LOGISTIC REGRESSION

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

In [2]:
```
var=pd.read_csv('C://Users/Gopi/Desktop/machine learning/csv files/claimants.csv')
```

In [3]:
```
var.head()
```

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

In [4]:
```
var.columns
```

Out[4]:
```
Index(['CASENUM', 'ATTORNEY', 'CLMSEX', 'CLMINSUR', 'SEATBELT', 'CLMAGE',
       'LOSS'],
      dtype='object')
```

In [5]:
```
import statsmodels.formula.api as smf
```

In [6]:
```
model=smf.logit('ATTORNEY~CLMAGE+LOSS+CLMINSUR+CLMSEX+SEATBELT',data=var).fit()
model.params
```

```
Optimization terminated successfully.
         Current function value: 0.587523
         Iterations 8
```

Out[6]:
```
Intercept   -0.199978
CLMAGE       0.006487
LOSS        -0.385044
CLMINSUR     0.602173
CLMSEX       0.432996
SEATBELT    -0.781079
dtype: float64
```

In [7]:
```
(np.exp(model.params))
```

Out[7]:
```
Intercept    0.818749
CLMAGE       1.006508
LOSS         0.680421
CLMINSUR     1.826083
CLMSEX       1.541870
SEATBELT     0.457912
dtype: float64
```

In [14]:
```
model.summary()
```

Out[14]:

Logit Regression Results

Dep. Variable:	ATTORNEY	**No. Observations:**	1096
Model:	Logit	**Df Residuals:**	1090
Method:	MLE	**Df Model:**	5
Date:	Sat, 21 Dec 2019	**Pseudo R-squ.:**	0.1505
Time:	18:08:22	**Log-Likelihood:**	-643.92
converged:	True	**LL-Null:**	-758.05
Covariance Type:	nonrobust	**LLR p-value:**	2.546e-47

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-0.2000	0.247	-0.810	0.418	-0.684	0.284
CLMAGE	0.0065	0.003	1.952	0.051	-2.75e-05	0.013
LOSS	-0.3850	0.035	-11.050	0.000	-0.453	-0.317
CLMINSUR	0.6022	0.231	2.606	0.009	0.149	1.055
CLMSEX	0.4330	0.136	3.191	0.001	0.167	0.699
SEATBELT	-0.7811	0.566	-1.380	0.168	-1.891	0.329

In [8]:
```
predict=model.predict(pd.DataFrame(var[['CLMAGE','LOSS','CLMINSUR','CLMSEX','SEATBELT']]))
predict
```

Out[8]:
```
0       0.000003
1       0.501679
2       0.576291
3       0.452142
4       0.641570
          ...
1335         NaN
1336    0.427275
1337    0.740790
1338    0.281230
1339    0.682417
Length: 1340, dtype: float64
```

In [9]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
confusionmatrix = confusion_matrix(var['ATTORNEY'],predict > 0.5 )
confusionmatrix
```

Out[9]:
```
array([[487, 198],
       [262, 393]], dtype=int64)
```

In [11]:
```
Accuracy_Score = accuracy_score(var['ATTORNEY'],predict > 0.5)
Accuracy_Score
```

Out[11]:
```
0.6567164179104478
```