IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Julio

Jan 26, 2024

Outline

Executive Summary

- Objective: Determine Falcon 9 first stage landing success to predict launch costs.
- Methodology: Utilize a standard Data Science pipeline including:
- Data Collection: Aggregate historical launch records.
- Data Processing: Clean and structure data for analysis.
- Feature Engineering: Identify key predictors of landing outcomes.
- Model Selection: Test various algorithms for best predictive accuracy.
- Validation: Employ cross-validation techniques to ensure model robustness.
- Impact: Enable cost estimation for SpaceX and potential competitors.
- Outcome: Aid in competitive bidding strategy for rocket launch services.

Introduction

- Project background and context
- In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX 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 SpaceX 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 SpaceX for a rocket launch.

Data Collection – SpaceX API

- The data is collected via the request to the SpaceX API
- GitHub URL
- https://github.com/JulioWangUS/DS-jupyter/blob/main/1-spacex-data-collection-api.ipynb

Data Wrangling

- Performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident.
- Converted those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.
- URL
- https://github.com/JulioWangUS/DS-jupyter/blob/main/2-webscraping.ipynb
- https://github.com/JulioWangUS/DS-jupyter/blob/main/3-spacex-Data%20wrangling.ipynb

EDA and interactive visual analytics methodology

- performed EDA using a database.
- checked if the data can be used to automatically determine if the Falcon 9's first stage will land. Some attributes can be used to determine if the first stage can be reused.
- used these features with machine learning to automatically predict if the first stage can land successfully
- determined what attributes are correlated with successful landings. The categorical variables will be converted using one hot encoding, preparing the data for a machine learning model that will predict if the first stage will successfully land.

EDA with Data Visualization

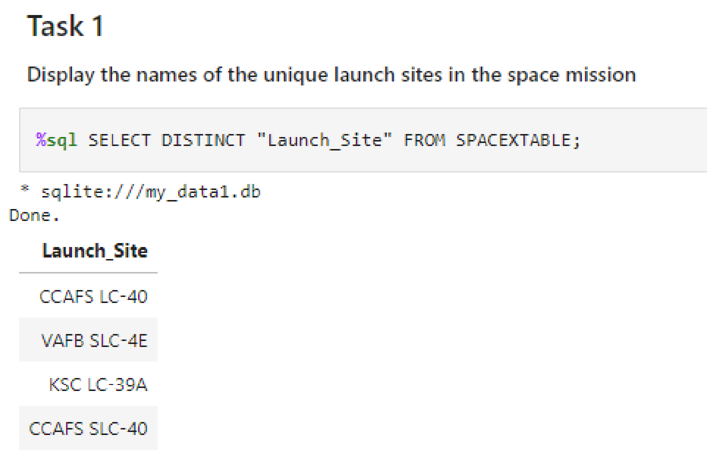
- Plotted scatter, bar and line charts, and conducted Feature Engineering to explore the relationships between the features and outcome.
- Add the GitHub URL
- https://github.com/JulioWangUS/DS-jupyter/blob/main/5-eda-dataviz.ipynb

EDA with SQL

- **Display the names of the unique launch sites in the space mission.**
- **Display 5 records where launch sites begin with the string 'CCA'**
- **Display the total payload mass carried by boosters launched by NASA (CRS)**
- **Display average payload mass carried by booster version F9 v1.1**
- **List the date when the first succesful landing outcome in ground pad was acheived.**
- **List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000**
- **List the total number of successful and failure mission outcomes**
- **List the names of the booster_versions which have carried the maximum payload mass. Use a subquery**
- **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.**
- **Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.**

- Add the GitHub URL
- https://github.com/JulioWangUS/DS-jupyter/blob/main/4-eda-sql.ipynb

