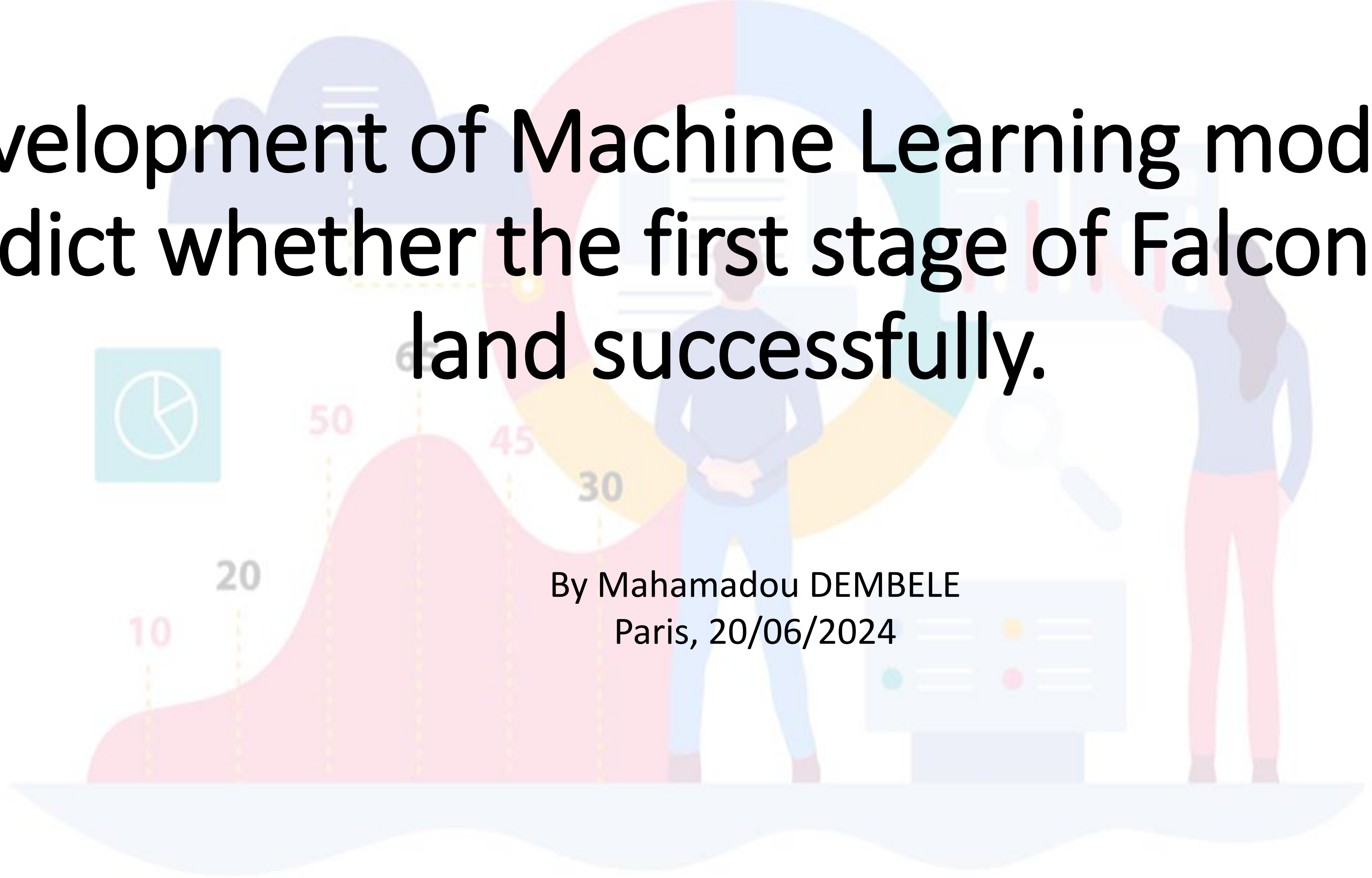


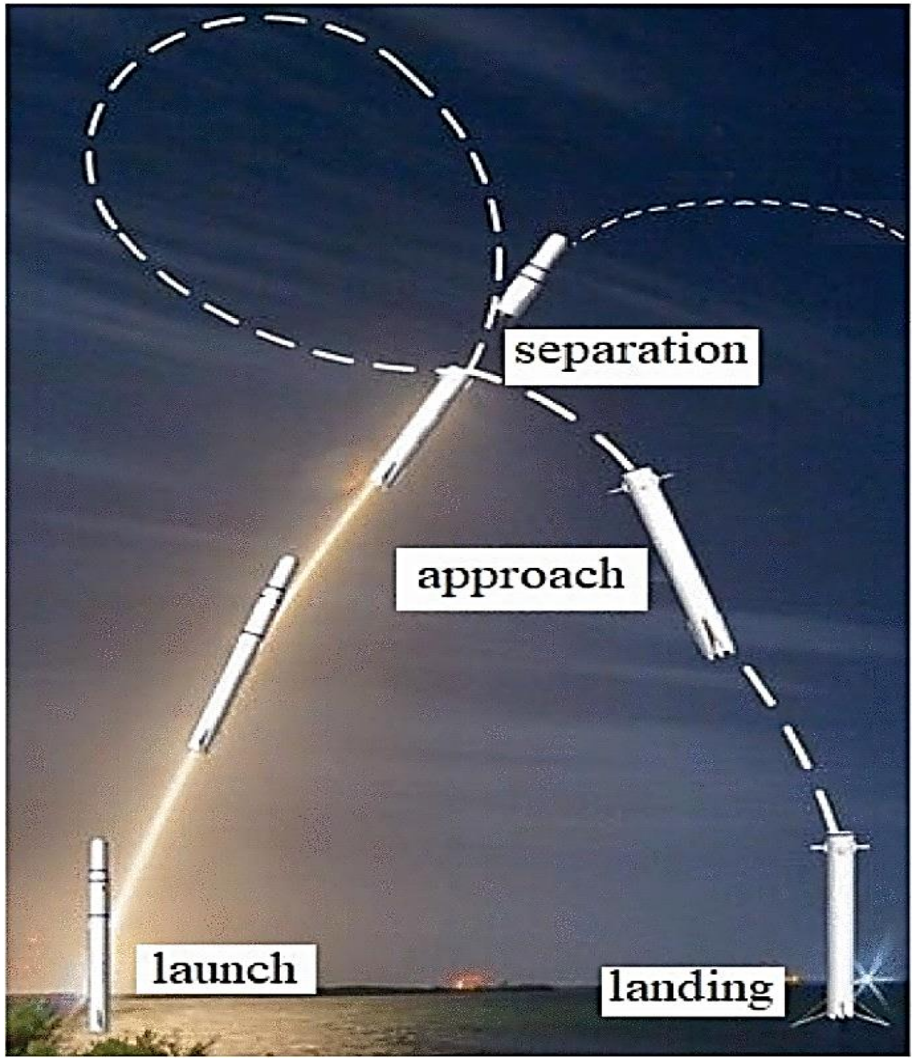
Development of Machine Learning models to predict whether the first stage of Falcon 9 will land successfully.

By Mahamadou DEMBELE
Paris, 20/06/2024

Summary

- According to SPACEX, the Falcon 9 rocket launches cost $62 million;
- other providers cost more than $165 million each, with much of the savings due to SpaceX being able to reuse the first stage.
- Therefore, if we can determine whether the first stage will land, we can determine the cost of a launch.
- This information can be used if another company wants to bid against SpaceX for a rocket launch.
- In this work, I developed different prediction algorithms that can be used to predict whether the Falcon 9 first stage will land successfully.

Introduction



- In this work, I developed **models of machine learning** to **predic**t whether the Falcon 9 first stage **will land successfully** :

- **Logistic regression :** A statistical model for studying the relationships between a set of qualitative variables Xi and a qualitative variable Y. It is a generalized linear model using a logistic function as a link function.

- **SVM (Support Vector Machine):** SVMs are a family of machine learning algorithms that solve classification, regression or anomaly detection problems.

