Applied Data Science

Hanna Hrekova

30.11.2024

Skills Network

IBM

Executive Summary

- Slide 3: Introduction
- Slides 4-5: Data Collection & Data Wrangling Methodology
- Slides 6-7: EDA & Interactive Visual Analytics Methodology
- Slides 8-9: Predictive Analysis Methodology
- Slide 10: EDA with Visualization Results
- Slide 11: EDA with SQL Results
- Slide 12: Interactive Map with Folium Results
- Slides 13-14: Plotly Dash Dashboard Results
- Slide 15: Predictive Analysis
- Slide 16: Conclusion
- Slide 17: Creativity & Insights

Introduction

Goal:

· Apply data science methods to solve a real-world problem.

· Analysis of successful and failed SpaceX rocket landings.

What was done:

· Collection, processing, and analysis of data.

· Building machine learning models for forecasting.

Data Collection & Data Wrangling Methodology

Data sources:

- *SpaceX API*
- *Web scraping for additional data*

Out[21]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.07B0003.18	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.07B0004.18	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.07B0005.18	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.07B0006.18	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.07B0007.18	No attempt\n	1 March 2013	15:10

In [27]:

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Out[27]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False
...	...	...	...	...	...	...	...	...	...	...
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True
90	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True
91	88	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True
92	89	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True
93	90	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False

90 rows × 17 columns

Data wrangling:

- *Data cleaning, gap filling*
- *Variable transformation for further analysis*

In [6]:

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Out[6]:

```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

In [13]:

```
df.head(5)
```

Out[13]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False

In [8]:

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

Out[8]:

```
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: count, dtype: int64
```

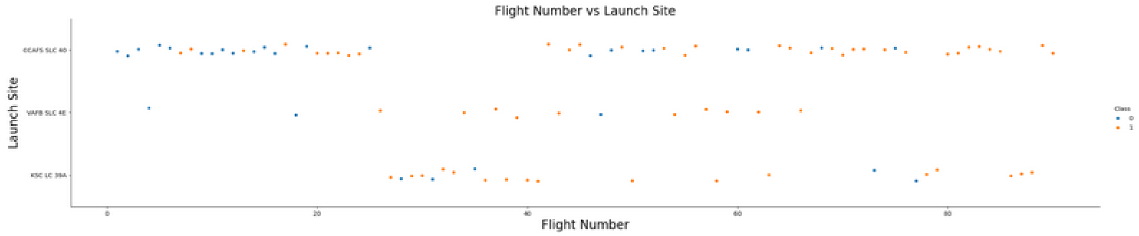
EDA & Interactive Visual Analytics Methodology

EDA & Visualization Methods

In [5]:

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be t
import seaborn as sns
import matplotlib.pyplot as plt

sns.catplot(
    x="FlightNumber",
    y="LaunchSite",
    hue="Class",
    data=df,
    aspect=5,
    kind="strip"
)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.title("Flight Number vs Launch Site", fontsize=20)
plt.show()
```



Exploratory Data Analysis

First, let's read the SpaceX dataset into a Pandas dataframe and print its summary

In [3]:

```
from js import fetch
import io

URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets
resp = await fetch(URL)
dataset_part_2_csv = io.BytesIO((await resp.arrayBuffer()).to_py())
df=pd.read_csv(dataset_part_2_csv)
df.head(5)
```

Out[3]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False

Interactive analysis tools (Folium, Dash)

```python
import pandas as pd
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.express as px

spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

app = dash.Dash(__name__)

app.layout = html.Div(children=[
    html.H1(
        'SpaceX Launch Records Dashboard',
        style={'textAlign': 'center', 'color': '#077533', 'font-size': 40}
    ),
```

