IBM Capstone Basic

Julio Diaz Delgado

22/07/22

This is not the final version but i could not do better in the shorter spawn that i got before my subscription ends i hope you enjoy the reading

OUTLINE

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

EXECUTIVE SUMMARY

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data

- Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction

- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

INTRODUCTION

- Project background and context Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully
- Problems you want to find answers - What factors determine if the rocket will land successfully? - The interaction amongst various features that determine the success rate of a successful landing. - What operating conditions needs to be in place to ensure a successful landing program.

METHODOLOGY

Executive Summary

- Data collection methodology:
- Data was collected using SpaceX API and web scraping from Wikipedia
- Perform data wrangling
- One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
- How to build, tune, evaluate classification models

Data Collection

- • The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

- This is the permalink to the data.
- https://github.com/kylekk76/IBM-Data-Science/blob/9639f0a8049ac9ff53977186bdfc87104d63609c/capstone%20in%20progress/spacex-data-collection.ipynb

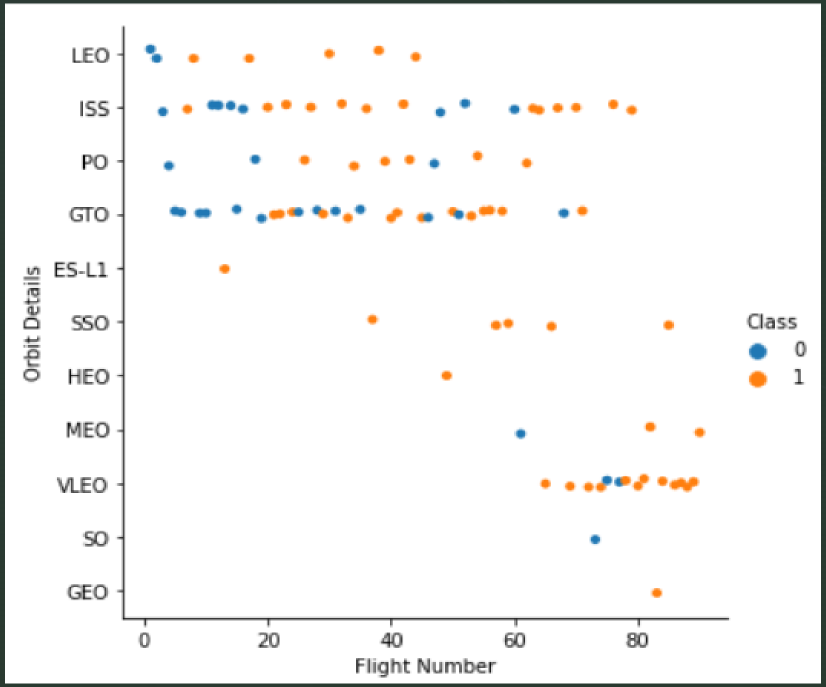
Data Wrangling

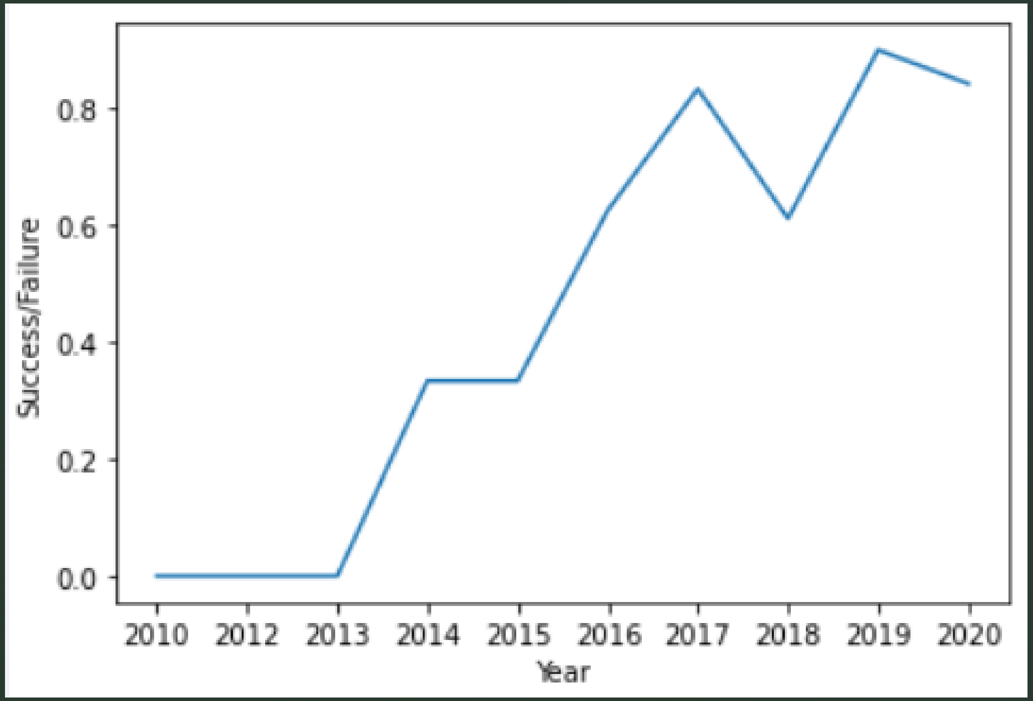
- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site, and the number and occurrence of each orbits
- Created landing outcome label from outcome column and exported the results to csv.

- This is the link to the Notebook and rest of the work:
- https://github.com/kylekk76/IBM-Data-Science/blob/9639f0a8049ac9ff53977186bdfc87104d63609c/capstone%20in%20progress/Spacex-Data%20wrangling.ipynb

EDA- With Data Visualization

Explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.

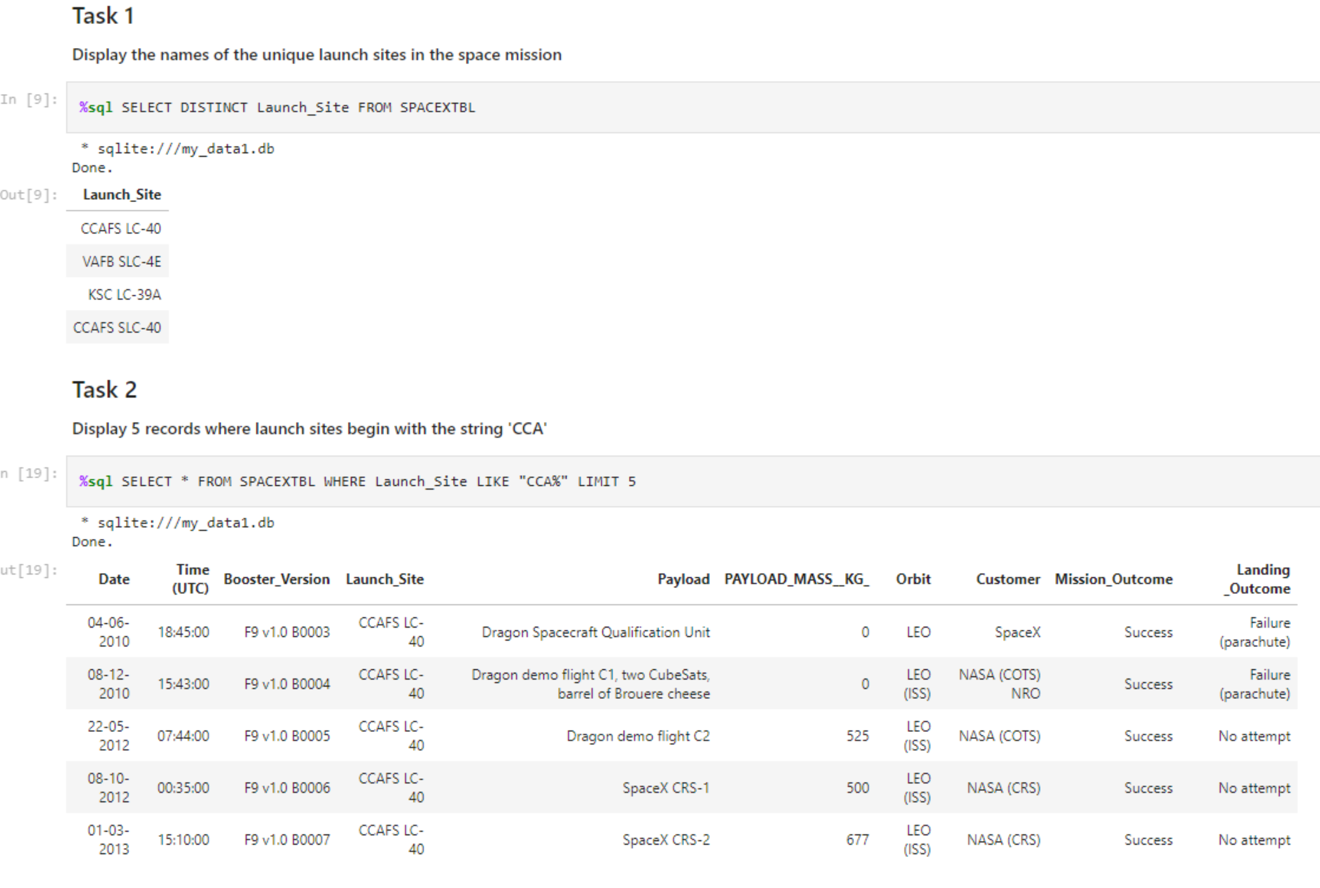




The Link To the Repository is:

https://github.com/kylekk76/IBM-Data-Science/blob/9639f0a8049ac9ff53977186bdfc87104d63609c/capstone%20in%20progress/Exploratory_Data_Base_Visualization.ipynb

