In [1]:

```
import pandas as pd
```

In [2]:

```
df  = pd.read_csv("claimants.csv")
```

In [3]:

```
df
```

Out[3]:

| | CASENUM | ATTORNEY | CLMSEX | CLMINSUR | SEATBELT | CLMAGE | LOSS |
|---|---|---|---|---|---|---|---|
| **0** | 5 | 0 | 0.0 | 1.0 | 0.0 | 50.0 | 34.940 |
| **1** | 3 | 1 | 1.0 | 0.0 | 0.0 | 18.0 | 0.891 |
| **2** | 66 | 1 | 0.0 | 1.0 | 0.0 | 5.0 | 0.330 |
| **3** | 70 | 0 | 0.0 | 1.0 | 1.0 | 31.0 | 0.037 |
| **4** | 96 | 1 | 0.0 | 1.0 | 0.0 | 30.0 | 0.038 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **1335** | 34100 | 1 | 0.0 | 1.0 | 0.0 | NaN | 0.576 |
| **1336** | 34110 | 0 | 1.0 | 1.0 | 0.0 | 46.0 | 3.705 |
| **1337** | 34113 | 1 | 1.0 | 1.0 | 0.0 | 39.0 | 0.099 |
| **1338** | 34145 | 0 | 1.0 | 0.0 | 0.0 | 8.0 | 3.177 |
| **1339** | 34153 | 1 | 1.0 | 1.0 | 0.0 | 30.0 | 0.688 |

1340 rows × 7 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1340 entries, 0 to 1339
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   CASENUM   1340 non-null   int64  
 1   ATTORNEY  1340 non-null   int64  
 2   CLMSEX    1328 non-null   float64
 3   CLMINSUR  1299 non-null   float64
 4   SEATBELT  1292 non-null   float64
 5   CLMAGE    1151 non-null   float64
 6   LOSS      1340 non-null   float64
dtypes: float64(5), int64(2)
memory usage: 73.4 KB
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
CASENUM       0
ATTORNEY      0
CLMSEX       12
CLMINSUR     41
SEATBELT     48
CLMAGE      189
LOSS          0
dtype: int64
```

Here CLMSEX, CLMINSUR, SEATBELT, CLMAGE column have null value .

In [6]:

```
df['CLMAGE'] = df['CLMAGE'].fillna(df['CLMAGE'].mean())
df['CLMSEX'] = df['CLMSEX'].fillna(df['CLMSEX'].mode()[0])
df['CLMINSUR'] = df['CLMINSUR'].fillna(df['CLMINSUR'].mode()[0])
df['SEATBELT'] = df['SEATBELT'].fillna(df['SEATBELT'].mode()[0])
```

# Classify using decision tree

In [7]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix ,accuracy_score
from sklearn.model_selection import train_test_split
```

In [8]:

```
x = df.drop(['ATTORNEY'],axis = 1)
y  = df['ATTORNEY']
```

In [9]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = .2,random_sta
```

In [10]:

```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [11]:

```
model = DecisionTreeClassifier()
model.fit(x_train_scaled ,y_train)
y_pred = model.predict(x_test_scaled)
```

In [12]:

```
accuracy_score(y_test,y_pred)
```

Out[12]:

```
0.6156716417910447
```

# Feature Selection Method

# (1)Univariate Selection

In [13]:

```
import numpy as np
from sklearn.feature_selection import chi2, SelectKBest
selectBestFeature = SelectKBest(score_func=chi2, k=5)
bestFeatures = selectBestFeature.fit(x, y)
```

In [14]:

```
bestscore_df = pd.DataFrame([x.columns, bestFeatures.scores_]).T
bestscore_df.columns = ['Features', 'Scores']
bestscore_df['Scores'] = bestscore_df['Scores'].astype(np.float64)
bestscore_df
```

Out[14]:

| | Features | Scores |
|---|---|---|
| **0** | CASENUM | 1485.004335 |
| **1** | CLMSEX | 3.435190 |
| **2** | CLMINSUR | 0.704679 |
| **3** | SEATBELT | 4.110781 |
| **4** | CLMAGE | 1.630615 |
| **5** | LOSS | 1921.439566 |

In [15]:

```
bestscore_df.nlargest(5, 'Scores')
```

Out[15]:

| | Features | Scores |
|---|---|---|
| **5** | LOSS | 1921.439566 |
| **0** | CASENUM | 1485.004335 |
| **3** | SEATBELT | 4.110781 |
| **1** | CLMSEX | 3.435190 |
| **4** | CLMAGE | 1.630615 |

