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

In [2]:
```
data = pd.read_csv("diabetes.csv")
data.head()
```

Out[2]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | Pedigree | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

In [3]:
```
data.isnull().any()
```

Out[3]:
```
Pregnancies      False
Glucose          False
BloodPressure    False
SkinThickness    False
Insulin          False
BMI              False
Pedigree         False
Age              False
Outcome          False
dtype: bool
```

In [4]:
```
data.describe().T
```

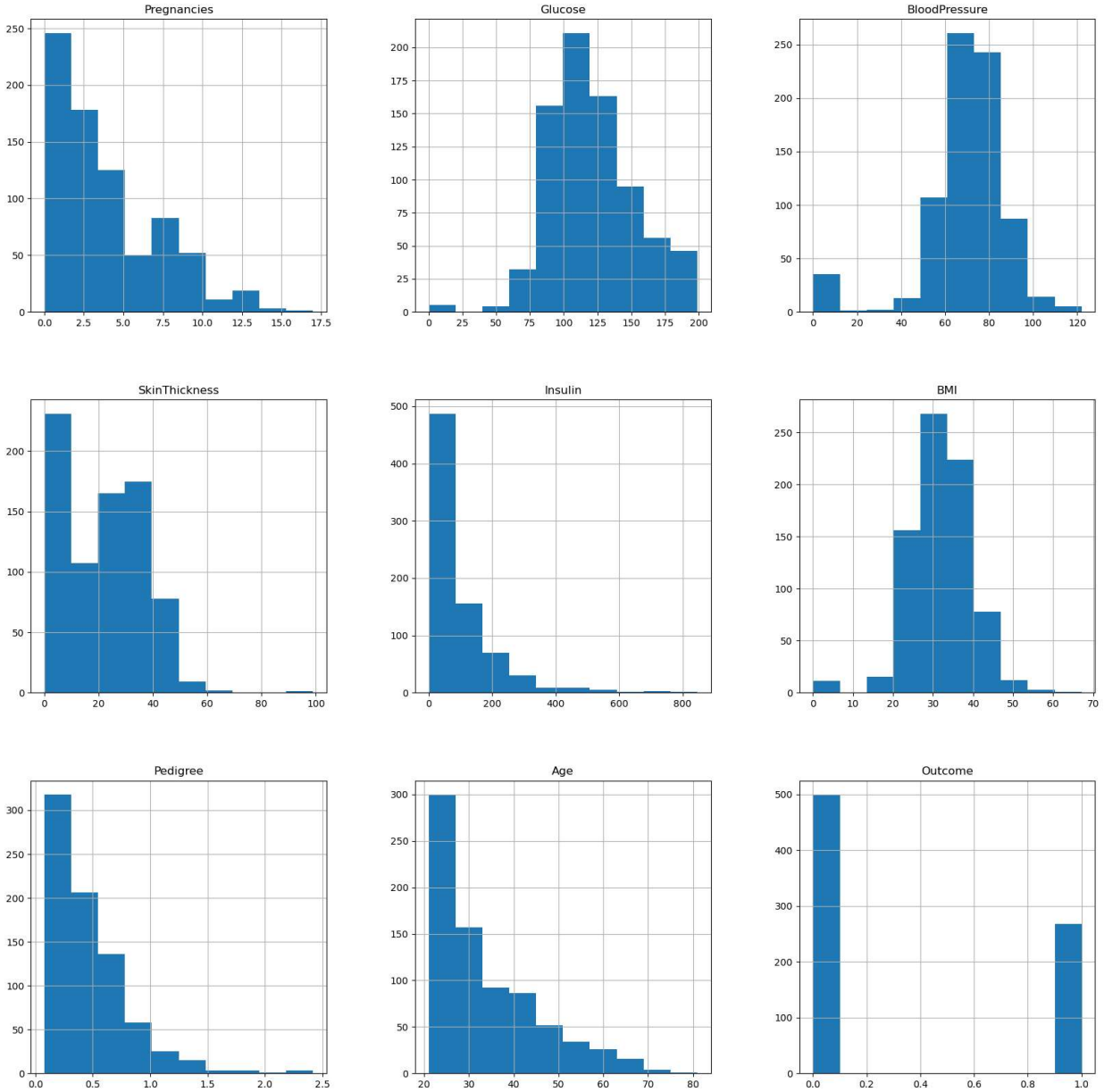
Out[4]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Pregnancies** | 768.0 | 3.845052 | 3.369578 | 0.000 | 1.00000 | 3.0000 | 6.00000 | 17.00 |
| **Glucose** | 768.0 | 120.894531 | 31.972618 | 0.000 | 99.00000 | 117.0000 | 140.25000 | 199.00 |
| **BloodPressure** | 768.0 | 69.105469 | 19.355807 | 0.000 | 62.00000 | 72.0000 | 80.00000 | 122.00 |
| **SkinThickness** | 768.0 | 20.536458 | 15.952218 | 0.000 | 0.00000 | 23.0000 | 32.00000 | 99.00 |
| **Insulin** | 768.0 | 79.799479 | 115.244002 | 0.000 | 0.00000 | 30.5000 | 127.25000 | 846.00 |
| **BMI** | 768.0 | 31.992578 | 7.884160 | 0.000 | 27.30000 | 32.0000 | 36.60000 | 67.10 |
| **Pedigree** | 768.0 | 0.471876 | 0.331329 | 0.078 | 0.24375 | 0.3725 | 0.62625 | 2.42 |
| **Age** | 768.0 | 33.240885 | 11.760232 | 21.000 | 24.00000 | 29.0000 | 41.00000 | 81.00 |
| **Outcome** | 768.0 | 0.348958 | 0.476951 | 0.000 | 0.00000 | 0.0000 | 1.00000 | 1.00 |

