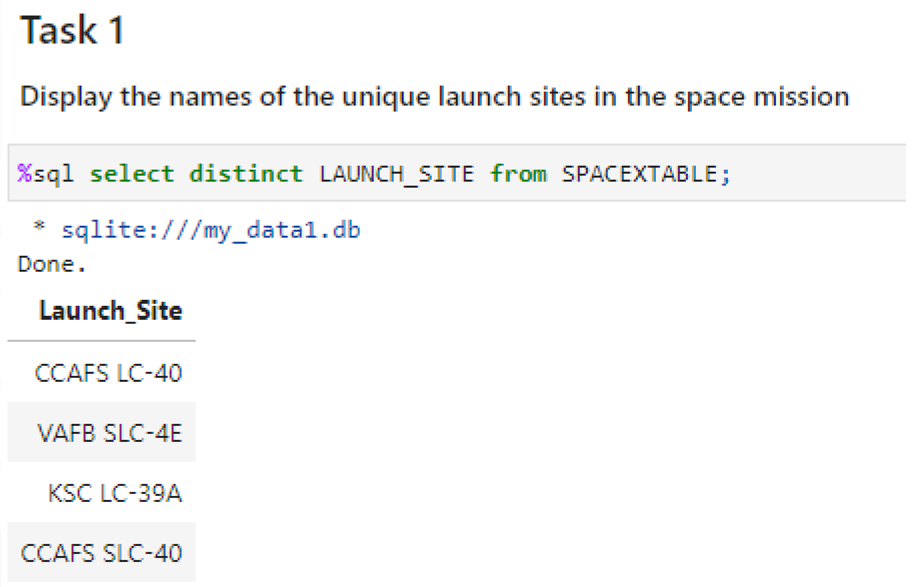
- There were no landing successes before 2014
- Landing success rate increased steadily between 2013 and 2017
- Landing success rate dipped in 2018 and 2020



EDA With SQL 1

Finding the names of all launch sites:

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



Task 1

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

```
%sql select distinct LAUNCH_SITE from SPACEXTABLE;
```

```
 * sqlite:///my_data1.db
Done.
```

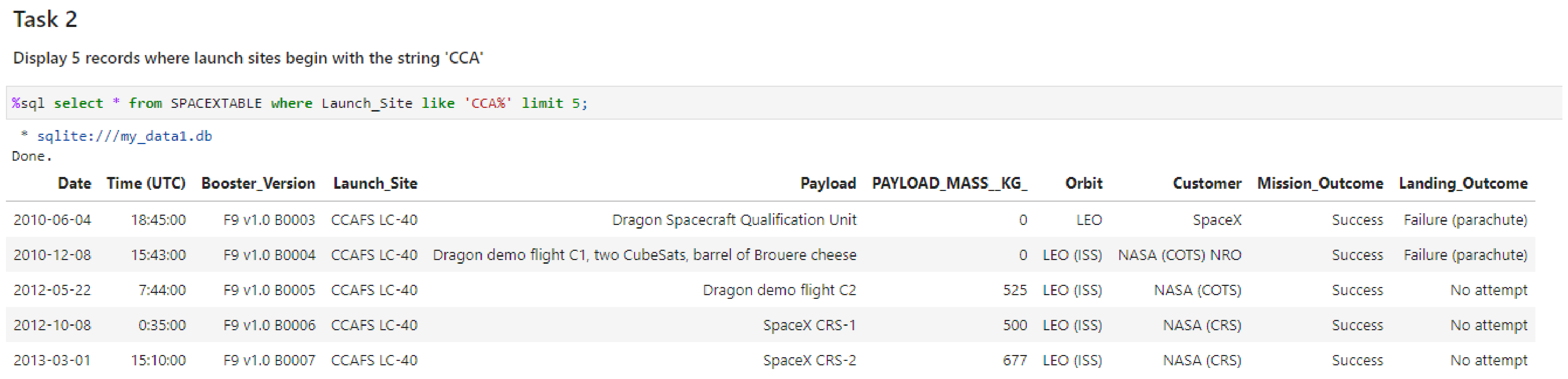
Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

IBM Developer

SKILLS NETWORK

EDA With SQL 2

Displaying the first five records where launch sites begin with 'CCA':



Task 2

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

```
%sql select * from SPACEXTABLE where Launch_Site like 'CCA%' limit 5;
```

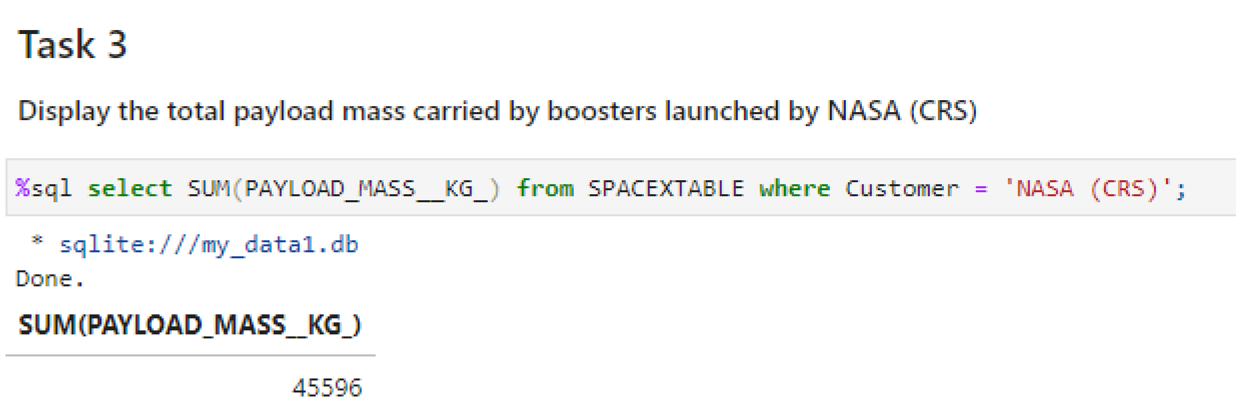
```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

EDA With SQL 3

Finding the total sum of the payload mass carried by boosters launch by NASA (CRS):

45596 kg



Task 3

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

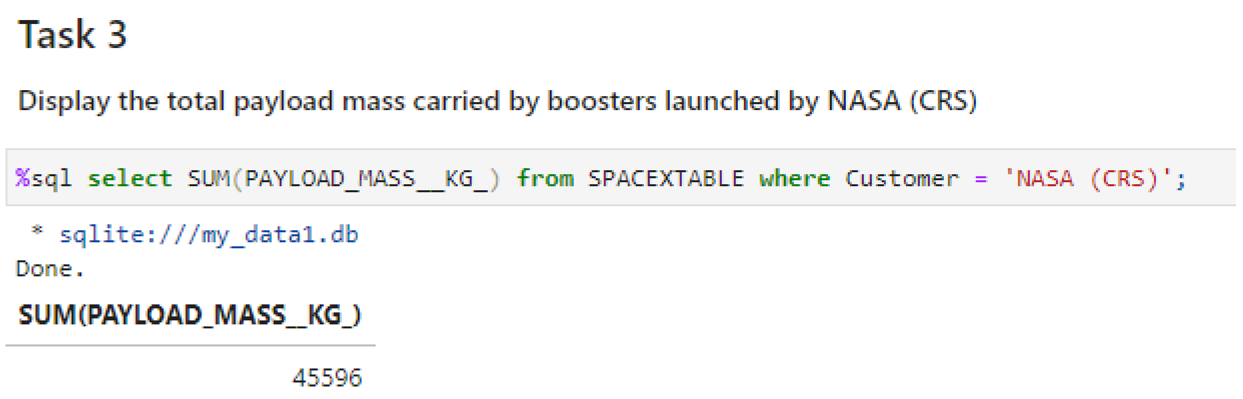
```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = 'NASA (CRS)';
```

```
 * sqlite:///my_data1.db
Done.
```

SUM(PAYLOAD_MASS__KG_)
45596

EDA With SQL 4

Finding the total sum of the payload mass carried by boosters launched by NASA (CRS): 45596 kg



Task 3

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

```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = 'NASA (CRS)';
```

```
 * sqlite:///my_data1.db
Done.
```

SUM(PAYLOAD_MASS__KG_)
45596

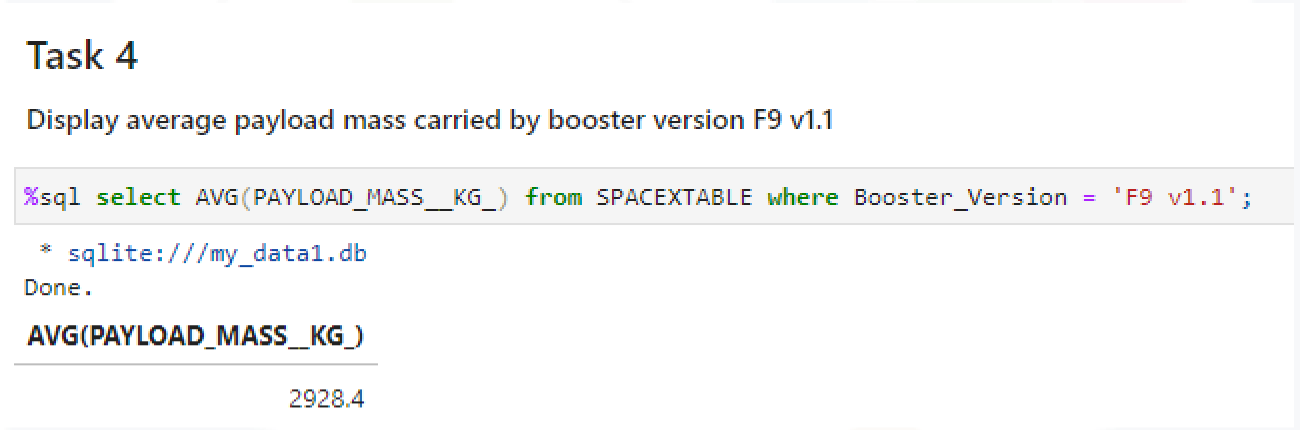
IBM Developer

SKILLS NETWORK

EDA With SQL 5

Finding the average payload mass carried by boosters version F9 v1.1:

2928.4 kg



Task 4

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

```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = 'F9 v1.1';
```

```
 * sqlite:///my_data1.db
Done.
```

AVG(PAYLOAD_MASS__KG_)
2928.4

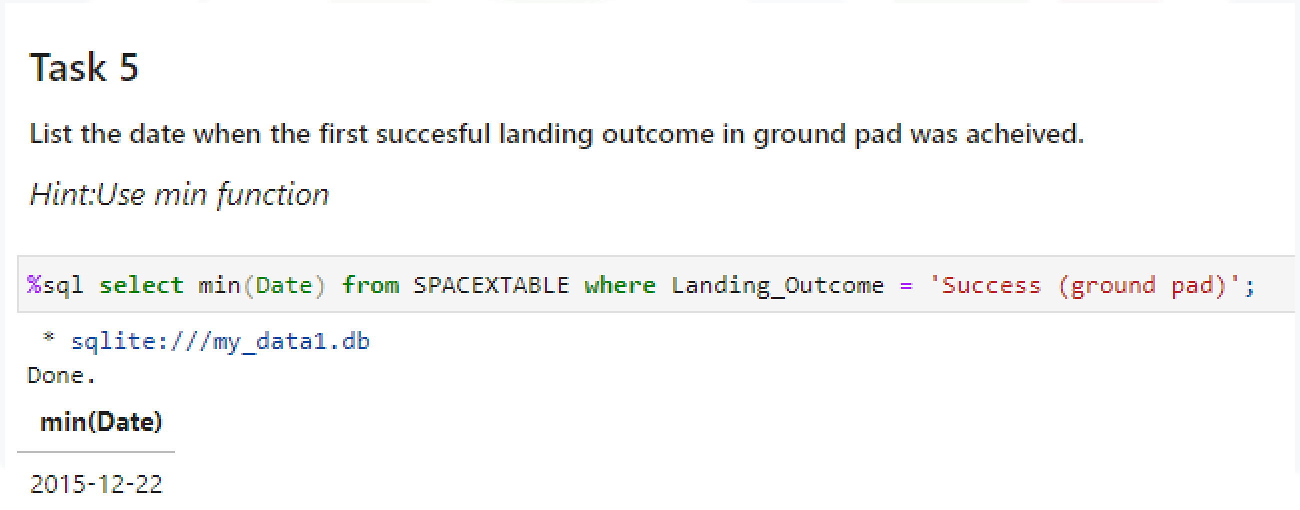
IBM Developer

SKILLS NETWORK

EDA With SQL 6

Finding the date when the first successful landing outcome on ground pad occurred:

22nd December 2015



Task 5

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

Hint:Use min function

```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)';
```

```
 * sqlite:///my_data1.db
Done.
```

min(Date)
2015-12-22

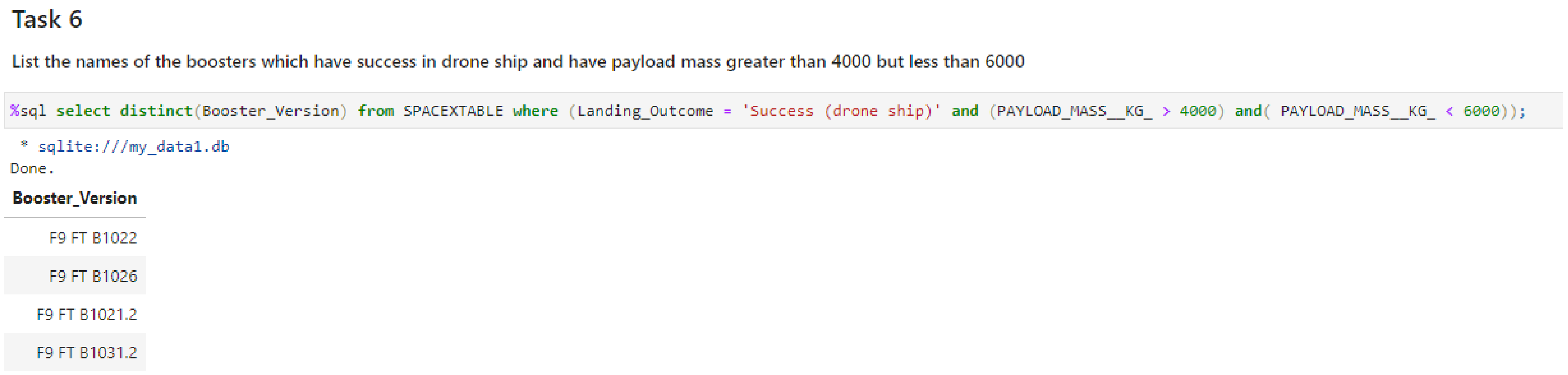
IBM Developer

SKILLS NETWORK

EDA With SQL 7

Finding the unique boosters which have achieved landing success on a drone ship while the payload mass is between 4000 kg and 6000 kg:

F9 FT B1022, F9 FT B1026, F9 FT B1021.2, F9 FT B1031.2



Task 6

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

```
%sql select distinct(Booster_Version) from SPACEXTABLE where (Landing_Outcome = 'Success (drone ship)' and (PAYLOAD_MASS__KG_ > 4000) and( PAYLOAD_MASS__KG_ < 6000));
```

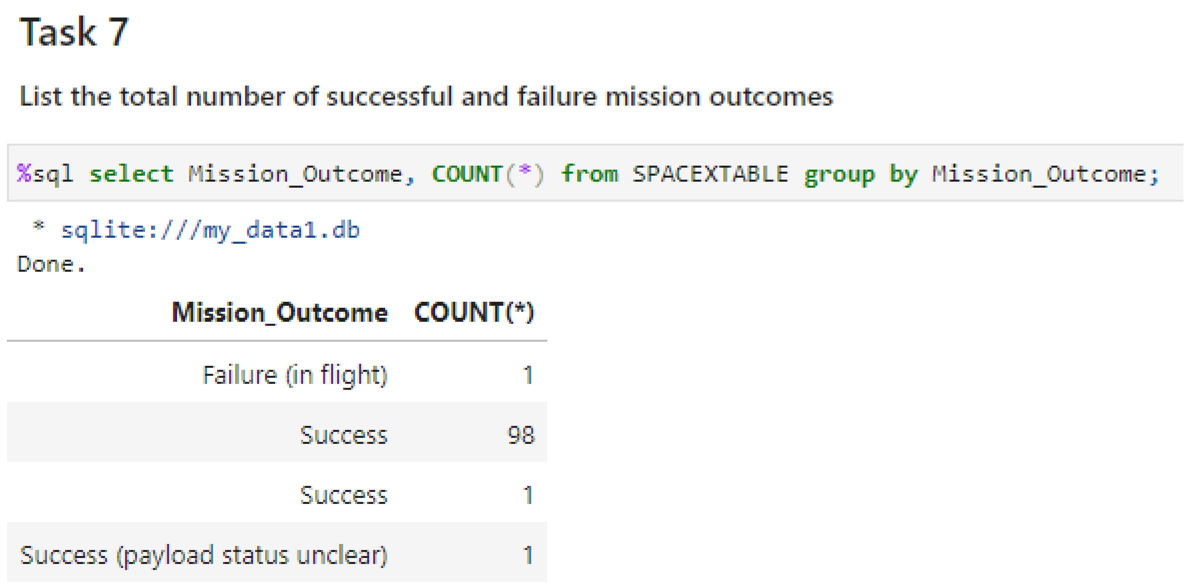
```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

