Full-name of dear student: Thai Thi Kim Yen

Dear student code: 47.01.104.250

# Import Libraries

In [18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix
```

# Loading Data

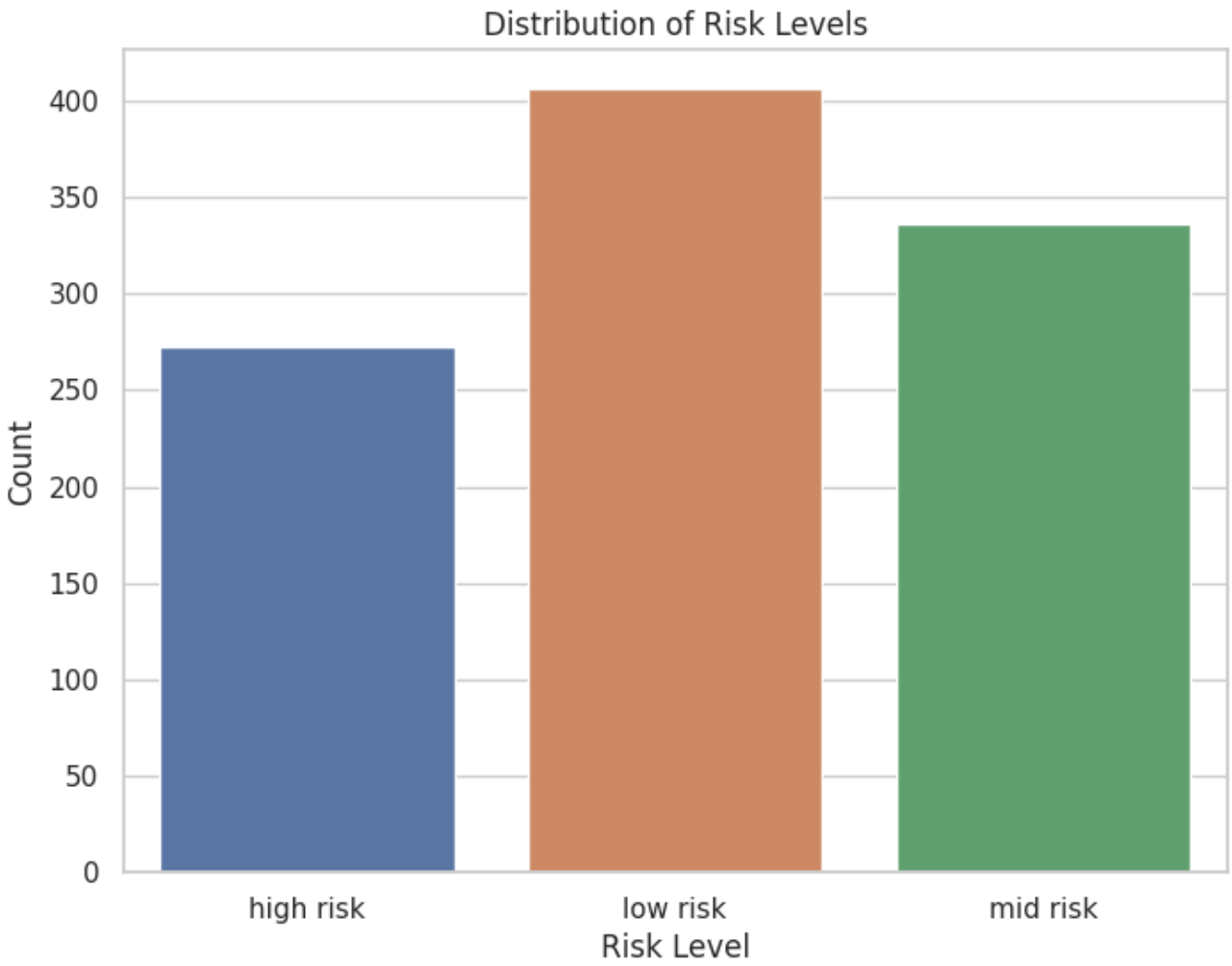
In [19]:
```
data = pd.read_csv('Maternal Health Risk Data Set.csv')
data.head()
```

Out[19]:

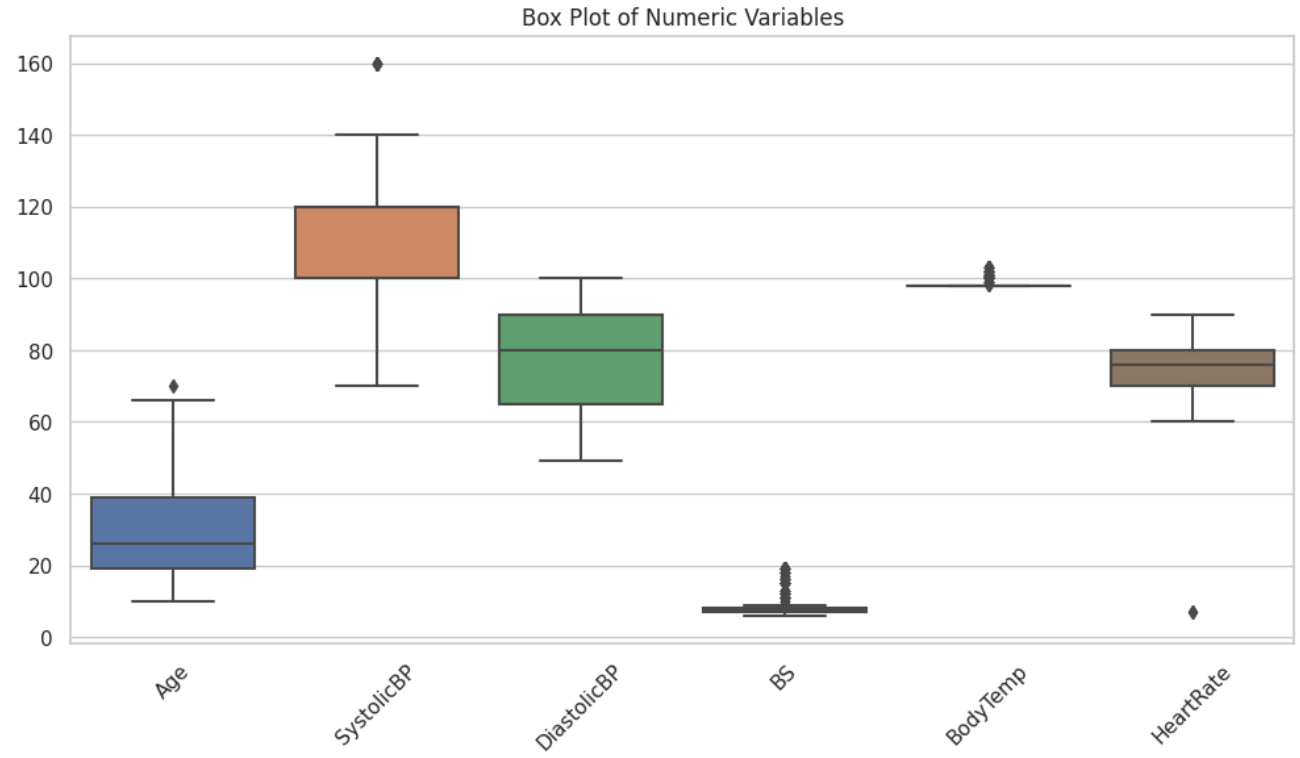
| | Age | SystolicBP | DiastolicBP | BS | BodyTemp | HeartRate | RiskLevel |
|---|---|---|---|---|---|---|---|
| **0** | 25 | 130 | 80 | 15.0 | 98.0 | 86 | high risk |
| **1** | 35 | 140 | 90 | 13.0 | 98.0 | 70 | high risk |
| **2** | 29 | 90 | 70 | 8.0 | 100.0 | 80 | high risk |
| **3** | 30 | 140 | 85 | 7.0 | 98.0 | 70 | high risk |
| **4** | 35 | 120 | 60 | 6.1 | 98.0 | 76 | low risk |

# Visualize the Data

In [20]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(8, 6))
sns.countplot(x="RiskLevel", data=data)
plt.title("Distribution of Risk Levels")
plt.xlabel("Risk Level")
plt.ylabel("Count")
plt.show()
```