In [5]:
```
data_copy = data.copy(deep = True)
data_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = data_copy[['G
data_copy.isnull().sum()
```

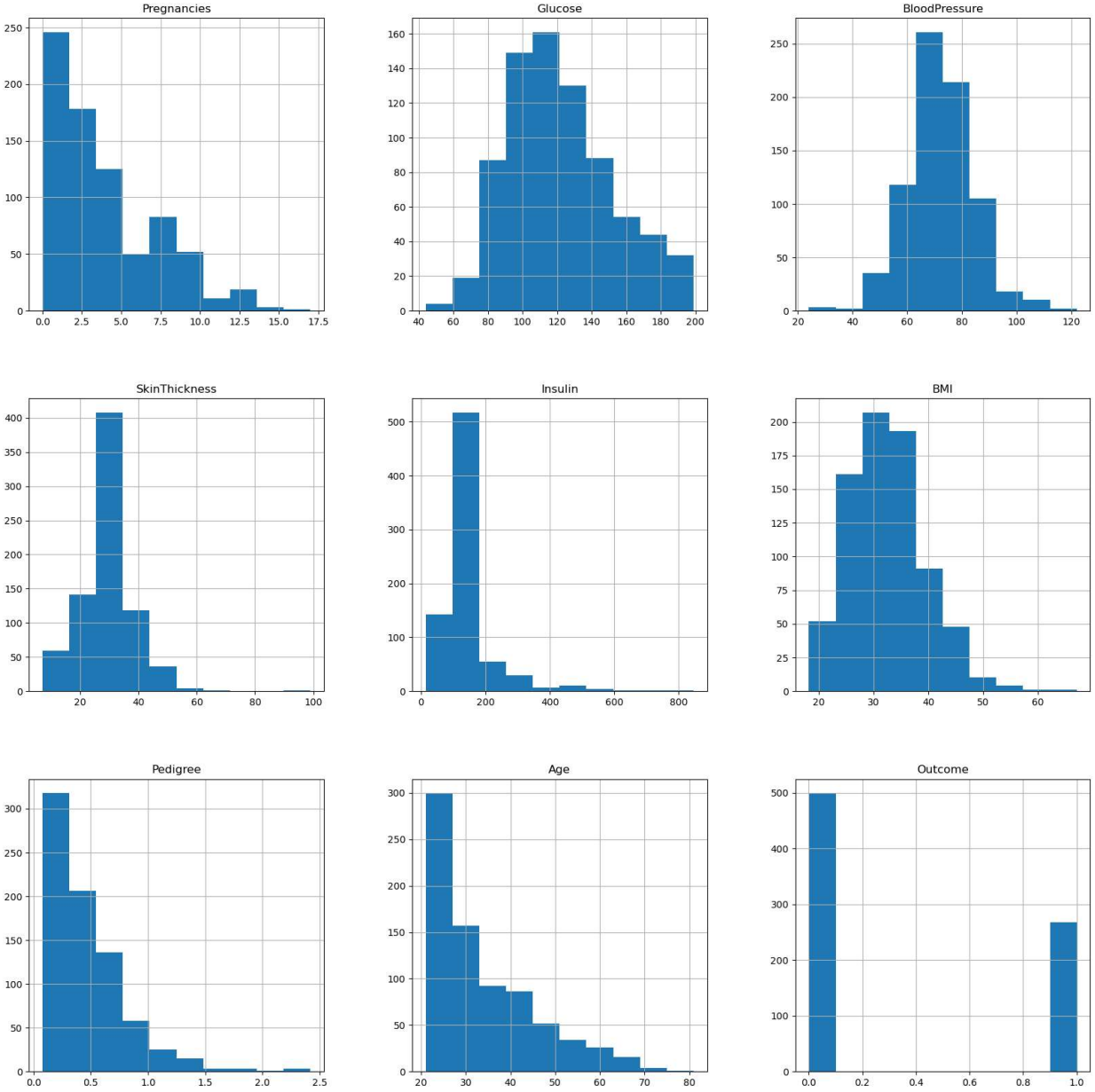
Out[5]:
```
Pregnancies         0
Glucose             5
BloodPressure      35
SkinThickness     227
Insulin           374
BMI                11
Pedigree            0
Age                 0
Outcome             0
dtype: int64
```

