- Implement KNN algorithm from sklearn to predict Fraud Flag for using the dataset MotorInsuranceFraud.csv., display accuracy score and confusion matrix.

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [14]:
```
datas = pd.read_csv("MotorInsuranceFraud.csv")
datas.head()
```

Out[14]:

	ID	Insurance Type	Income of Policy Holder	Marital Status	Num Claimants	Injury Type	Overnight Hospital Stay	Claim Amount	Total Claimed	Num Claims	Num Soft Tissue	% Soft Tissue	Claim Amount Received	Fraud Flag
0	1	CI	0	NaN	2	Soft Tissue	No	1625	3250	2	2.0	1.0	0	1
1	2	CI	0	NaN	2	Back	Yes	15028	60112	1	0.0	0.0	15028	0
2	3	CI	54613	Married	1	Broken Limb	No	-99999	0	0	0.0	0.0	572	0
3	4	CI	0	NaN	3	Serious	Yes	270200	0	0	0.0	0.0	270200	0
4	5	CI	0	NaN	4	Soft Tissue	No	8869	0	0	0.0	0.0	0	1

In [15]:
```
datas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   ID                      500 non-null    int64  
 1   Insurance Type          500 non-null    object 
 2   Income of Policy Holder  500 non-null    int64  
 3   Marital Status          170 non-null    object 
 4   Num Claimants           500 non-null    int64  
 5   Injury Type             500 non-null    object 
 6   Overnight Hospital Stay  500 non-null    object 
 7   Claim Amount            500 non-null    int64  
 8   Total Claimed           500 non-null    int64  
 9   Num Claims              500 non-null    int64  
 10  Num Soft Tissue         490 non-null    float64
 11  % Soft Tissue           500 non-null    float64
 12  Claim Amount Received   500 non-null    int64  
 13  Fraud Flag              500 non-null    int64  
dtypes: float64(2), int64(8), object(4)
memory usage: 54.8+ KB
```

In [16]:
```
datas = datas.rename(columns={'Insurance Type': 'IT', 'Income of Policy Holder': 'IPH', 'Marital Status': 'MS',
                              'Claim Amount': 'CA','Total Claimed': 'TC','Num Claims': 'NC','Num Soft Tissue':
```

In [17]:
```
datas
```

Out[17]:

	ID	Insurance Type	IPH	MS	NUC	InT	OHS	CA	TC	NC	NST	PST	CAR	Fraud Flag
0	1	CI	0	NaN	2	Soft Tissue	No	1625	3250	2	2.0	1.0	0	1
1	2	CI	0	NaN	2	Back	Yes	15028	60112	1	0.0	0.0	15028	0
2	3	CI	54613	Married	1	Broken Limb	No	-99999	0	0	0.0	0.0	572	0
3	4	CI	0	NaN	3	Serious	Yes	270200	0	0	0.0	0.0	270200	0
4	5	CI	0	NaN	4	Soft Tissue	No	8869	0	0	0.0	0.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	496	CI	0	NaN	1	Soft Tissue	No	2118	0	0	0.0	0.0	0	1
496	497	CI	29280	Married	4	Broken Limb	Yes	3199	0	0	NaN	0.0	0	1
497	498	CI	0	NaN	1	Broken Limb	Yes	32469	0	0	0.0	0.0	16763	0
498	499	CI	46683	Married	1	Broken Limb	No	179448	0	0	0.0	0.0	179448	0
499	500	CI	0	NaN	1	Broken Limb	No	8259	0	0	0.0	0.0	0	1

500 rows × 14 columns

In [18]:
```
datas['Fraud Flag'].value_counts()
```

Out[18]:
```
0    332
1    168
Name: Fraud Flag, dtype: int64
```

In [19]:
```
datas = datas.drop(columns = [
    'ID',
    'Insurance Type ',
    'MS'])
```

