In [8]:
```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, Randomiz
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f
from sklearn.preprocessing import StandardScaler

data = pd.read_csv('filtered_dataset.csv')

X = data[['url_length', 'extension', 'tls', 'special_chars_count', 'tld']]
y = data['is_malicious']

X = pd.get_dummies(X, columns=['extension', 'tld'])

scaler = StandardScaler()
X[['url_length', 'special_chars_count']] = scaler.fit_transform(X[['url_leng

# Разделение на train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rar
```

In [9]:
```
# K=5
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print("Базовая модель (K=5):")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print(f"Precision: {precision_score(y_test, y_pred):.4f}")
print(f"Recall: {recall_score(y_test, y_pred):.4f}")
print(f"F1-score: {f1_score(y_test, y_pred):.4f}")
print("Confusion matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Базовая модель (K=5):
Accuracy: 0.7913
Precision: 0.8133
Recall: 0.7561
F1-score: 0.7836
Confusion matrix:
[[2409  506]
 [ 711 2204]]
```

In [10]:
```
param_grid = {
    'n_neighbors': range(1, 21),
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan', 'minkowski']
}

grid_search = GridSearchCV(
    KNeighborsClassifier(),
    param_grid,
    cv=StratifiedKFold(n_splits=5),
    scoring='f1',
    n_jobs=-1
)
```

```
grid_search.fit(X_train, y_train)

best_knn_grid = grid_search.best_estimator_
y_pred_grid = best_knn_grid.predict(X_test)

print("\nGridSearchCV results:")
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best F1-score (CV): {grid_search.best_score_:.4f}")
print(f"Test Accuracy: {accuracy_score(y_test, y_pred_grid):.4f}")
print(f"Test F1-score: {f1_score(y_test, y_pred_grid):.4f}")
```

```
GridSearchCV results:
Best parameters: {'metric': 'euclidean', 'n_neighbors': 20, 'weights': 'dist
ance'}
Best F1-score (CV): 0.8281
Test Accuracy: 0.8413
Test F1-score: 0.8341
```

In [11]:
```
from scipy.stats import randint

param_dist = {
    'n_neighbors': randint(1, 50),
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan', 'minkowski'],
    'p': [1, 2, 3]
}

random_search = RandomizedSearchCV(
    KNeighborsClassifier(),
    param_dist,
    n_iter=50,
    cv=KFold(n_splits=5, shuffle=True),
    scoring='f1',
    random_state=42,
    n_jobs=-1
)
random_search.fit(X_train, y_train)

best_knn_random = random_search.best_estimator_
y_pred_random = best_knn_random.predict(X_test)

print("\nRandomizedSearchCV results:")
print(f"Best parameters: {random_search.best_params_}")
print(f"Best F1-score (CV): {random_search.best_score_:.4f}")
print(f"Test Accuracy: {accuracy_score(y_test, y_pred_random):.4f}")
print(f"Test F1-score: {f1_score(y_test, y_pred_random):.4f}")
```

```
RandomizedSearchCV results:
Best parameters: {'metric': 'manhattan', 'n_neighbors': 31, 'p': 3, 'weight
s': 'distance'}
Best F1-score (CV): 0.8322
Test Accuracy: 0.8420
Test F1-score: 0.8345
```

In [12]:
```
results = pd.DataFrame({
    'Model': ['Baseline (K=5)', 'GridSearchCV', 'RandomizedSearchCV'],
    'Accuracy': [
```

```
        accuracy_score(y_test, y_pred),
        accuracy_score(y_test, y_pred_grid),
        accuracy_score(y_test, y_pred_random)
    ],
    'Precision': [
        precision_score(y_test, y_pred),
        precision_score(y_test, y_pred_grid),
        precision_score(y_test, y_pred_random)
    ],
    'Recall': [
        recall_score(y_test, y_pred),
        recall_score(y_test, y_pred_grid),
        recall_score(y_test, y_pred_random)
    ],
    'F1-score': [
        f1_score(y_test, y_pred),
        f1_score(y_test, y_pred_grid),
        f1_score(y_test, y_pred_random)
    ]
})

print("\nСравнение моделей:")
print(results)
```

```
Сравнение моделей:
                Model  Accuracy  Precision    Recall  F1-score
0      Baseline (K=5)  0.791252   0.813284  0.756089  0.783644
1        GridSearchCV  0.841338   0.874060  0.797599  0.834081
2  RandomizedSearchCV  0.842024   0.876226  0.796569  0.834501
```

In [ ]: