```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn.linear_model import LogisticRegression
from optbinning import OptimalBinning
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from IPython.display import Image
import pydotplus
df = pd.read_csv('2022.csv')
```

Logistic Regression

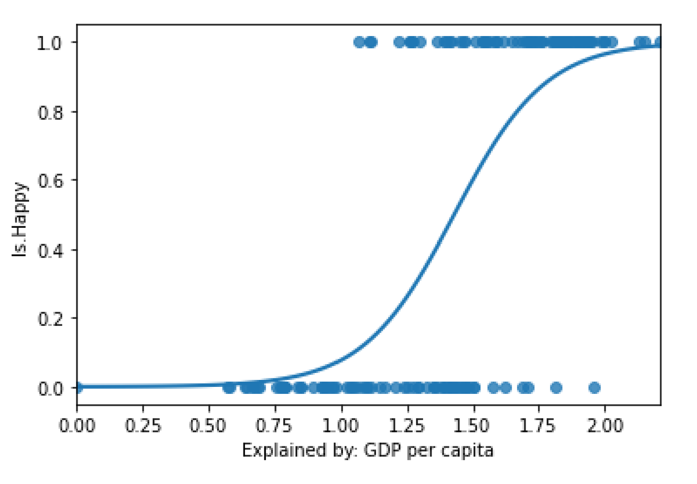
In [64]:
```python
df = df.rename(columns = {"Happiness score":"Score"})
df['Is.Happy'] =[1 if each>np.mean(df.Score) else 0 for each in df.Score]

feature_cols = ['Explained by: GDP per capita', 'Explained by: Social support', 'Explained by: Healthy life exp
X = df[feature_cols].values
y = df['Is.Happy']
```

In [65]:
```python
df['Is.Happy'].value_counts()
```

Out[65]:
```
1    74
0    72
Name: Is.Happy, dtype: int64
```

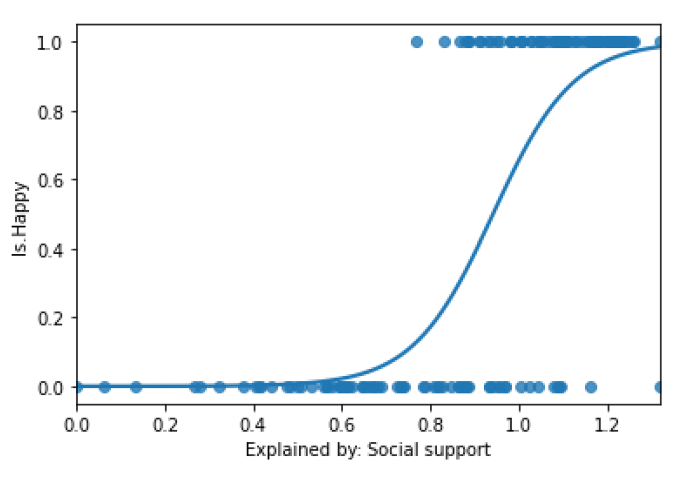
In [105]:
```python
sns.regplot(x=df['Explained by: GDP per capita'], y=y, data=df, logistic=True, ci=None)
```

Out[105]: `<AxesSubplot:xlabel='Explained by: GDP per capita', ylabel='Is.Happy'>`



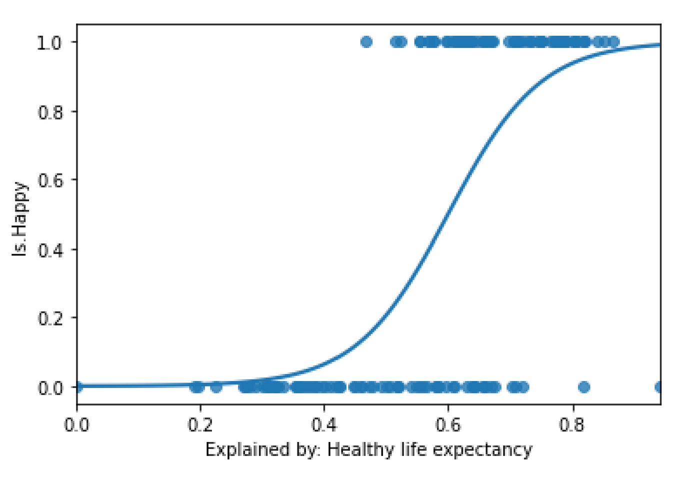
In [106]:
```python
sns.regplot(x=df['Explained by: Social support'], y=y, data=df, logistic=True, ci=None)
```

Out[106]: `<AxesSubplot:xlabel='Explained by: Social support', ylabel='Is.Happy'>`



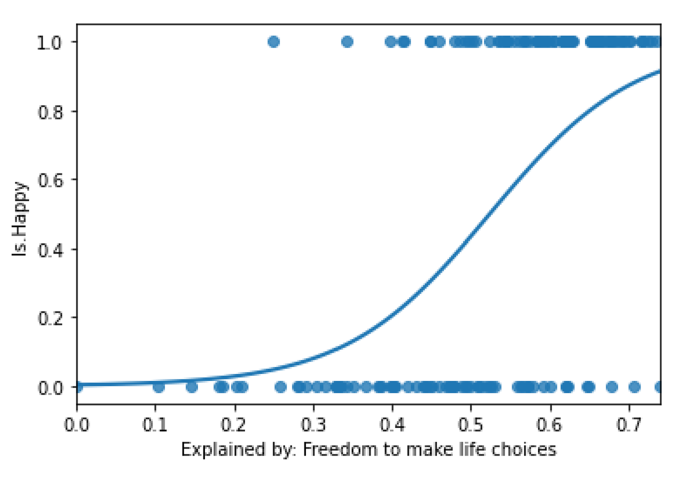
In [107]:
```python
sns.regplot(x=df['Explained by: Healthy life expectancy'], y=y, data=df, logistic=True, ci=None)
```

Out[107]: `<AxesSubplot:xlabel='Explained by: Healthy life expectancy', ylabel='Is.Happy'>`



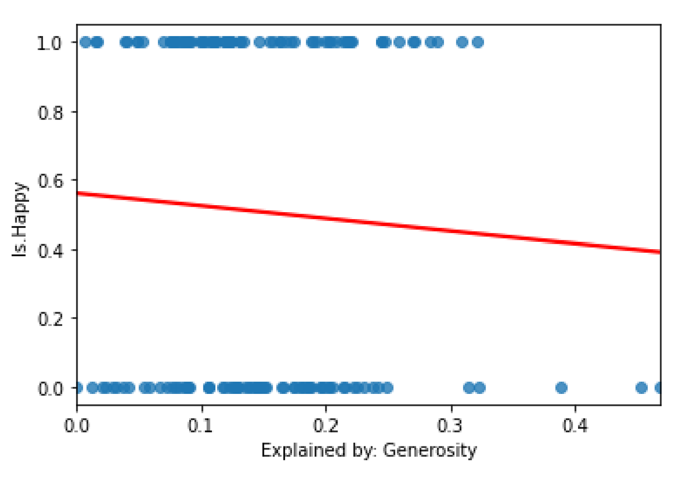
In [108]:
```python
sns.regplot(x=df['Explained by: Freedom to make life choices'], y=y, data=df, logistic=True, ci=None)
```

Out[108]: `<AxesSubplot:xlabel='Explained by: Freedom to make life choices', ylabel='Is.Happy'>`