EDA with SQL



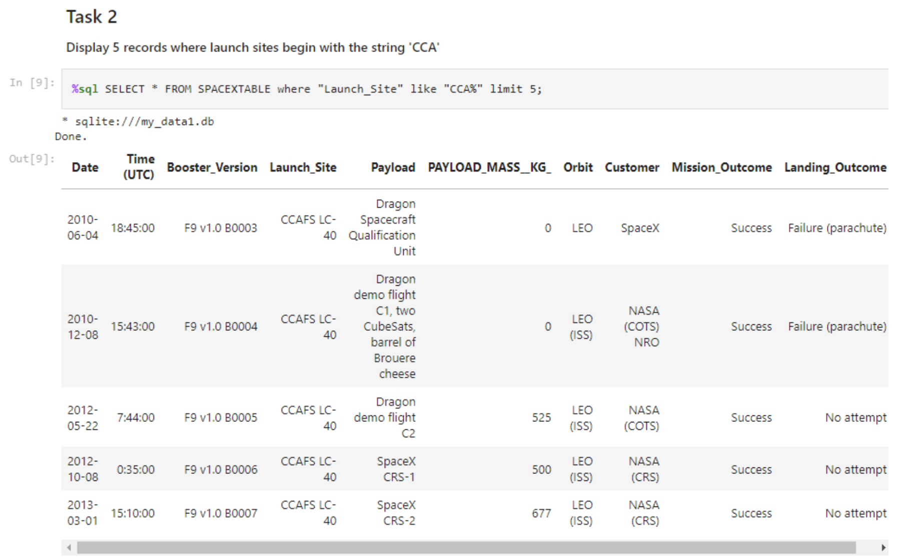
Task 1

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



Task 2

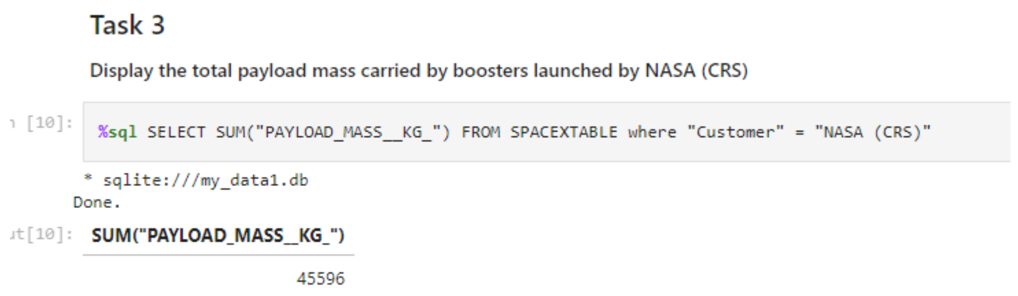
Display 5 records where launch sites begin with the string 'CCA'

```
In [9]: %sql SELECT * FROM SPACEXTABLE where "Launch_Site" like "CCA%" limit 5;
```

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

Out[9]:

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



Task 3

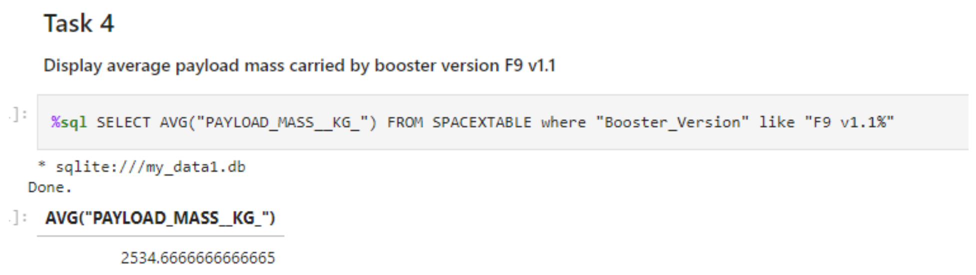
Display the total payload mass carried by boosters launched by NASA (CRS)

```
n [10]: %sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE where "Customer" = "NASA (CRS)"
```

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

ut[10]:

SUM("PAYLOAD_MASS__KG_")
45596



Task 4

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

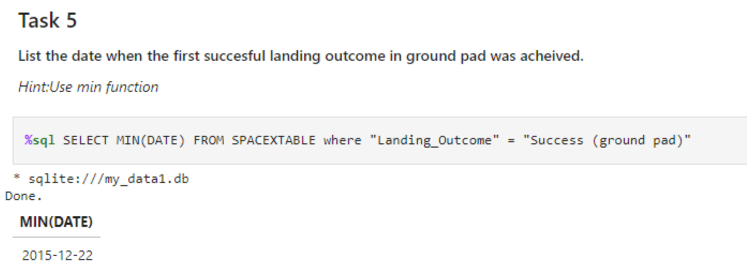
```
]: %sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTABLE where "Booster_Version" like "F9 v1.1%"
```

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

]:

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

EDA with SQL



Task 5

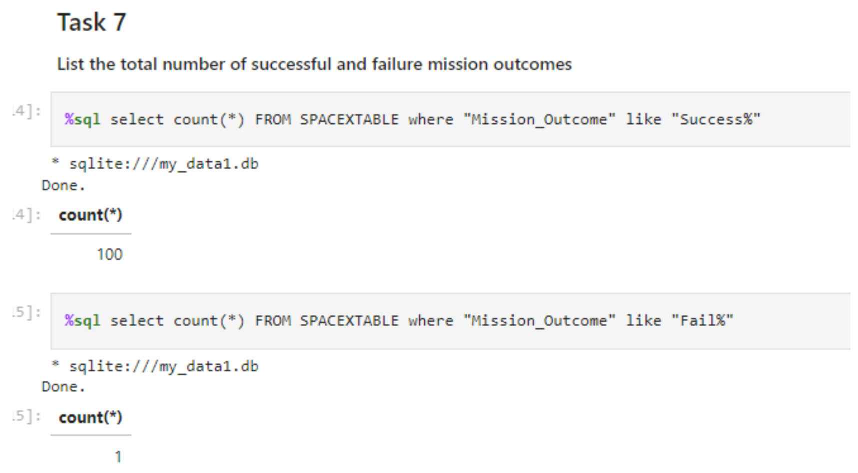
List the date when the first succesful landing outcome in ground pad was acheived.

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



Task 7

List the total number of successful and failure mission outcomes

```
.4]:  %sql select count(*) FROM SPACEXTABLE where "Mission_Outcome" like "Success%"
```

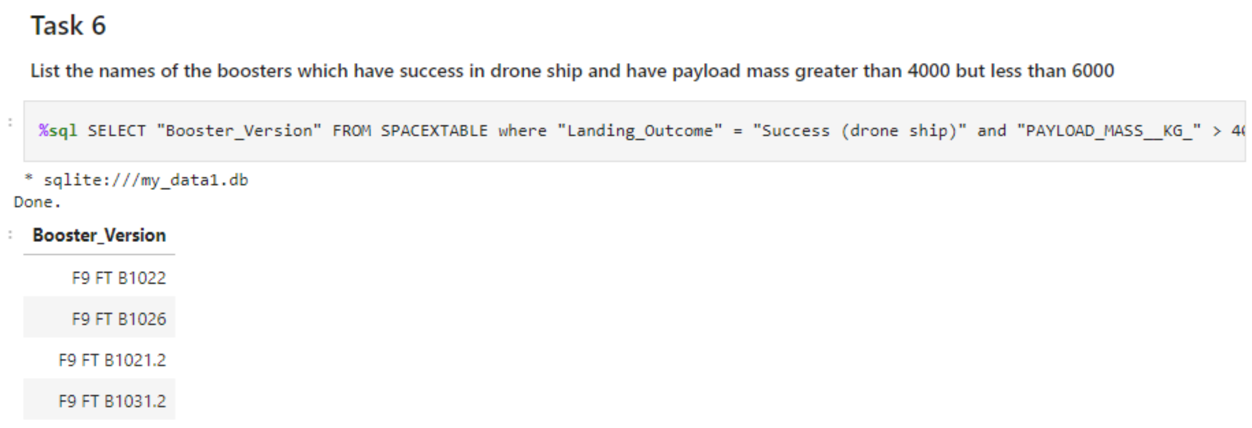
```
 * sqlite:///my_data1.db
Done.
```

count(*)
100

```
.5]:  %sql select count(*) FROM SPACEXTABLE where "Mission_Outcome" like "Fail%"
```

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

count(*)
1



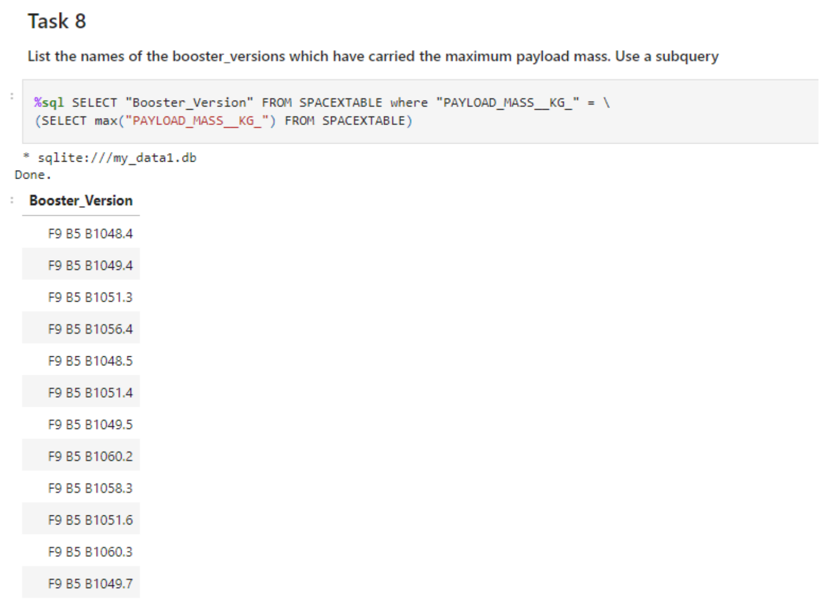
Task 6

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 "Booster_Version" FROM SPACEXTABLE where "Landing_Outcome" = "Success (drone ship)" and "PAYLOAD_MASS__KG_" > 4
```

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

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



Task 8

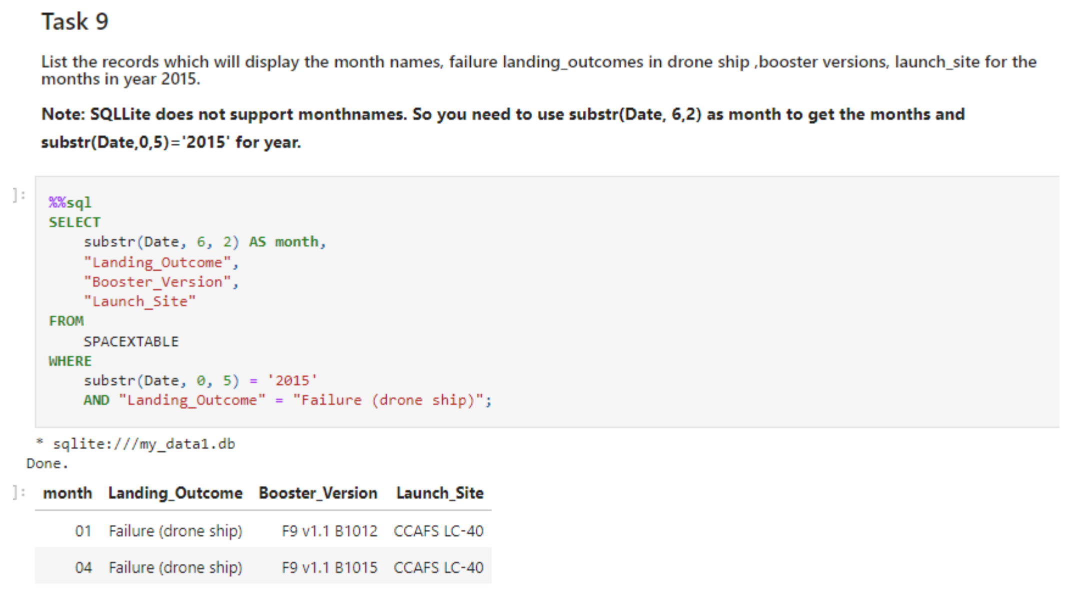
List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql SELECT "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

EDA with SQL



Task 9

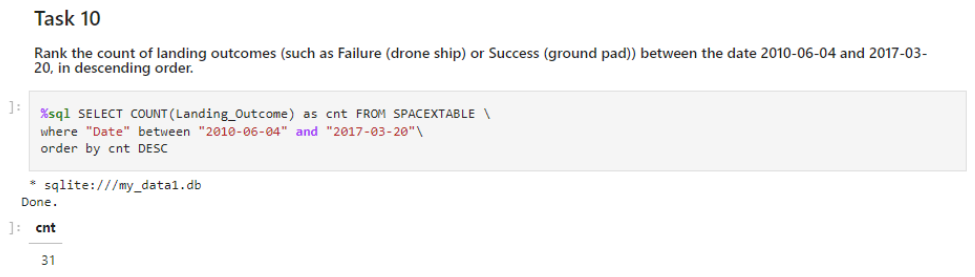
List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

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
    substr(Date, 0, 5) = '2015'
    AND "Landing_Outcome" = "Failure (drone ship)";
```

```
 * 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



Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%sql SELECT COUNT(Landing_Outcome) as cnt FROM SPACEXTABLE \
where "Date" between "2010-06-04" and "2017-03-20"\
order by cnt DESC
```