## Recursive Feature Elimination

In [16]:

```
from sklearn.feature_selection import RFE
import matplotlib.pyplot as plt

rfe = RFE(estimator=model, n_features_to_select=1)
rfe.fit(x_train, y_train)
```

Out[16]:

```
▸                  RFE
▸ estimator: DecisionTreeClassifier
     ▸ DecisionTreeClassifier
```

In [17]:

```
rfe.support_
```

Out[17]:

```
array([False, False, False, False, False,  True])
```

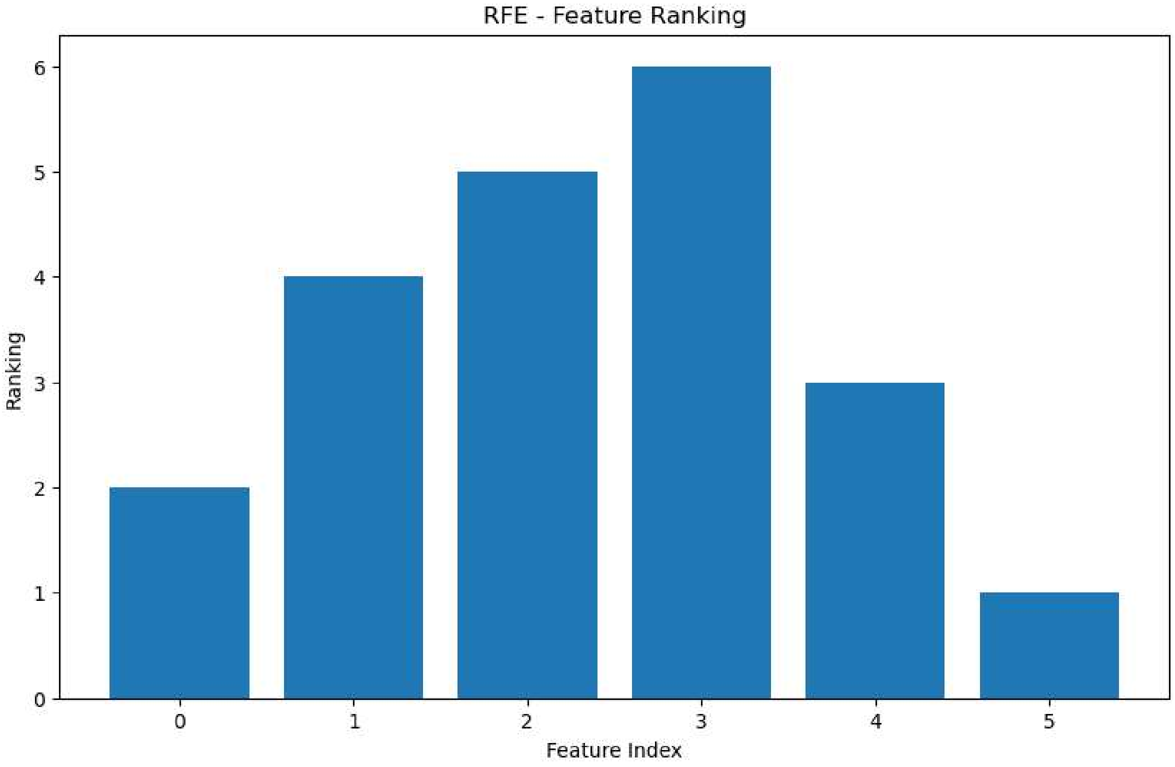
In [18]:

```
feature_ranking = rfe.ranking_
```

In [19]:

```
selected_feature = np.where(feature_ranking == 1)[0]


plt.figure(figsize=(10, 6))
plt.title("RFE - Feature Ranking")
plt.xlabel("Feature Index")
plt.ylabel("Ranking")
plt.bar(range(len(feature_ranking)), feature_ranking)
plt.show()
```



In [20]:

```
print("Selected Features:", selected_feature)

# Train final model
model.fit(x_train.iloc[:, selected_feature], y_train)

# Evaluate the model on the test set
rfe_y_pred = model.predict(x_test.iloc[:, selected_feature])
print("Accuracy on the Test Set:", accuracy_score(y_test, rfe_y_pred))
```

```
Selected Features: [5]
Accuracy on the Test Set: 0.6007462686567164
```