SpaceX-Falcon9

Reshma R

21-12-2022

SKILLS NETWORK

OUTLINE

XECUTIVE SUMMARY

Data science is the domain of study that deals with vast volumes of data using modern tools and techniques to find unseen patterns, derive meaningful information, and make business decisions. Data science uses machine learning algorithms to build predictive models.

The data used for analysis can come from many different sources and presented in various formats.

INTRODUCTION

In this project 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 dollar; other providers cost upward of 165 million dollar 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.

METHODOLOGY

Data Collection : The process of gathering and analyzing accurate data from various sources to find answers to research problems, trends and probabilities to evaluate possible outcomes.

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
```

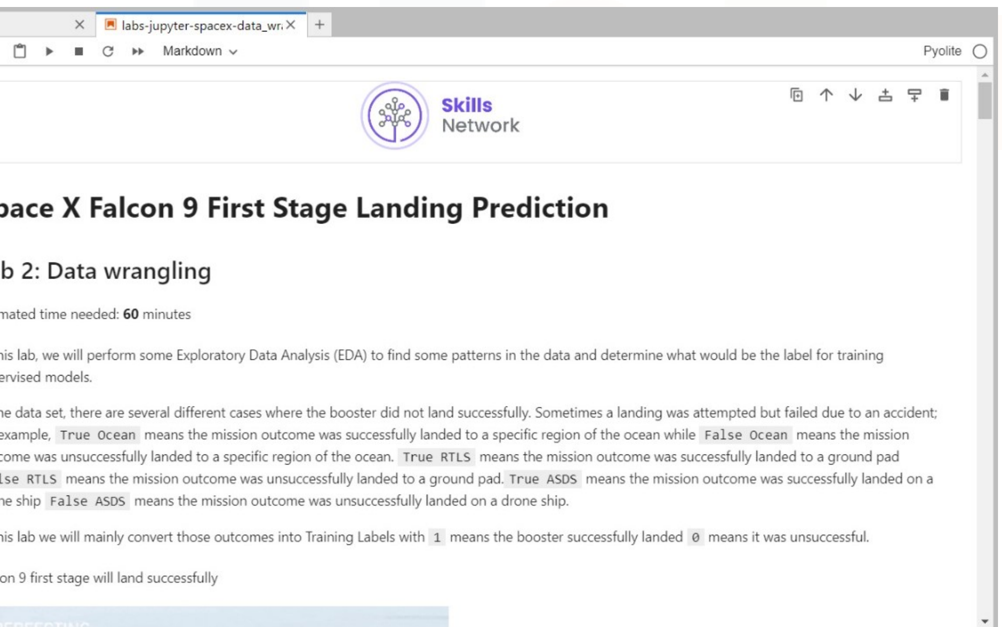
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

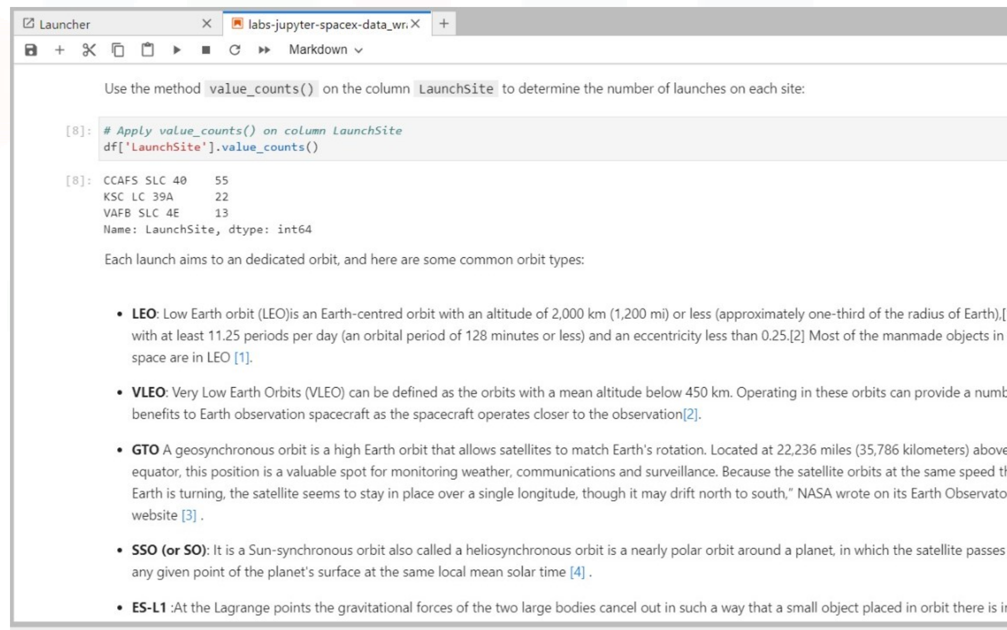
From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
```

METHODOLOGY

Data Wrangling: The process of removing errors and combining complex data to make them more accessible and easier to analyze.





Developer

SKILLS NETWORK

Methodology

Data Wrangling:

```
[18]: df.head(5)
```

[18]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	S
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B

We can use the following line of code to determine the success rate:

Methodology

EDA and interactive visual analytics:

Data scientists implement Exploratory Data Analysis(EDA) tools and techniques to investigate, analyze and summarize the main characteristics of the dataset often utilizing data visualization methodologies.