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

cnt
31

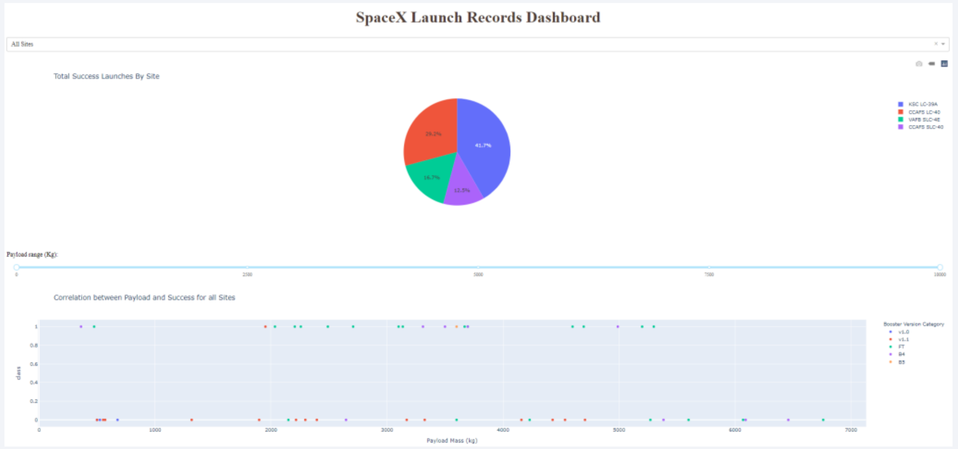
Build an Interactive Map with Folium

- Marked all launch sites on a map
- Marked the success/failed launches for each site on the map
- Calculated the distances between a launch site to its proximities

- Add the GitHub URL
- https://github.com/JulioWangUS/DS-jupyter/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart.