In [9]:

```python
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add La
for i in range (len(launch_sites_df.index)):
    coordinate = [launch_sites_df["Lat"][i], launch_sites_df["Long"][i]]
    circle = folium.Circle(coordinate, radius=100, color='#0aa153', fill=True).add_child(folium.Popup(launc
    marker = folium.map.Marker(
        coordinate,
        icon=DivIcon(
            icon_size=(20, 20),
            icon_anchor=(0, 0),
            html='<div style="font-size: 12; color:#620aa1;"><b>%s</b></div>' % launch_sites_df["Launch Sit
            )
        )
    site_map.add_child(circle)
    site_map.add_child(marker)

site_map
```

Predictive Analysis Methodology

Used machine learning algorithms (KNN, decision tree, logistic regression, support vector machine)

```
In [12]: parameters ={'C':[0.01,0.1,1],
                      'penalty':['l2'],
                      'solver':['lbfgs']}
```

```
In [13]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
         lr=LogisticRegression()
         logreg_cv = GridSearchCV(lr, parameters, cv=10)
         logreg_cv.fit(X_train, Y_train)

Out[13]: GridSearchCV(cv=10, estimator=LogisticRegression(),
                      param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                                  'solver': ['lbfgs']})
```

```
In [19]: parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
                       'C': np.logspace(-3, 3, 5),
                       'gamma':np.logspace(-3, 3, 5)}
         svm = SVC()
```

```
In [20]: svm_cv = GridSearchCV(svm, parameters, cv=10)
         svm_cv.fit(X_train, Y_train)

Out[20]: GridSearchCV(cv=10, estimator=SVC(),
                      param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.1622776
         6e+01,
                1.00000000e+03]),
                                  'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.162
         27766e+01,
                1.00000000e+03]),
                                  'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

```
In [27]: parameters = {'criterion': ['gini', 'entropy'],
              'splitter': ['best', 'random'],
              'max_depth': [2*n for n in range(1,10)],
              'max_features': ['sqrt', 'log2', None],
              'min_samples_leaf': [1, 2, 4],
              'min_samples_split': [2, 5, 10]}

         tree = DecisionTreeClassifier()
```

```
In [28]: tree_cv = GridSearchCV(tree, parameters, cv=10)
         tree_cv.fit(X_train, Y_train)

Out[28]: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
                      param_grid={'criterion': ['gini', 'entropy'],
                                  'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                                  'max_features': ['sqrt', 'log2', None],
                                  'min_samples_leaf': [1, 2, 4],
                                  'min_samples_split': [2, 5, 10],
                                  'splitter': ['best', 'random']})
```

```
In [32]: 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()
```

```
In [33]: knn_cv = GridSearchCV(KNN, parameters, cv=10)
         knn_cv.fit(X_train, Y_train)

Out[33]: 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]})
```

Evaluated the accuracy of the models

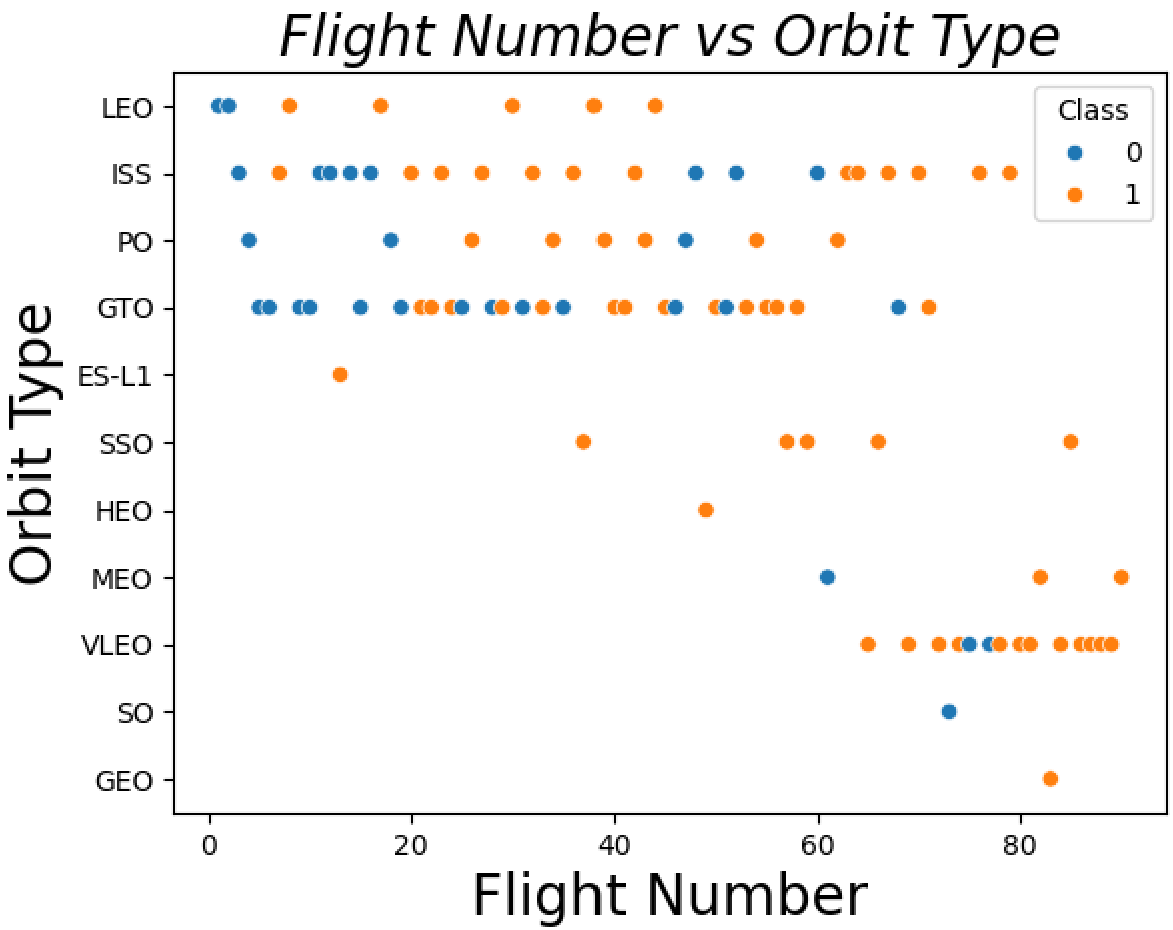
In [14]:
```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```