Interactive data visualization is the use of tools and processes to produce a visual representation of data which can be explored and analyzed directly within the visualization itself.

Methodology

Predictive Ananlysis: Predictive analytics is a branch of advanced analytics that makes predictions about future outcomes using historical data combined with statistical modeling, data mining techniques and machine learning.

Predictive analytics models are designed to assess historical data, discover patterns, observe trends, and use that information to predict future trends.

EDA with Visualization Results

ory Data Analysis

the SpaceX dataset into a Pandas dataframe and print its summary

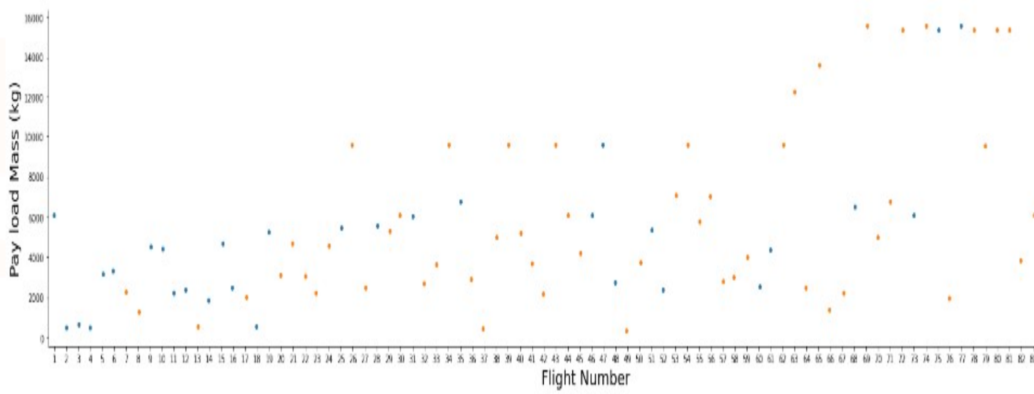
```
csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")

e unable to complete the previous lab correctly you can uncomment and load this csv

_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_pa
```

er	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	La
1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.5
2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.5
3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.5
4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.6
5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.5

```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



We see that different launch sites have different success rates. `CCAFS LC-40`, has a success rate of 60 %, while `KSC LC-39A` and `VAFB SLC 4E` has a s 77%.

Next, let's drill down to each site visualize its detailed launch records.

Developer

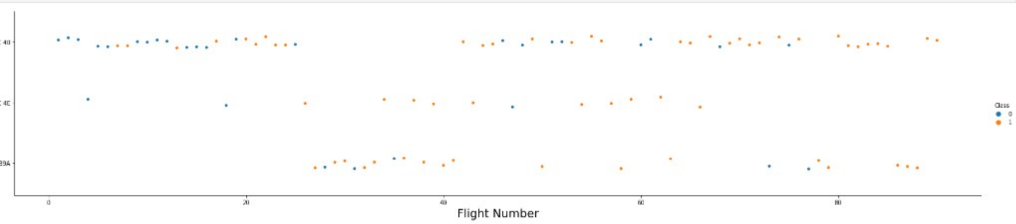
SKILLS NETWORK

EDA with Visualization Results

K 1: Visualize the relationship between Flight Number and Launch Site

e function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`, set the `y` to `Launch Site` and set the
eter `hue` to `'class'`

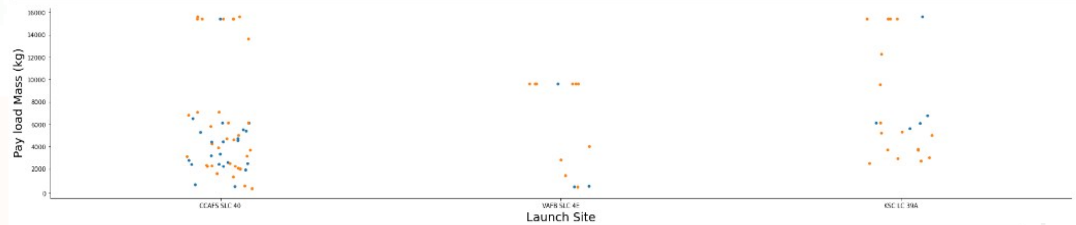
```
t a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
atplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
label("Flight Number",fontsize=20)
label("Launch Site",fontsize=20)
how()
```



TASK 2: Visualize the relationship between Payload and Launch Site

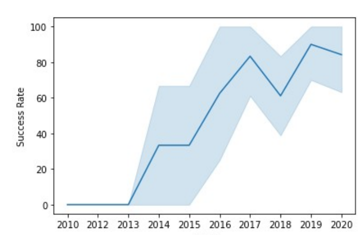
We also want to observe if there is any relationship between launch sites and their payload mass.

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="PayloadMass", x="LaunchSite", hue="Class", data=df, aspect = 5)
plt.xlabel("Launch Site",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```