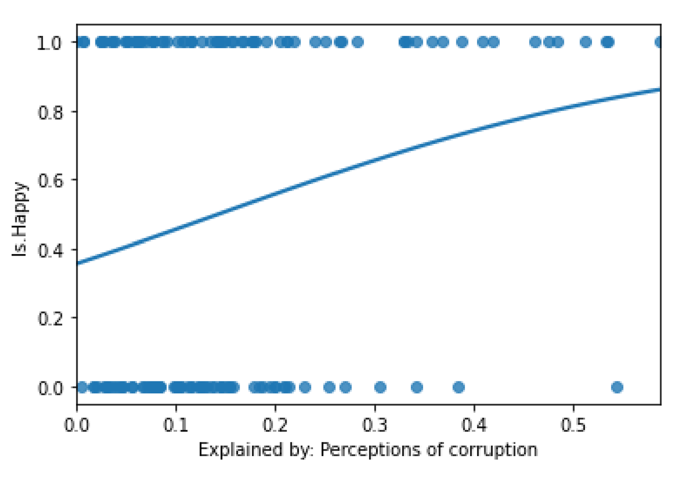
In [109]:
```python
sns.regplot(x=df['Explained by: Generosity'], y=y, data=df,line_kws={'color': 'red'},logistic=True, ci=None)
```

Out[109]: `<AxesSubplot:xlabel='Explained by: Generosity', ylabel='Is.Happy'>`



In [111]:
```python
sns.regplot(x=df['Explained by: Perceptions of corruption'], y=y, data=df, logistic=True, ci=None)
```

Out[111]: `<AxesSubplot:xlabel='Explained by: Perceptions of corruption', ylabel='Is.Happy'>`



In [116]:
```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
import statsmodels.api as sm
from statsmodels.api import Logit, add_constant
X_const = add_constant(X)
logit_model=sm.Logit(y,X_const)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.237120
         Iterations 9
                           Logit Regression Results
==============================================================================
Dep. Variable:               Is.Happy   No. Observations:                  146
Model:                          Logit   Df Residuals:                      139
Method:                           MLE   Df Model:                            6
Date:                Fri, 15 Apr 2022   Pseudo R-squ.:                  0.6579
Time:                        23:34:42   Log-Likelihood:                -34.620
converged:                       True   LL-Null:                       -101.19
Covariance Type:            nonrobust   LLR p-value:                 2.813e-26
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        -19.5489      3.809     -5.132      0.000     -27.015     -12.083
x1             1.1457      1.794      0.639      0.523      -2.370       4.661
x2             8.6156      2.612      3.298      0.001       3.496      13.735
x3             8.2294      3.923      2.098      0.036       0.541      15.918
x4            11.1951      3.192      3.507      0.000       4.939      17.451
x5            -5.1569      4.189     -1.231      0.218     -13.366       3.053
x6            -4.6588      3.378     -1.379      0.168     -11.280       1.963
==============================================================================
```

In [117]:
```python
feature = ['Explained by: Social support','Explained by: Healthy life expectancy', 'Explained by: Freedom to ma
X1 = df[feature]
X1_const = add_constant(X1)
logit_model=sm.Logit(y,X1_const)
result1=logit_model.fit()
print(result1.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.248417
         Iterations 9
                                Results: Logit
==============================================================================================
Model:                        Logit                      Pseudo R-squared:           0.642
Dependent Variable:           Is.Happy                   AIC:                        80.5378
Date:                         2022-04-15 23:34           BIC:                        92.4723
No. Observations:             146                        Log-Likelihood:             -36.269
Df Model:                     3                          LL-Null:                    -101.19
Df Residuals:                 142                        LLR p-value:                5.8735e-28
Converged:                    1.0000                     Scale:                      1.0000
No. Iterations:               9.0000
----------------------------------------------------------------------------------------------
                                          Coef.   Std.Err.    z     P>|z|   [0.025   0.975]
----------------------------------------------------------------------------------------------
const                                   -19.4225   3.7506 -5.1785 0.0000 -26.7736 -12.0714
Explained by: Social support              8.8260   2.2470  3.9279 0.0001   4.4219  13.2300
Explained by: Healthy life expectancy     9.5704   3.0354  3.1530 0.0016   3.6212  15.5196
Explained by: Freedom to make life choices 9.7304  2.9464  3.3025 0.0010   3.9556  15.5051
==============================================================================================
```

In [119]:
```python
X_train,X_test,y_train,y_test=train_test_split(X1,y,test_size=0.3,random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.89
```