```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

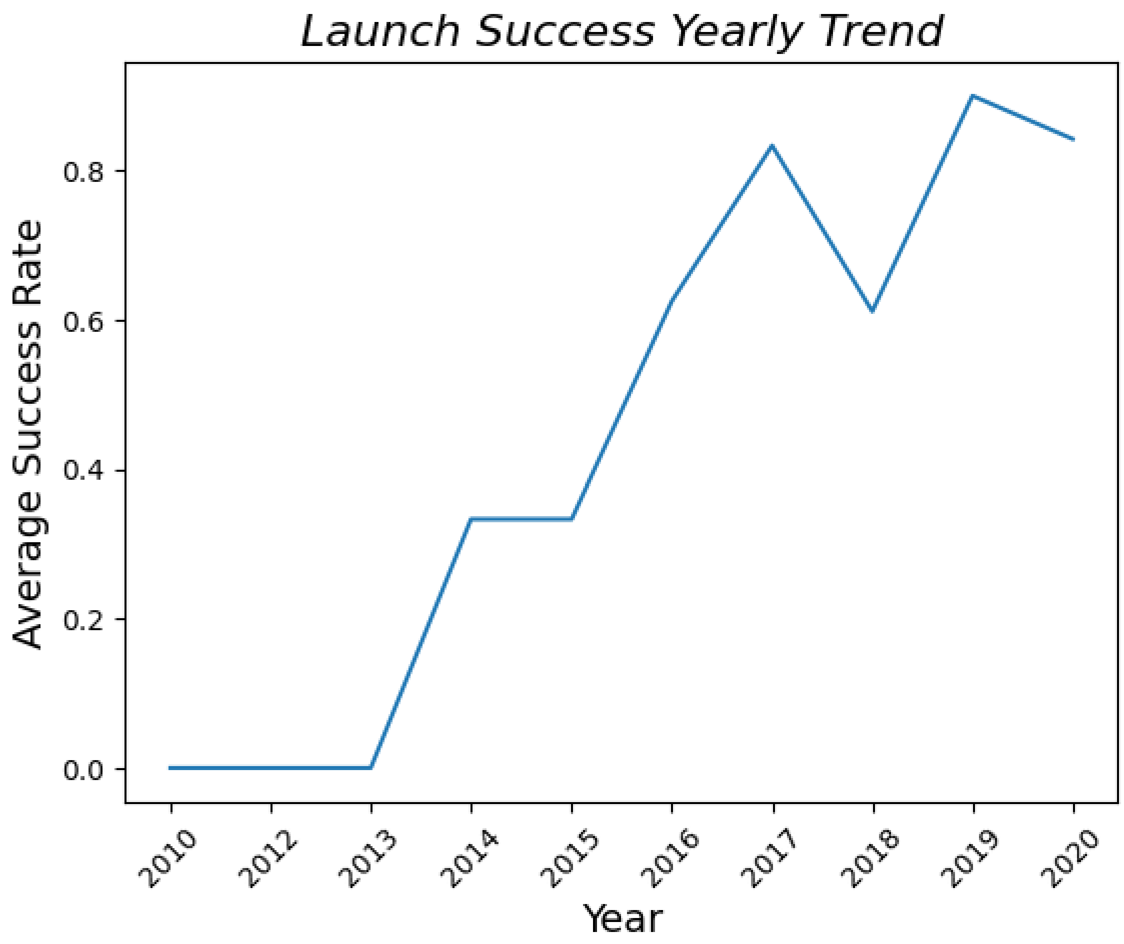
In [21]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```
```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

In [29]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```
```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 16, 'max_features': 'sqrt', 'mi
n_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.875
```

In [34]:
```
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
```

EDA with Visualization Results





EDA with SQL Results

In [17]:
```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number FROM SPACEXTABLE GROUP BY MISSION_OUTCOME
```
```
 * sqlite:///my_data1.db
Done.
```
Out[17]:

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

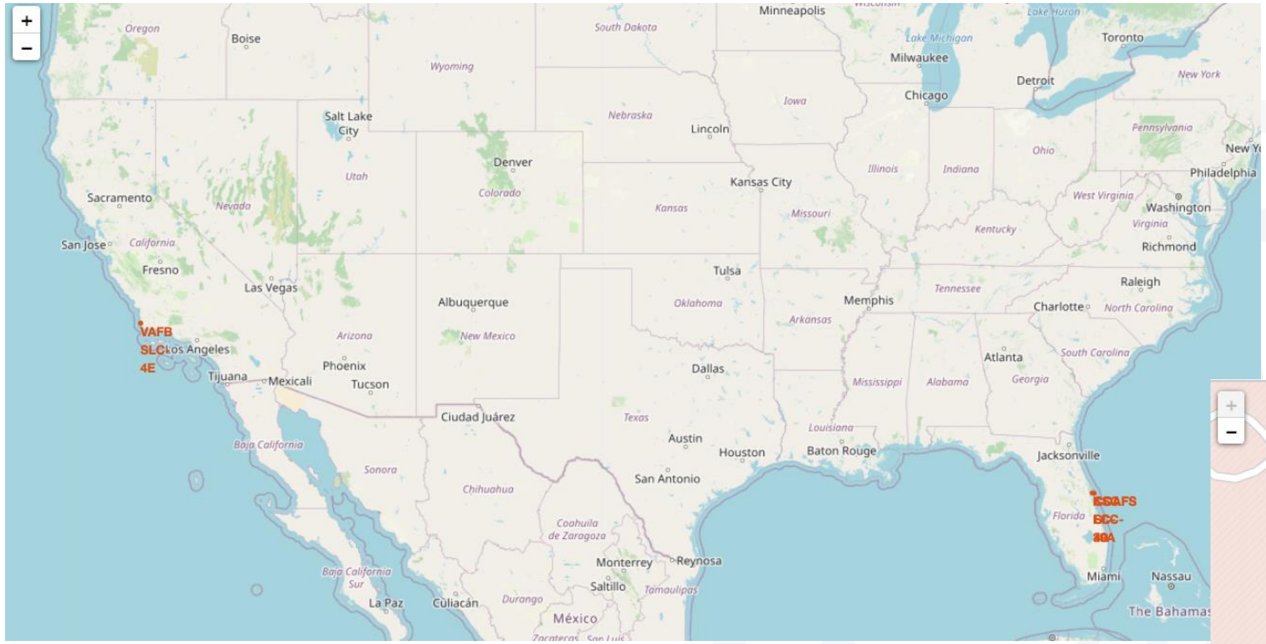
In [12]:
```
%sql SELECT * FROM SPACEXTABLE WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```
```
 * sqlite:///my_data1.db
Done.
```
Out[12]:

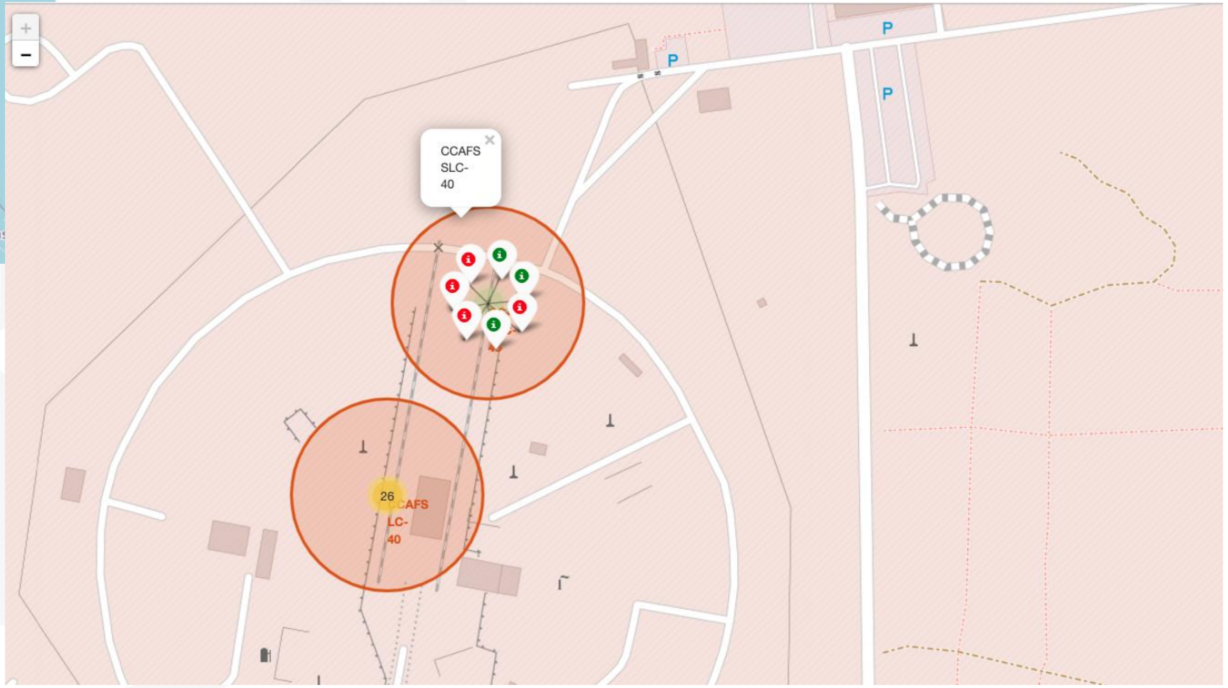
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outco