In [6]:
```
p = data.hist(figsize = (20,20))
```

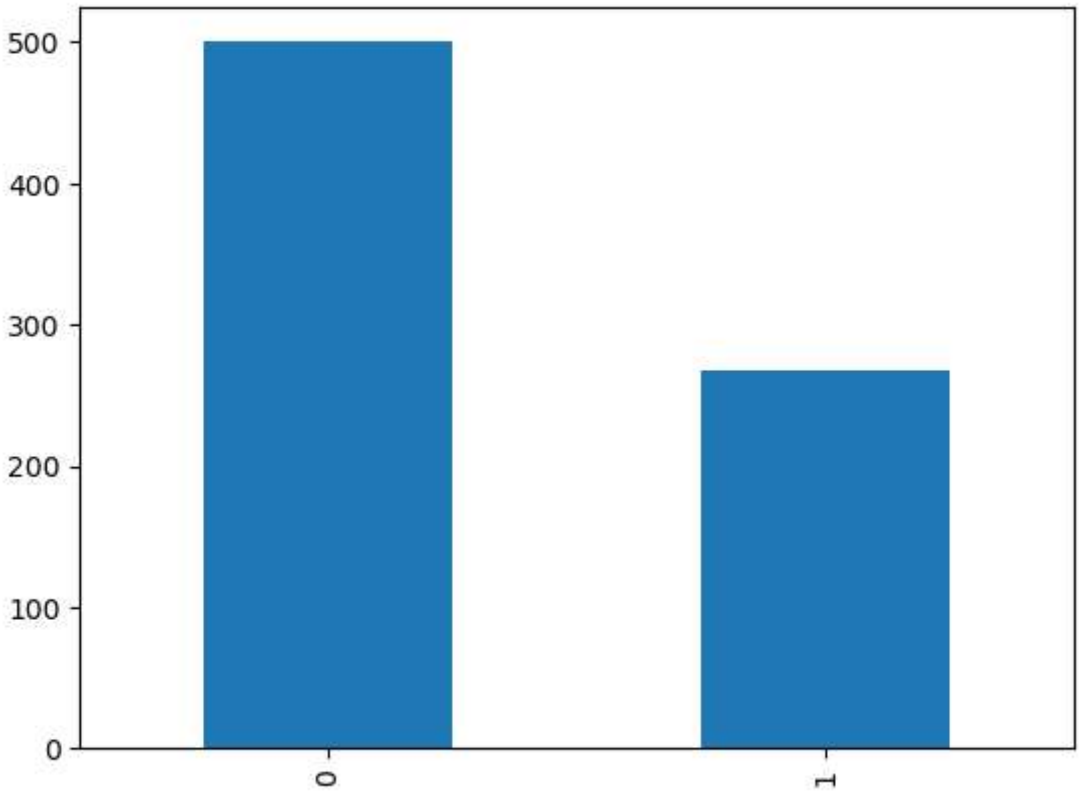



In [7]:
```
data_copy['Glucose'].fillna(data_copy['Glucose'].mean(), inplace = True)
data_copy['BloodPressure'].fillna(data_copy['BloodPressure'].mean(), inplace = True)
data_copy['SkinThickness'].fillna(data_copy['SkinThickness'].median(), inplace = True)
data_copy['Insulin'].fillna(data_copy['Insulin'].median(), inplace = True)
data_copy['BMI'].fillna(data_copy['BMI'].median(), inplace = True)
```

In [8]:
```
p = data_copy.hist(figsize = (20,20))
```




In [9]:
```
p=data.Outcome.value_counts().plot(kind="bar")
```




In [10]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X = pd.DataFrame(sc_X.fit_transform(data_copy.drop(["Outcome"], axis =1),),columns=['P
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
```

In [11]:
```
X.head()
```

Out[11]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunctio |
|---|---|---|---|---|---|---|---|
| **0** | 0.639947 | 0.865108 | -0.033518 | 0.670643 | -0.181541 | 0.166619 | 0.4684 |
| **1** | -0.844885 | -1.206162 | -0.529859 | -0.012301 | -0.181541 | -0.852200 | -0.3650 |
| **2** | 1.233880 | 2.015813 | -0.695306 | -0.012301 | -0.181541 | -1.332500 | 0.6043 |
| **3** | -0.844885 | -1.074652 | -0.529859 | -0.695245 | -0.540642 | -0.633881 | -0.9207 |
| **4** | -1.141852 | 0.503458 | -2.680669 | 0.670643 | 0.316566 | 1.549303 | 5.4849 |

In [12]:
```
y =data_copy.Outcome
```

In [13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_stat
```

In [14]:
```
from sklearn.neighbors import KNeighborsClassifier

train_scores = []
test_scores = []

for i in range(1,15):
    knn = KNeighborsClassifier(i)
```

```
    knn.fit(X_train, y_train)
    train_scores.append(knn.score(X_train, y_train))
    test_scores.append(knn.score(X_test, y_test))
```

In [15]:
```
max_test_score =max(test_scores)
```

In [17]:
```
test_score_index = [i for i, v in enumerate(test_scores) if v== max_test_score]
print('Max test score {} % and k = {}'.format(max_test_score*100,list(map(lambda x: x+
```

```
Max test score 76.5625 % and k = [11]
```