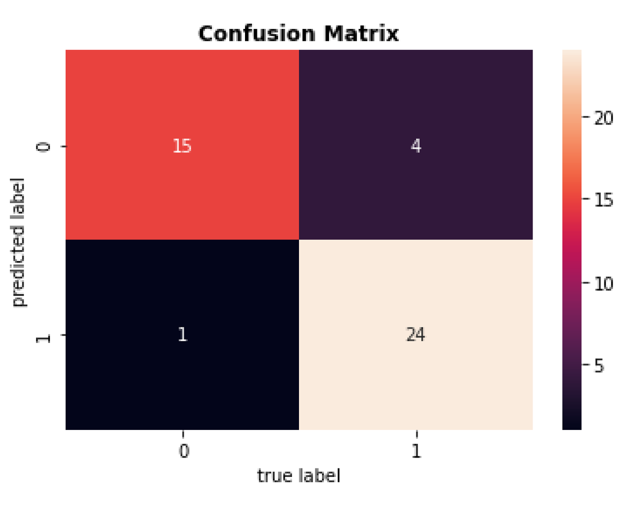
In [120]:
```python
from sklearn import metrics
y_pred = logreg.predict(X_test)
print(f'Accuracy Score:\n{accuracy_score(y_test, y_pred):0.3f}')
probs = logreg.predict_proba(X_test)
print('ROC AUC Score:')
print(roc_auc_score(y_test, probs[:,1]))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Accuracy Score:
0.886
ROC AUC Score:
0.9178947368421053
Precision: 0.8571428571428571
Recall: 0.96
```

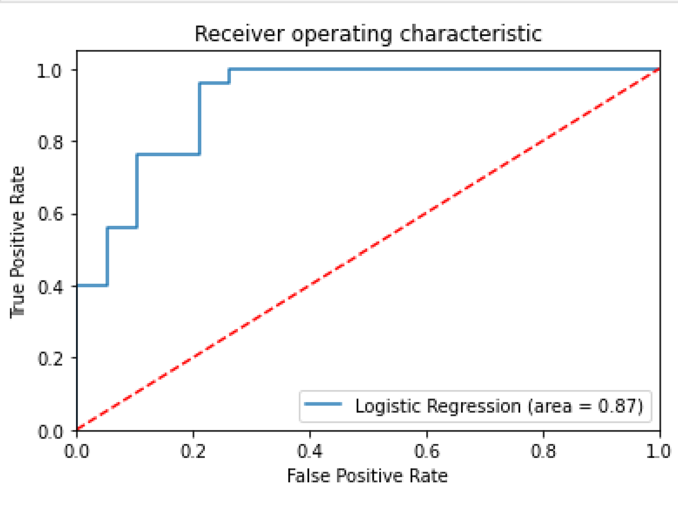
In [121]:
```python
print('\nClassification Report:')
print(classification_report(y_test, y_pred))
import seaborn as sns
mat=confusion_matrix(y_test, y_pred)
sns.heatmap(mat,annot=True)
plt.xlabel('true label')
plt.ylabel('predicted label')
plt.title( 'Confusion Matrix', weight='bold')
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.79      0.86        19
           1       0.86      0.96      0.91        25

    accuracy                           0.89        44
   macro avg       0.90      0.87      0.88        44
weighted avg       0.89      0.89      0.88        44
```