```
# A function to Extract years from the date
def Extract_year(year):
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
```

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
year = []
df["year"] = Extract_year(year)
df["Success Rate"] = df["Class"] * 100
sns.lineplot(data = df, x = "year", y = "Success Rate")
```



you can observe that the success rate since 2013 kept increasing till 2020

TASK 3: Visualize the relationship between success rate of each orbit type

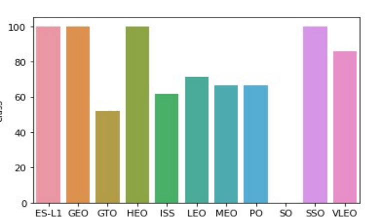
ext, we want to visually check if there are any relationship between success rate and orbit type.

et's create a `bar chart` for the sucess rate of each orbit

```
# HINT use groupby method on Orbit column and get the mean of Class column
temp = df.groupby(["Orbit"]).mean().reset_index()
temp2 = temp[["Orbit", "Class"]]
temp2["Class"] = temp2["Class"]*100
sns.barplot(x = "Orbit", y = "Class", data = temp2)
```

```
4: SettingWithCopyWarning:
 value is trying to be set on a copy of a slice from a DataFrame.
ry using .loc[row_indexer,col_indexer] = value instead

ee the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
 temp2["Class"] = temp2["Class"]*100
```



EDA with SQL Results

ames of the unique launch sites in the space mission

```
: DISTINCT LAUNCH_SITE from SPACEXDATASET
```

a://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB

ords where launch sites begin with the string 'CCA'

```
: * from SPACEXDATASET where launch_site like 'CCA%' limit 5
```

a://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB

e_utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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)
07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
00:35:00	F9 v1.0 B0006	CCAFS LC-	SpaceX CRS-1	500	LEO	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 sum from SPACEXDATASET where customer like 'NASA (CRS)'
```

* ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.

SUM
45596

Task 4

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

```
%sql select avg(payload_mass__kg_) as Average from SPACEXDATASET where booster_version like 'F9 v1.1%'
```

* ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.

average
2534

EDA with SQL Results

he first succesful landing outcome in ground pad was acheived.

n

```
late) as Date from SPACEXDATASET where mission_outcome like 'Success'
```

```
s27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
```

e boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
ter_version from SPACEXDATASET where (mission_outcome like 'Success')
_kg_ BETWEEN 4000 AND 6000) AND (landing__outcome like 'Success (drone ship)')
```

```
s27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
```


hich will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015

```
THNAME(DATE) as Month, landing__outcome, booster_version, launch_site
SET where DATE like '2015%' AND landing__outcome like 'Failure (drone ship)'
```

```
xs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
```

outcome	booster_version	launch_site
one ship)	F9 v1.1 B1012	CCAFS LC-40
one ship)	F9 v1.1 B1015	CCAFS LC-40

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT mission_outcome, count(*) as Count FROM SPACEXDATASET GROUP by mission_outcome ORDER BY mission_outcome
```

```
 * ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

mission_outcome	COUNT
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Task 8

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

```
maxm = %sql select max(payload_mass__kg_) from SPACEXDATASET
maxv = maxm[0][0]
%sql select booster_version from SPACEXDATASET where
payload_mass__kg_=(select max(payload_mass__kg_) from SPACEXDATASET)
```

```
 * ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
 * ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

booster_version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3

Task 10

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

```
%sql select landing__outcome, count(*) as count from SPACEXDATASET
where Date >= '2010-06-04' AND Date <= '2017-03-20'
GROUP by landing__outcome ORDER BY count Desc
```

```
 * ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

landing__outcome	COUNT
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

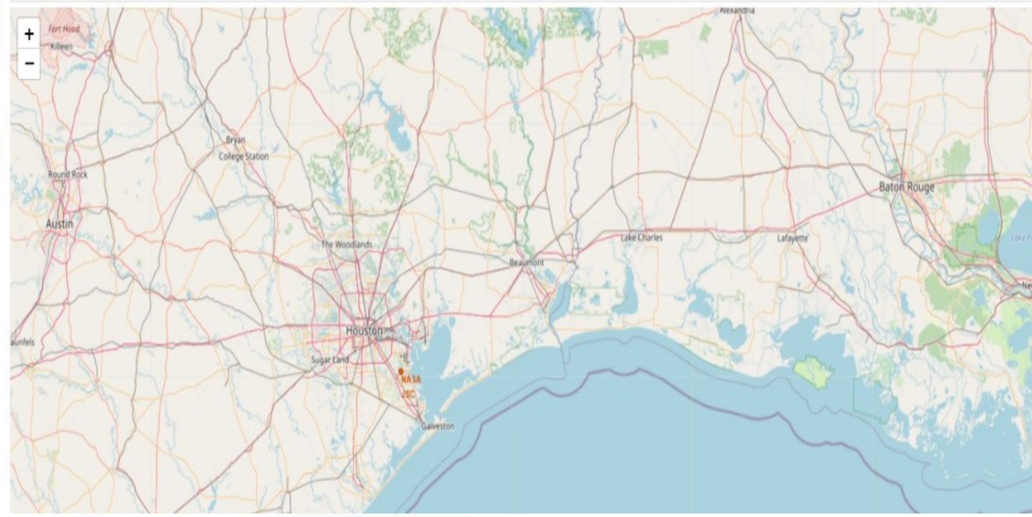
Interactive map with Folium Results

```
t sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
ex_df[['Launch Site', 'Lat', 'Long', 'class']]
= spacex_df.groupby(['Launch Site'], as_index=False).first()
= launch_sites_df[['Launch Site', 'Lat', 'Long', 'class']]
```

