In [15]:

```
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 23 14:30:30 2018

@author: filipe.luz
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import  classification_report,confusion_matrix, roc_curve



df = pd.read_csv('C:\\Users\\filipe.luz\\Desktop\\Desafio_BB\\census.csv')

df.isnull().sum()

df_new = df.dropna()

df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5532 entries, 0 to 5531
Data columns (total 14 columns):
age                5532 non-null int64
workclass          5532 non-null object
education_level    5532 non-null object
education-num      5532 non-null float64
marital-status     5532 non-null object
occupation         5532 non-null object
relationship       5532 non-null object
race               5532 non-null object
sex                5532 non-null object
capital-gain       5532 non-null float64
capital-loss       5532 non-null float64
hours-per-week     5532 non-null float64
native-country     5532 non-null object
income             5532 non-null object
dtypes: float64(4), int64(1), object(9)
memory usage: 648.3+ KB
```

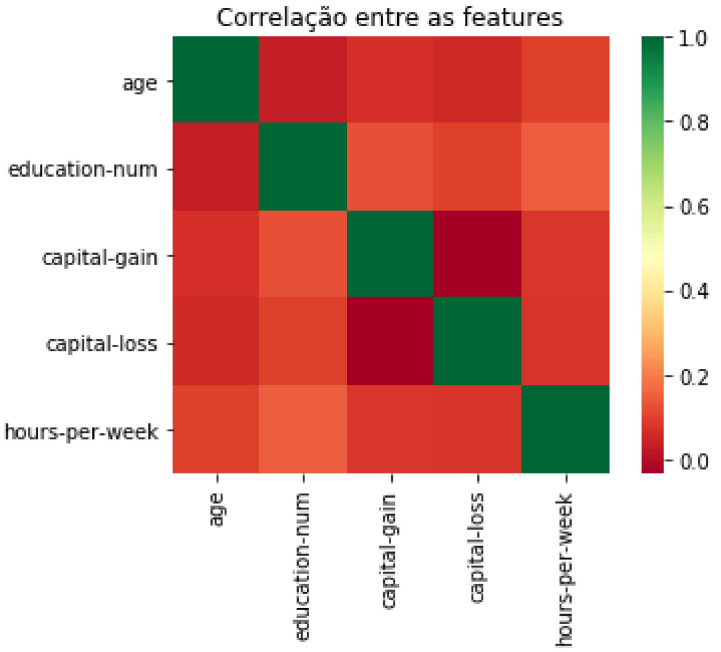
In [16]:

```
df_new = pd.concat([
        df_new.select_dtypes([], ['object']),
        df_new.select_dtypes(['object']).apply(pd.Series.astype, dtype='category')
        ], axis=1).reindex(df_new.columns, axis=1)

df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5532 entries, 0 to 5531
Data columns (total 14 columns):
age                5532 non-null int64
workclass          5532 non-null category
education_level    5532 non-null category
education-num      5532 non-null float64
marital-status     5532 non-null category
occupation         5532 non-null category
relationship       5532 non-null category
race               5532 non-null category
sex                5532 non-null category
capital-gain       5532 non-null float64
capital-loss       5532 non-null float64
hours-per-week     5532 non-null float64
native-country     5532 non-null category
income             5532 non-null category
dtypes: category(9), float64(4), int64(1)
memory usage: 312.3 KB
```

In [17]:

```
def heat_corr (data):
    '''Analisando a correlação entre as features
       do dataset
    '''
    sns.heatmap(data.corr(),square=True,cmap='RdYlGn')
    plt.title('Correlação entre as features')
    plt.show()


#Chamando função para analisar a correlação entre as features originais do dataset
heat_corr(df_new)
```



In [18]:

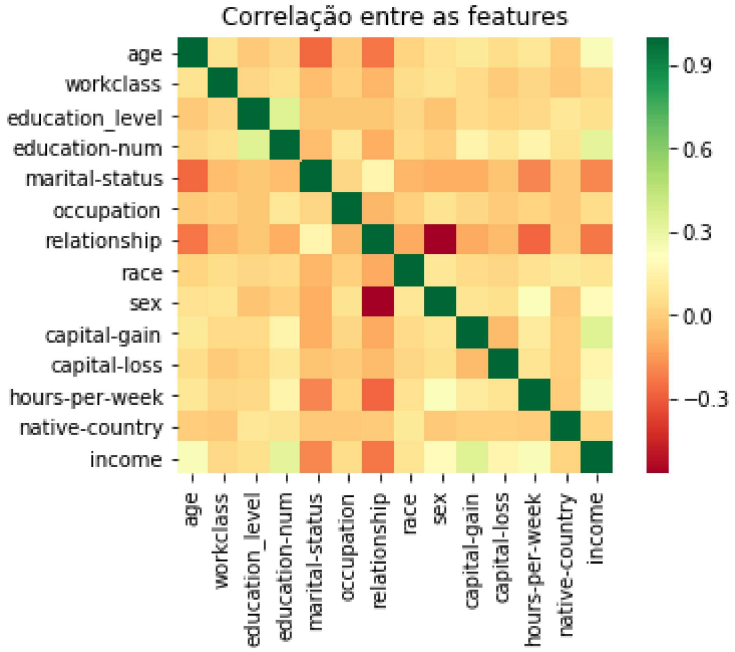
```
def compute_log_loss(predicted, actual, eps=1e-14):
    '''
    Computa a medida de avaliação sobre perda (log loss)
    Utilizada para avaliação do modelo
    '''

    predicted = np.clip( predicted , eps , 1- eps)
    loss = -1 * np.mean(  actual * np.log(predicted) + (1 - actual) * np.log(1 - predic
ted))

    return loss

def ac_desemp (cm):
    ac = 0
    ac = (cm[0,0] + cm[1,1]) / (cm[0,0] + cm[1,1] + cm[1,0] + cm[0,1])

    print("Acuracia do modelo é : " + str(ac))
```

In [19]:

```
#Generating the X and y datasets

X =  (df_new.drop(['income'], axis =1) )
y =   np.where(df_new.iloc[:,-1] == '>50K',1,0)


#One Hot Encoding fuel rail column
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelEncoder= LabelEncoder()
for i in range(0,len(X.columns)):
    X.iloc[:,i] = labelEncoder.fit_transform(X.iloc[:,i])


hotEncoder = OneHotEncoder()
for i in range(0,len(X.columns)):
    X.iloc[:,i] = labelEncoder.fit_transform(X.iloc[:,i])
```

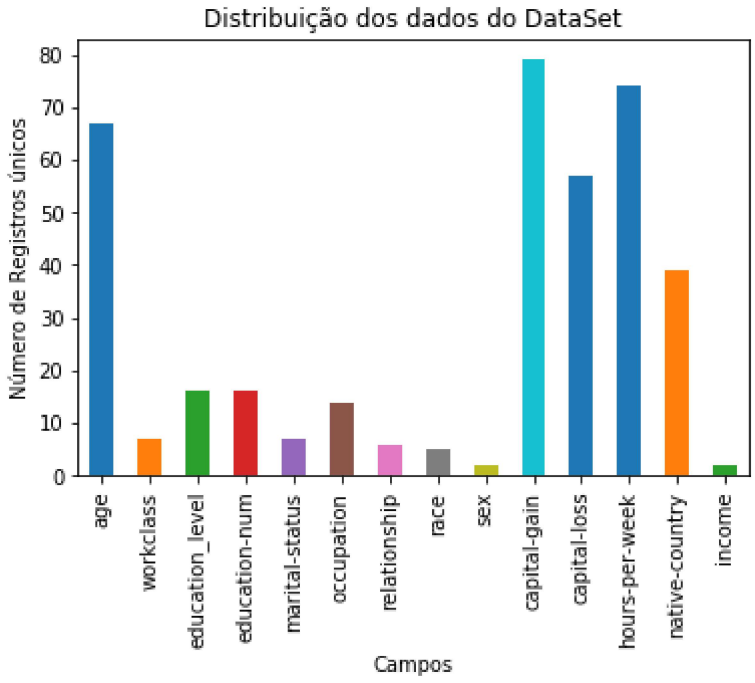
In [20]:

```
#Creating a new df to can analyse the correlation after OneHotEncoder
acc = X.copy()
acc['income'] = y.copy()
heat_corr(acc)
```



In [21]:

```
def distribuicao (data):
    '''
    Esta função exibirá a quantidade de registros únicos para cada coluna
    existente no dataset

    dataframe -> Histogram
    '''
    # Calculando valores unicos para cada label: num_unique_labels
    num_unique_labels = data.apply(pd.Series.nunique)

    # plotando valores
    num_unique_labels.plot( kind='bar')

    # Nomeando os eixos
    plt.xlabel('Campos')
    plt.ylabel('Número de Registros únicos')
    plt.title('Distribuição dos dados do DataSet')

    # Exibindo gráfico
    plt.show()

#Chamando função para analisar a distribuição dos dados no data set
distribuicao(acc)
```

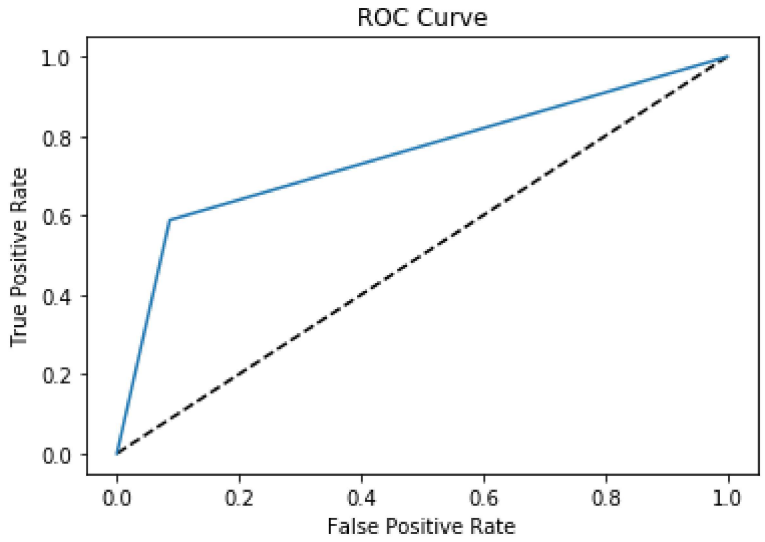


In [22]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y , test_size=0.4, random_state
= 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)



classifier = RandomForestClassifier(n_estimators=40, criterion='entropy', random_state=
0)

classifier.fit(X_train, y_train)

y_pred_train = classifier.predict_proba(X_train)


y_pred = classifier.predict_proba(X_test)


# Compute predicted probabilities: y_pred_prob
y_pred_prob = (y_pred)[:,1]
```

In [23]:

```
# Generate ROC curve values: fpr, tpr, thresholds
fpr, tpr,threshold = roc_curve(y_test,y_pred_prob.round())

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



In [24]:

```
# Making the Confusion Matrix
cm = confusion_matrix(y_test, y_pred[:,1].round())
ac_desemp(cm)
# Compute metrics
print(classification_report(y_test, y_pred[:,1].round()))
print("Log Loss Result: " + str( compute_log_loss(y_pred[:,1],y_test)))
```

```
Acuracia do modelo é : 0.8309986443741527
             precision    recall  f1-score   support

          0       0.87      0.91      0.89      1652
          1       0.70      0.59      0.64       561

avg / total       0.82      0.83      0.83      2213

Log Loss Result: 0.474954156448957
```