Out[121]: `Text(0.5, 1.0, 'Confusion Matrix')`



In [122]:
```python
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



Decision Tree

In [79]:
```python
df['Happy1'] =['Happy' if each>np.mean(df.Score) else "Nothappy" for each in df.Score]
df['Happy1'].value_counts()
```

Out[79]:
```
Happy       74
Nothappy    72
Name: Happy1, dtype: int64
```

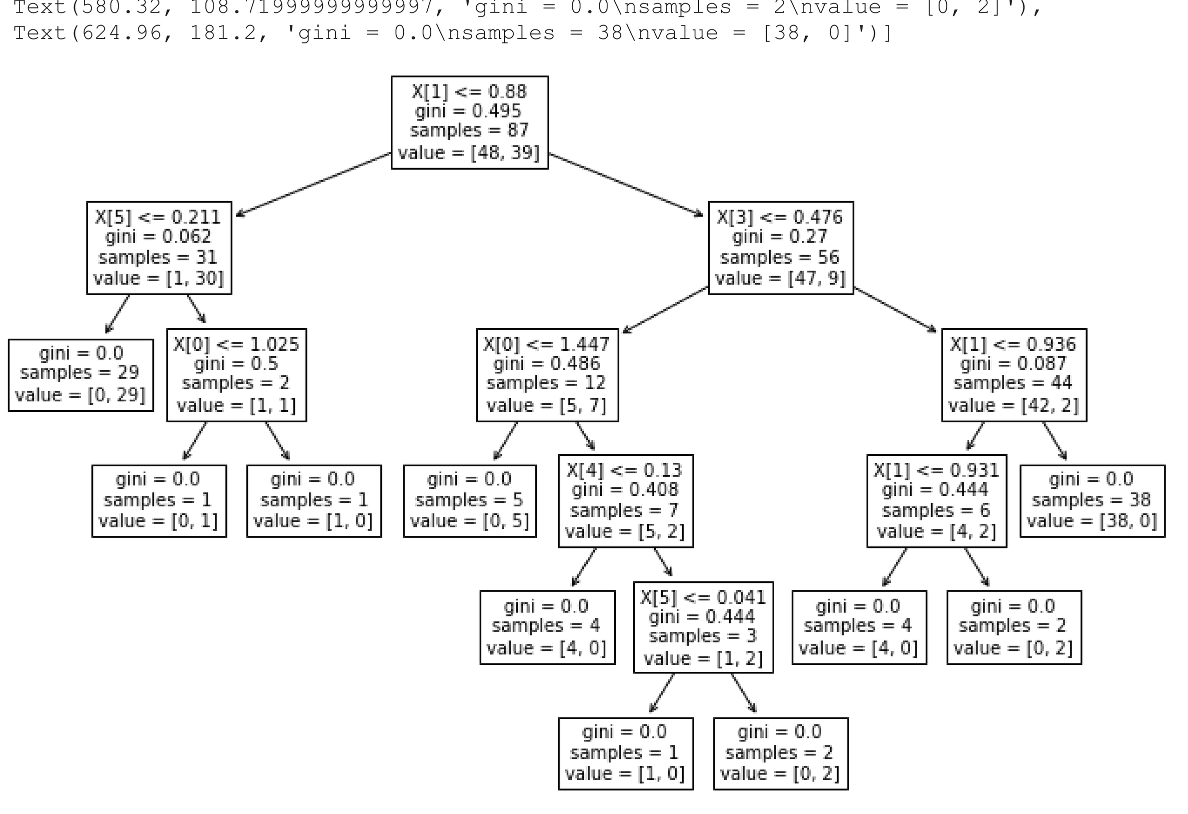
In [90]:
```python
predictors = df[['Explained by: GDP per capita', 'Explained by: Social support', 'Explained by: Healthy life ex
targets = df['Happy1']
pred_train, pred_test, tar_train, tar_test  =   train_test_split(predictors, targets, test_size=.4)
clf_gini = DecisionTreeClassifier(criterion='gini', random_state=0)

clf_gini.fit(pred_train,tar_train)
```

Out[90]: `DecisionTreeClassifier(random_state=0)`

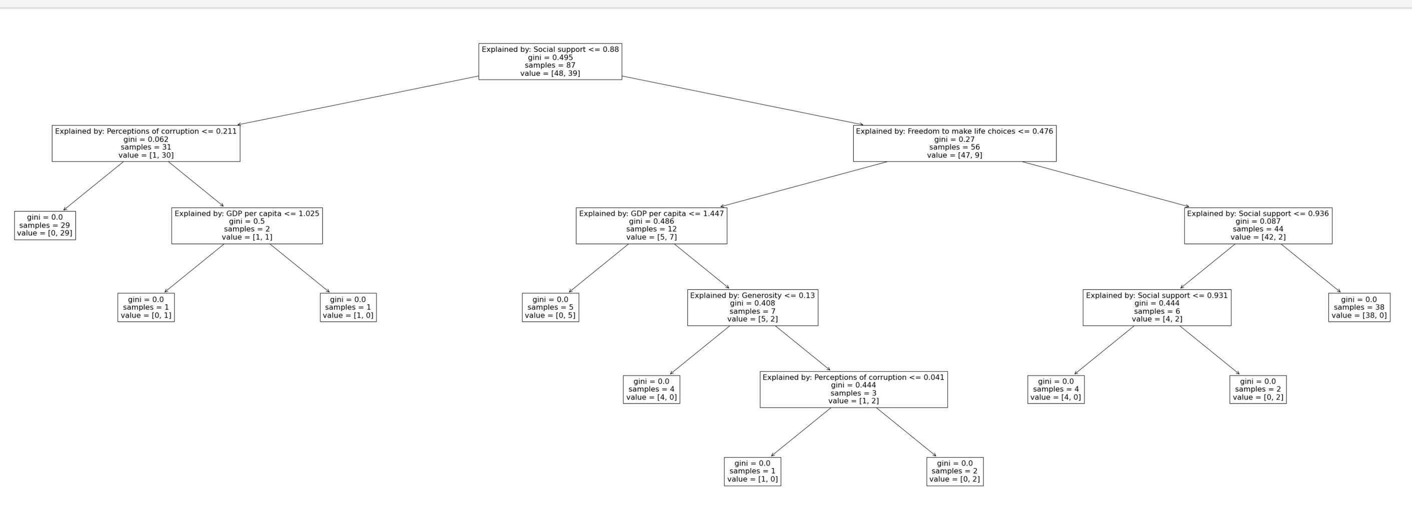
In [91]:
```python
y_pred_gini = clf_gini.predict(pred_test)
```

In [92]:
```python
plt.figure(figsize=(12,8))

from sklearn import tree

tree.plot_tree(clf_gini.fit(pred_train, tar_train))
```

Out[92]:
```
[Text(267.84000000000003, 398.64, 'X[1] <= 0.88\ngini = 0.495\nsamples = 87\nvalue = [48, 39]'),
 Text(89.28, 326.15999999999997, 'X[5] <= 0.211\ngini = 0.062\nsamples = 31\nvalue = [1, 30]'),
 Text(44.64, 253.67999999999998, 'gini = 0.0\nsamples = 29\nvalue = [0, 29]'),
 Text(133.92000000000002, 253.67999999999998, 'X[0] <= 1.025\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(89.28, 181.2, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(178.56, 181.2, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(446.4, 326.15999999999997, 'X[3] <= 0.476\ngini = 0.27\nsamples = 56\nvalue = [47, 9]'),
 Text(312.48, 253.67999999999998, 'X[0] <= 1.447\ngini = 0.486\nsamples = 12\nvalue = [5, 7]'),
 Text(267.84000000000003, 181.2, 'gini = 0.0\nsamples = 5\nvalue = [0, 5]'),
 Text(357.12, 181.2, 'X[4] <= 0.13\ngini = 0.408\nsamples = 7\nvalue = [5, 2]'),
 Text(312.48, 108.71999999999997, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(401.76, 108.71999999999997, 'X[5] <= 0.041\ngini = 0.444\nsamples = 3\nvalue = [1, 2]'),
 Text(357.12, 36.23999999999995, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(446.4, 36.23999999999995, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(580.32, 253.67999999999998, 'X[1] <= 0.936\ngini = 0.087\nsamples = 44\nvalue = [42, 2]'),
 Text(535.6800000000001, 181.2, 'X[1] <= 0.931\ngini = 0.444\nsamples = 6\nvalue = [4, 2]'),
 Text(491.04, 108.71999999999997, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(580.32, 108.71999999999997, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(624.96, 181.2, 'gini = 0.0\nsamples = 38\nvalue = [38, 0]')]
```



In [103]:
```python
from sklearn import tree
plt.figure(figsize=(60,20))
_ = tree.plot_tree(clf_gini, feature_names = pred_train.columns)
plt.show()
```



The best predictor of happiness score here would be social support