Naphon Santisukwongchot

Profile summary

Student

Thammasat business school

Business administration : Finance

Aug 2017 - May 2021

Present

Associate account manager

N-Squared eCommerce, Bangkok

Oct 2021 - May 2023

Personal statement

I became interested in data science through self-learning and working on personal projects using Excel, SQL, R and Python. These allowed me to build strong analytical and technical skills. Currently, I want to pursue a BSc. Data Science to deepen my knowledge, to build a strong academic foundation, to improve my practical abilities, and to prepare for a professional career.

Technical strengths

Business Intelligence :	Looker, Power BI, Tableau
Data Analysis :	Pandas, NumPy
Data Visualization :	Matplotlib, Seaborn
Machine Learning :	Scikit-Learn
Microsoft Office :	Excel, PowerPoint, Word
Programming :	Python, R, SQL

Project 1
Churn Rate prediction

Churn Rate prediction (1)

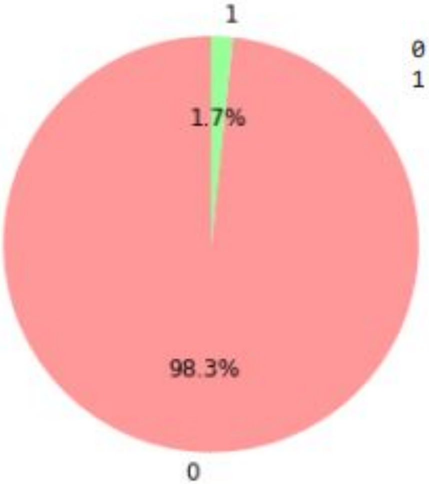
Overview

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('customer_dataset.csv')
df_postpaid_loyalty = df[(df['ACCT_TYPE']=='Postpaid') & (df['AOU_DAY']>1095)]
df_postpaid_loyalty.head()
```

```python
print(df_fs.isnull().sum())
```

```
CURR_MAIN_PKG_FEE     0
AOU_DAY               0
AOU_DVC               0
DVC_GRP               0
DVC_CLASS             0
DVC_SUPPORT           0
MOST_USED_4W_REGION   0
DTAC_RWRD_SEGMENT     0
Churn_Flag            0
REVENUE               0
AVG3M_REVENUE         0
VC_DOM_MOU            0
VC_DOM_CNT            0
AVG3M_VC_DOM_MOU      0
AVG3M_VC_DOM_CNT      0
DATA_MB               0
AVG3M_DATA_MB         0
CIN_CALLCNT           0
VOC_ACTIVEDAY         0
DATA_ACTIVEDAY        0
PM_PMMTHDCOMMON       0
PCT_CALL_DROP         0
dtype: int64
```




```
0    175494
1      3009
     178503
```

Company target

- ◊ Churn rate = 1.70%
- ◊ Retention rate = 80%

In **Postpaid loyalty customers (3 years)**

Data preparation

- ◊ Importing frameworks
- ◊ **Filter relevant data** points
- ◊ **Check data types** and **missing value**

Exploratory data analysis

- ◊ Create pie chart of **Churn_Flag proportion**
- ◊ (0) Loyal customer = 98.30%
- ◊ (1) Churner = 1.70%

Since the current churn rate is on target at 1.70%, it's not a major concern at this stage, but proactive measures should still be in place to prevent potential causes.

Churn Rate prediction (2)

Machine Learning Models

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
y_test_pred = rf.predict(X_test)

print("Accuracy Score:\n", accuracy_score(y_test, y_test_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
print("Classification Report:\n", classification_report(y_test, y_test_pred))
```

```
Accuracy Score:
 0.9994397916024761
Confsuion Matrix:
 [[35079    20]
 [    0   602]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     35099
           1       0.97      1.00      0.98       602

    accuracy                           1.00     35701
   macro avg       0.98      1.00      0.99     35701
weighted avg       1.00      1.00      1.00     35701
```

Features selection

◊ (X) Features : **relevant features**

For **categorical features** —> **label encoder**

◊ (Y) Target : Churn_Flag

For (0) Loyal customer (1) Churner

Data implementation

◊ Recheck data shape

◊ Create **train test split**

◊ Use **SMOTE** (oversampling technique)

Model evaluation

◊ Conduct model : **Random Forest**

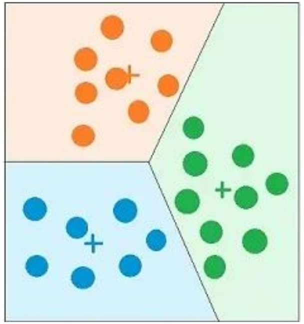
◊ Nearly **perfect performance**

This outcome provides a strong starting point for making proactive business decisions. However, we should be mindful of potential overfitting.