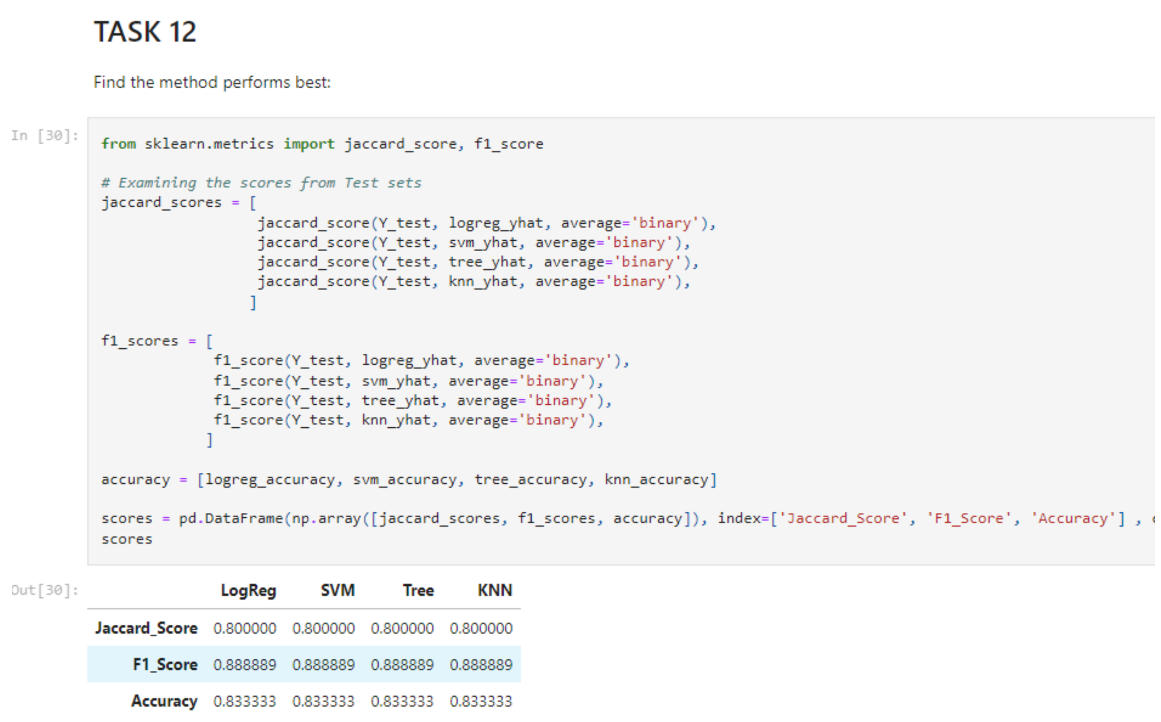


- Add the GitHub URL
- https://github.com/JulioWangUS/DS-jupyter/blob/main/6_spacex_dash_app.py

Predictive Analysis (Classification)

- Perform exploratory Data Analysis and determine Training Labels
  - create a column for the class
  - Standardize the data
  - Split into training data and test dataYou need present your model development process using key phrases and flowchart
- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
  - Find the method performs best using test data
- Add the GitHub URL
- https://github.com/JulioWangUS/DS-jupyter/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Predictive Analysis (Classification)



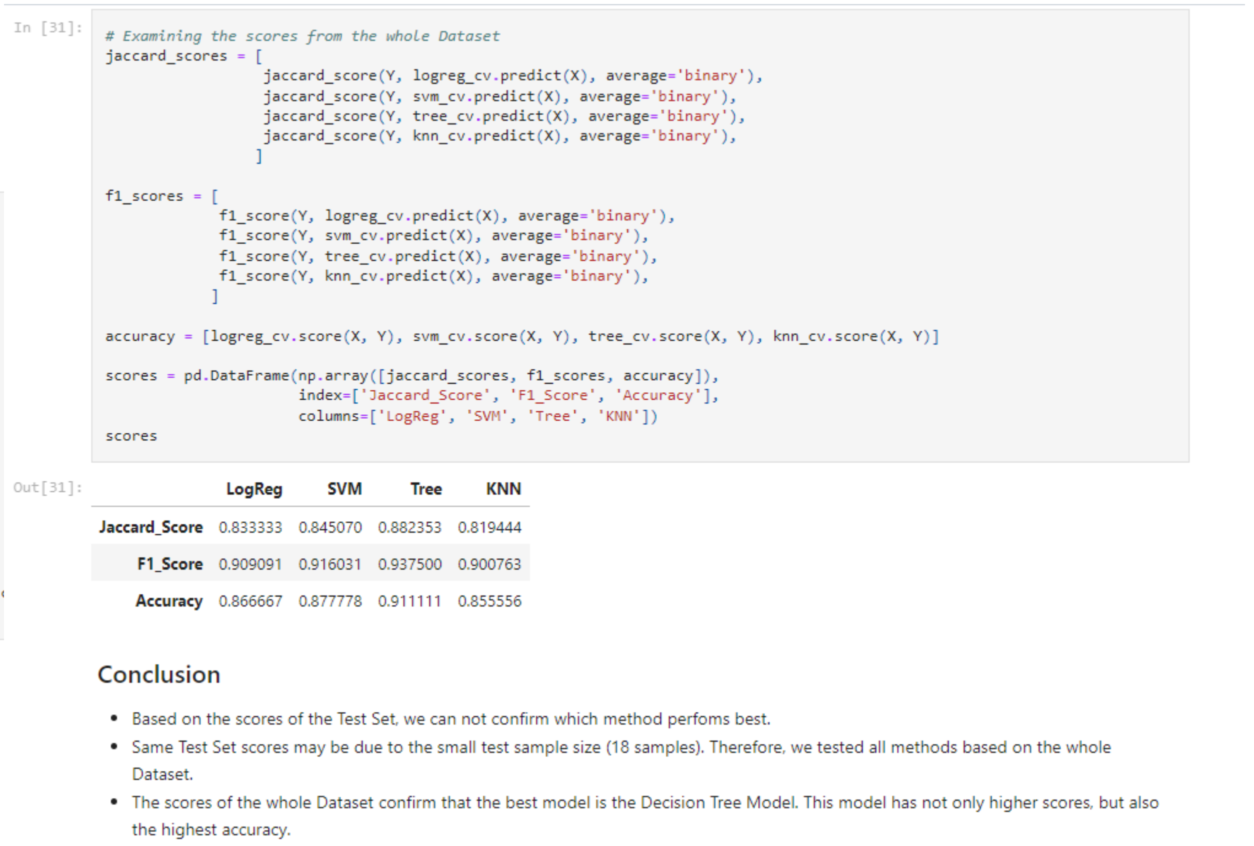
TASK 12

Find the method performs best:

In [30]:
```
from sklearn.metrics import jaccard_score, f1_score

# Examining the scores from Test sets
jaccard_scores = [
                  jaccard_score(Y_test, logreg_yhat, average='binary'),
                  jaccard_score(Y_test, svm_yhat, average='binary'),
                  jaccard_score(Y_test, tree_yhat, average='binary'),
                  jaccard_score(Y_test, knn_yhat, average='binary'),
                 ]

f1_scores = [
             f1_score(Y_test, logreg_yhat, average='binary'),
             f1_score(Y_test, svm_yhat, average='binary'),
             f1_score(Y_test, tree_yhat, average='binary'),
             f1_score(Y_test, knn_yhat, average='binary'),
            ]

accuracy = [logreg_accuracy, svm_accuracy, tree_accuracy, knn_accuracy]

scores = pd.DataFrame(np.array([jaccard_scores, f1_scores, accuracy]), index=['Jaccard_Score', 'F1_Score', 'Accuracy'] ,
scores
```

Out[30]:

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.800000	0.800000	0.800000	0.800000
F1_Score	0.888889	0.888889	0.888889	0.888889
Accuracy	0.833333	0.833333	0.833333	0.833333



In [31]:
```
# Examining the scores from the whole Dataset
jaccard_scores = [
                  jaccard_score(Y, logreg_cv.predict(X), average='binary'),
                  jaccard_score(Y, svm_cv.predict(X), average='binary'),
                  jaccard_score(Y, tree_cv.predict(X), average='binary'),
                  jaccard_score(Y, knn_cv.predict(X), average='binary'),
                 ]

f1_scores = [
             f1_score(Y, logreg_cv.predict(X), average='binary'),
             f1_score(Y, svm_cv.predict(X), average='binary'),
             f1_score(Y, tree_cv.predict(X), average='binary'),
             f1_score(Y, knn_cv.predict(X), average='binary'),
            ]

accuracy = [logreg_cv.score(X, Y), svm_cv.score(X, Y), tree_cv.score(X, Y), knn_cv.score(X, Y)]

scores = pd.DataFrame(np.array([jaccard_scores, f1_scores, accuracy]),
                      index=['Jaccard_Score', 'F1_Score', 'Accuracy'],
                      columns=['LogReg', 'SVM', 'Tree', 'KNN'])

scores
```

