Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине

«Бинарная классификация фактографических данных»

Студент Косенков В.Д.

Группа М-ИАП-23-1

Руководитель Кургасов В. В.

доцент, канд. пед. наук

Липецк 2023 г.

Цель работы:

Получить практические навыки решения задачи бинарной классификации данных в среде Jupiter Notebook. Научиться загружать данные, обучать классификаторы и проводить классификацию. Научиться оценивать точность полученных моделей.

Задание кафедры

1. В среде Jupiter Notebook создать новый ноутбук (Notebook)
2. Импортировать необходимые для работы библиотеки и модули
3. Загрузить данные в соответствии с вариантом
4. Вывести первые 15 элементов выборки (координаты точек и метки класса)
5. Отобразить на графике сгенерированную выборку. Объекты разных классов должны иметь разные цвета.
6. Разбить данные на обучающую (train) и тестовую (test) выборки в пропорции 75% — 25% соответственно.
7. Отобразить на графике обучающую и тестовую выборки. Объекты разных классов должны иметь разные цвета.
8. Реализовать модели классификаторов, обучить их на обучающем множестве. Применить модели на тестовой выборке, вывести результаты классификации:
    - Истинные и предсказанные метки классов
    - Матрицу ошибок (confusion matrix)
    - Значения полноты, точности, f1-меры и аккуратности
    - Значение площади под кривой ошибок (AUC ROC)
    - Отобразить на графике область принятия решений по каждому классу

    В качестве методов классификации использовать:

    a) Метод к-ближайших соседей (n_neighbors = {1, 3, 5, 9})
    b) Наивный байесовский метод
    c) Случайный лес (n_estimators = {5, 10, 15, 20, 50})
9. По каждому пункту работы занести в отчет программный код и результат вывода.
10. По результатам п.8 занести в отчет таблицу с результатами классификации всеми методами и выводы о наиболее подходящем методе классификации ваших данных.
11. Изучить, как изменится качество классификации, если на тестовую часть выделить 10% выборки, 35% выборки. Для этого повторить п.п. 6 – 10.

Ход работы

На Листинге 1 представлены все необходимые и импортированные библиотеки.

```
from sklearn.datasets import make_classification
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import classification_report
```

Загрузка выборки

```
X, y = make_classification(
    n_samples=100,  # Количество образцов
    n_features=2,   # Количество признаков
    n_redundant=0,   # Количество избыточных признаков
    n_informative=1, # Количество информативных признаков
    n_clusters_per_class=1,  # Количество кластеров на класс
    class_sep=0.45,  # Разделение классов
    random_state=78  # Random State для воспроизводимости
)

Вывод первых 15 элементов выборки
data = pd.DataFrame({'Feature 1': X[:, 0], 'Feature 2': X[:, 1], 'Label':
y})
print(data.head(15))
```