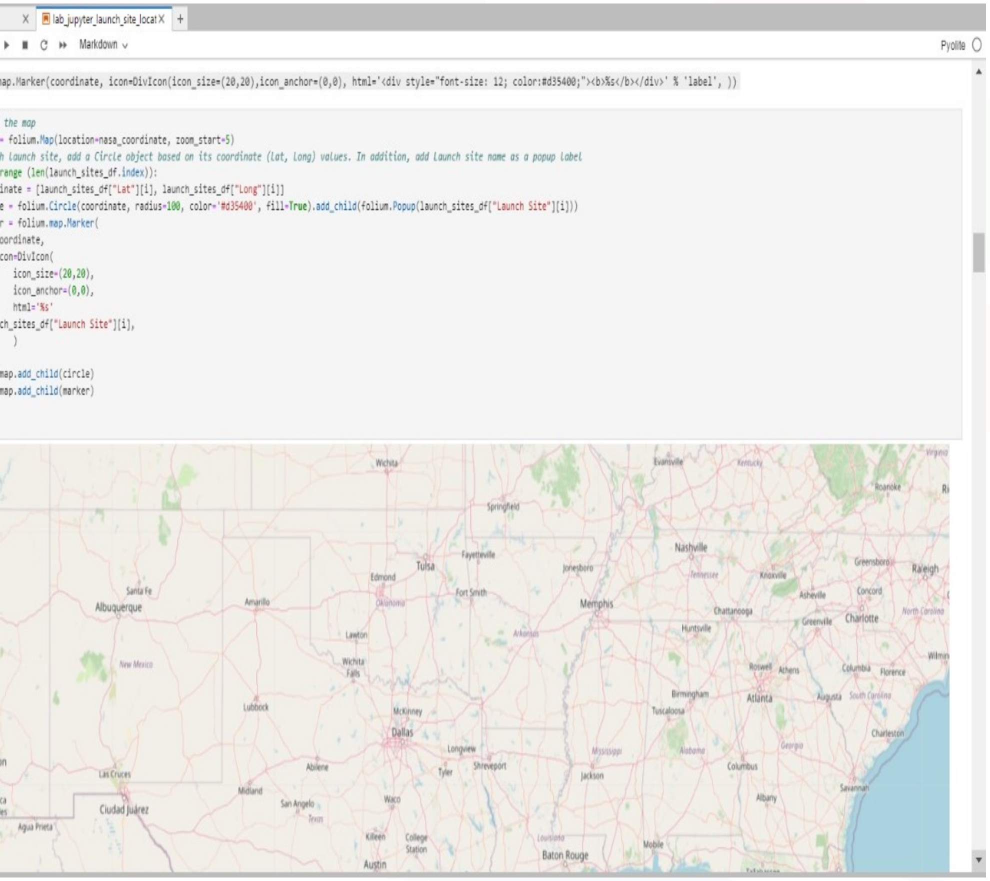
Lat	Long	class
.562302	-80.577356	0
.563197	-80.576820	1
.573255	-80.646895	1
.632834	-120.610746	0