EDA With SQL



Task 1

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

```
In [9]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTBL
```

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

Out[9]:

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

Task 2

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

```
n [19]: %sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE "CCA%" LIMIT 5
```

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

ut[19]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Loaded sample SpaceX dataset into a PostgreSQL database.
- Applied EDA with SQL to get insight from the data. Wrote queries to find out for instance:
  The names of unique launch sites in the space mission.

The permalink to the Notebook:

https://github.com/kylekk76/IBM-Data-Science/blob/9639f0a8049ac9ff53977186bdfc87104d63609c/capstone%20in%20progress/Exploratory_Data_Base_With_SQL.ipynb

Interactive Map with Folium

- Marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, identified which launch sites have relatively high success rate.
- Calculated the distances between a launch site to its proximities. Answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

This is the permalink if you wanna give a check:

https://github.com/kylekk76/IBM-Data-Science/blob/9639f0a8049ac9ff53977186bdfc87104d63609c/capstone%20in%20progress/Launch%20sites%20location.ipynb

Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

Permalink:

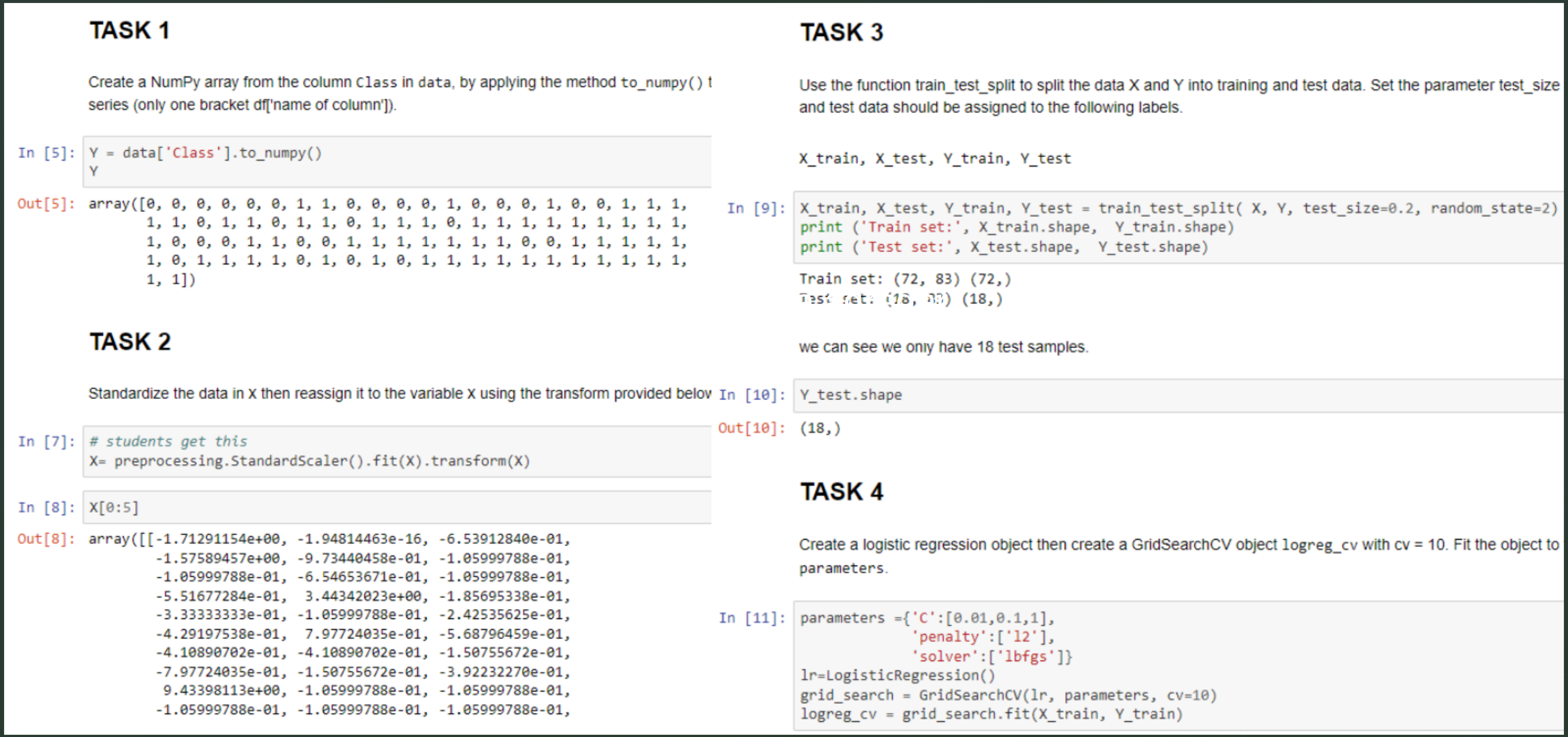
https://github.com/kylekk76/IBM-Data-Science/blob/9639f0a8049ac9ff53977186bdfc87104d63609c/capstone%20in%20progress/spacex_dash_app.py

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Found the best performing classification model.

- Permalink:
- https://github.com/kylekk76/IBM-Data-Science/blob/9788f7501cfb320fa3f5b3eac1e79b37c419b78c/capstone%20in%20progress/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Predictive Analysis (Classification)



TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` t series (only one bracket df['name of column']).

```
In [5]: Y = data['Class'].to_numpy()
        Y

Out[5]: array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
               1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
               1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1])
```

TASK 2

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

```
In [7]: # students get this
        X= preprocessing.StandardScaler().fit(X).transform(X)

In [8]: X[0:5]

Out[8]: array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01,
                -1.57589457e+00, -9.73440458e-01, -1.05999788e-01,
                -1.05999788e-01, -6.54653671e-01, -1.05999788e-01,
                -5.51677284e-01,  3.44342023e+00, -1.85695338e-01,
                -3.33333333e-01, -1.05999788e-01, -2.42535625e-01,
                -4.29197538e-01,  7.97724035e-01, -5.68796459e-01,
                -4.10890702e-01, -4.10890702e-01, -1.50755672e-01,
                -7.97724035e-01, -1.50755672e-01, -3.92232270e-01,
                 9.43398113e+00, -1.05999788e-01, -1.05999788e-01,
                -1.05999788e-01, -1.05999788e-01, -1.05999788e-01,
```

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

```
In [9]: X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
        print ('Train set:', X_train.shape,  Y_train.shape)
        print ('Test set:', X_test.shape,  Y_test.shape)

        Train set: (72, 83) (72,)
        Test set: (18, 83) (18,)
```

we can see we only have 18 test samples.

```
In [10]: Y_test.shape

Out[10]: (18,)
```