Out[31]:

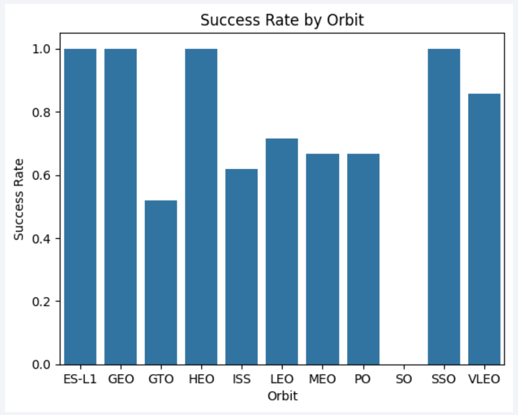
	LogReg	SVM	Tree	KNN
Jaccard_Score	0.833333	0.845070	0.882353	0.819444
F1_Score	0.909091	0.916031	0.937500	0.900763
Accuracy	0.866667	0.877778	0.911111	0.855556

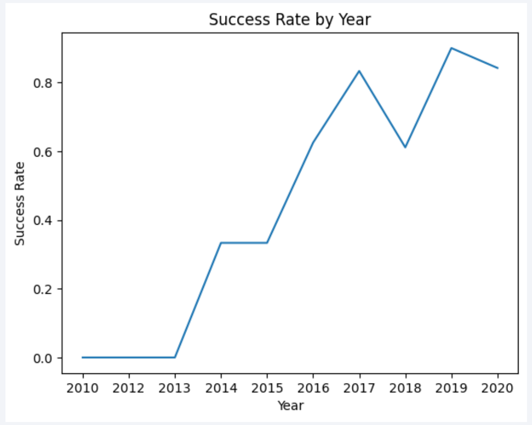
Conclusion

- Based on the scores of the Test Set, we can not confirm which method perfoms best.
- Same Test Set scores may be due to the small test sample size (18 samples). Therefore, we tested all methods based on the whole Dataset.
- The scores of the whole Dataset confirm that the best model is the Decision Tree Model. This model has not only higher scores, but also the highest accuracy.

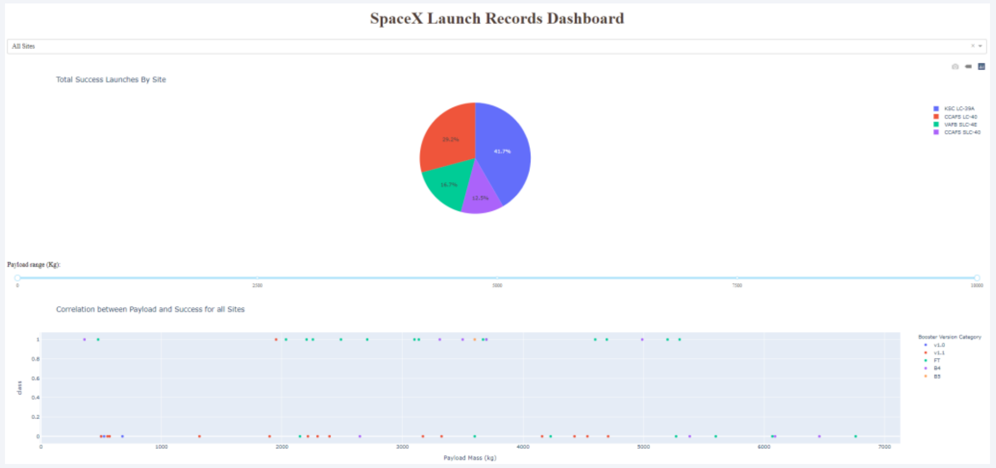
Results

- Exploratory data analysis results





- Interactive analytics demo in screenshots



- Predictive analysis results
  - Whole sample size: 90
  - Test size: 18
  - Best SVM kernel: ‘sigmoid’
  - Highest accuracy by decision tree via GridSearch: 0.83

Thank you!