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Succ
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Succ
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Succ
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Succ
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Succ

In [15]:
```
%sql SELECT MIN(DATE) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'
```
```
 * sqlite:///my_data1.db
Done.
```
Out[15]:

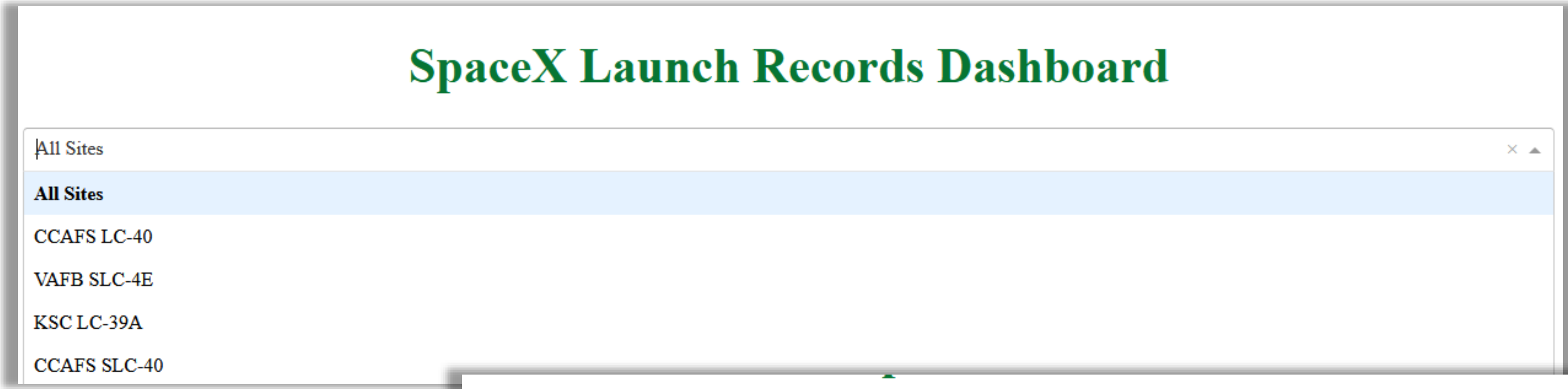
MIN(DATE)
2015-12-22

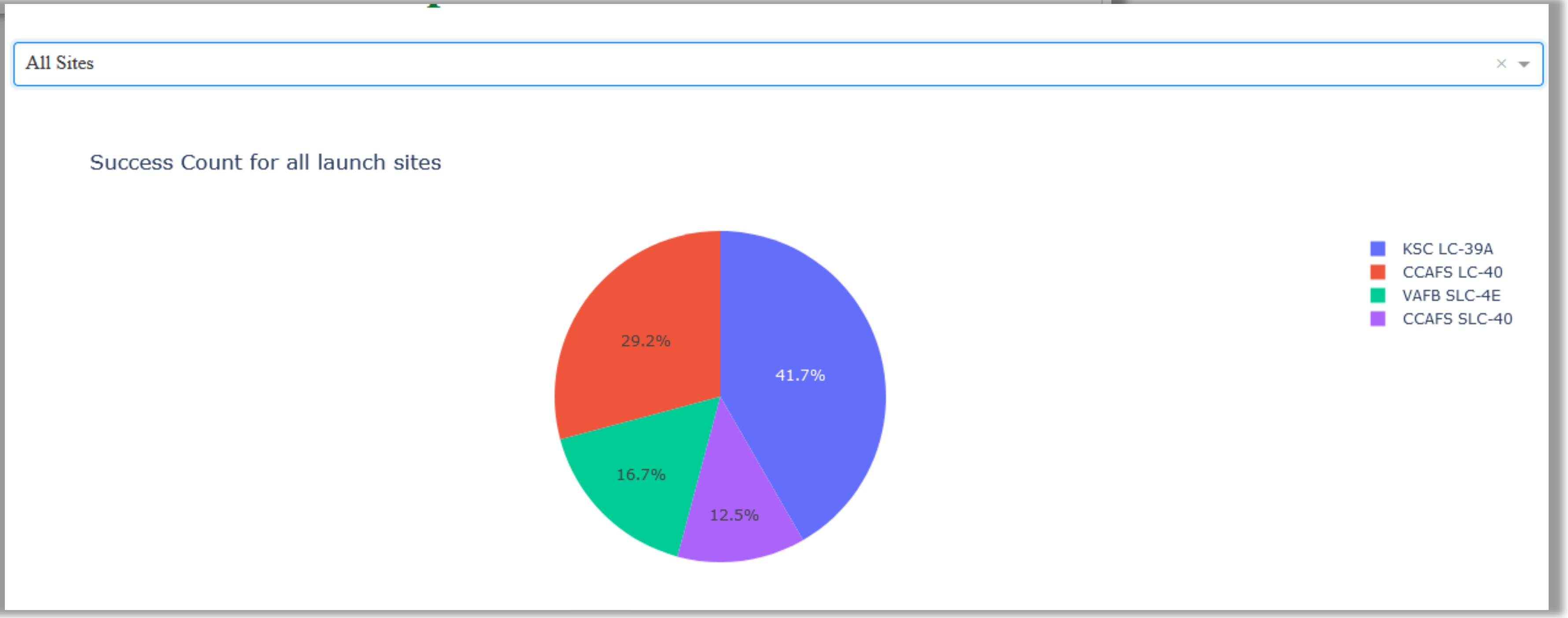
Interactive Map with Folium Results





Plotly Dash Dashboard Results





Plotly Dash Dashboard Results

Predictive Analysis

Model	Accuracy
Logistic Regression	0.846
Decision Tree	0.875
SVM	0.848
KNN	0.848

The results of the model comparison showed that the Decision Tree algorithm demonstrated the best accuracy in predicting successful SpaceX rocket landings with an accuracy of 87.5%. The Logistic Regression, SVM and KNN algorithms have similar results, around 84.6-84.8%, which indicates their reliability, but slightly lower accuracy compared to decision trees.

Decision Tree was found to be the most effective model for prediction, with an accuracy of 87.5%. This algorithm works well due to its ability to handle categorical variables and detect important patterns in the data.

Conclusion

- Key insights from the analysis:

The collected and analyzed data allowed us to identify key factors influencing the success of SpaceX rocket landings.

Deep analysis was performed using visualizations, SQL, and interactive tools, which helped better understand launch dynamics.

The best forecasting method turned out to be Decision Tree, which achieved 87.5% accuracy, confirming its effectiveness in similar tasks.

Interactive dashboards and maps significantly simplify the visualization of results for a non-technical audience.

- Meaning for SpaceX:

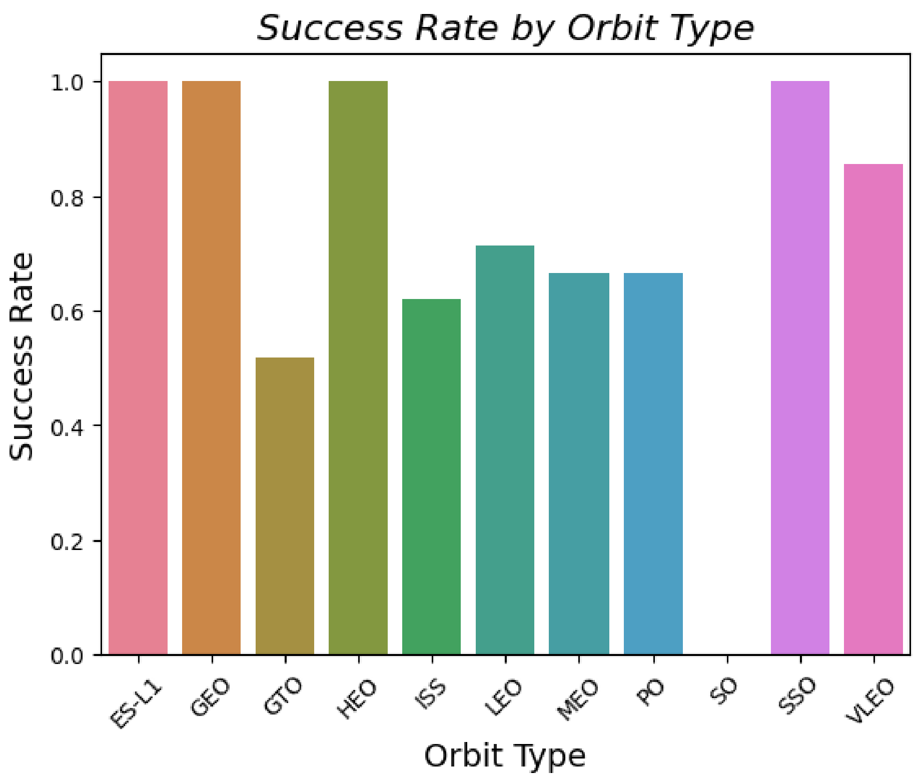
The results can help SpaceX improve launch planning and reduce the risk of unsuccessful landings.

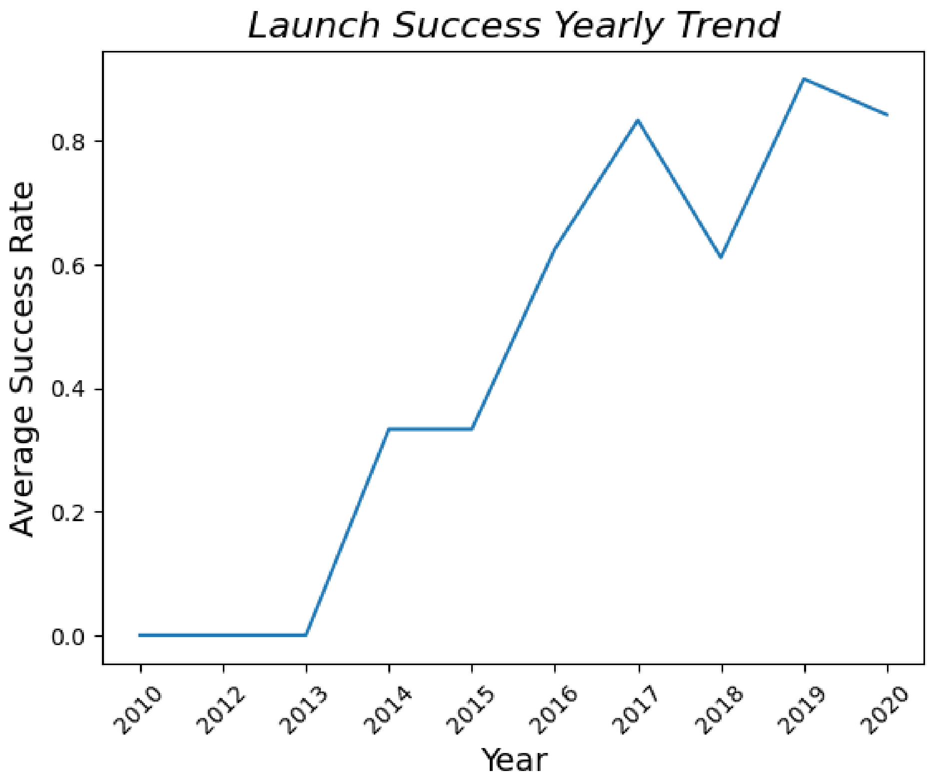
Identifying critical success factors will allow us to optimize mission preparation processes and test new technologies.

Forecasting and analysis provide a basis for automating decision-making and developing more sustainable engineering solutions.

Creativity & Insights

I decided to add color to some of the graphs and changed the font and angle a bit for better readability. This approach improved the overall look of the graphs and emphasized the attention to detail that is important for presenting complex information.





Thank you for your attention!