Churn Rate prediction (3)

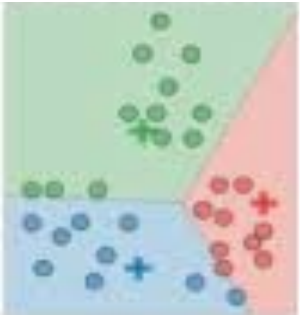
Customer Segmentation

For example only



Clustering of
(0) loyal customer

Effective offer with
high accuracy



Clustering of
(1) Churner

Likely to show
retention rate > 80%

- ◊ *Group 1 : **Price sensitivity** —> **Discount plan***
- ◊ *Group 2 : **High data usage** —> **Free extra data***
- ◊ *Group 3 : **Loyalty customer** —> **Exclusive deal***
- ◊ *Another promotions : bundle plan, gift, co-credit card, etc.*

Retention rate definition

◊ **Retention rate** is **customers who accept offers** when **they request to terminate services**

Customer segmentation

- ◊ (0) Loyal customer : **Customer segmentation**
- ◊ Conduct model : **K-means clustering**
- ◊ Find an **optimal point** : **Elbow method**

Implementation

- ◊ **Tailored offers** to each group of (0) Loyal customer
- ◊ **Implement their acceptance** in each group
- ◊ **Conduct** another **classification model**
- ◊ (0) decline, (1) accept
- ◊ **Offer the most effective promotion** to **potential churners** in each similar group

A/B testing is easier method to implement.

Contact

Naphon Santisukwongchot
emoney_euro@hotmail.com
(+66)89 738 3632

https://www.linkedin.com/in/naphon1999/
https://github.com/naphon1999
https://www.datacamp.com/portfolio/naphon1999
https://drive.google.com/drive/folders/1-3x_-Xmho03z5u3PA6VKZi2-nY90oixK?usp=sharing

Certifications & Developments

Data Science Bootcamp 10 : DataRockie
Data Analyst in SQL & Python : DataCamp
Google Advanced Data Analytics : Google
IBM Data Science: IBM
Machine Learning : DeepLearning.AI

Project 2
Predicting Movie Rental Durations

Project Predicting Movie Rental Durations (1)

```python
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15861 entries, 0 to 15860
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   rental_date       15861 non-null  object
 1   return_date       15861 non-null  object
 2   amount            15861 non-null  float64
 3   release_year      15861 non-null  float64
 4   rental_rate       15861 non-null  float64
 5   length            15861 non-null  float64
 6   replacement_cost  15861 non-null  float64
 7   special_features  15861 non-null  object
 8   NC-17             15861 non-null  int64
 9   PG                15861 non-null  int64
 10  PG-13             15861 non-null  int64
 11  R                 15861 non-null  int64
 12  amount_2          15861 non-null  float64
 13  length_2          15861 non-null  float64
 14  rental_rate_2     15861 non-null  float64
dtypes: float64(8), int64(4), object(3)
memory usage: 1.8+ MB
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15861 entries, 0 to 15860
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   rental_date       15861 non-null  datetime64[ns, UTC]
 1   return_date       15861 non-null  datetime64[ns, UTC]
 2   amount            15861 non-null  float64
 3   release_year      15861 non-null  float64
 4   rental_rate       15861 non-null  float64
 5   length            15861 non-null  float64
 6   replacement_cost  15861 non-null  float64
 7   special_features  15861 non-null  object
 8   NC-17             15861 non-null  int64
 9   PG                15861 non-null  int64
 10  PG-13             15861 non-null  int64
 11  R                 15861 non-null  int64
 12  amount_2          15861 non-null  float64
 13  length_2          15861 non-null  float64
 14  rental_rate_2     15861 non-null  float64
 15  rental_length     15861 non-null  timedelta64[ns]
 16  rental_days       15861 non-null  int64
 17  deleted_scenes    15861 non-null  int64
 18  behind_the_scenes 15861 non-null  int64
dtypes: datetime64[ns, UTC](2), float64(8), int64(7), object(1), timedelta64[ns](1)
memory usage: 2.3+ MB
```

A DVD rental company needs your help! They want to figure out how many days a customer will rent a DVD for based on some features. They want you to try out some regression models which will help predict the number of days a customer will rent a DVD. **The company wants a model which yields a MSE** of 3 or less on a test set. The model you make will help the company become more efficient inventory planning.