TASK 4

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

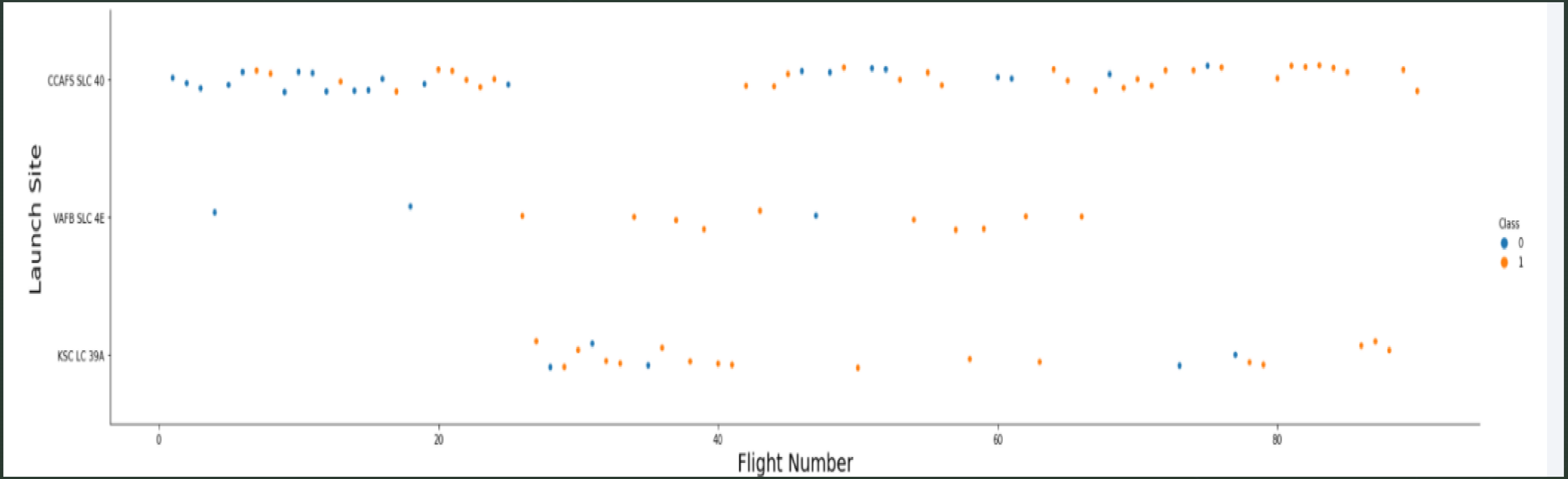
```
In [11]: parameters ={'C':[0.01,0.1,1],
                      'penalty':['l2'],
                      'solver':['lbfgs']}
         lr=LogisticRegression()
         grid_search = GridSearchCV(lr, parameters, cv=10)
         logreg_cv = grid_search.fit(X_train, Y_train)
```

RESULTS

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

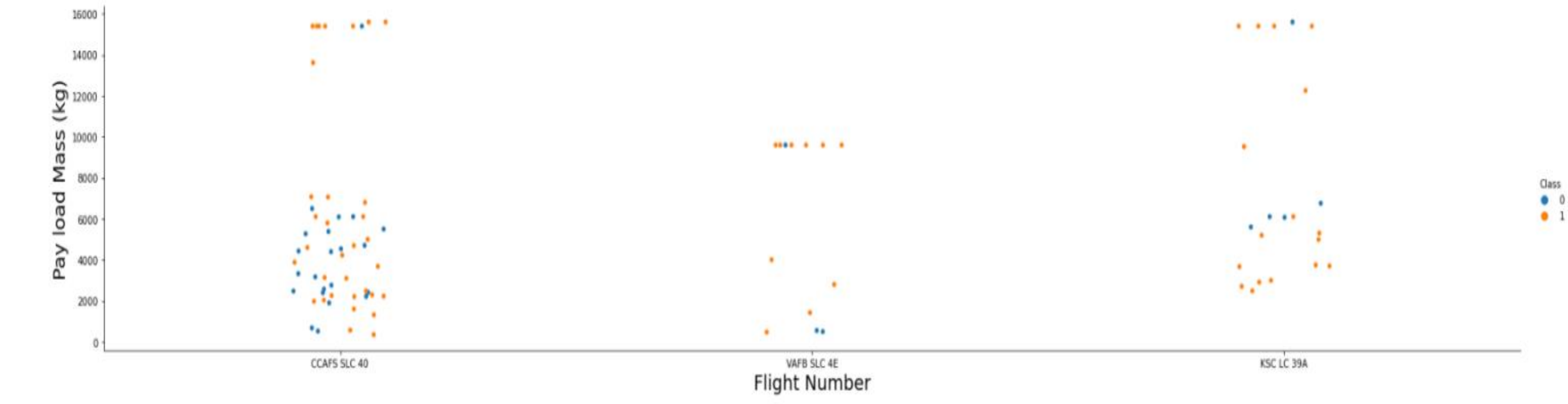
Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



Payload vs. Launch Site

If we observe Payload Vs. Launch Site scatter point chart you will find for the VAFBSLC launchsite there are no rockets launched for heavypayload mass(greater than 10000)



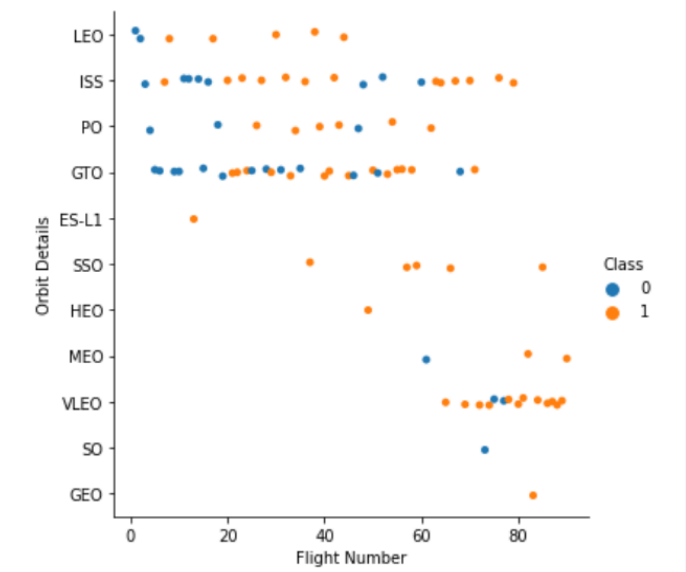
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

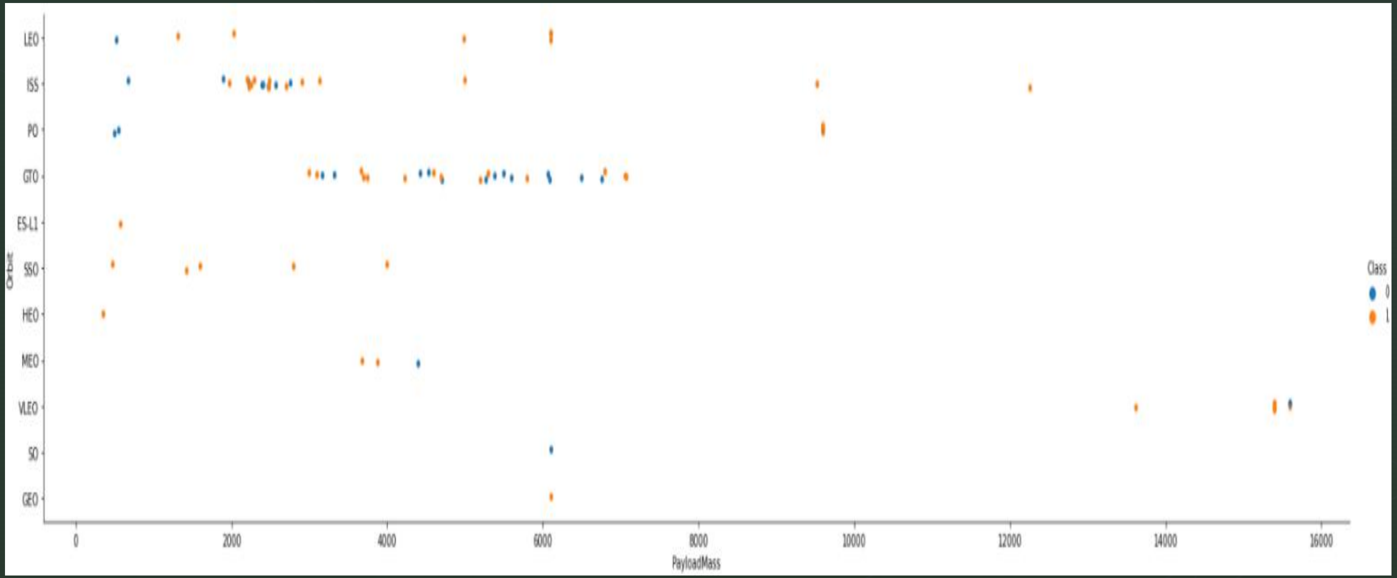
Success Rate vs. Orbit Type Flight Number vs. Orbit Type Payload vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO had the most success rate.

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

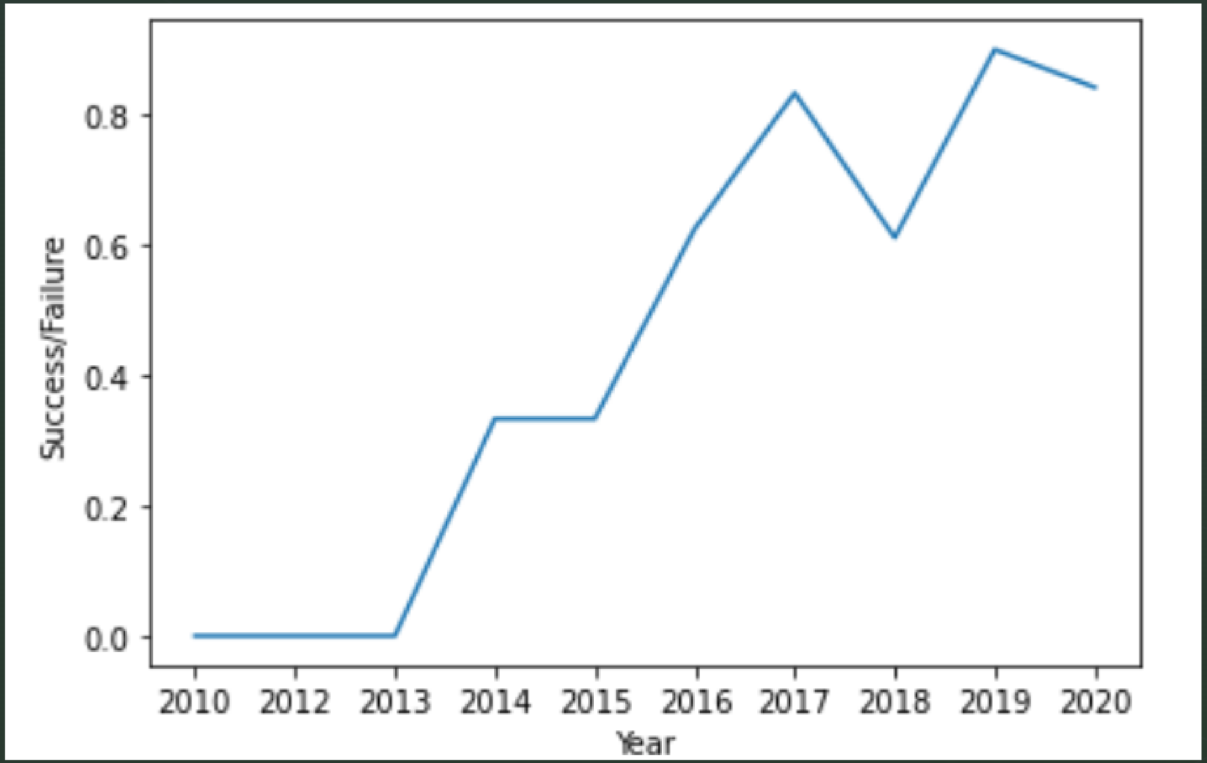
- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.





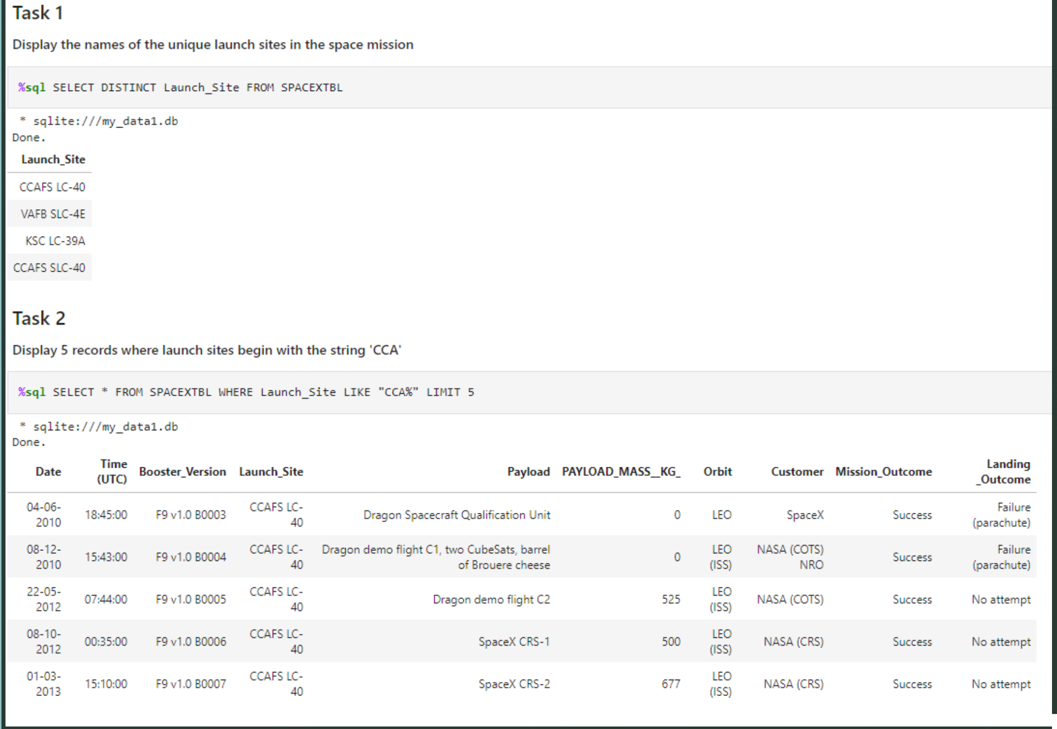
Launch Success Yearly Trend

From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



Getting some insights from the Data Base

- We used the key word DISTINCT to show only unique launch sites from the SpaceX data.
- Used the query above to display 2 records where launch sites begin with `CCA`
- Can calculate the total payload carried by boosters from NASA with the below query.
- Can calculate the average payload mass carried by booster version F9 v1.1 using the below query
- The date of the first successful landing outcome on ground pad can be fetched using the below query.
- Used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000
- Used wildcard like '%' to filter for WHERE LANDING__OUTCOME was a success or a failure.
- Determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.
- Used a combinations of the WHERE clause AND conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015
- We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes **BETWEEN** 06-04-2010' **AND** '03-20-2017. • We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order



Task 1

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

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

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

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

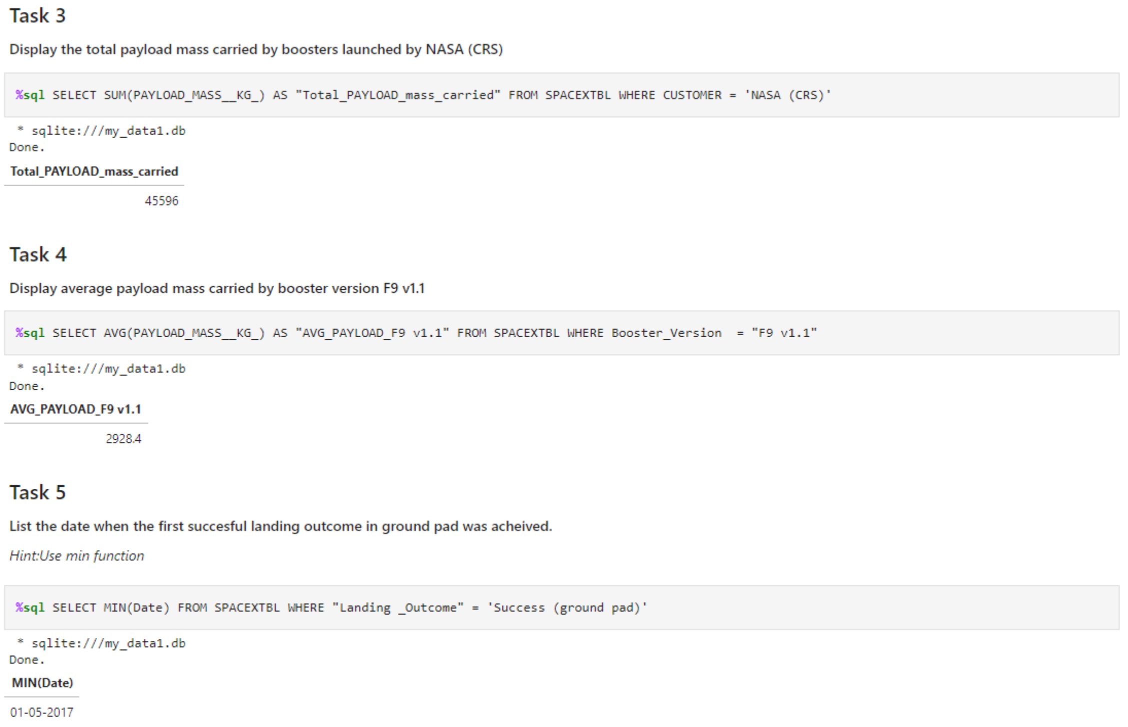
Task 2

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

```
%sql SELECT * FROM SPACEXTBL 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
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt



Task 3

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total_PAYLOAD_mass_carried" FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)'
```

