In [38]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score
```

In [2]:

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

In [16]:

```
#convert string to number
df = df.replace('y', 1).replace('n', 0).replace('?', np.nan).dropna().replace('democrat',0).repl
ace('republican',1)
```

In [17]:

```
df.head(5)
```

Out[17]:

sue	crime	duty_free_exports	export_administration_act_south_africa	political_par
	1.0	1.0	1.0	0
	1.0	0.0	1.0	1
	0.0	1.0	1.0	0
	0.0	1.0	1.0	0
	0.0	1.0	1.0	0

In [18]:

```
#define features and label
y = df['political_party'].values
X = df.drop('political_party', axis = 1).values
```

In [19]:

```
#seperate training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state = 121, s
tratify = y)
```

In [20]:

```
#set k-value of knn model
knn = KNeighborsClassifier(n_neighbors = 8)
```

In [21]:

```
#fitting on training data set
knn.fit(X_train,y_train)
```

Out[21]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=8, p=2,
           weights='uniform')
```

In [22]:

```
#make prediction on test data set
y_pred = knn.predict(X_test)
```

In [23]:

```
#get matrix: [[tp,fn],[fp,tn]]
print(confusion_matrix(y_test,y_pred))
```

```
[[44  6]
 [ 1 42]]
```

In [24]:

```
print(classification_report(y_test,y_pred))
```

```
             precision    recall  f1-score   support

          0       0.98      0.88      0.93        50
          1       0.88      0.98      0.92        43

avg / total       0.93      0.92      0.92        93
```

In [36]:

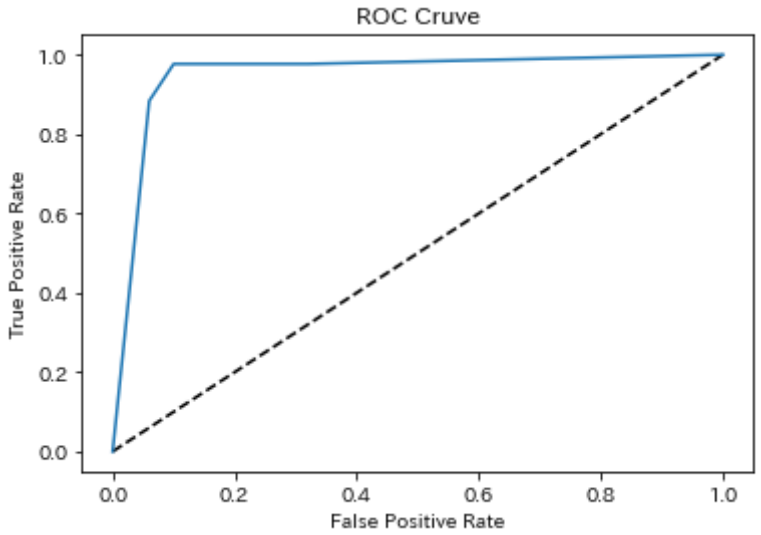
```
y_pred_prob = knn.predict_proba(X_test)[:,1]
```

In [26]:

```
fpr,tpr,threshold = roc_curve(y_test, y_pred_prob)
```

In [28]:

```
plt.plot([0,1],[0,1], 'k--')
plt.plot(fpr,tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Cruve')
plt.show()
```



In [30]:

```
#calculate AUC(area under curve)
roc_auc_score(y_test, y_pred_prob)
```

Out[30]:

0.9506976744186046

In [35]:

```
#use cross-fold validation to get 5 auc scores
cv_scores = cross_val_score(knn, X, y,cv = 5, scoring = 'roc_auc')
print(cv_scores)
```

[0.94909091 0.94636364 0.97454545 1. 0.91765873]

In [115]:

```
#tuning hyperparameter
param_grid = {'n_neighbors': list(range(1,50))} #define hyperparameter range
knn2 = KNeighborsClassifier()
knn2_cv = GridSearchCV(knn2, param_grid, cv = 5, scoring='accuracy')#define cross-fold validatio
n parameters
```

In [116]:

```
knn2_cv.fit(X_train,y_train) #test k-value from 1 to 50
```

Out[116]:

```
GridSearchCV(cv=5, error_score='raise',
       estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='mink
owski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform'),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 3
5, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [117]:

```
print("Best k-value:{}".format(knn2_cv.best_params_)) #find the best k-value
```

```
Best k-value:{'n_neighbors': 5}
```

In [118]:

```
print("Best Score:{}".format(knn2_cv.best_score_)) #find the best performance socre of the model
```

```
Best Score:0.9280575539568345
```