- **Decision Tree** : A non-parametric supervised learning algorithm, which is used for both classification and regression tasks

- **Algorithm of K closest neighbours :** also known as KNN or k-NN, is a non-parametric supervised learning discriminant, which uses proximity to perform classifications or predictions on the clustering of an individual data point

Material and Methods

Step 1

- Collecting the data
- Data Wrangling

Step 2

- Analysis with SQL, pandas and Matplolib
- Visual analytics

Step 3

- Machine learning models development

Results

STEP1 : Collecting and Wrangling data


```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

```
data = requests.get(static_json_url)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.047721
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.047721
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.047721
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.047721
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89	102	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1060	-80.603956	28.608058
90	103	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	13	B1058	-80.603956	28.608058
91	104	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1051	-80.603956	28.608058

Results

STEP2 : Analysis with SQL, pandas and Matplolib a

```
%sql select * from SPACEXTBL
```

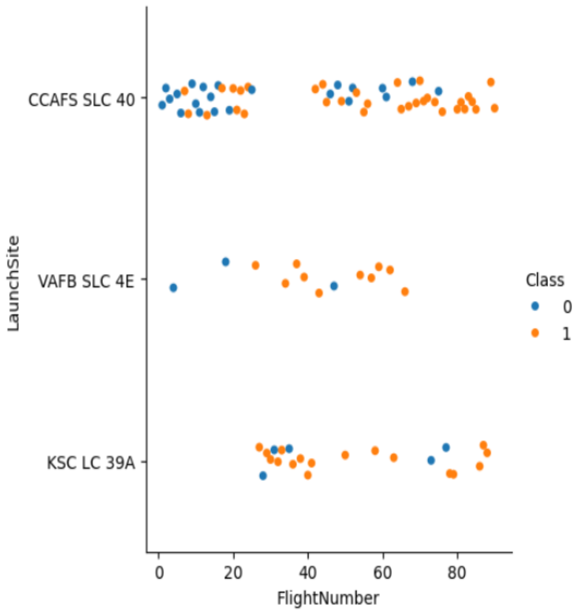
* sqlite:///my_data1.db
Done.

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
2013-09-29	16:00:00	F9 v1.1 B1003	VAFB SLC-4E	CASSIOPE	500	Polar LEO	MDA	Success	Uncontrolled (ocean)

Would you like to receive official Jupyter

```
# Plot 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
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 1)
```

<seaborn.axisgrid.FacetGrid at 0x7e3d848>



Results

STEP2 : Visual analytics folium

```
from folium import plugins
```

```
# Task 2: Mark the success/failed launches for each s
a = folium.Map(location=[28.562302,-80.577356], zoom_start=10)
launch_sites = folium.map.FeatureGroup()
launch_sites = plugins.MarkerCluster().add_to(a)



for lat, lon, label  in zip(spacex_df.Lat, spacex_df.Long, spacex_df["class"]):
    folium.Marker(
        location=[lat, lon],
        popup=label,
        icon=None).add_to(launch_sites)

a
```


```
# TASK 3: Calculate the distances between a launch site to its proximities
```

Results

STEP3 : Logistic regression model development

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

▾ Régression logistique

Régression logistique()

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
#"RR = Ridge()
logreg_cv = GridSearchCV(lr,parameters, cv=10).fit(X_train, Y_train)
```

TÂCHE 5

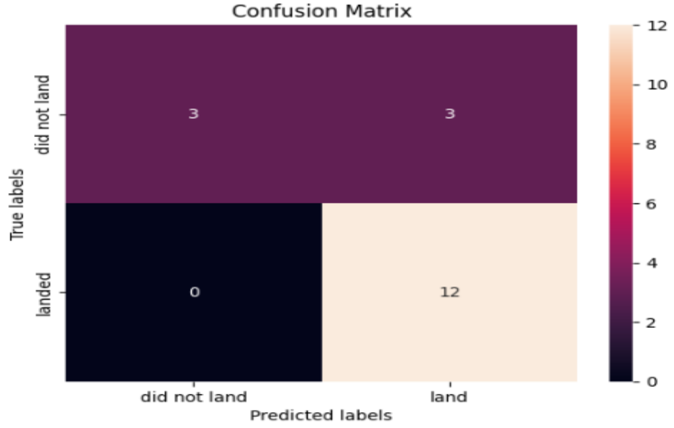
Calculez la précision sur les données de test en utilisant la méthode `score` :

```
BestRR=logreg_cv.best_estimator_
print(BestRR.score(X_test, Y_test))
```

0,8333333333333334

Regardons la matrice de confusion :

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



Results

STEP3 : SVM (Support Vector Machine) model development

Créez un objet machine à vecteurs de support puis créez un `GridSearchCV` objet `svm_cv` avec cv = 10. Ajustez l'objet pour trouver les m

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

```
svm_cv = GridSearchCV(svm, parameters, cv=10).fit(X_train, Y_train)
```

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

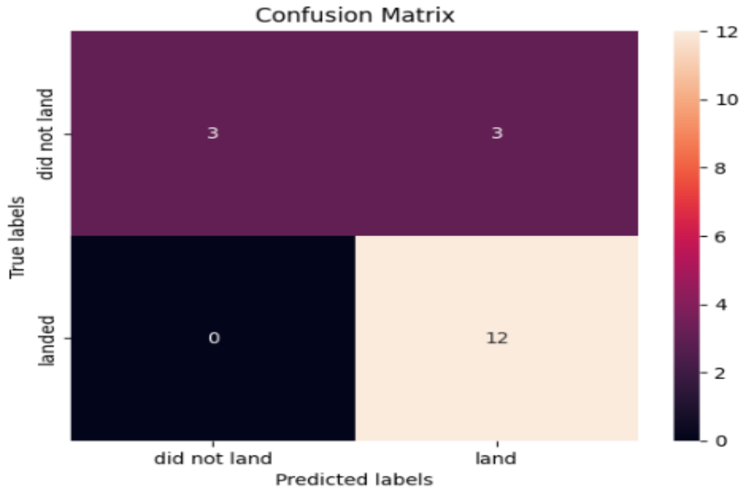
```
paramètres hpyer réglés : (meilleurs paramètres) {'C' : 1,0, 'gamma' : 0,03162277660168379, 'noyau' : 'sigmoïde'}
précision : 0.8482142857142856
```

```
BestRR2=svm_cv.best_estimator_
print(BestRR2.score(X_test, Y_test))
```

```
0,8333333333333334
```

Nous pouvons tracer la matrice de confusion

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



Results

STEP3 : Decision Tree model development

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the

```
import warnings
warnings.filterwarnings("ignore")
from sklearn.exceptions import FitFailedWarning
```

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

tree = DecisionTreeClassifier()
```

```
grid_search = GridSearchCV(tree, parameters, cv=10)
tree_cv= grid_search.fit(X_train, Y_train)
```

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

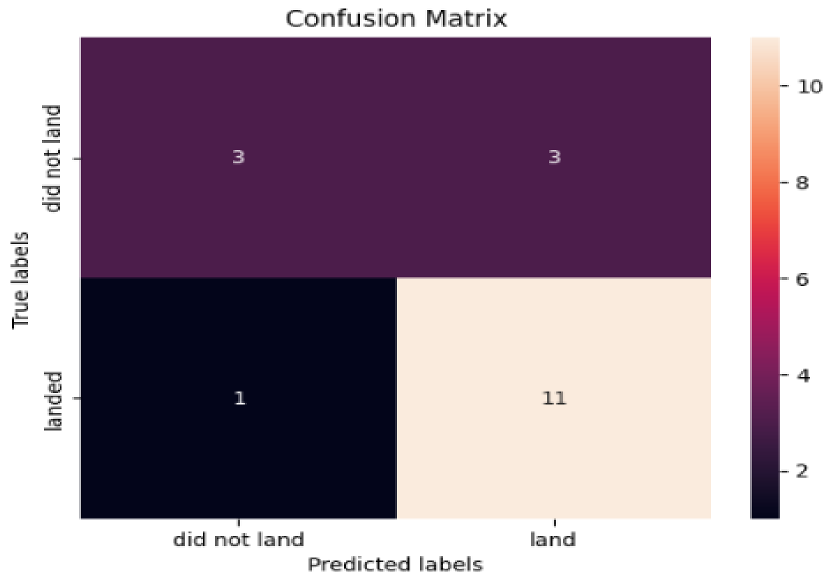
```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 2, 'max_features': 'sqrt', '
accuracy : 0.8339285714285714
```

```
BestRR3=tree_cv.best_estimator_
print(BestRR3.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)
```



Results

STEP3 : Algorithm of K closest neighbours model development

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

```
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()
```

```
grid_search = GridSearchCV(KNN, parameters, cv=10)
knn_cv= grid_search.fit(X_train, Y_train)
```

```
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
```

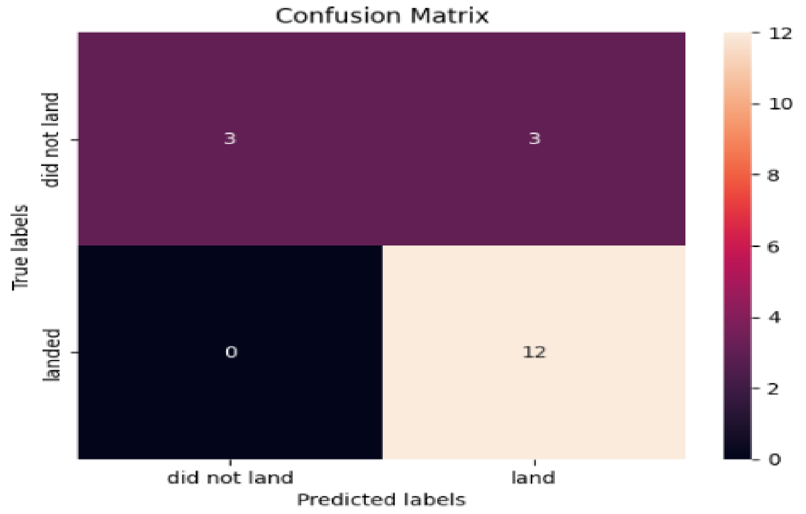
Calculate the accuracy of knn_cv on the test data using the method `score` :

```
BestRR4=knn_cv.best_estimator_
print(BestRR4.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)
```



Discussion

- In this work we collected data on the spaceX site
- We analyzed them to observe the tandaces for concerning the successful landing of the first stage of falcon9.
- We have also developed four machine models to predict whether the first stage of falcon 9 will land successfully
- All these models showed a prediction score of over 75%.

Conclusion

- All agorithms appear to be good models for predicting the outcome of the falcon 9 landing.
- However, the SVM has a relatively higher prediction score than the others.
- These models still need to be trained on a large data set to improve their performance.

Appendices

Sources :
SPACEX: https://www.spacex.com/vehicles/falcon-9/