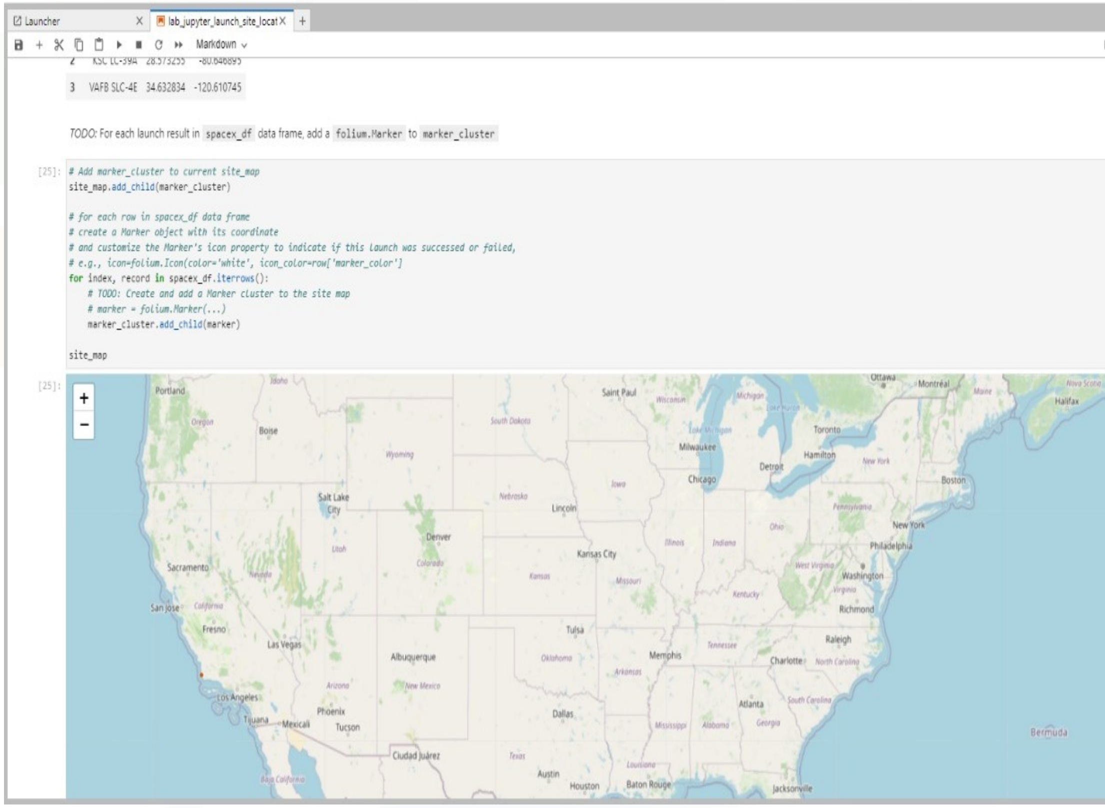
We could use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate. For example,

```python
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```



Interactive map with Folium results

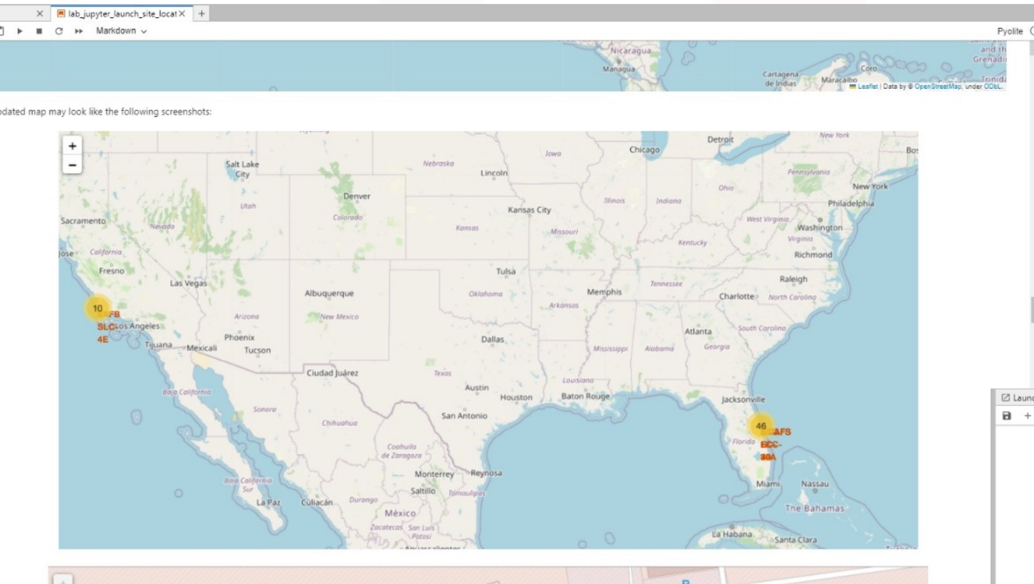


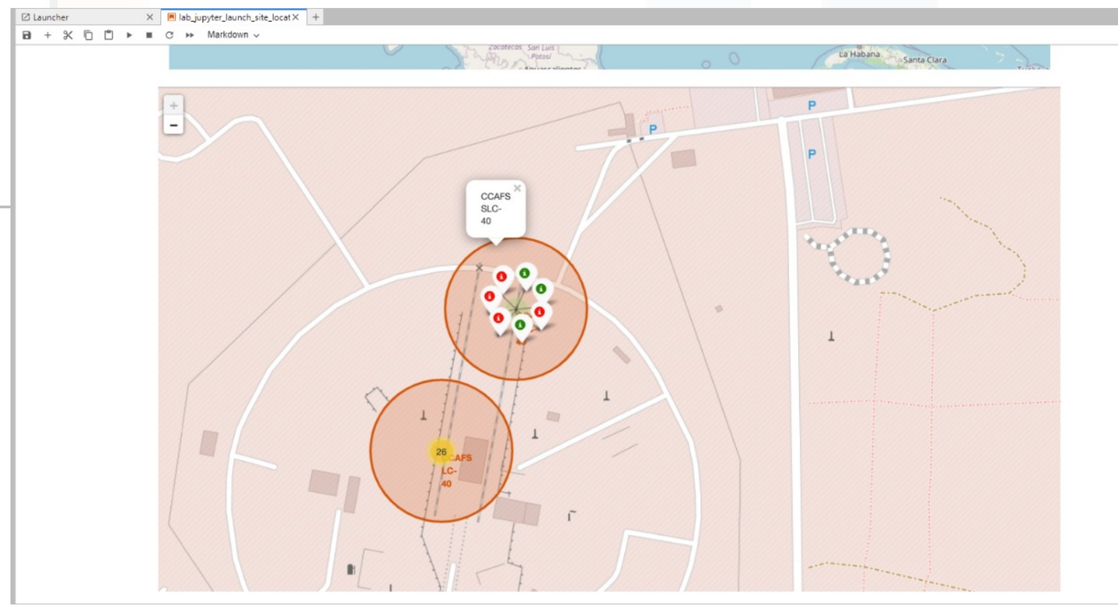


Developer

SKILLS NETWORK

Interactive map with Folium results

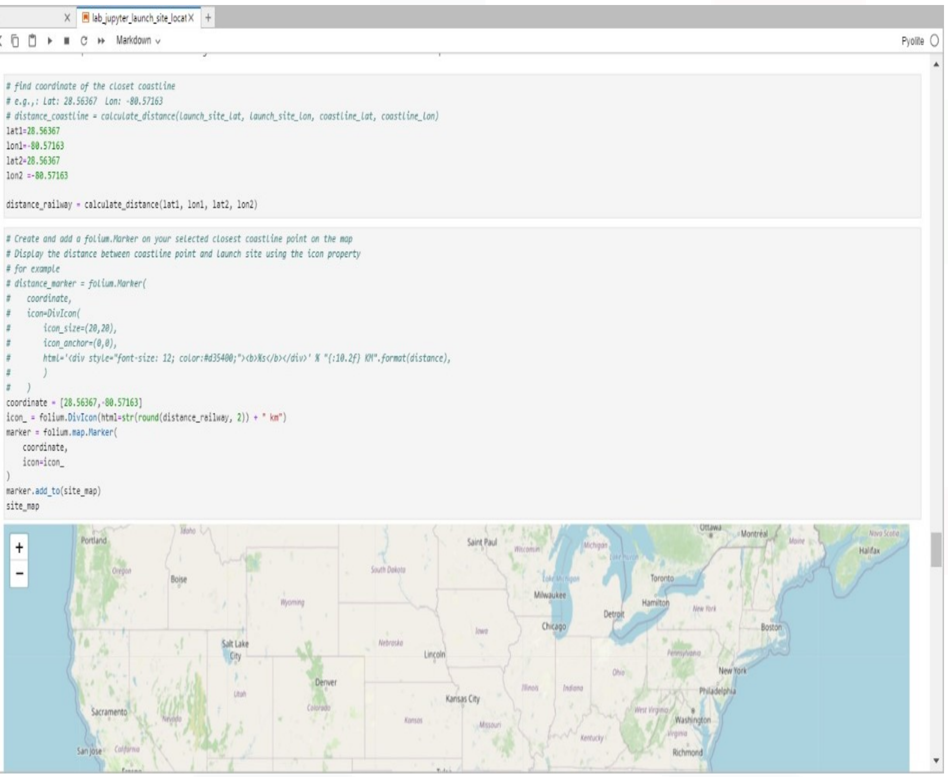


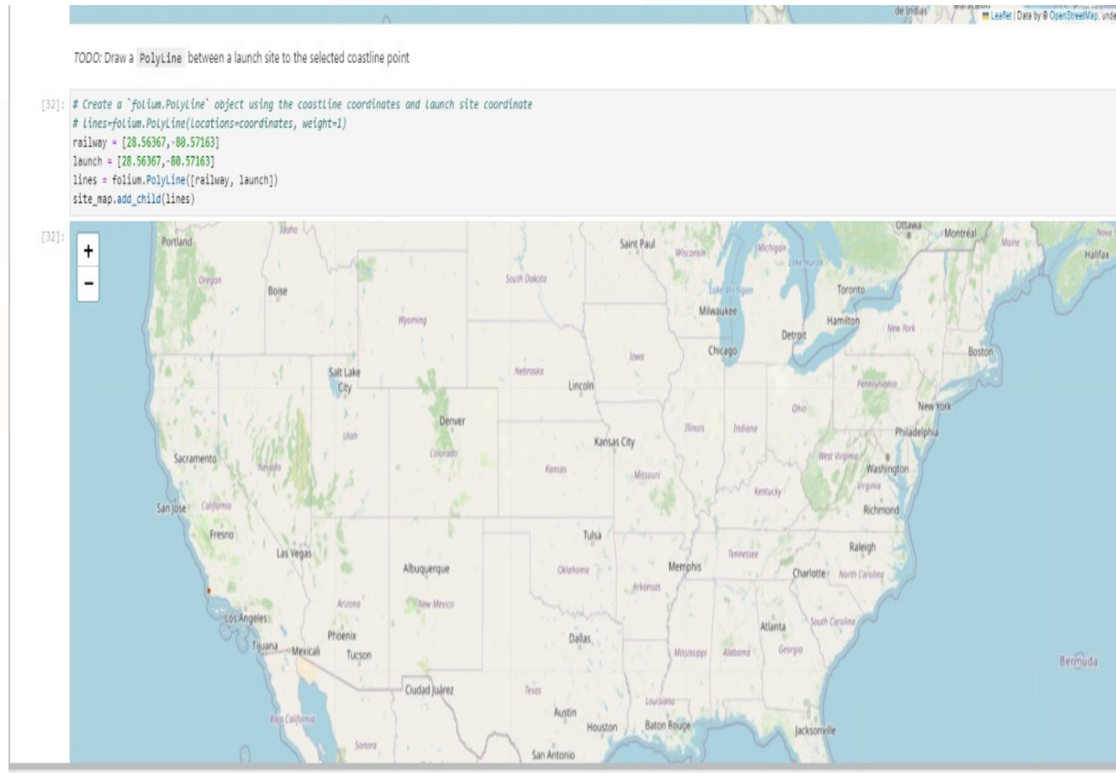


Developer

SKILLS NETWORK

Interactive map with Folium results

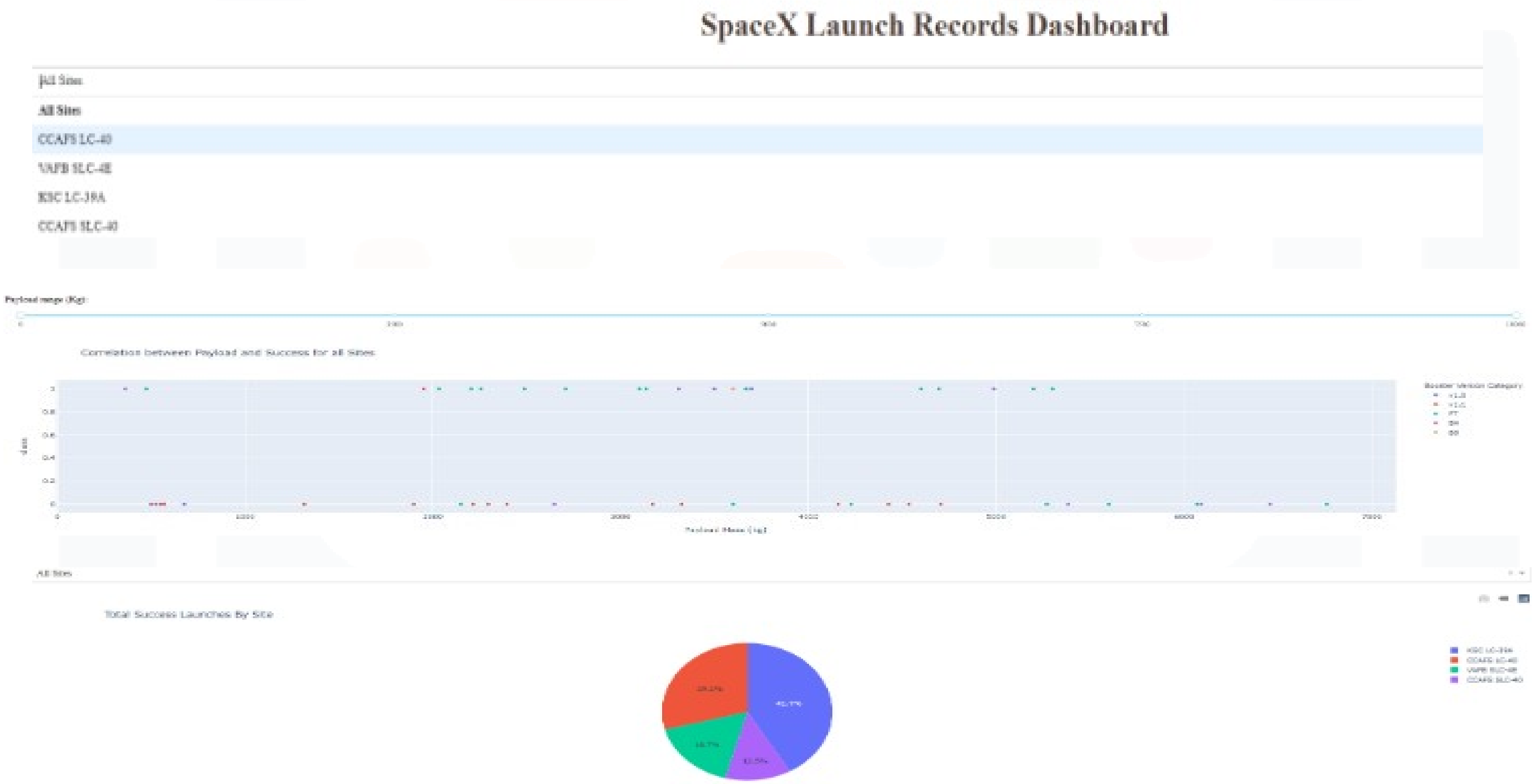




Developer

SKILLS NETWORK

Plotly dashboard results



Developer SKILLS NETWORK

Predictive Analysis Results

uncher | SpaceX_Machine_Learning_P

Markdown | Pyolite

[9]:
```python