In [18]:
```
knn = KNeighborsClassifier(11)

knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

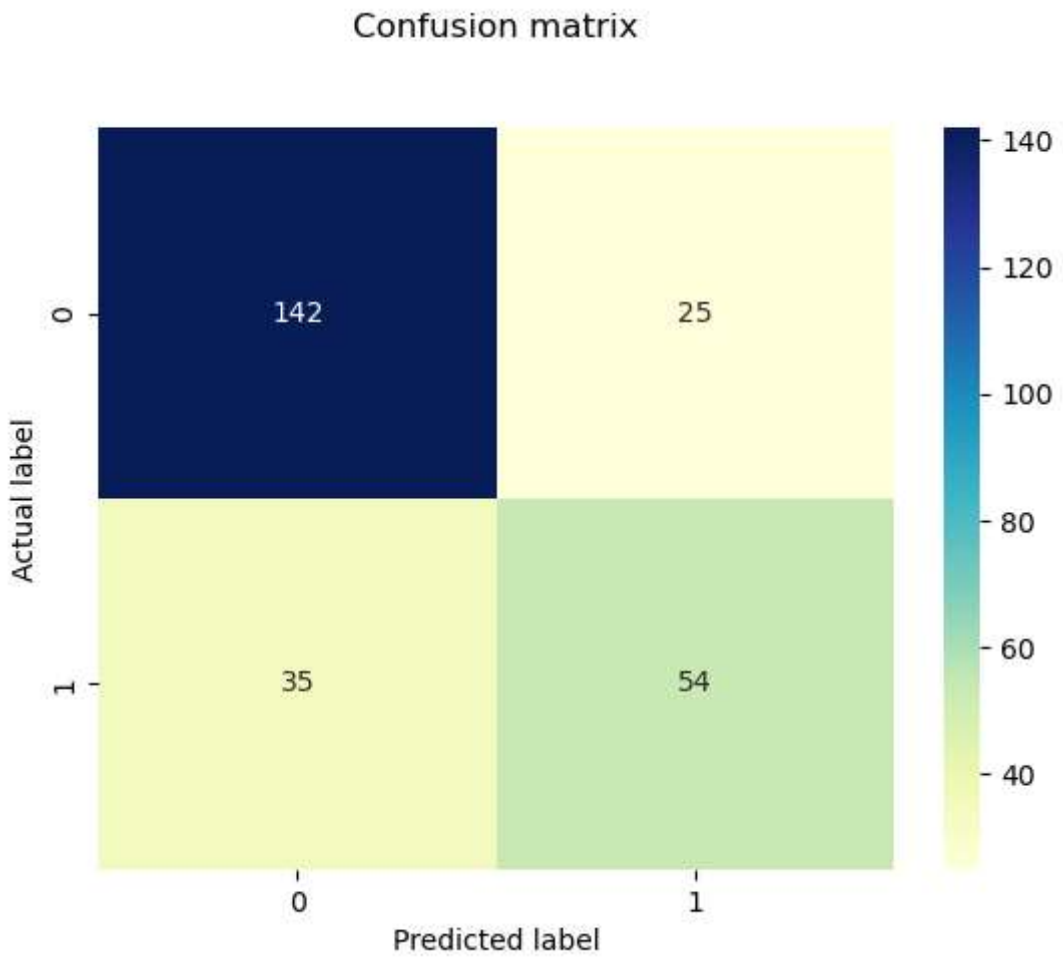
Out[18]:
```
0.765625
```

In [19]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, f
y_pred = knn.predict(X_test)

cnf_matrix = confusion_matrix(y_test, y_pred)
```

In [22]:
```
import seaborn as sns
import matplotlib.pyplot as plt
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[22]:
```
Text(0.5, 23.52222222222222, 'Predicted label')
```




In [23]:
```
def model_evaluation(y_test, y_pred, model_name):
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    f2 = fbeta_score(y_test, y_pred, beta = 2.0)

    results = pd.DataFrame([[model_name, acc, prec, rec, f1, f2]],
                      columns = ["Model", "Accuracy", "Precision", "Recall",
                                 "F1 SCore", "F2 Score"])
    results = results.sort_values(["Precision", "Recall", "F2 Score"], ascending = Fal
    return results



model_evaluation(y_test, y_pred, "KNN")
```

Out[23]:

| | Model | Accuracy | Precision | Recall | F1 SCore | F2 Score |
|---|---|---|---|---|---|---|
| **0** | KNN | 0.765625 | 0.683544 | 0.606742 | 0.642857 | 0.62069 |

In [24]:
```
# Alternate way
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