EDA With SQL 8

Finding the total number of successful and failed mission outcomes:

- 99 Success
- 1 Success with payload status unclear
- 1 Failure in flight



Task 7

List the total number of successful and failure mission outcomes

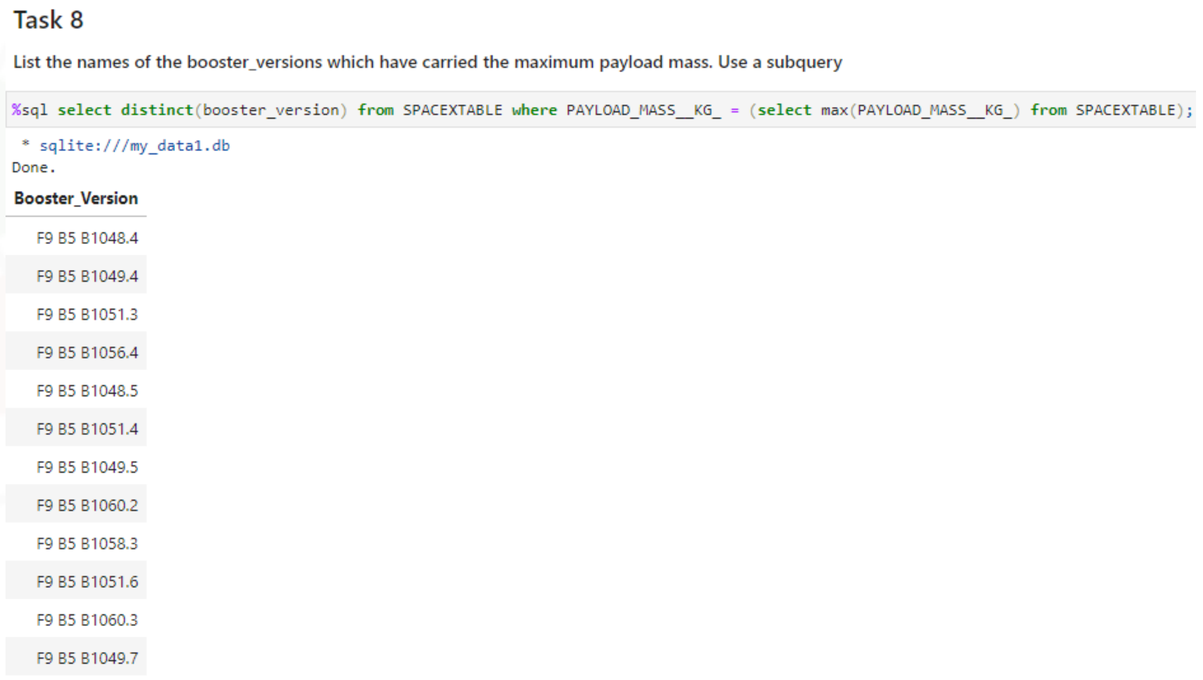
```
%sql select Mission_Outcome, COUNT(*) from SPACEXTABLE group by Mission_Outcome;
```

```
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	COUNT(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

EDA With SQL 9

Finding the names of the booster versions which have carried the maximum payload mass



Task 8

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

```
%sql select distinct(booster_version) from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE);
```

```
 * sqlite:///my_data1.db
Done.
```

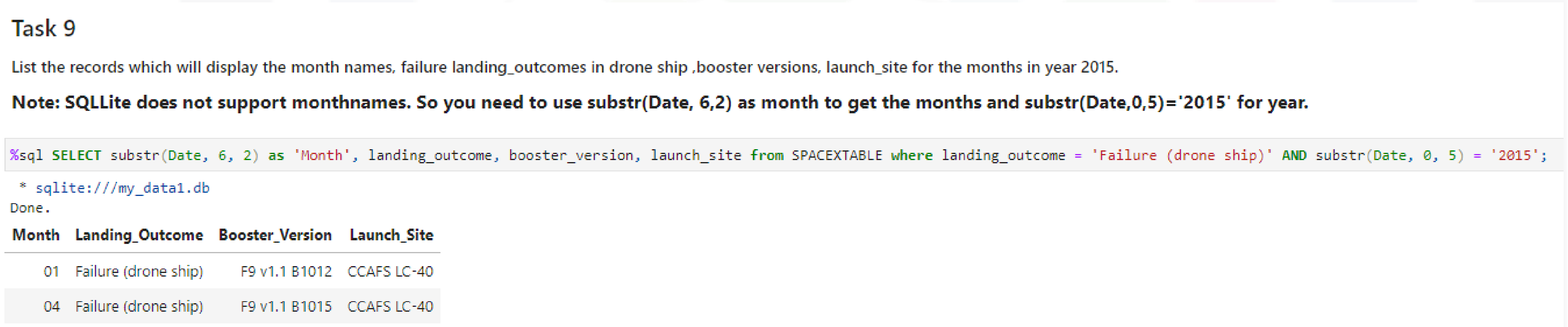
Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

IBM Developer

SKILLS NETWORK

EDA With SQL 10

Finding the months, booster versions and launch sites for failures by drone ship in the year 2015:



Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql SELECT substr(Date, 6, 2) as 'Month', landing_outcome, booster_version, launch_site from SPACEXTABLE where landing_outcome = 'Failure (drone ship)' AND substr(Date, 0, 5) = '2015';
```

```
 * sqlite:///my_data1.db
Done.
```

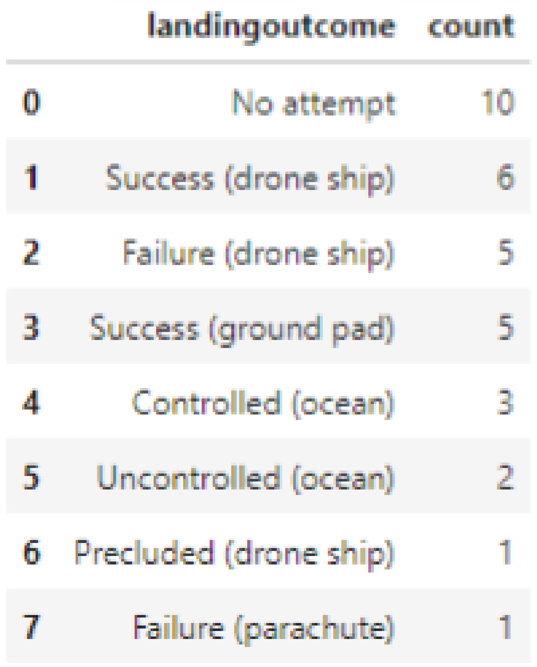
Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

IBM Developer

SKILLS NETWORK

EDA With SQL 10

Ranking the total count of each landing outcome between 04/06/2010 and 20/03/2017:

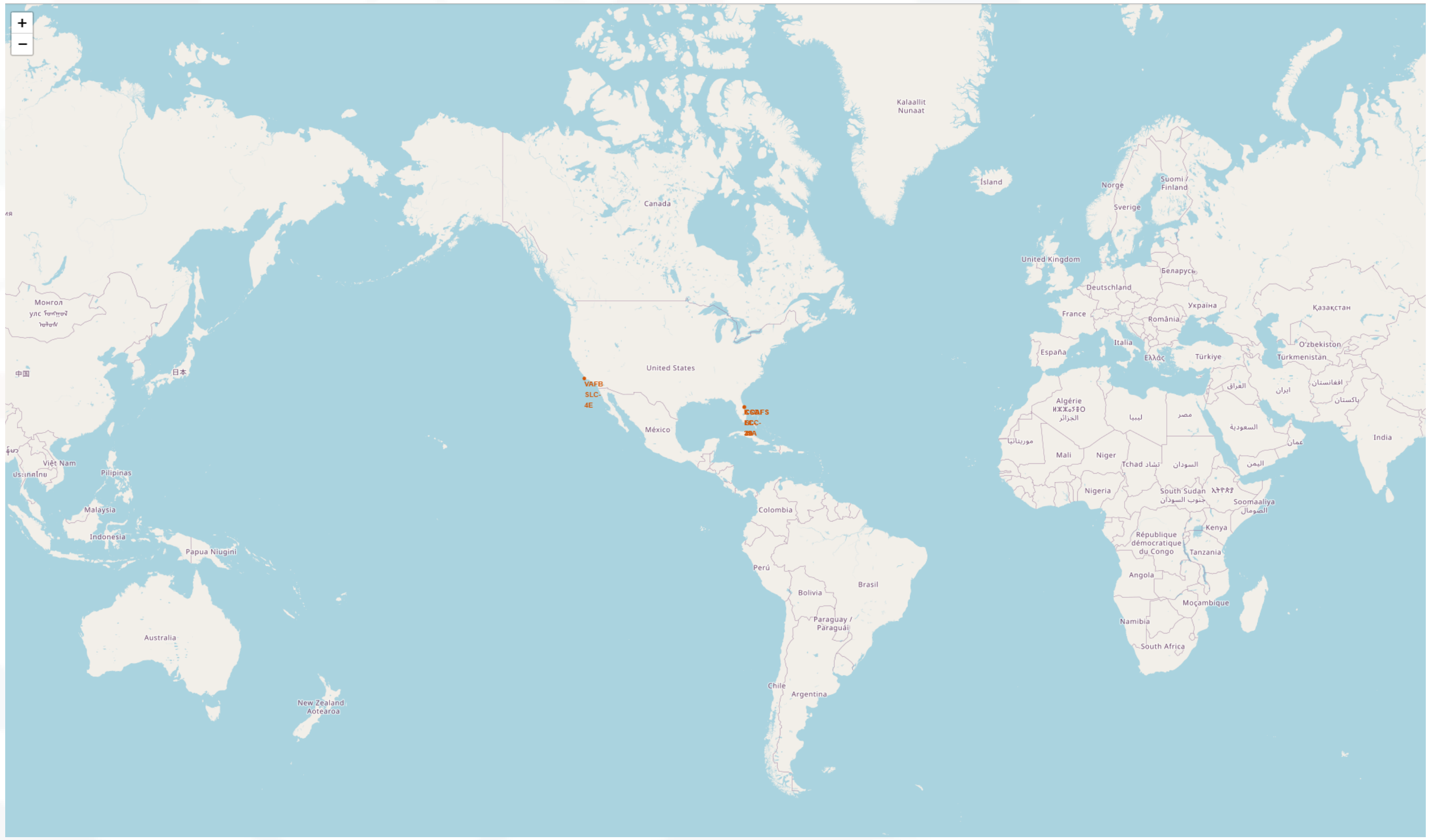


	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Interactive Map With Folium 1

Adding each launch site to the world map, we can see:

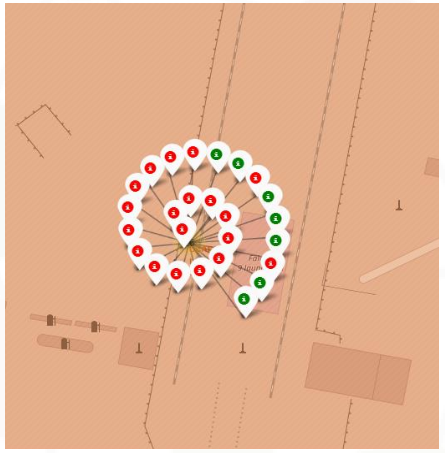
- All launch sites are in the USA
- All launch sites are close to the coast
- All launch sites are relatively close to the equator

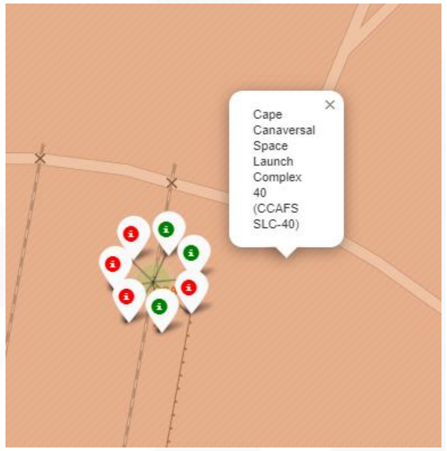


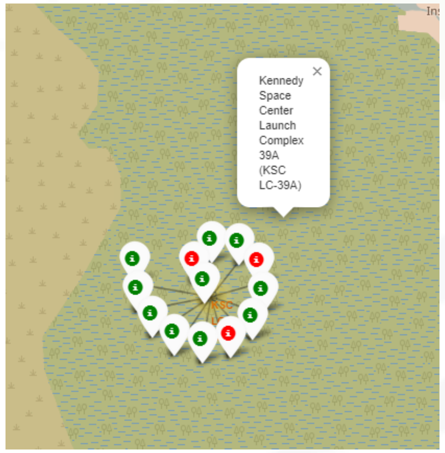
Interactive Map With Folium 2

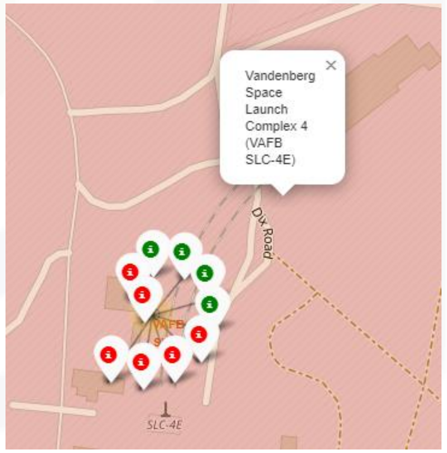
Marking the successful and failed landings from flights launched at each site, we can see :

- The most launches happened at CCAFS LC-40
- The fewest launches happened at CCAFS SLC-40
- KSC LC-39A has the highest landing success rate





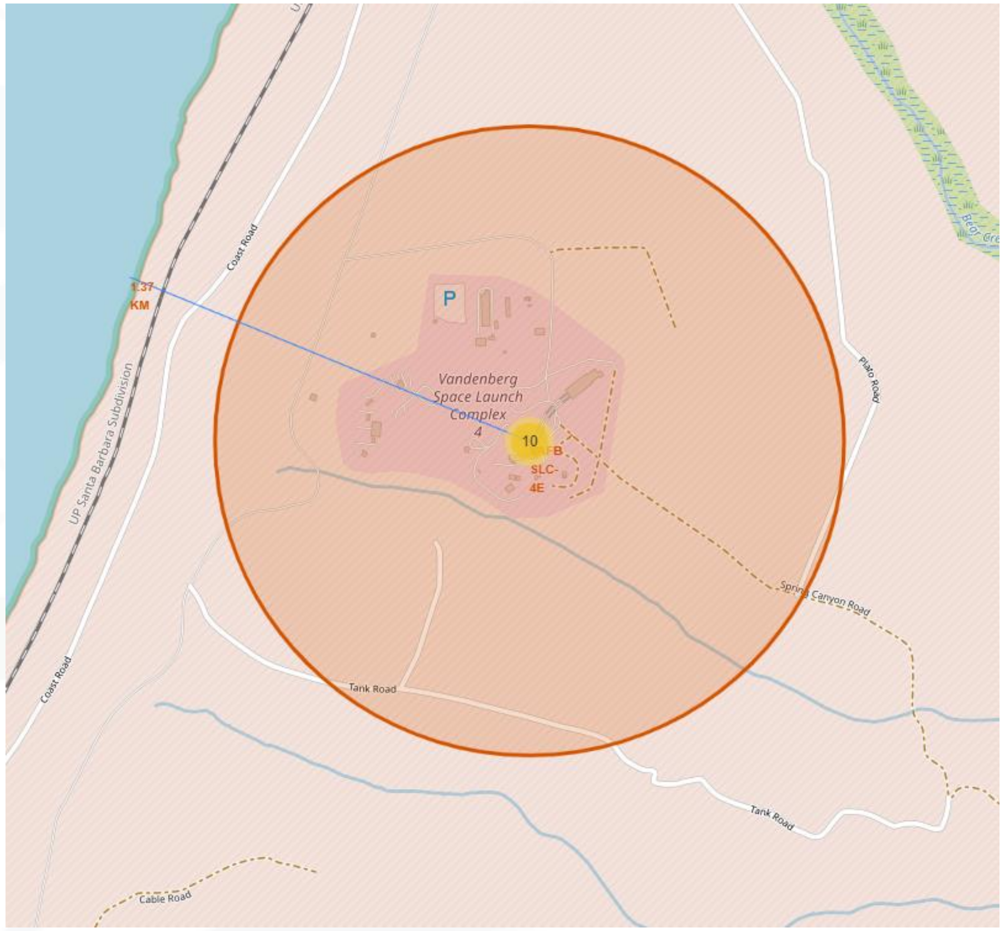




Interactive Map With Folium 3

Calculating distances between the launch sites and their closest points of interest, we see that:

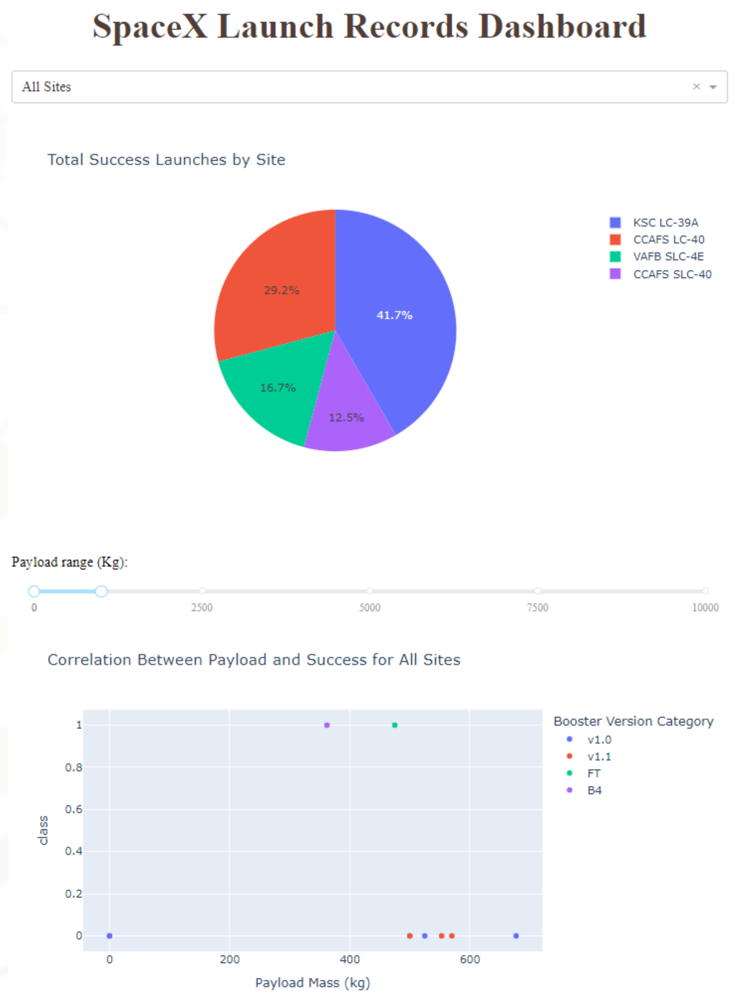
- The launch sites are close to railways
- The launch sites are far away from major highways
- The launch sites are very near the coast
- The launch sites are all far away from large cities



Plotly Dash Dashboard 1

The Plotly Dash dashboard includes:

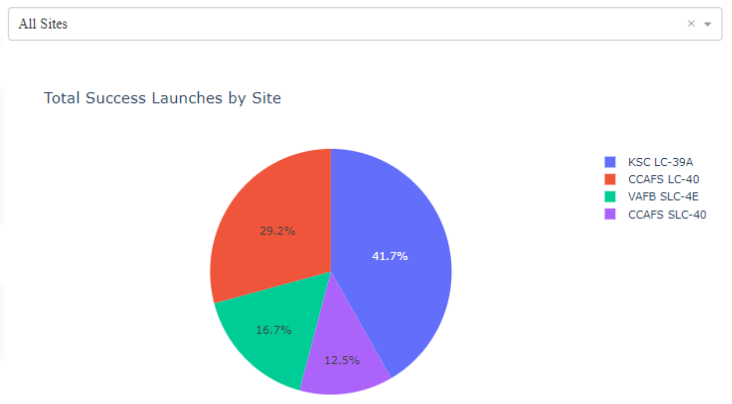
- A dropdown with each of the launch sites and the option to select all sites
- A range slider to select a range for the payload mass
- A pie chart
- A scatter chart

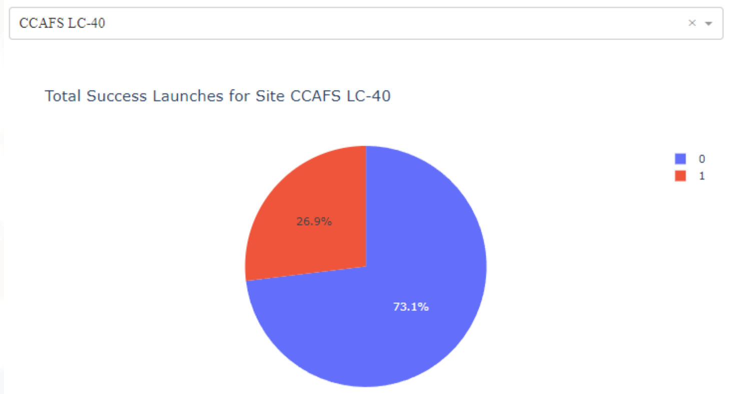


Plotly Dash Dashboard 2

- When 'All Sites' is selected, the pie chart shows the proportion of total landing successes seen by each site

- When a single launch site is selected, the pie chart shows the proportion of landing successes and failures seen by each site



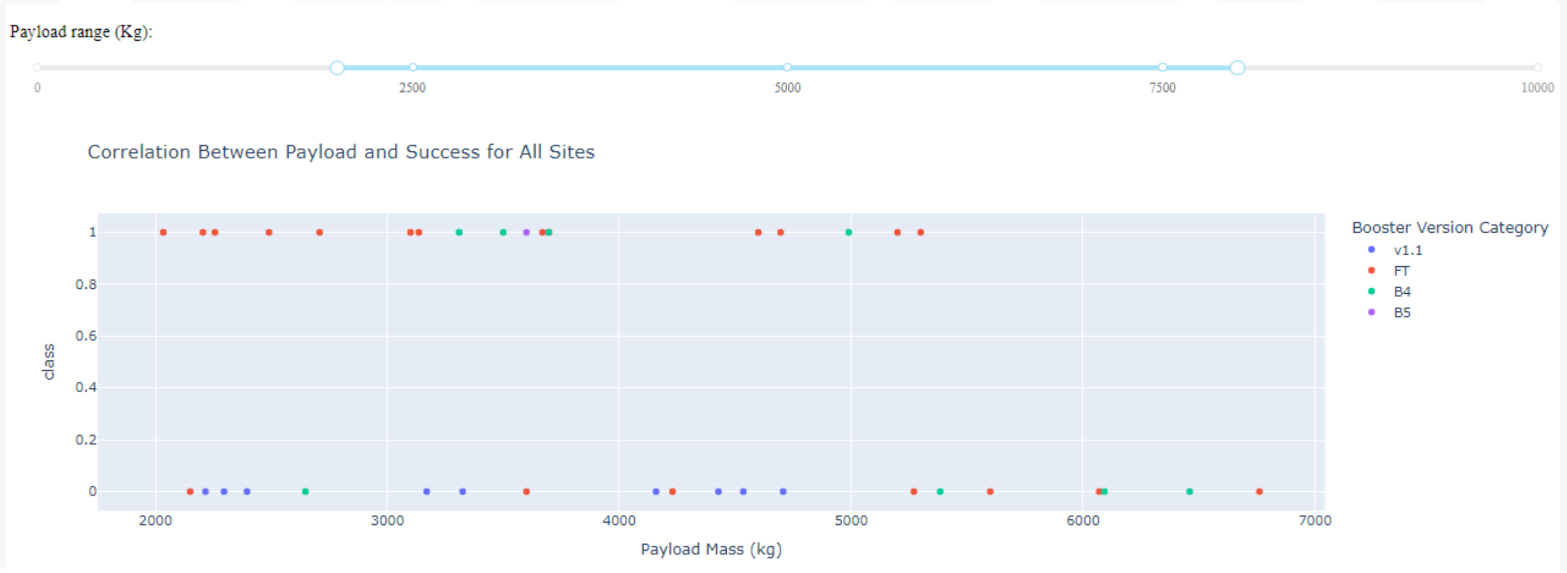


IBM Developer

SKILLS NETWORK

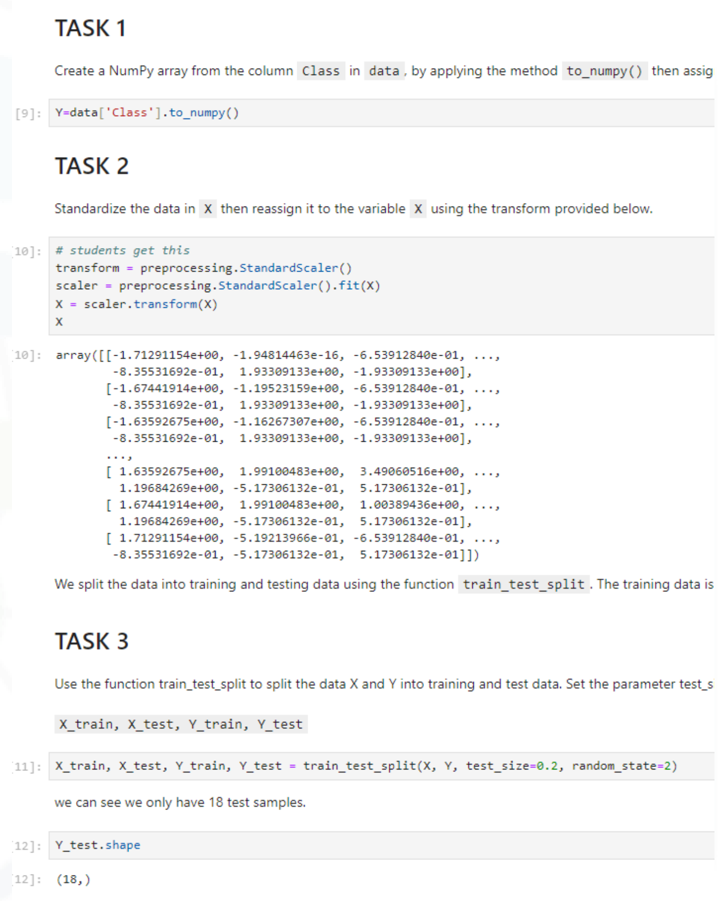
Plotly Dash Dashboard 3

- The scatter chart shows the landing successes for different payload masses and booster versions
- The scatter chart displays different information depending on your choice of landing site and payload range



Predictive Analysis 1

- Using SKLearn, we performed preprocessing and split the data into train and test sets
- There were 18 observations in the test set



TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assig

```
[9]: Y=data['Class'].to_numpy()
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

```
10]: # students get this
     transform = preprocessing.StandardScaler()
     scaler = preprocessing.StandardScaler().fit(X)
     X = scaler.transform(X)
     X

10]: array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01, ...,
             -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
            [-1.67441914e+00, -1.19523159e+00, -6.53912840e-01, ...,
             -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
            [-1.63592675e+00, -1.16267307e+00, -6.53912840e-01, ...,
             -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
            ...,
            [ 1.63592675e+00,  1.99100483e+00,  3.49060516e+00, ...,
              1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
            [ 1.67441914e+00,  1.99100483e+00,  1.00389436e+00, ...,
              1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
            [ 1.71291154e+00, -5.19213966e-01, -6.53912840e-01, ...,
             -8.35531692e-01, -5.17306132e-01,  5.17306132e-01]])
```

We split the data into training and testing data using the function `train_test_split`. The training data is

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_s

`X_train, X_test, Y_train, Y_test`

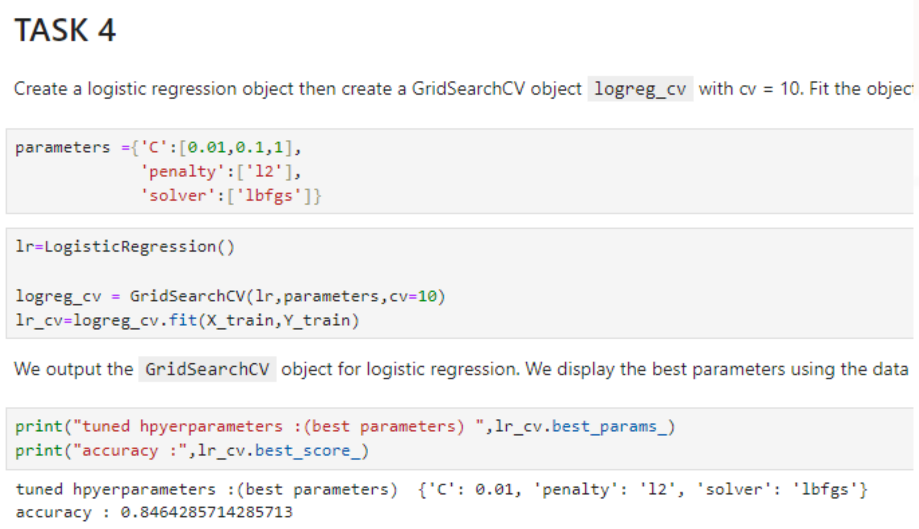
```
11]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

```
12]: Y_test.shape

12]: (18,)
```

Predictive Analysis 2

- We found the accuracy with the best hyperparameters for a logistic regression model, and found the confusion matrix
- The model correctly identifies all landing successes, but has a problem with false positives



TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the objec

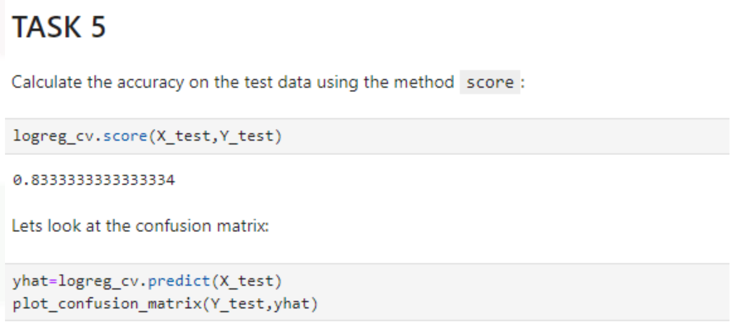
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
lr=LogisticRegression()

logreg_cv = GridSearchCV(lr,parameters,cv=10)
lr_cv=logreg_cv.fit(X_train,Y_train)
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data

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

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



TASK 5

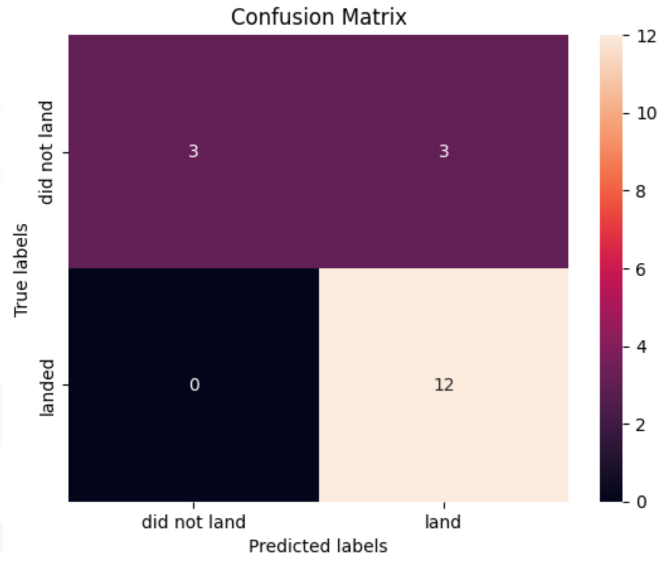
Calculate the accuracy on the test data using the method `score` :

```
logreg_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

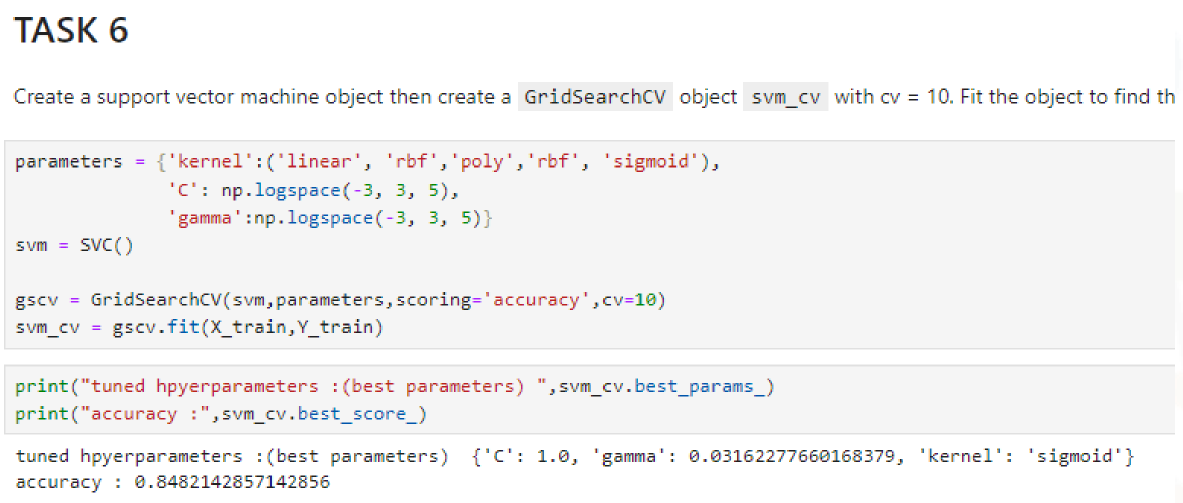
Lets look at the confusion matrix:

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



Predictive Analysis 3

- We had similar results when running a support vector machine model, but with different optimal hyperparameters



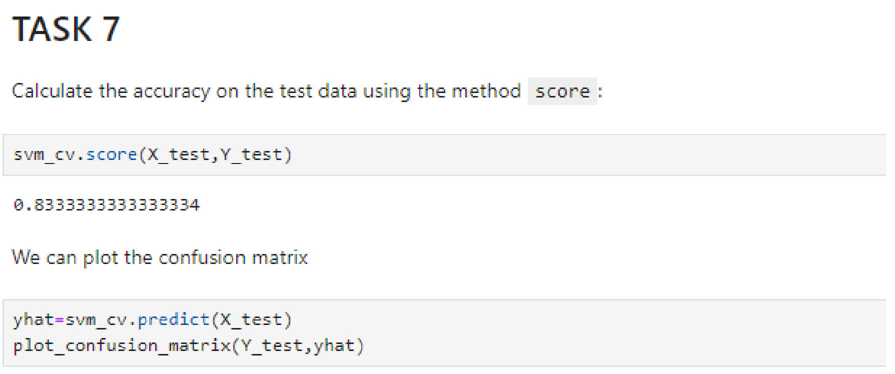
TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv = 10. Fit the object to find th

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

gscv = GridSearchCV(svm,parameters,scoring='accuracy',cv=10)
svm_cv = gscv.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
```



TASK 7

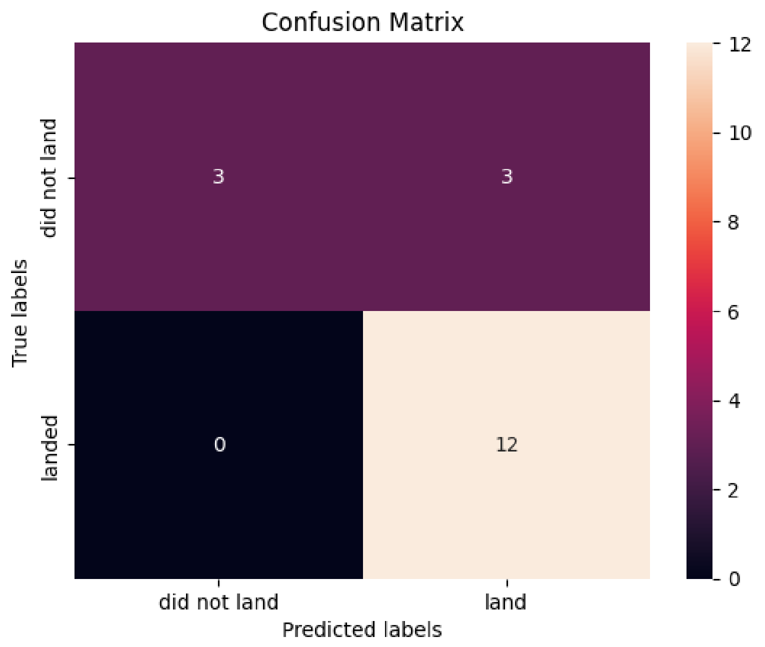
Calculate the accuracy on the test data using the method `score` :

```
svm_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

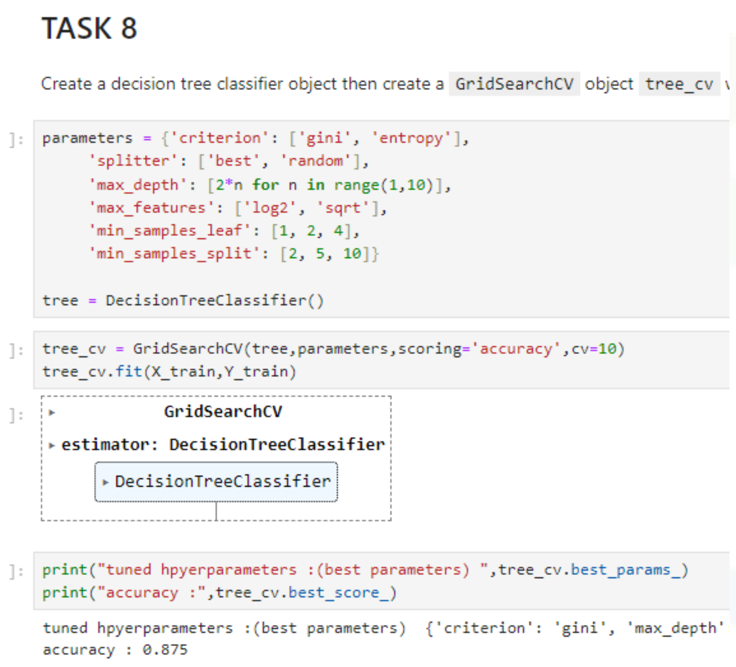
We can plot the confusion matrix

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



Predictive Analysis 4

- We had similar results when running a decision tree classifier model, but with different optimal hyperparameters



TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv`

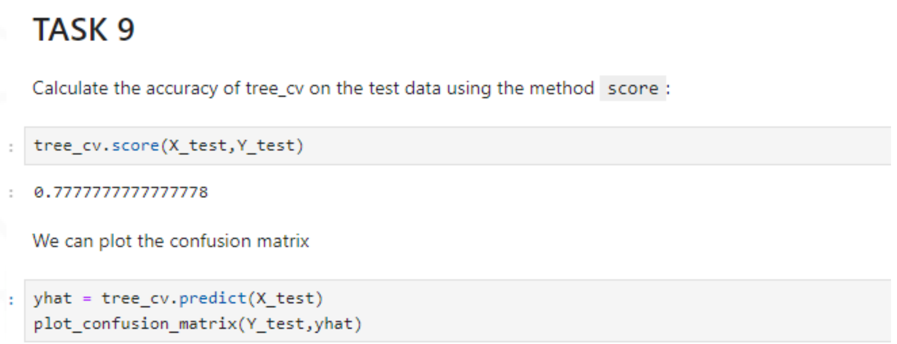
```
]: parameters = {'criterion': ['gini', 'entropy'],
        'splitter': ['best', 'random'],
        'max_depth': [2*n for n in range(1,10)],
        'max_features': ['log2', 'sqrt'],
        'min_samples_leaf': [1, 2, 4],
        'min_samples_split': [2, 5, 10]}

   tree = DecisionTreeClassifier()
```

```
]: tree_cv = GridSearchCV(tree,parameters,scoring='accuracy',cv=10)
   tree_cv.fit(X_train,Y_train)
```

```
]: GridSearchCV
   estimator: DecisionTreeClassifier
       DecisionTreeClassifier
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth'
accuracy : 0.875
```



TASK 9

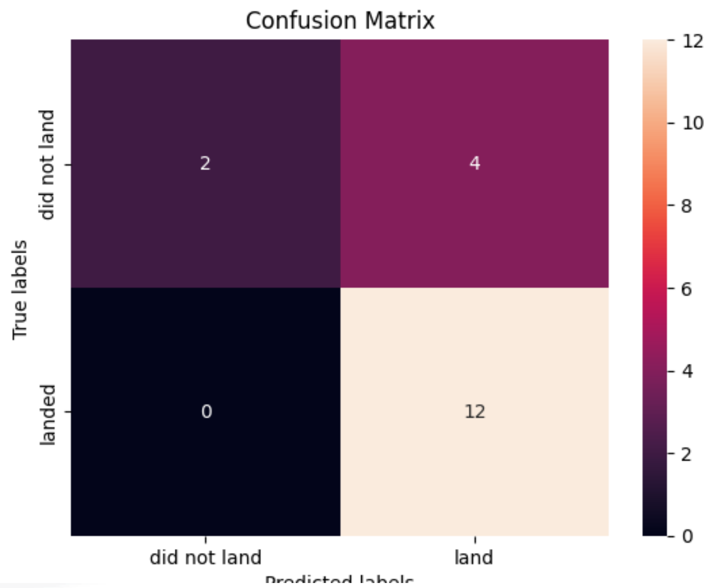
Calculate the accuracy of tree_cv on the test data using the method `score` :

```
: tree_cv.score(X_test,Y_test)
```

```
: 0.7777777777777778
```

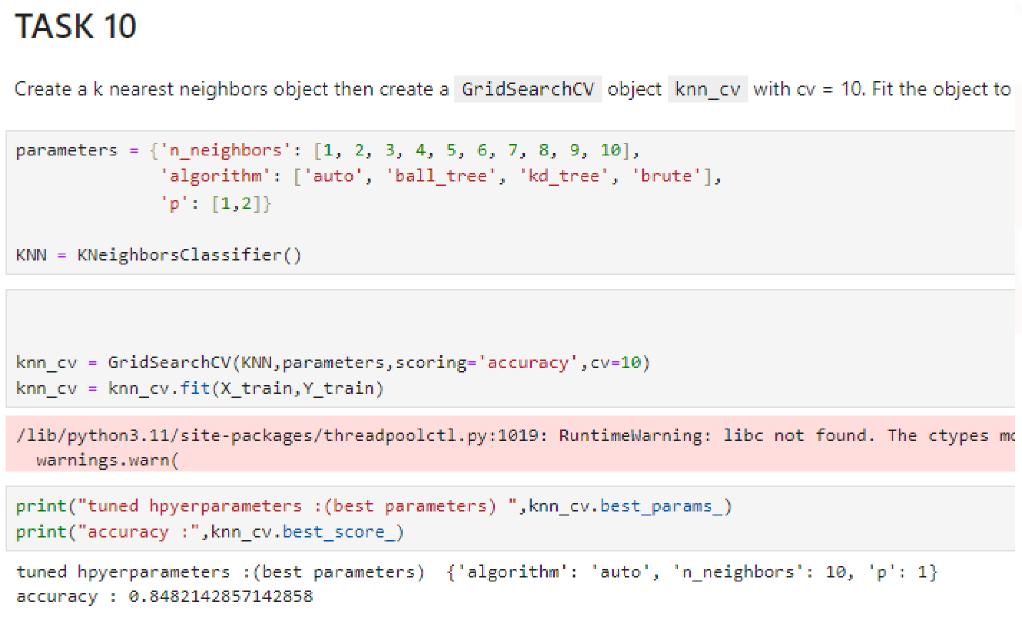
We can plot the confusion matrix

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



Predictive Analysis 5

- We had similar results when running a k nearest neighbours model, but with different optimal hyperparameters



TASK 10

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to

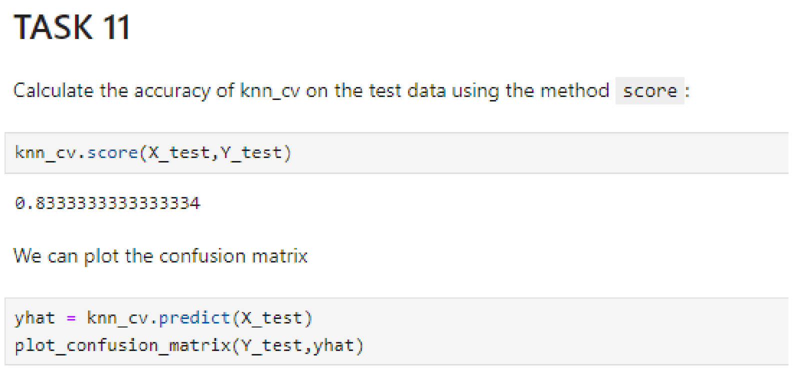
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

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

```
/lib/python3.11/site-packages/threadpoolctl.py:1019: RuntimeWarning: libc not found. The ctypes mo
  warnings.warn(
```

```
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
```



TASK 11

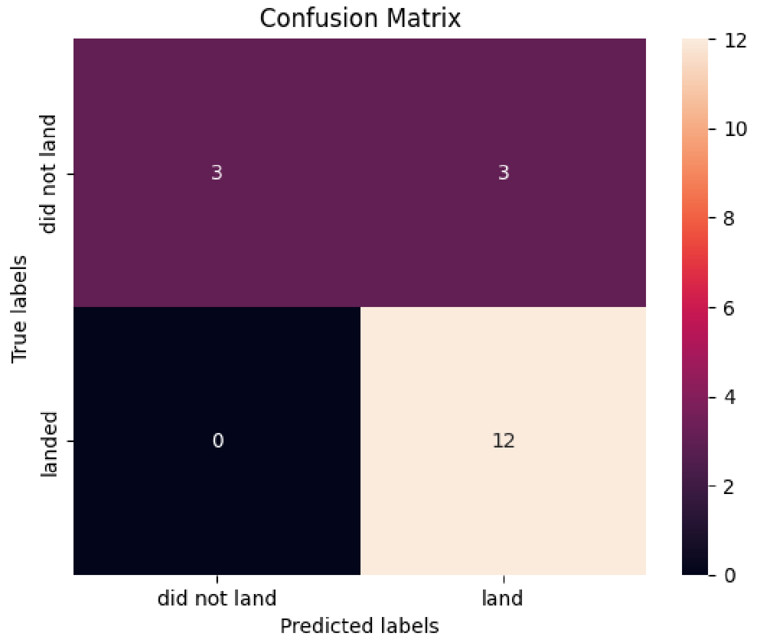
Calculate the accuracy of knn_cv on the test data using the method `score` :

```
knn_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

We can plot the confusion matrix

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



Predictive Analysis 6

Comparisons

- After comparing the accuracy of each model, we can see that the decision tree classifier model is the most accurate
- After comparing the confusion matrices of each model, we can see that all of the models performed well at identifying successful landings, but struggled with false negatives

SECTION 3: CONCLUSION

- The Stage 1 landing success rate has increased drastically over time and with experience
- The highest landing success rate (100%) has been observed for orbit types ES-L1, GEO, HEO and SSO
- KSC LC 39A has a complete landing success rate for flights with a payload mass less than 5500 kg
- There is a strong correlation between landing success rate and orbit type for larger payload masses
- The decision tree classifier model is the most accurate predictor of landing success rate