In [21]:
```
datas['NST'].fillna(value=datas['NST'].mean(), inplace=True)
```

In [47]:
```
datas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   IPH         500 non-null    int64  
 1   NUC         500 non-null    int64  
 2   InT         500 non-null    int32  
 3   OHS         500 non-null    int32  
 4   CA          500 non-null    int64  
 5   TC          500 non-null    int64  
 6   NC          500 non-null    int64  
 7   NST         500 non-null    float64
 8   PST         500 non-null    float64
 9   CAR         500 non-null    int64  
 10  Fraud Flag  500 non-null    int64  
dtypes: float64(2), int32(2), int64(7)
memory usage: 39.2 KB
```

Data Preprocessing / Cleaning

In [23]:
```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()
```

In [26]:
```
datas['InT']= label_encoder.fit_transform(datas['InT'])
```

In [27]:
```
datas['OHS']= label_encoder.fit_transform(datas['OHS'])
```

In [28]:
```
datas
```

Out[28]:

	IPH	NUC	InT	OHS	CA	TC	NC	NST	PST	CAR	Fraud Flag
0	0	2	3	0	1625	3250	2	2.000000	1.0	0	1
1	0	2	0	1	15028	60112	1	0.000000	0.0	15028	0
2	54613	1	1	0	-99999	0	0	0.000000	0.0	572	0
3	0	3	2	1	270200	0	0	0.000000	0.0	270200	0
4	0	4	3	0	8869	0	0	0.000000	0.0	0	1
...	...	...	...	...	...	...	...	...	...	...	...
495	0	1	3	0	2118	0	0	0.000000	0.0	0	1
496	29280	4	1	1	3199	0	0	0.234694	0.0	0	1
497	0	1	1	1	32469	0	0	0.000000	0.0	16763	0
498	46683	1	1	0	179448	0	0	0.000000	0.0	179448	0
499	0	1	1	0	8259	0	0	0.000000	0.0	0	1

500 rows × 11 columns

In [36]:
```
X = datas.iloc[:, 0:-1]
Y = datas.iloc[:, -1]
```

In [37]:
```
X
```

Out[37]:

	IPH	NUC	InT	OHS	CA	TC	NC	NST	PST	CAR
0	0	2	3	0	1625	3250	2	2.000000	1.0	0
1	0	2	0	1	15028	60112	1	0.000000	0.0	15028
2	54613	1	1	0	-99999	0	0	0.000000	0.0	572
3	0	3	2	1	270200	0	0	0.000000	0.0	270200
4	0	4	3	0	8869	0	0	0.000000	0.0	0
...	...	...	...	...	...	...	...	...	...	...
495	0	1	3	0	2118	0	0	0.000000	0.0	0
496	29280	4	1	1	3199	0	0	0.234694	0.0	0
497	0	1	1	1	32469	0	0	0.000000	0.0	16763
498	46683	1	1	0	179448	0	0	0.000000	0.0	179448
499	0	1	1	0	8259	0	0	0.000000	0.0	0

500 rows × 10 columns

In [38]:
```
Y
```

Out[38]:
```
0      1
1      0
2      0
3      0
4      1
      ..
495    1
496    1
497    0
498    0
499    1
Name: Fraud Flag, Length: 500, dtype: int64
```

In [39]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size= 0.25,random_state = 65)
```

In [40]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

In [41]:
```
#Classifier Creation
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5,metric = 'minkowski',p = 2)
classifier.fit(X_train,Y_train)
```

Out[41]:
```
KNeighborsClassifier()
```

In [42]:
```
Y_pred = classifier.predict(X_test)
```

In [43]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test,Y_pred)
cm
```

Out[43]:
```
array([[70, 18],
       [16, 21]], dtype=int64)
```

With 72.8% accuracy, we take a closer look at the confusion matrix:

- 70 transactions were classified as valid that were actually valid
- 18 transactions were classified as fraud that were actually valid (type 1 error)
- 16 transactions were classified as valid that were fraud (type 2 error)
- 21 transactions were classified as fraud that were fraud

- Error = ((FP+FN))/(TP+TN+FN+FP)
  = ((18+16))/(70+18+16+21)
  = 34/125
  = 0.272
- Error % = 0.272 * 100
  = 27.2%

So, the algorithm misclassified 27.2% fraudulent transactions.

In [44]:
```
#accuracy score
classifier.score(X_test, Y_test)
```

Out[44]:
```
0.728
```