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

Total_PAYLOAD_mass_carried
45596

Task 4

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "AVG_PAYLOAD_F9 v1.1" FROM SPACEXTBL WHERE Booster_Version  = "F9 v1.1"
```

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

AVG_PAYLOAD_F9 v1.1
2928.4

Task 5

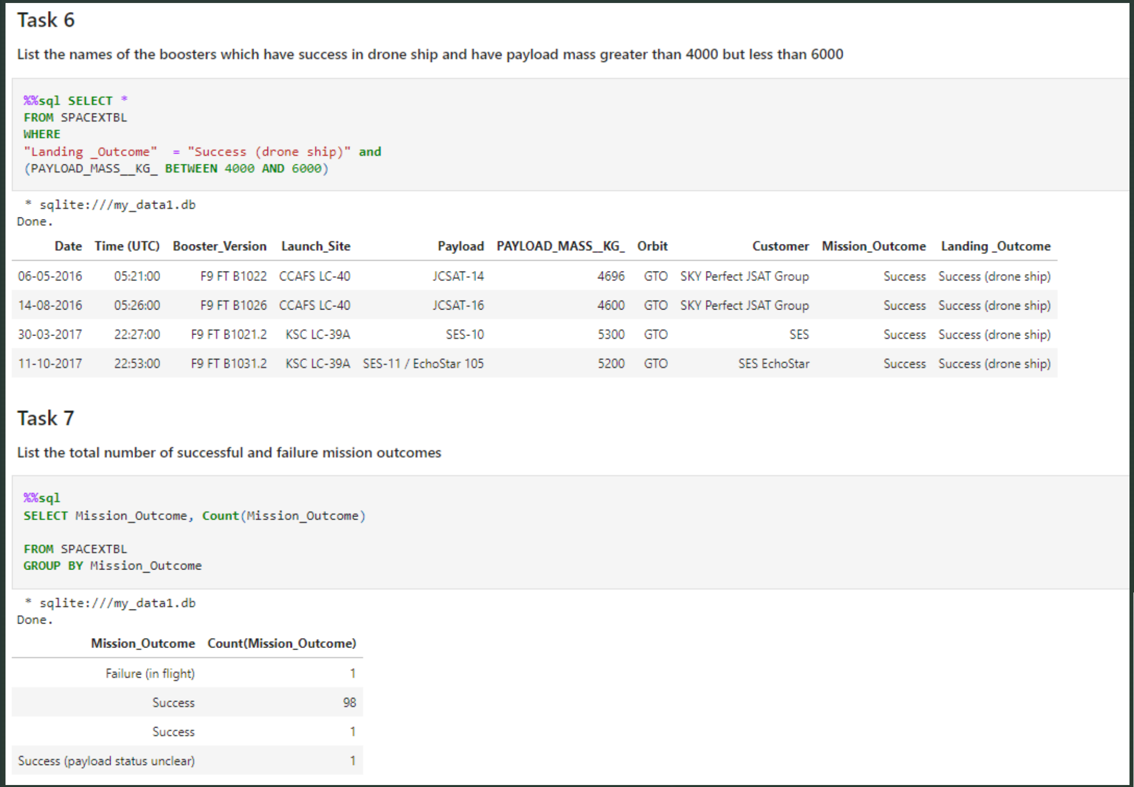
List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE "Landing _Outcome" = 'Success (ground pad)'
```

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

MIN(Date)
01-05-2017



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 *
FROM SPACEXTBL
WHERE
"Landing _Outcome"  = "Success (drone ship)" and
(PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
06-05-2016	05:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
14-08-2016	05:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
30-03-2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
11-10-2017	22:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200	GTO	SES EchoStar	Success	Success (drone ship)

Task 7

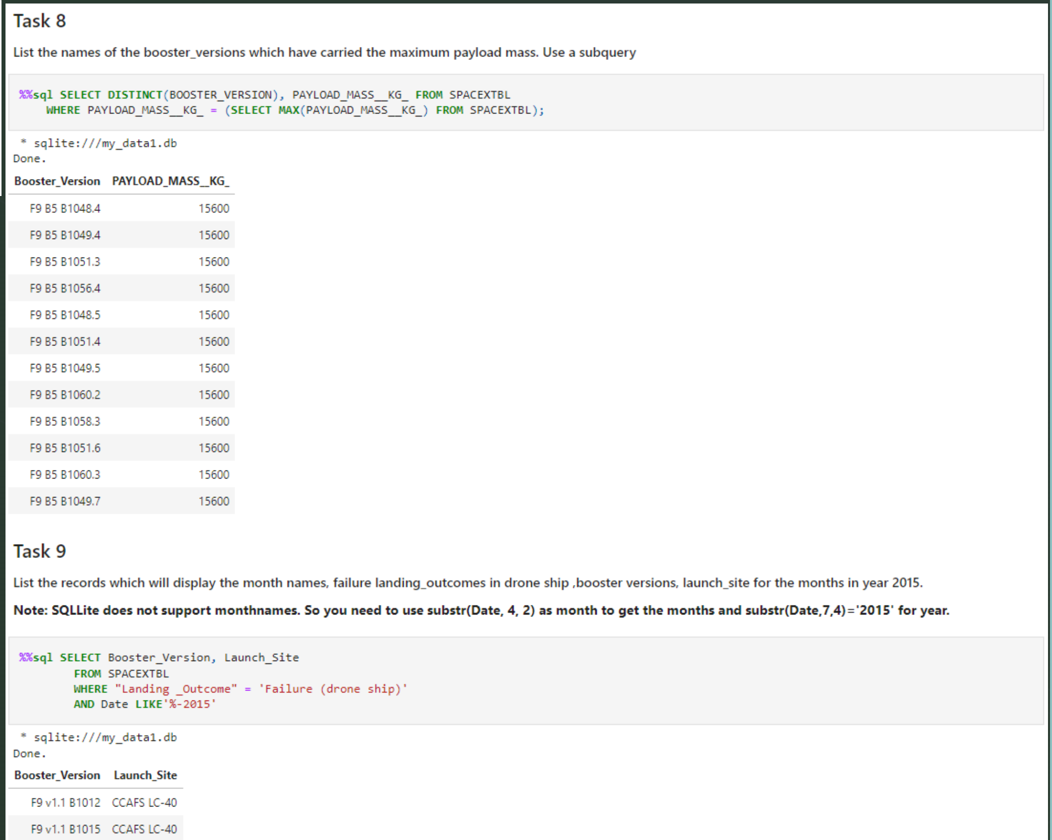
List the total number of successful and failure mission outcomes

```
%%sql
SELECT Mission_Outcome, Count(Mission_Outcome)

FROM SPACEXTBL
GROUP BY Mission_Outcome
```

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

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



Task 8

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

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

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

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

Task 9

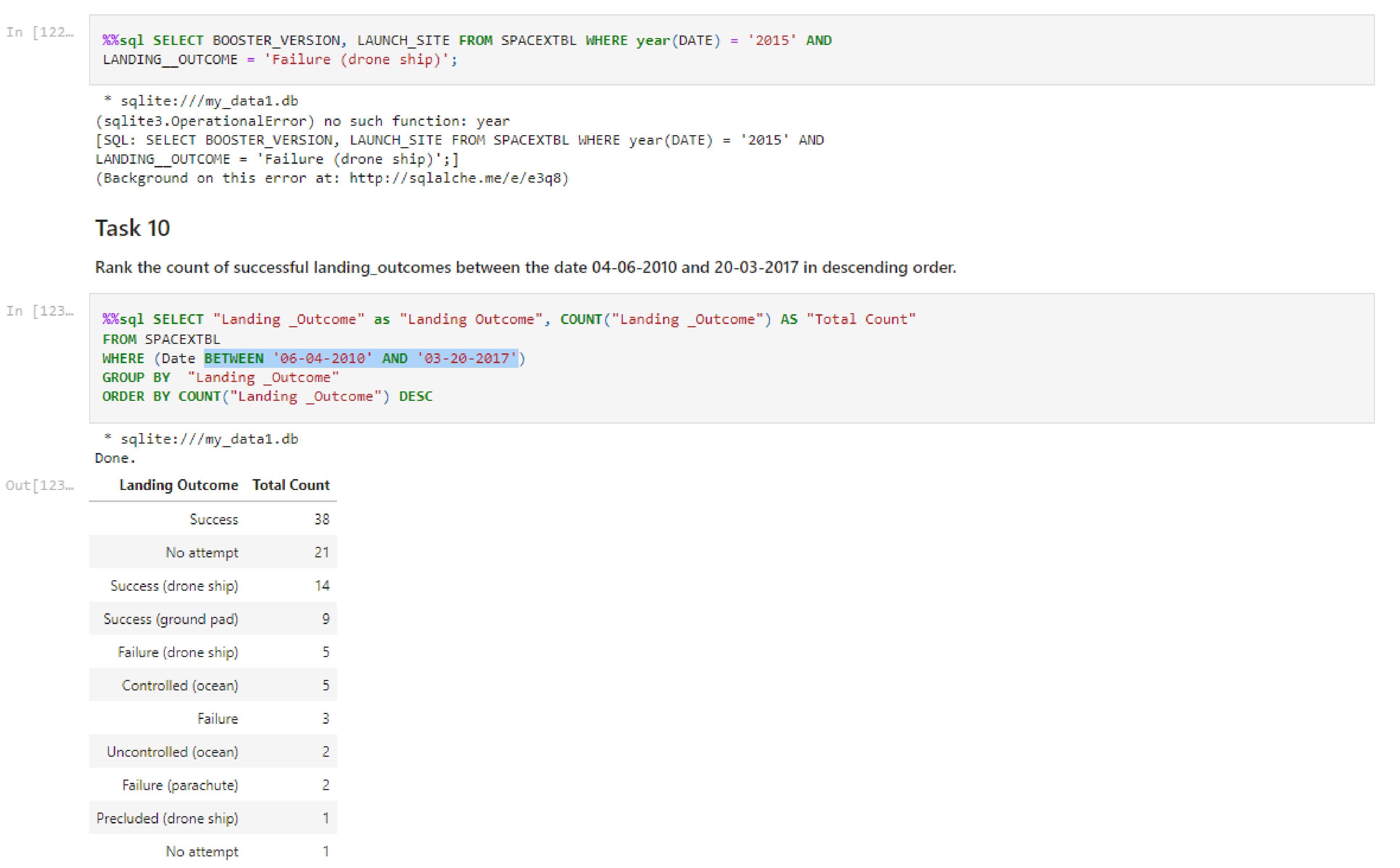
List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%%sql SELECT Booster_Version, Launch_Site
       FROM SPACEXTBL
       WHERE "Landing _Outcome" = 'Failure (drone ship)'
       AND Date LIKE'%-2015'
```

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

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40



```
In [122…
%%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE year(DATE) = '2015' AND
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such function: year
[SQL: SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE year(DATE) = '2015' AND
LANDING__OUTCOME = 'Failure (drone ship)';]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Task 10

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

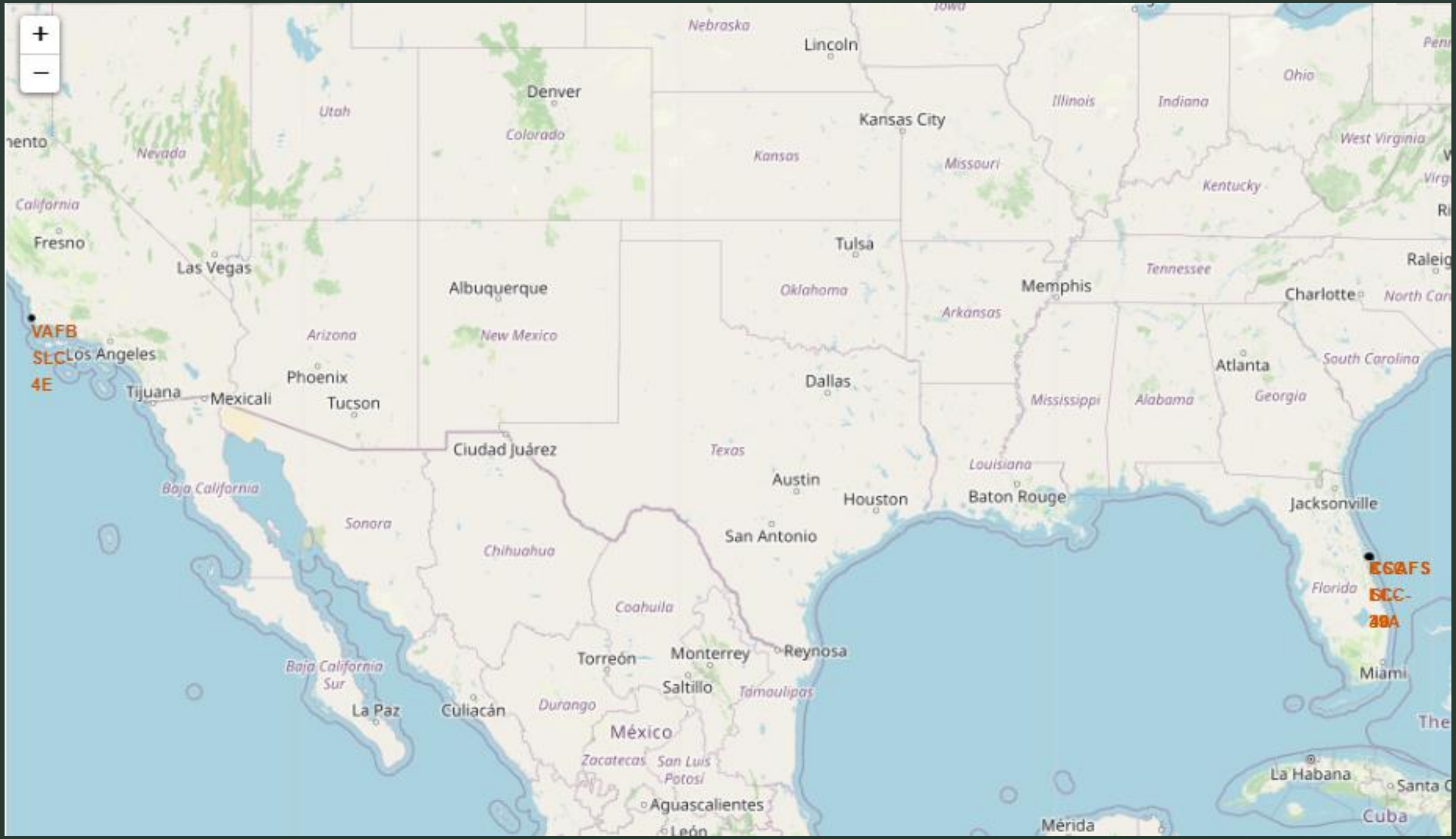
```
In [123…
%%sql SELECT "Landing _Outcome" as "Landing Outcome", COUNT("Landing _Outcome") AS "Total Count"
FROM SPACEXTBL
WHERE (Date BETWEEN '06-04-2010' AND '03-20-2017')
GROUP BY  "Landing _Outcome"
ORDER BY COUNT("Landing _Outcome") DESC
```