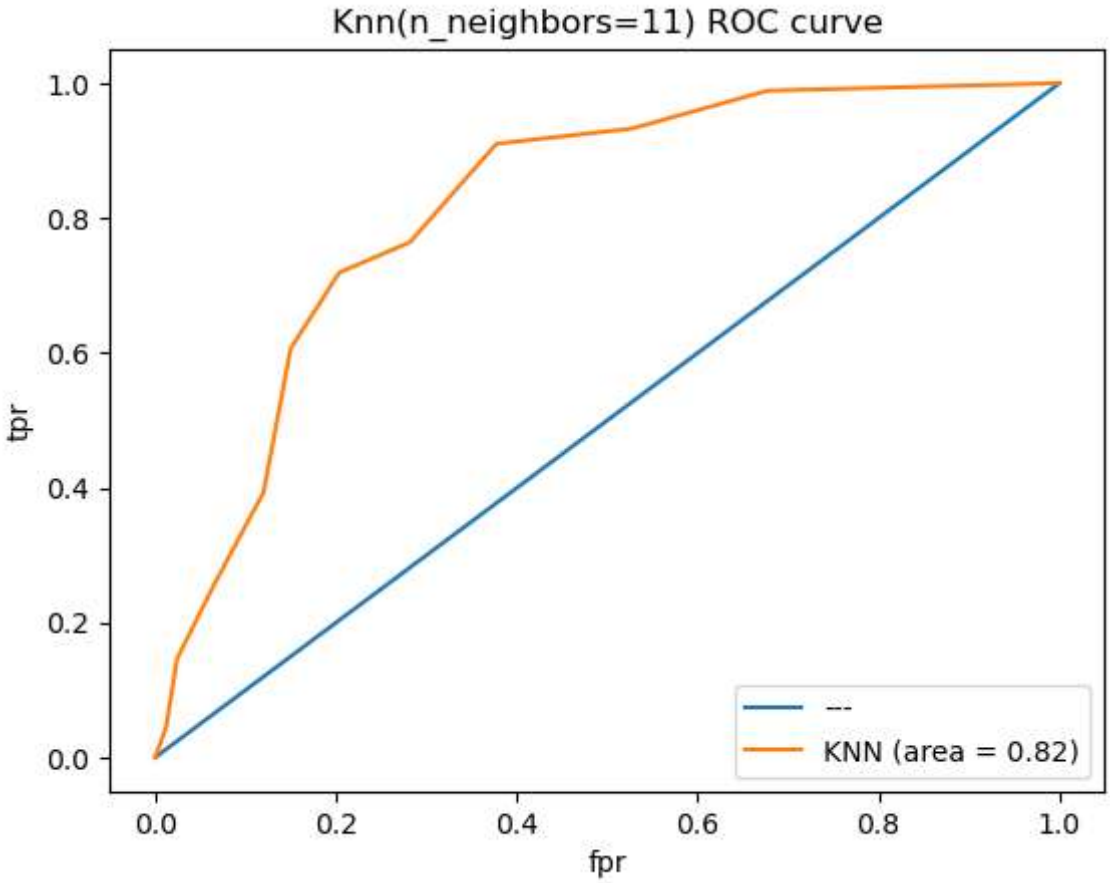
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.80 | 0.85 | 0.83 | 167 |
| 1 | 0.68 | 0.61 | 0.64 | 89 |
| accuracy | | | 0.77 | 256 |
| macro avg | 0.74 | 0.73 | 0.73 | 256 |
| weighted avg | 0.76 | 0.77 | 0.76 | 256 |

In [25]:
```
from sklearn.metrics import auc, roc_auc_score, roc_curve

y_pred_proba = knn.predict_proba(X_test)[:,-1]
fpr, tpr, threshold = roc_curve(y_test, y_pred_proba)
```

In [26]:
```
classifier_roc_auc = roc_auc_score(y_test, y_pred_proba)
plt.plot([0,1],[0,1], label = "---")

plt.plot(fpr, tpr, label ='KNN (area = %0.2f)' % classifier_roc_auc)
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.title('Knn(n_neighbors=11) ROC curve')
plt.legend(loc="lower right", fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
plt.show()
```




In [27]:
```
from sklearn.model_selection import GridSearchCV
parameters_grid = {"n_neighbors": np.arange(0,50)}
knn= KNeighborsClassifier()
```

```
knn_GSV = GridSearchCV(knn, param_grid=parameters_grid, cv = 5)
knn_GSV.fit(X, y)
```

Out[27]:

```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': array([ 0,  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, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])})
```

In [28]:

```
print("Best Params" ,knn_GSV.best_params_)
print("Best score" ,knn_GSV.best_score_)
```

```
Best Params {'n_neighbors': 25}
Best score 0.7721840251252015
```

In [ ]: