In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
data= pd.read_csv('afghanistan_conflict_displacements_2021.csv')
```

In [3]:
```
data = data.iloc[1: , :]
data
```

Out[3]:

	DISP_DATE	ORIG_PROV_CODE	ORIG_PROV_NAME	ORIG_DIST_CODE	ORIG_DIST_NAME	DISP_PROV_CODE	DISP_PROV_NAME	DISP_DIST_C
1	01-01-2021	6	Nangarhar	604	Khogyani	6	Nangarhar	
2	01-01-2021	28	Faryab	2806	Qaysar	28	Faryab	2
3	01-01-2021	28	Faryab	2802	Khwajasabzposh	28	Faryab	2
4	01-01-2021	28	Faryab	2805	Almar	28	Faryab	2
5	01-01-2021	28	Faryab	2803	Pashtunkot	28	Faryab	2
...	...	...	...	...	...	...	...	
1174	18-10-2021	21	Ghor	2101	Chaghcharan	30	Hirat	3
1175	19-10-2021	21	Ghor	2109	Tolak	30	Hirat	3
1176	20-10-2021	11	Ghazni	1101	Ghazni	1	Kabul	
1177	09-11-2021	4	Wardak	404	Hesa-e-Awal-e-Behsud	1	Kabul	
1178	22-11-2021	2	Kapisa	205	Tagab	2	Kapisa	

1178 rows × 14 columns

In [4]:
```
data.head()
```

Out[4]:

	DISP_DATE	ORIG_PROV_CODE	ORIG_PROV_NAME	ORIG_DIST_CODE	ORIG_DIST_NAME	DISP_PROV_CODE	DISP_PROV_NAME	DISP_DIST_CODI
1	01-01-2021	6	Nangarhar	604	Khogyani	6	Nangarhar	60
2	01-01-2021	28	Faryab	2806	Qaysar	28	Faryab	280
3	01-01-2021	28	Faryab	2802	Khwajasabzposh	28	Faryab	280
4	01-01-2021	28	Faryab	2805	Almar	28	Faryab	280
5	01-01-2021	28	Faryab	2803	Pashtunkot	28	Faryab	280

In [5]:
```
data.tail()
```

Out[5]:

	DISP_DATE	ORIG_PROV_CODE	ORIG_PROV_NAME	ORIG_DIST_CODE	ORIG_DIST_NAME	DISP_PROV_CODE	DISP_PROV_NAME	DISP_DIST_C
1174	18-10-2021	21	Ghor	2101	Chaghcharan	30	Hirat	3
1175	19-10-2021	21	Ghor	2109	Tolak	30	Hirat	3
1176	20-10-2021	11	Ghazni	1101	Ghazni	1	Kabul	
1177	09-11-2021	4	Wardak	404	Hesa-e-Awal-e-Behsud	1	Kabul	
1178	22-11-2021	2	Kapisa	205	Tagab	2	Kapisa	

In [6]:
```
data.describe()
```

Out[6]:

	DISP_DATE	ORIG_PROV_CODE	ORIG_PROV_NAME	ORIG_DIST_CODE	ORIG_DIST_NAME	DISP_PROV_CODE	DISP_PROV_NAME	DISP_DIST_
count	1178	1178	1178	1176	1177	1178	1178	
unique	275	33	34	233	227	34	34	
top	23-05-2021	21	Ghor	2101	Chaghcharan	30	Hirat	
freq	30	145	144	29	29	243	243	

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1178 entries, 1 to 1178
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   DISP_DATE          1178 non-null   object
 1   ORIG_PROV_CODE     1178 non-null   object
 2   ORIG_PROV_NAME     1178 non-null   object
 3   ORIG_DIST_CODE     1176 non-null   object
 4   ORIG_DIST_NAME     1177 non-null   object
 5   DISP_PROV_CODE     1178 non-null   object
 6   DISP_PROV_NAME     1178 non-null   object
 7   DISP_DIST_CODE     1178 non-null   object
 8   DISP_DIST_NAME     1178 non-null   object
 9   DISP_IND           1178 non-null   object
 10  DISP_FAM           1178 non-null   object
 11  DISP_ADULT_MALE    61 non-null     object
 12  DISP_ADULT_FEMALE  61 non-null     object
 13  DISP_CHILDREN_U18  61 non-null     object
dtypes: object(14)
memory usage: 129.0+ KB
```

In [8]:
```
data.isna().sum()
```

Out[8]:
```
DISP_DATE              0
ORIG_PROV_CODE         0
ORIG_PROV_NAME         0
ORIG_DIST_CODE         2
ORIG_DIST_NAME         1
DISP_PROV_CODE         0
DISP_PROV_NAME         0
DISP_DIST_CODE         0
DISP_DIST_NAME         0
DISP_IND               0
DISP_FAM               0
DISP_ADULT_MALE     1117
DISP_ADULT_FEMALE   1117
DISP_CHILDREN_U18   1117
dtype: int64
```

In [9]:
```
data['ORIG_DIST_NAME'].replace('', np.nan, inplace=True)
data.dropna(subset=['ORIG_DIST_NAME'], inplace=True)
data['ORIG_DIST_CODE'].replace('', np.nan, inplace=True)
data.dropna(subset=['ORIG_DIST_CODE'], inplace=True)
data['DISP_ADULT_MALE'].replace('', np.nan, inplace=True)
data.dropna(subset=['DISP_ADULT_MALE'], inplace=True)
data['DISP_ADULT_FEMALE'].replace('', np.nan, inplace=True)
data.dropna(subset=['DISP_ADULT_FEMALE'], inplace=True)
data['DISP_CHILDREN_U18'].replace('', np.nan, inplace=True)
data.dropna(subset=['DISP_CHILDREN_U18'], inplace=True)
```

In [10]:
```
data.isnull().any().any()
```

Out[10]: False

In [11]:
```
data['DISP_MONTH'] = pd.DatetimeIndex(data['DISP_DATE']).month
```

In [12]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [13]:
```
data.ORIG_PROV_NAME = le.fit_transform(data.ORIG_PROV_NAME)
data.ORIG_DIST_NAME = le.fit_transform(data.ORIG_DIST_NAME)
data.DISP_PROV_NAME = le.fit_transform(data.DISP_PROV_NAME)
data.DISP_DIST_NAME = le.fit_transform(data.DISP_DIST_NAME)
```

In [25]:
```
x = data.drop(['DISP_IND', 'DISP_DATE'], axis = 1)
```

In [26]:
```
y = data.DISP_IND
```

In [27]:
```
x.shape
```

Out[27]: (61, 13)

In [28]:
```
y.shape
```

Out[28]: (61,)

In [29]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [30]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
c:\python\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preproc
essing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/sta
ble/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[30]: LogisticRegression()

In [31]:
```
y_pred = model.predict(X_test)
```

In [32]:
```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.7083333333333334
Testing Accuracy : 0.07692307692307693
```

In [33]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [34]:
```
model1 = LinearRegression()
model1.fit(X_train, y_train)
```

Out[34]: LinearRegression()

In [35]:
```
y_pred = model1.predict(X_test)
```

In [36]:
```
print("Training Accuracy :", model1.score(X_train, y_train))
print("Testing Accuracy :", model1.score(X_test, y_test))
```

```
Training Accuracy : 0.9999827695230145
Testing Accuracy : 0.9971842170917591
```

In [42]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [43]:
```
model3 = DecisionTreeRegressor(max_depth=6)
model3.fit(X_train, y_train)
```

Out[43]: DecisionTreeRegressor(max_depth=6)

In [44]:
```
y_pred = model3.predict(X_test)
```

In [45]:
```
print("Training Accuracy :", model3.score(X_train, y_train))
print("Testing Accuracy :", model3.score(X_test, y_test))
```

```
Training Accuracy : 0.9999723814628593
Testing Accuracy : 0.7305243393498684
```

In [46]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [47]:
```
model4 = RandomForestRegressor(n_estimators = 100, random_state = 0)
model4.fit(X_train, y_train)
```

Out[47]:
```
RandomForestRegressor(random_state=0)
```

In [48]:
```
y_pred = model4.predict(X_test)
```

In [49]:
```
print("Training Accuracy :", model4.score(X_train, y_train))
print("Testing Accuracy :", model4.score(X_test, y_test))
```

```
Training Accuracy : 0.9578801941063431
Testing Accuracy : 0.9724425203206172
```