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

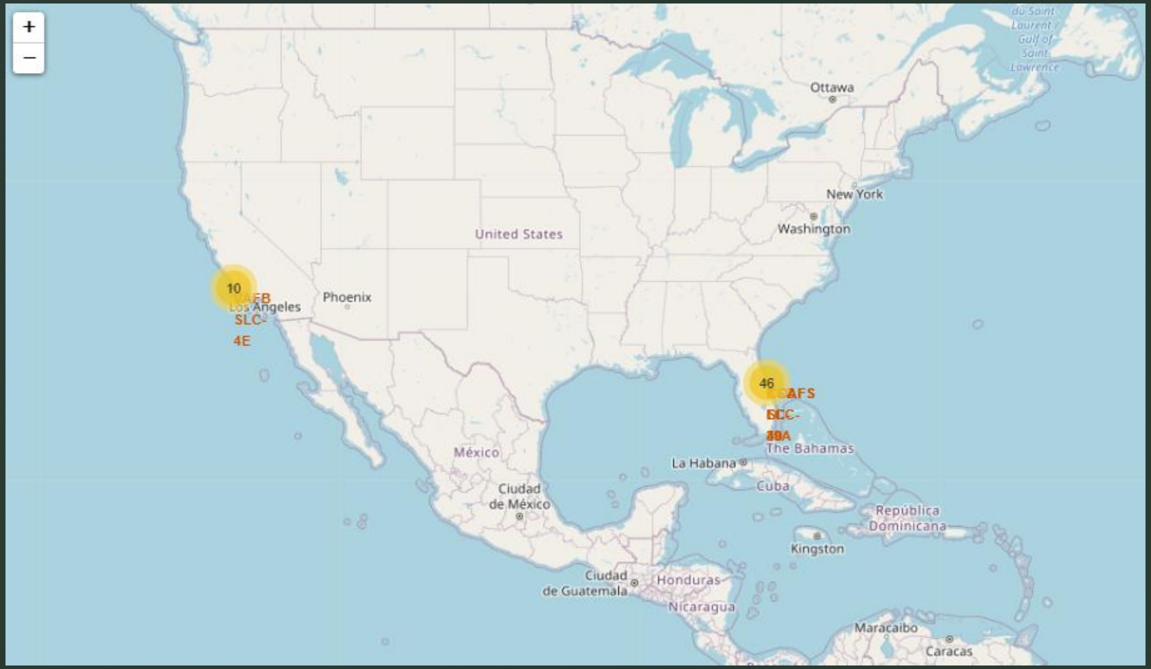
Out[123…

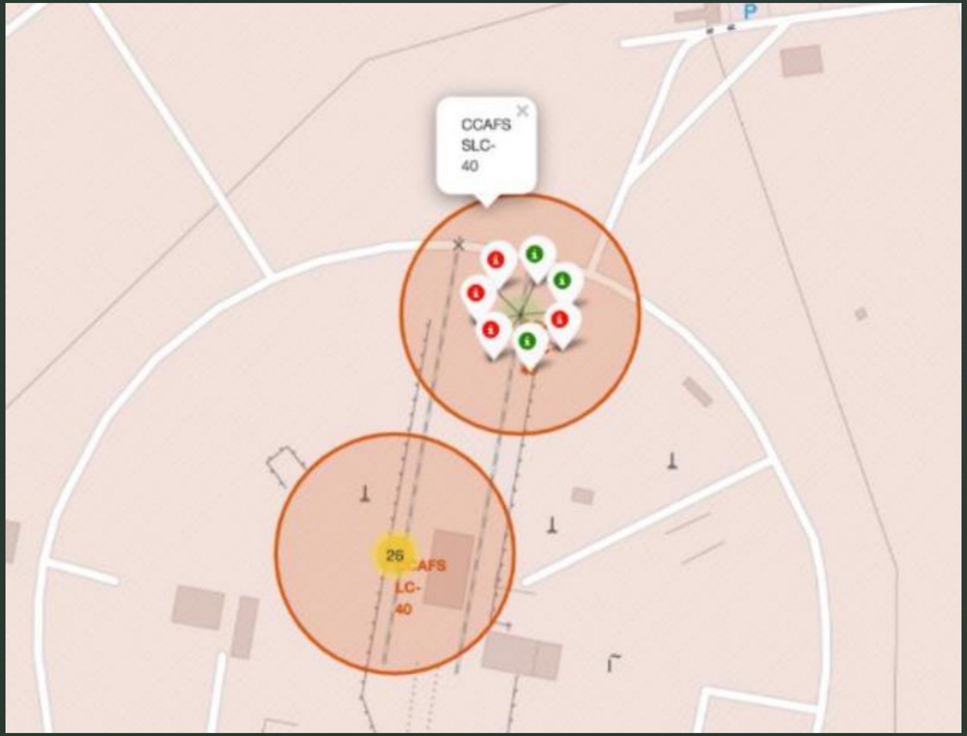
Landing Outcome	Total Count
Success	38
No attempt	21
Success (drone ship)	14
Success (ground pad)	9
Failure (drone ship)	5
Controlled (ocean)	5
Failure	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1
No attempt	1

All launch sites on a map

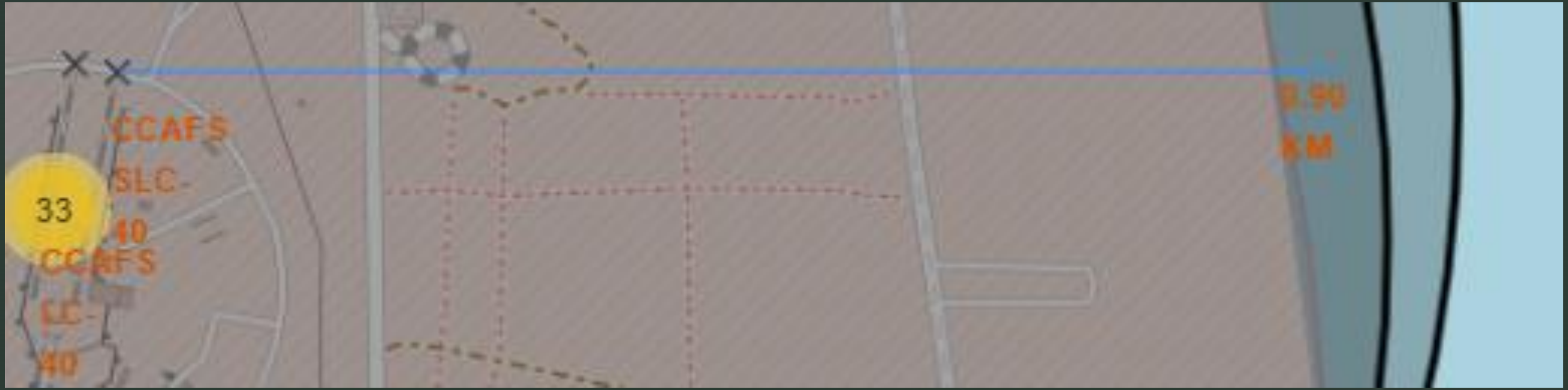


Success/failed launches for each site on the map and Success/failed launches for each site on the map

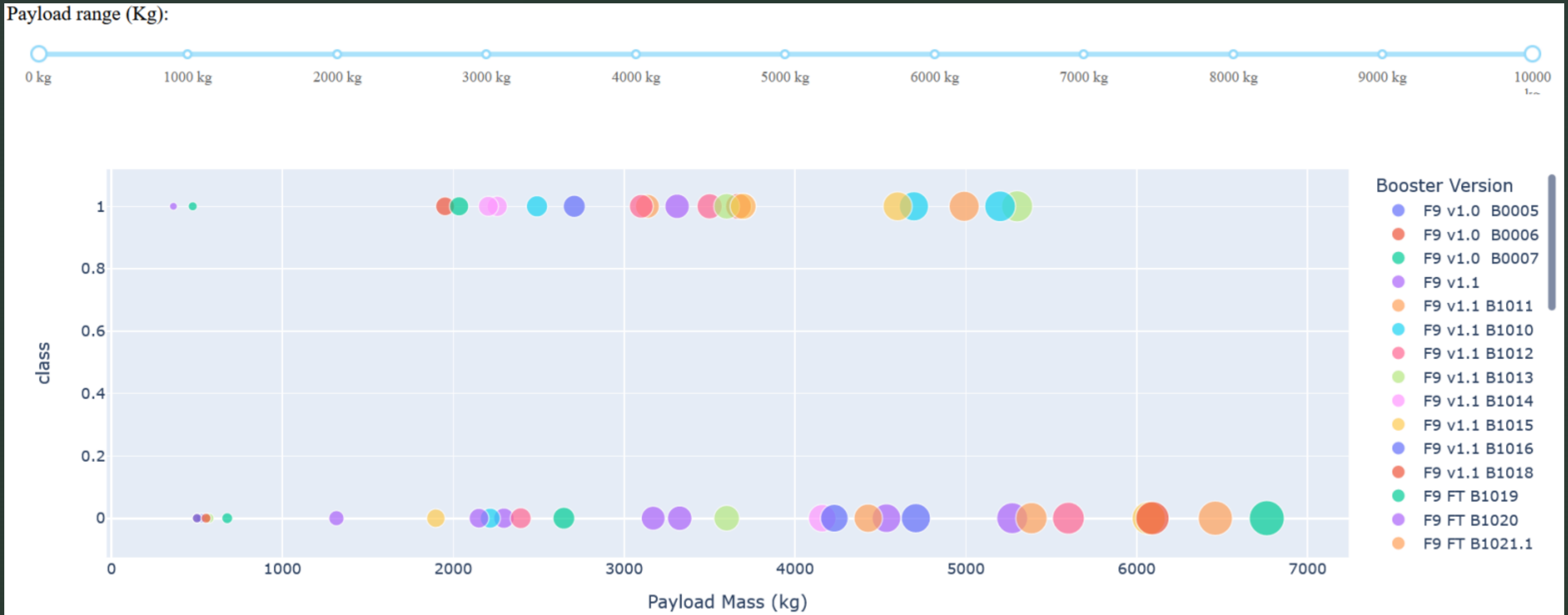


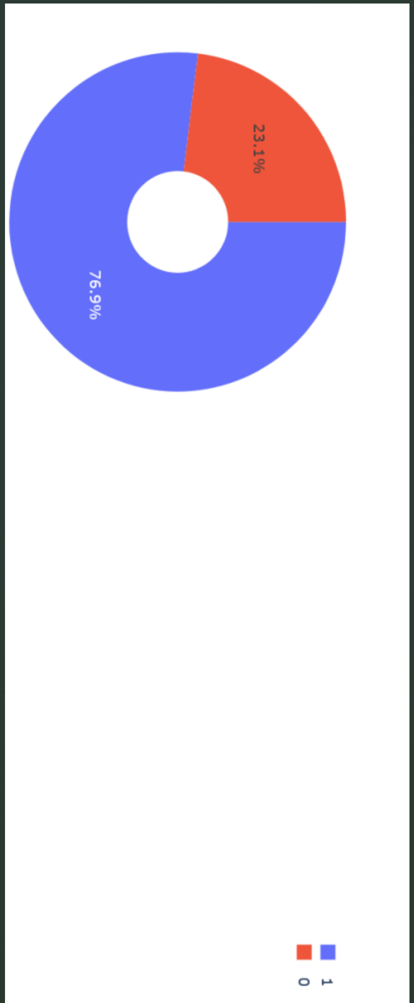


Distances between a launch site to his proximities



Pie chart showing the Launch site with the highest launch success ratio and Scatter plot of Payload vs Launch Outcome for all sites

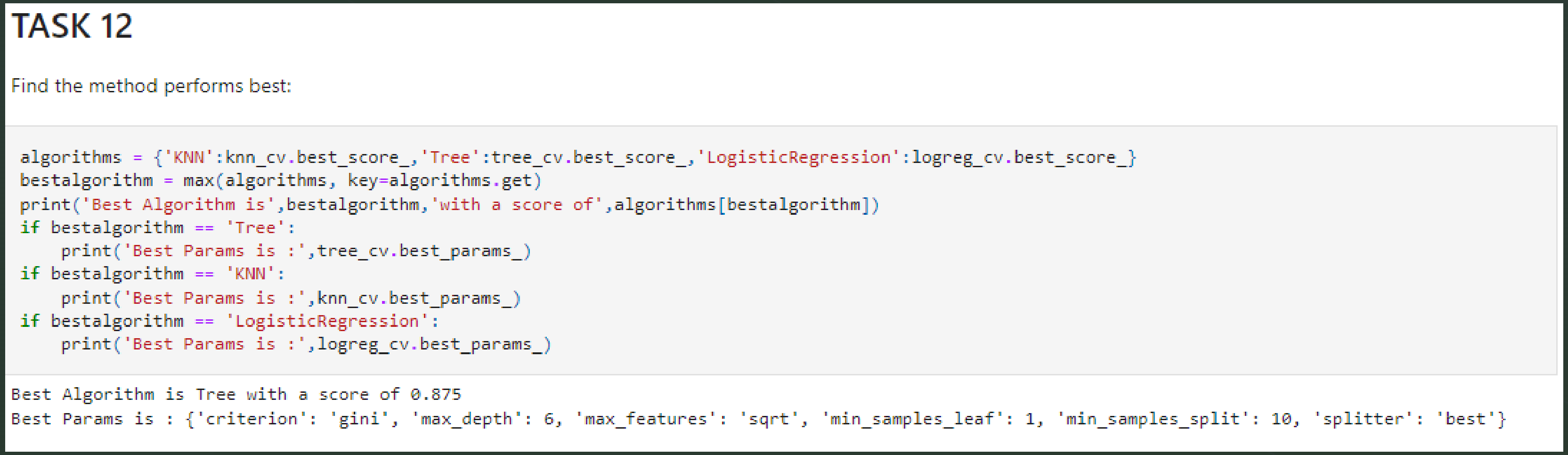




Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

- It gives us a 87% of confidence probably could be improve with more data.



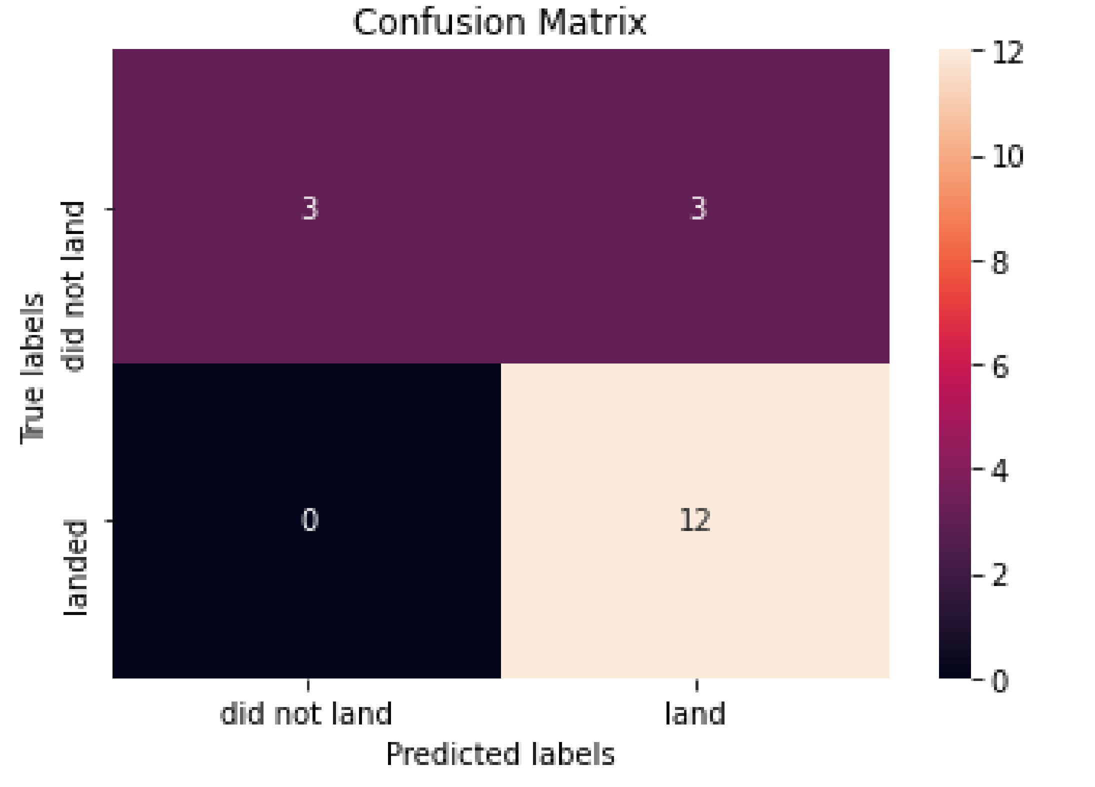
TASK 12

Find the method performs best:

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.875
Best Params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'best'}
```

Confusion Matrix


Confusion Matrix

True labels \ Predicted labels	did not land	land
did not land	3	3
landed	0	12

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes.
- The major problem is the false positives. i.e. unsuccessful landing marked as successful landing by the classifier.

Conclusions

- It can be concluded that: • The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

……………………..Note……………………..

After This Presentation i will expend more time in improve this but at the moment i got few hours until my subscripcion ends so this is the best i could do with the time i had, i gope you apreciate and enjoy the work, see you soon