# students get this
transform = preprocessing.StandardScaler()
x_scaled = transform.fit_transform(X)
X = pd.DataFrame(x_scaled)
X
```

[9]:

	0	1	2	3	4	5	6	7	8	9	...	73	74	
0	-1.712912	-1.948145e-16	-0.653913	-1.575895	-0.973440	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666	-0.18
1	-1.674419	-1.195232e+00	-0.653913	-1.575895	-0.973440	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666	-0.18
2	-1.635927	-1.162673e+00	-0.653913	-1.575895	-0.973440	-0.106	-0.106	-0.654654	-0.106	1.812654	...	-0.185695	-0.215666	-0.18
3	-1.597434	-1.200587e+00	-0.653913	-1.575895	-0.973440	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666	-0.18
4	-1.558942	-6.286706e-01	-0.653913	-1.575895	-0.973440	-0.106	-0.106	1.527525	-0.106	-0.551677	...	-0.185695	-0.215666	-0.18
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
85	1.558942	1.991005e+00	0.174991	0.945537	0.202528	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666	5.38
86	1.597434	1.991005e+00	1.003894	0.945537	0.202528	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	5.385165	-0.215666	-0.18
87	1.635927	1.991005e+00	3.490605	0.945537	1.966480	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666	-0.18
88	1.674419	1.991005e+00	1.003894	0.945537	0.202528	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666	5.38
89	1.712912	-5.192140e-01	-0.653913	0.945537	-0.973440	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666	-0.18

90 rows × 83 columns

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 to 0.2 and random_
2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

```python
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.

```python
Y_test.shape
```

```
(18,)
```

Predictive Analysis Results

4

ogistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters dictionary `parameters`.

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

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

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

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

```
ned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
curacy :",logreg_cv.best_score_)
```

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

Launcher | SpaceX_Machine_Learning_P | Markdown

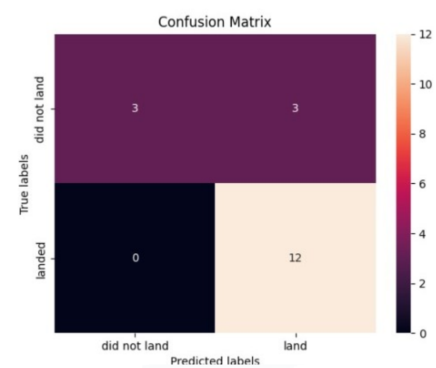
Calculate the accuracy on the test data using the method `score`:

```
[15]: accu=[]
methods=[]
accu.append(logreg_cv.score(X_test,Y_test))
methods.append('logistic regression')
logreg_cv.score(X_test,Y_test)
```

```
[15]: 0.8333333333333334
```

Lets look at the confusion matrix:

```
[16]: yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
plt.show()
```



Developer

SKILLS NETWORK

Predictive Analysis Results

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

```
ernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
': np.logspace(-3, 3, 5),
amma':np.logspace(-3, 3, 5)}
```

```
rchCV(svm, parameters, cv = 10)
in, Y_train)
```

```
10, estimator=SVC(),
am_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
0e+03]),
            'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
0e+03]),
            'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

```
erparameters :(best parameters) ",svm_cv.best_params_)
:",svm_cv.best_score_)
```

```
eters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
2142857142856
```

racy on the test data using the method `score` :

```
cv.score(X_test,Y_test))
support vector machine')
est,Y_test)
```

```
34
```

nfusion matrix

TASK 7

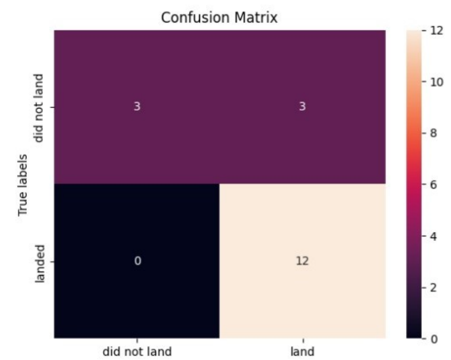
Calculate the accuracy on the test data using the method `score` :

```
accu.append(svm_cv.score(X_test,Y_test))
methods.append('support vector machine')
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)
plt.show()
```



Developer

SKILLS NETWORK

Predictive Analysis Results

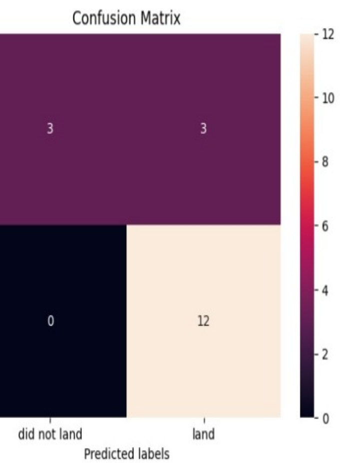
racy of tree_cv on the test data using the method `score` :

```
e_cv.score(X_test,Y_test))
'decision tree classifier')
_test,Y_test)
```

444

onfusion matrix

```
redict(X_test)
matrix(Y_test,yhat)
```



TASK 10

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` .

```
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 Results

TASK 12

Find the method performs best:

```
print(methods)
print(accu)
```

```
['logistic regression', 'support vector machine', 'decision tree classifier', 'k nearest neighbors']
[0.8333333333333334, 0.8333333333333334, 0.9444444444444444, 0.8333333333333334]
```

Authors

Conclusion

Thus the models have been built using the SpaceX data after performing exploratory data analysis and also visualizing the transformed data. The machine learning models were built(classification) algorithms such as Logistic regression, support vector machine, decission tree classifier and k-NN algorithm and we obtained an accuracy of 83.34%.i.e., the launch of the first stage of the spaceX falcon9.