In [21]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=data.drop(columns=["RiskLevel"]))
plt.title("Box Plot of Numeric Variables")
plt.xticks(rotation=45)
plt.show()
```




# kNN classification algorithm

## Split data into train and test sets

In [22]:
```
X = data[['Age', 'SystolicBP', 'DiastolicBP', 'BS', 'BodyTemp', 'HeartRate']]
y = data['RiskLevel']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
=42)
```

## Fitting and Evaluating the Model

With k = 4

In [40]:
```
knn = KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train, y_train)
```

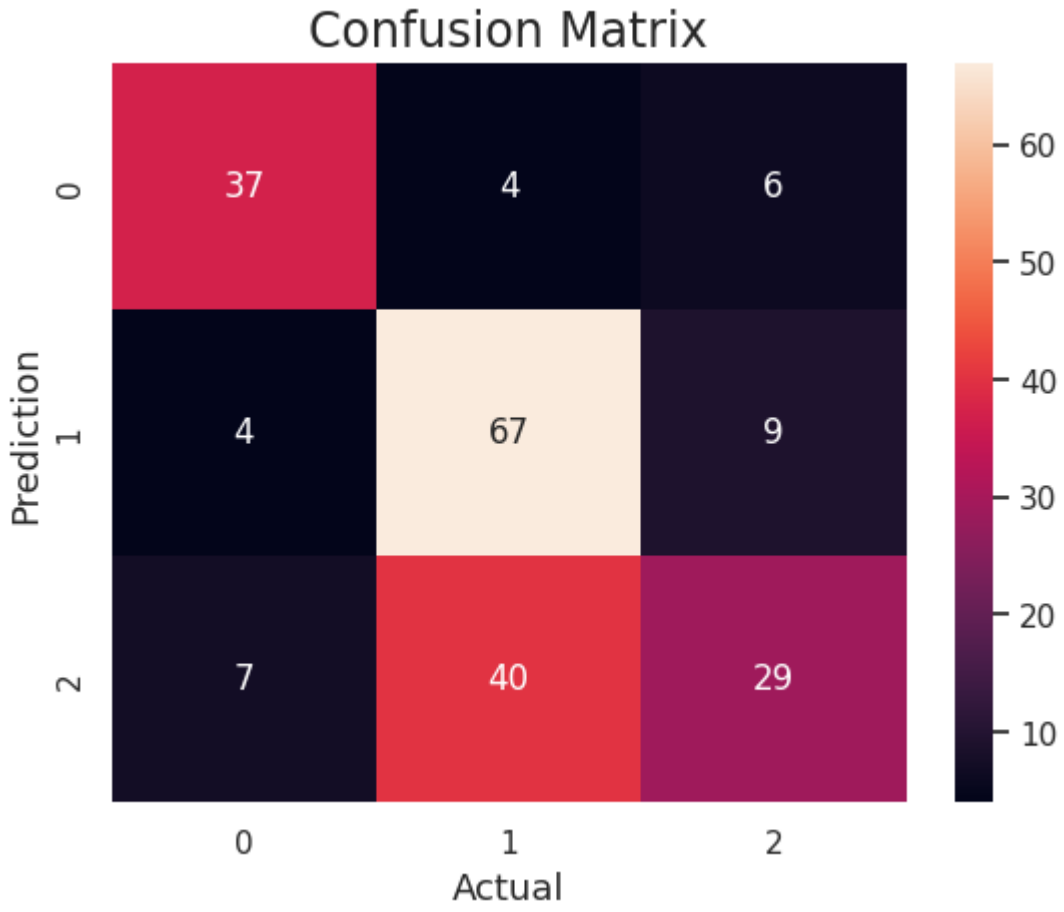
Out[40]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

## Evaluate the model

In [41]:
```
y_predict = knn.predict(X_test)
print('Recall:', recall_score(y_test, y_predict, average='weighted'))
print('F1-score: ', f1_score(y_test, y_predict, average='weighted'))
```

```
Recall: 0.6551724137931034
F1-score:  0.6377808576324364
```

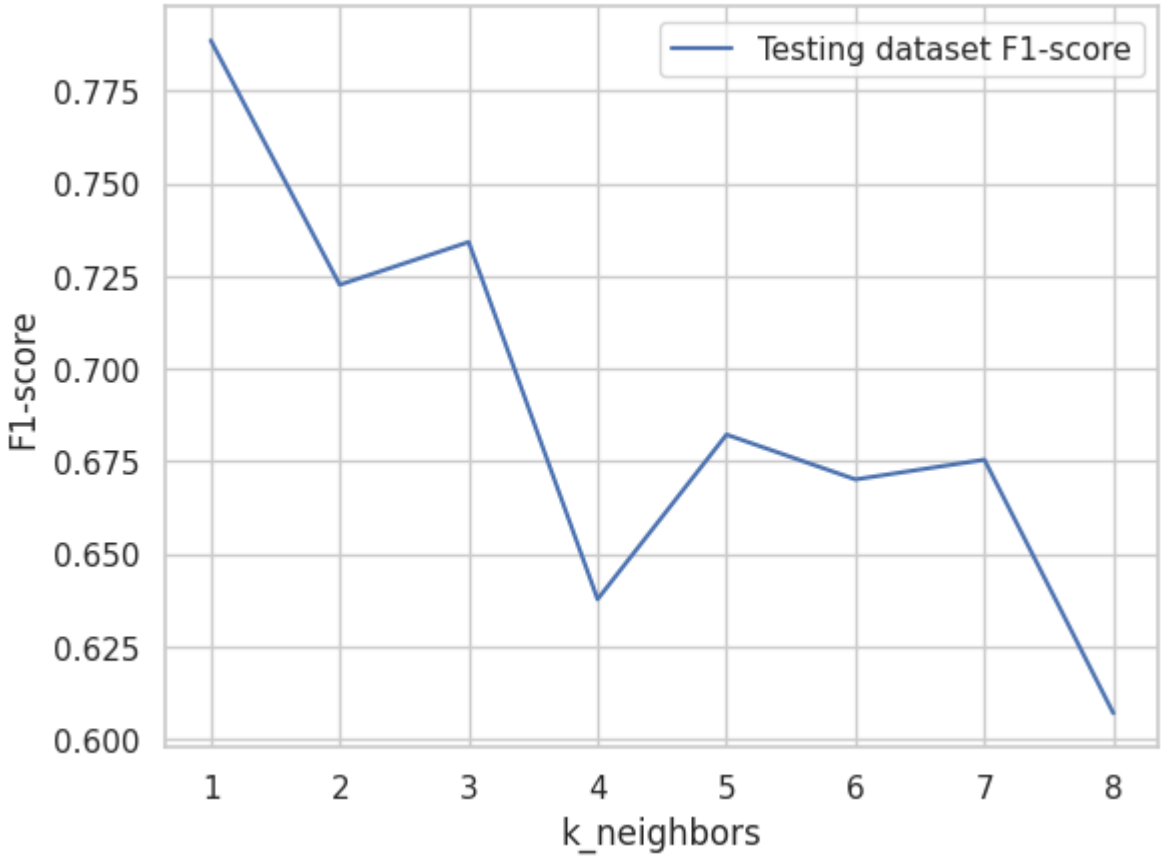
In [42]:
```
cm = confusion_matrix(y_test, y_predict)

sns.heatmap(cm,
            annot=True,
            fmt='g')
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```




## Choosing the best number of k

With k = 0 --> 9

In [43]:
```
neighbors = np.arange(1, 9)
test_f1 = np.empty(len(neighbors))

for i, k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_predict = knn.predict(X_test)

    test_f1[i] = f1_score(y_test, y_predict, average='weighted')

plt.plot(neighbors, test_f1, label = 'Testing dataset F1-score')

plt.legend()
plt.xlabel('k_neighbors')
plt.ylabel('F1-score')
plt.show()
```




Looking at the chart, "con" see that should choose k = 3 to have the highest F1-score.

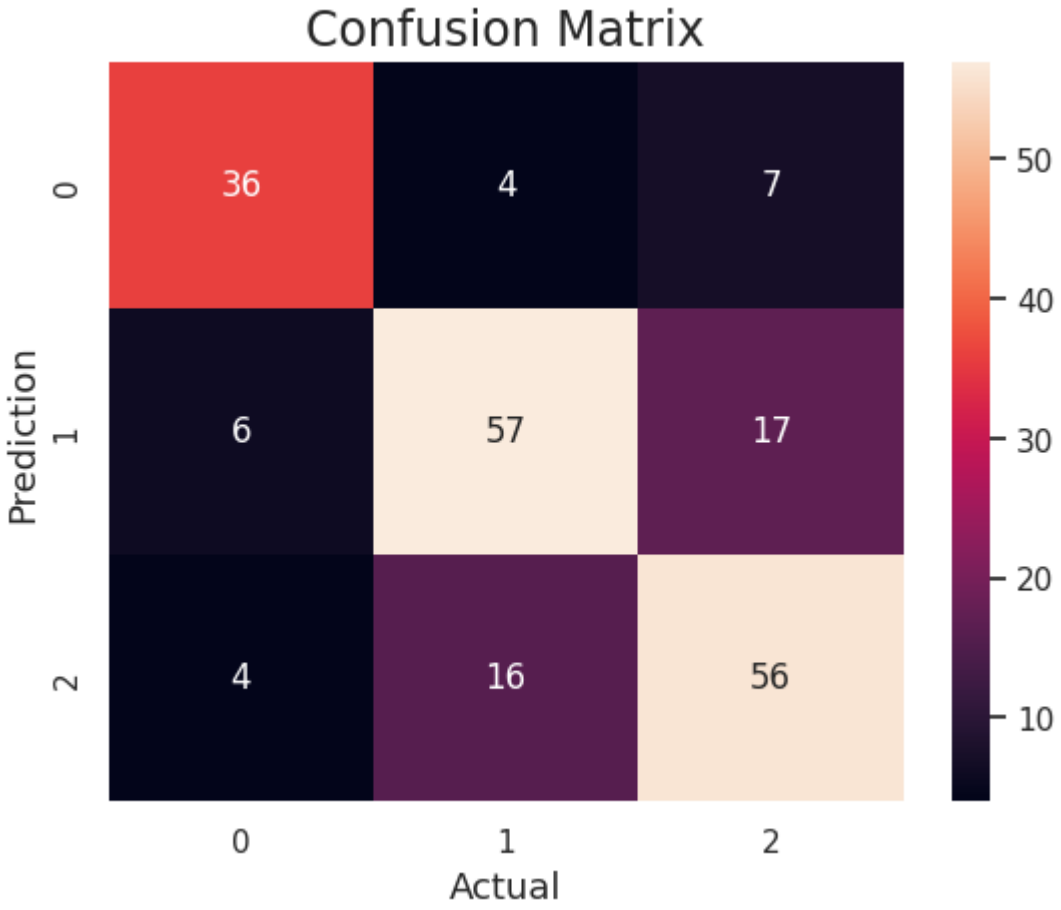
In [44]:
```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

Out[44]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [45]:
```
y_predict = knn.predict(X_test)
print('Recall:', recall_score(y_test, y_predict, average='weighted'))
print('F1-score: ', f1_score(y_test, y_predict, average='weighted'))
```

```
Recall: 0.7339901477832512
F1-score:  0.734189021245411
```

In [46]:
```
cm = confusion_matrix(y_test, y_predict)

sns.heatmap(cm,
            annot=True,
            fmt='g')
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```




# Naive Bayes classification algorithm

## Split data into train and test sets

In [47]:
```
X = data[['Age', 'SystolicBP', 'DiastolicBP', 'BS', 'BodyTemp', 'HeartRate']]
y = data['RiskLevel']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
=42)
```

## Fitting and Evaluating the Model

In [48]:
```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

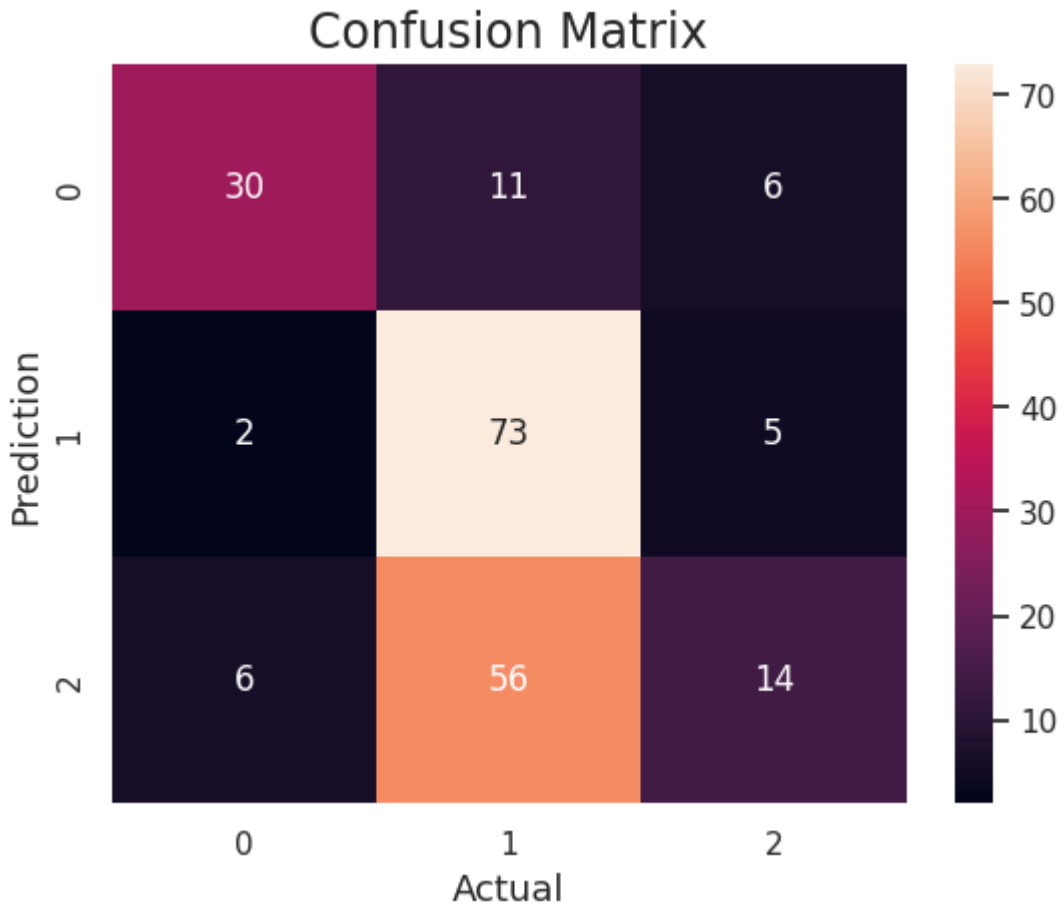
Out[48]:
```
▾ GaussianNB

GaussianNB()
```

## Evaluate the model

In [49]:
```
y_predict = gnb.predict(X_test)
print('Recall:', recall_score(y_test, y_predict, average='weighted'))
print('F1-score: ', f1_score(y_test, y_predict, average='weighted'))
```

```
Recall: 0.5763546798029556
F1-score:  0.5287521507873766
```

In [50]:
```
cm = confusion_matrix(y_test, y_predict)

sns.heatmap(cm,
            annot=True,
            fmt='g')
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```



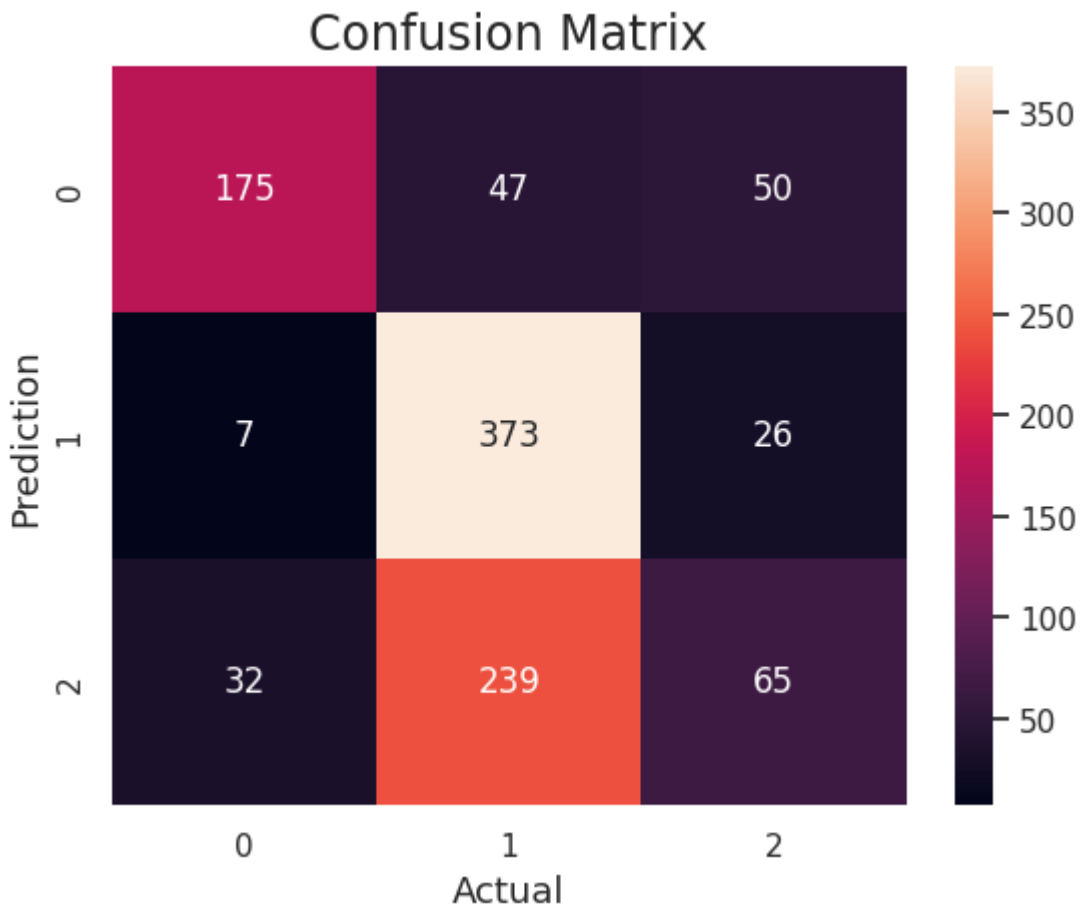
# Not included in the request: KFold implementation

In [51]:
```
X = data[['Age', 'SystolicBP', 'DiastolicBP', 'BS', 'BodyTemp', 'HeartRate']]
y = data['RiskLevel']
```

In [52]:
```
from sklearn.model_selection import KFold

num_folds = 5

kf = KFold(n_splits=num_folds, shuffle=True, random_state=42)

f1_scores = []
all_y_test = []
all_y_pred = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    gnb = GaussianNB()
    gnb.fit(X_train, y_train)
    y_pred = gnb.predict(X_test)

    f1 = f1_score(y_test, y_pred, average='weighted')
    f1_scores.append(f1)

    # Collect the y_test and y_pred for each fold
    all_y_test.extend(y_test)
    all_y_pred.extend(y_pred)

mean_f1 = np.mean(f1_scores)
print(f"Mean F1-score: {mean_f1}")

# Compute and display the confusion matrix for the entire test set
cm = confusion_matrix(all_y_test, all_y_pred)
print(f"Confusion Matrix for Test Set:\n{cm}")
```

```
Mean F1-score: 0.5644064560931644
Confusion Matrix for Test Set:
[[175  47  50]
 [  7 373  26]
 [ 32 239  65]]
```

In [53]:
```
sns.heatmap(cm,
            annot=True,
            fmt='g')
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```




# Reference


[1] Ngo Quoc Viet, Machine Learning lecture slides. (Bài giảng của thầy Việt dzễ thương)

[2] GeeksforGeeks. k-nearest neighbor algorithm in Python. Link: https://www.geeksforgeeks.org/k-nearest-neighbor-algorithm-in-python/ (https://www.geeksforgeeks.org/k-nearest-neighbor-algorithm-in-python/)

[3] GeeksforGeeks. Naive Bayes Classifiers. Link: https://www.geeksforgeeks.org/naive-bayes-classifiers/ (https://www.geeksforgeeks.org/naive-bayes-classifiers/)

[4] GeeksforGeeks. Confusion Matrix in Machine Learning. Link: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ (https://www.geeksforgeeks.org/confusion-matrix-machine-learning/)

[5] W3Schools. Machine Learning - Cross Validation. Link: https://www.w3schools.com/python/python_ml_cross_validation.asp (https://www.w3schools.com/python/python_ml_cross_validation.asp)