Exploratory data analysis

◊ Import frameworks and csv file
◊ Perform EDA : df.head(), df.info(), df.describe
◊ Set a target variable
  - Add rental_length column
  - Add rental_days column : Target
◊ Categorize special features into one hot encoder
  - Add deleted_scenes column : Feature
  - Add behind_the_scenes column : Feature

Project Predicting Movie Rental Durations (2)

Feature Selection

- ◊ removing irrelevant features
  - Assign relevant features into X
- ◊ Assign rental_days (target) into Y

```
X = df.drop(['rental_days','rental_date','return_date','rental_length','special_features'], axis=1)
y = df['rental_days']
```

Data implementation

- ◊ Checking data set dimension
- ◊ Perform train test split

```
print(X.shape)
print(y.shape)
```

```
(15861, 14)
(15861,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=9)
```

Project Predicting Movie Rental Durations (3)

Linear (lasso)

```
# Perform feature selectino by choosing columns with positive coefficients

lasso = Lasso(alpha=0.3, random_state=9)
lasso.fit(X_train, y_train)
lasso_coef = lasso.coef_
X_lasso_train, X_lasso_test = X_train.iloc[:, lasso_coef > 0], X_test.iloc[:, lasso_coef > 0]
```

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X_lasso_train, y_train)
lr_pred = lr.predict(X_lasso_test)
lr_mse = mean_squared_error(y_test, lr_pred)
lr_mse
```

```
4.812297241276244
```

Decision tree

```
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor(max_depth = 4,
                           min_samples_leaf=0.1,
                           random_state = 3)
dt.fit(X_train, y_train)
dt_pred = dt.predict(X_test)
dt_mse = mean_squared_error(y_test, dt_pred)
dt_mse
```

```
3.2717707577851667
```

Random forest

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV

param_dist = {'n_estimators': np.arange(1,101,1),
              'max_depth': np.arange(1,11,1)}

rf = RandomForestRegressor()
random_search = RandomizedSearchCV(rf,
                                   param_distributions = param_dist,
                                   cv=5,
                                   random_state=9)
random_search.fit(X_train, y_train)

hyper_params = random_search.best_params_

rf = RandomForestRegressor(n_estimators = hyper_params['n_estimators'],
                           max_depth = hyper_params['max_depth'],
                           random_state=9)

rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)
rf_mse = mean_squared_error(y_test, rf_pred)
rf_mse
```

```
2.225667528098759
```

MSE calculation

◊ Perform machine learning

- Linear (lasso) : MSE = 4.812
- Decision tree : MSE = 3.271
- Random Forest : MSE = 2.225

Contact

Naphon Santisukwongchot
emoney_euro@hotmail.com
(+66)89 738 3632

https://www.linkedin.com/in/naphon1999/
https://github.com/naphon1999
https://www.datacamp.com/portfolio/naphon1999
https://drive.google.com/drive/folders/1-3x_-Xmho03z5u3PA6VKZi2-nY90oixK?usp=sharing

Portfolio reference

https://drive.google.com/file/d/1nS9qUg9F65z3MXSeZQoGUH-U2ZeuNgZq/view?usp=drive_link

Certifications & Developments

Data Science Bootcamp 10 :	DataRockie
Data Analyst in SQL & Python :	DataCamp
Google Advanced Data Analytics :	Google
IBM Data Science:	IBM
Machine Learning :	DeepLearning.AI

Work achievement

- ◊ Achieve campaign sales target
- ◊ Completely release aging stock

Project 3
Predicting Landing Success Rate

SpaceX Falcon 9 (I)

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
X.head(100)
```

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS	...	Serial_B1058
0	1.0	6104.959412	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
1	2.0	525.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
2	3.0	677.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	...	0.0
3	4.0	500.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
4	5.0	3170.000000	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0

Import library

◊ Import frameworks

pandas, numpy : Data manipulation
matplotlib, seaborn : Data visualization
Sklearn : Machine learning

◊ Exploring data: df.head(), df.info(), df.describe

SpaceX Falcon 9 (II)

```
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

Feature Selection

◊ ‘Class’ column : **Target**

◊ All columns (except ‘Class’) : **Feature**

Standardization

◊ Ensuring all features contribute equally and improves convergence

Data splitting

◊ Split the data into training and test set, 20%

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

SpaceX Falcon 9 (III)

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

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

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

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

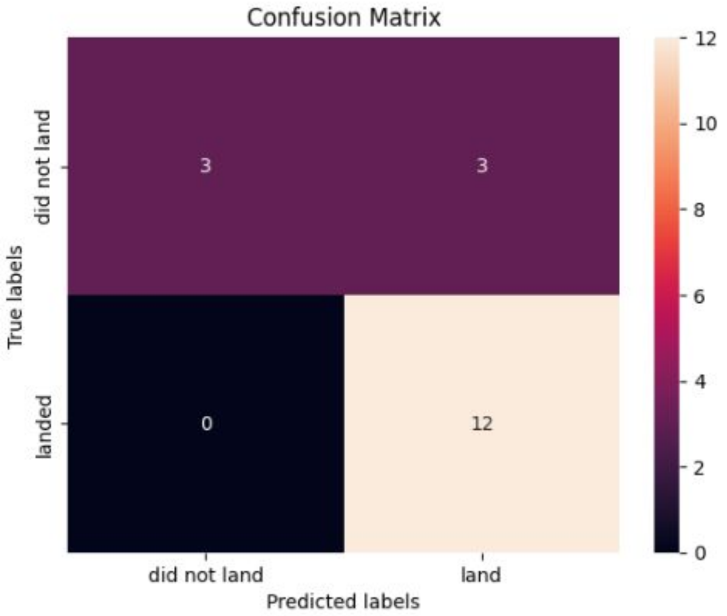
```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

Conducting model (1)

- Conduct Logistic regression model
- GridSearchCV, cv=10 : find the best parameters
  - C = 0.01
  - penalty : '12' (L2 regularization)
  - solver : 'lbfgs'
- Accuracy best params = 0.846

SpaceX Falcon 9 (IV)

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



Confusion matrix (1)

- ◊ Accuracy test data = 0.83
- ◊ Precision = 1.00
- ◊ Recall = 0.50
- ◊ F1 Score = 0.67

SpaceX Falcon 9 (V)

```
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)
svm_cv.fit(X_train, Y_train)
```

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

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

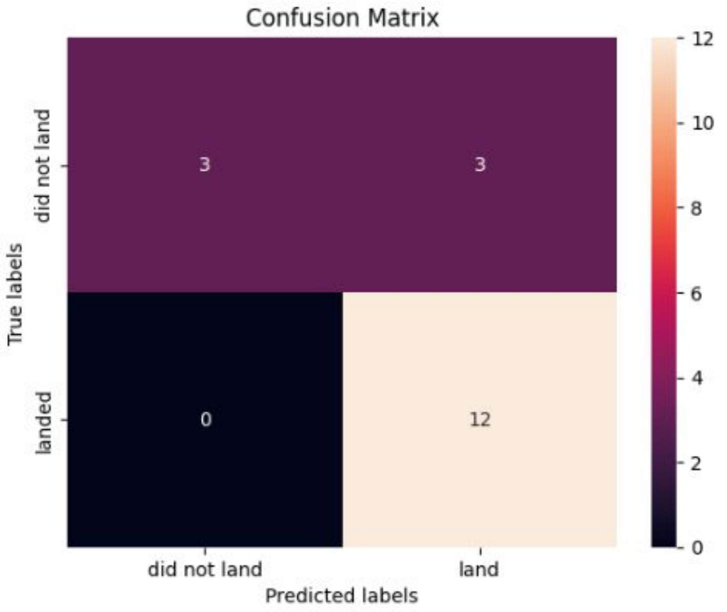
```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

Conducting model (2)

- Create support vector machine
- GridSearchCV, svm_cv=10 : find the best parameters
  - C :=1.00
  - gamma = 0.032
  - kernel : ‘sigmoid’
- Accuracy best params = 0.848

SpaceX Falcon 9 (VI)

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



Confusion matrix (2)

- ◊ Accuracy test data = 0.83
- ◊ Precision = 1.00
- ◊ Recall = 0.50
- ◊ F1 Score = 0.67

SpaceX Falcon 9 (VII)

Conducting model (3)

- ◊ Create decision tree classifier
- ◊ GridSearchCV, tree_cv=10 : find the best parameters
- ◊ Accuracy best params = 0.9

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

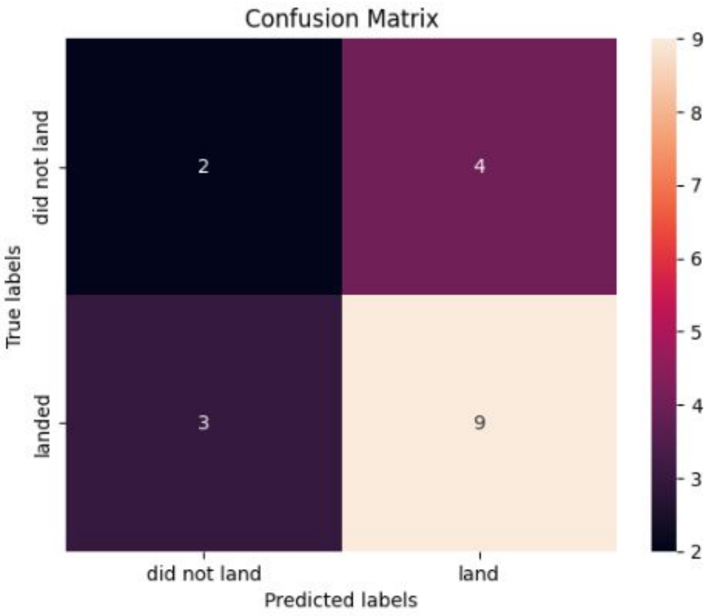
```
tree_cv = GridSearchCV(estimator=tree, param_grid=parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_lea
f': 4, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.9
```

SpaceX Falcon 9 (VIII)

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



Confusion matrix (3)

◊ Accuracy test data	=	0.61
◊ Precision	=	0.40
◊ Recall	=	0.33
◊ F1 Score	=	0.36

SpaceX Falcon 9 (IX)

```
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(estimator=KNN, param_grid=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_)
```

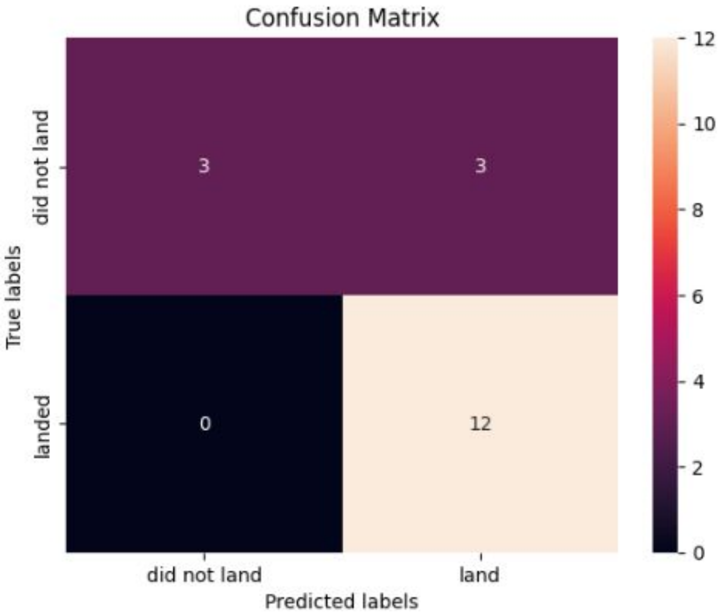
```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

Conducting model (4)

- ◊ Create k nearest neighbors classifier
- ◊ GridSearchCV, tree_cv=10 : find the best parameters
- ◊ Accuracy best params = 0.848

SpaceX Falcon 9 (X)

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



Confusion matrix (4)

- Accuracy test data = 0.83
- Precision = 1.00
- Recall = 0.50
- F1 Score = 0.67

Contact

Naphon Santisukwongchot
emoney_euro@hotmail.com
(+66)89 738 3632

https://www.linkedin.com/in/naphon1999/
https://github.com/naphon1999
https://www.datacamp.com/portfolio/naphon1999
https://drive.google.com/drive/folders/1-3x_-Xmho03z5u3PA6VKZi2-nY90oixK?usp=sharing

Data Source

https://drive.google.com/file/d/1arTdtj3HaGGeVVVU6CgdzYKqkC___VaFk/view?usp=drive_link

https://drive.google.com/file/d/16aoYQ-j5AAbTy445m4_3LmysiAZRi_vD/view?usp=drive_link

Certifications & Developments

Data Science Bootcamp 10 :	DataRockie
Data Analyst in SQL & Python :	DataCamp
Google Advanced Data Analytics :	Google
IBM Data Science:	IBM
Machine Learning :	DeepLearning.AI