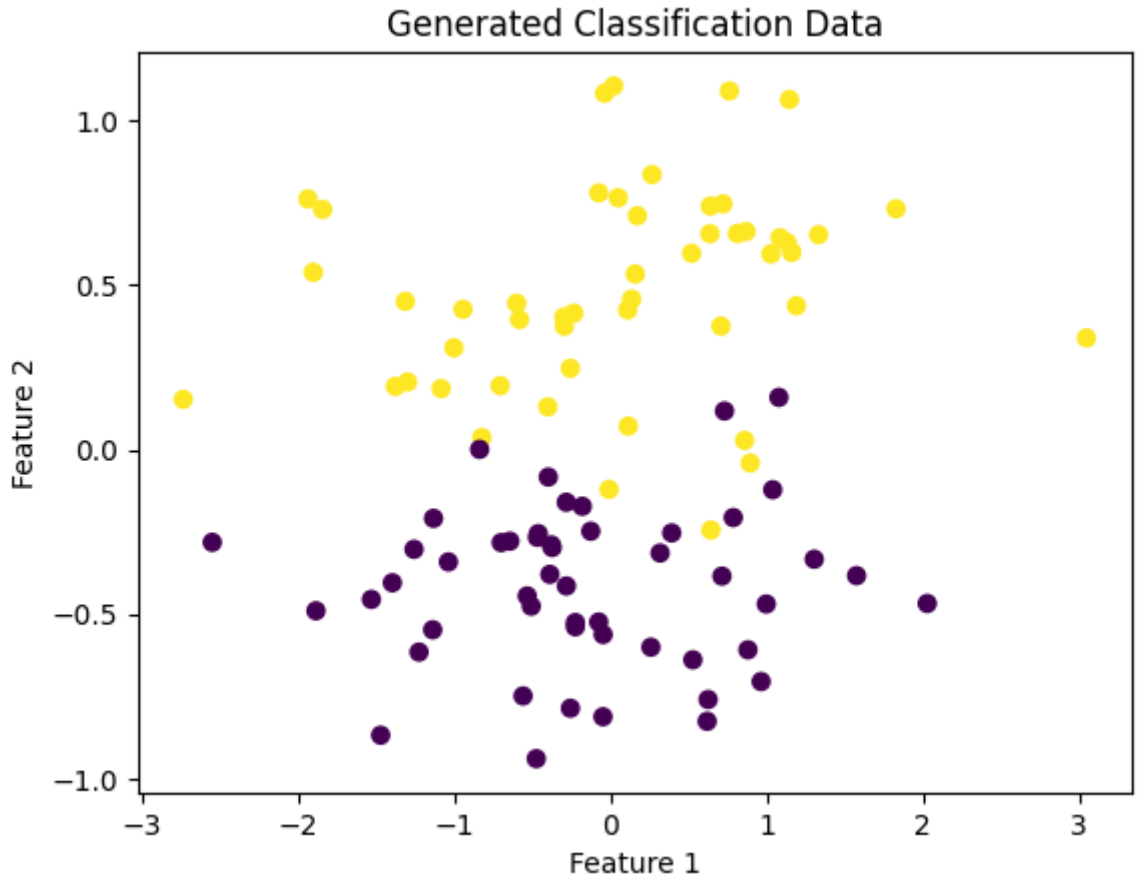
Вывод первых 15 элементов выборки

	Feature 1	Feature 2	Label
0	0.618400	-0.825987	0
1	-1.002505	0.309745	1
2	1.077837	0.159075	0
3	0.158532	0.534370	1
4	0.265539	0.836729	1
5	0.518891	0.597110	1
6	0.636534	0.656617	1
7	1.145333	1.064834	1
8	-1.037819	-0.341347	0
9	-0.462668	-0.255397	0
10	1.303928	-0.333062	0
11	-1.133073	-0.208877	0
12	-2.735665	0.152848	1
13	-0.047600	-0.562068	0
14	-0.048519	-0.812140	0

Рисунок 1 – Результат

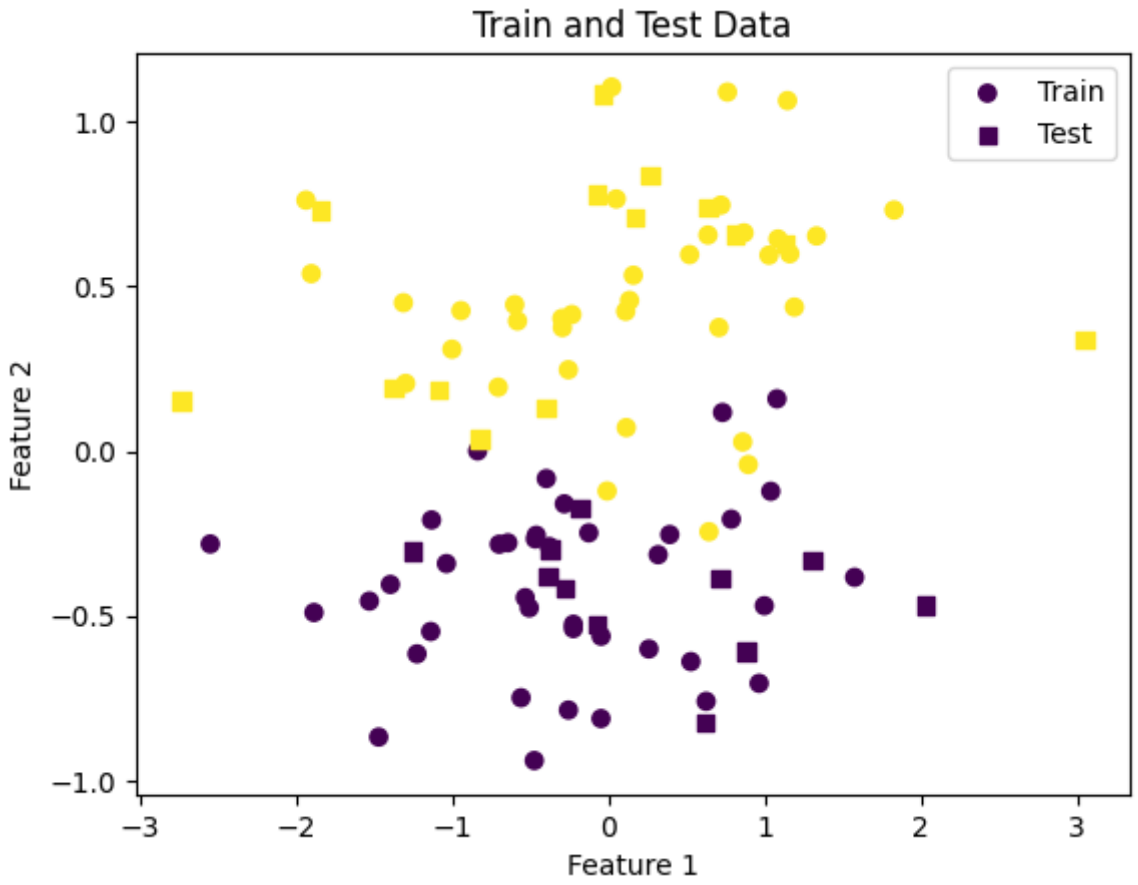
Отобразить на графике сгенерированную выборку

```
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis', marker='o')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Generated Classification Data')
plt.show()
```



Разделение выборки

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)

# Отобразить обучающую и тестовую выборки
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap='viridis',
marker='o', label='Train')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap='viridis',
marker='s', label='Test')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Train and Test Data')
plt.legend()
plt.show()
```

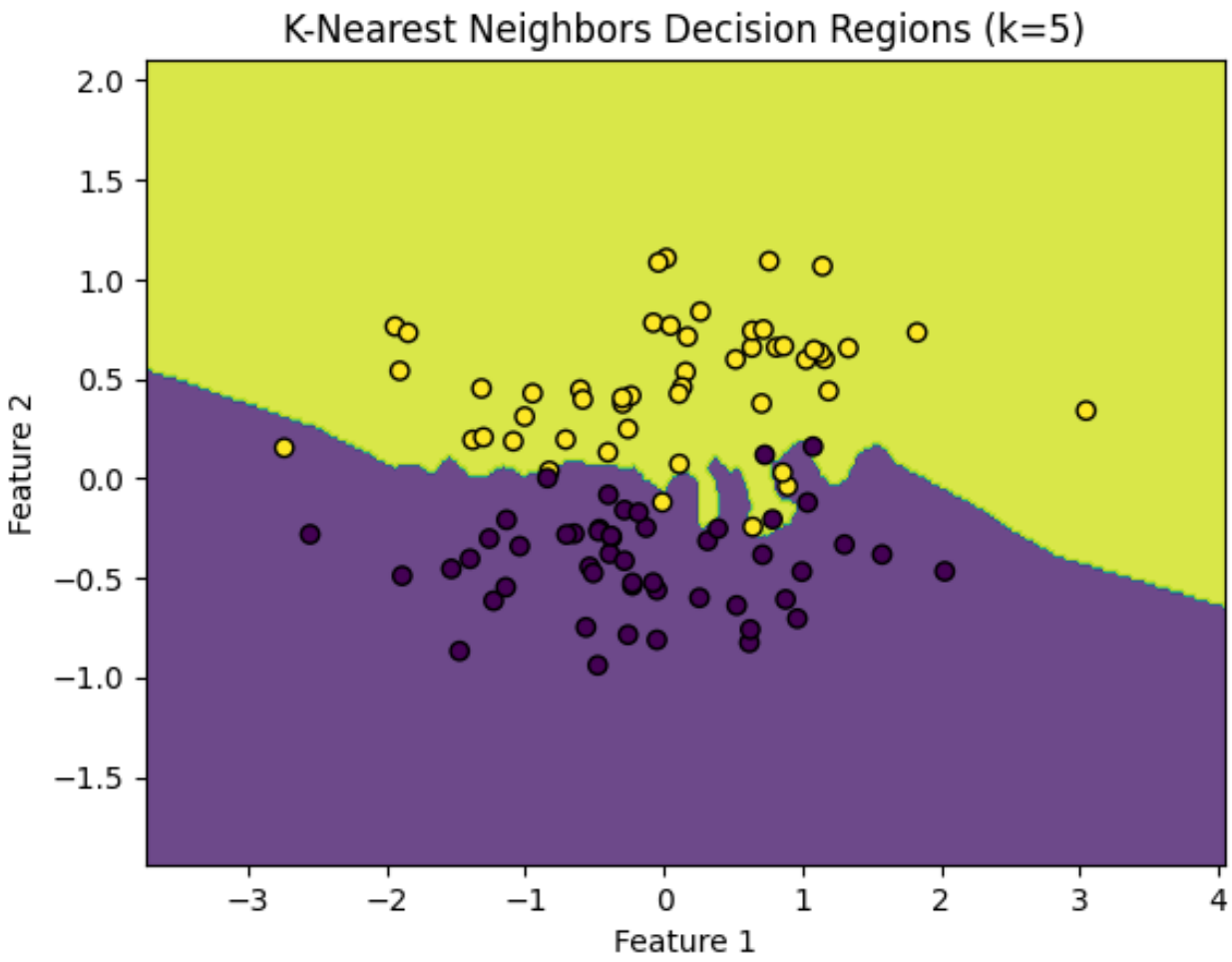


Реализовать и обучить модели классификаторов

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score, confusion_matrix
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_predict
import numpy as np

# Функция для вывода результатов классификации
def print_classification_results(y_true, y_pred, model_name):
    print(f"Results for {model_name}:")
    print("Confusion Matrix:")
    print(confusion_matrix(y_true, y_pred))
    print("\nClassification Report:")
    print(f"Accuracy: {accuracy_score(y_true, y_pred)}")
    print(f"Precision: {precision_score(y_true, y_pred)}")
    print(f"Recall: {recall_score(y_true, y_pred)}")
    print(f"F1 Score: {f1_score(y_true, y_pred)}")
    print(f"AUC-ROC Score: {roc_auc_score(y_true, y_pred)}\n")

# a) Метод к-ближайших соседей
k_values = [1, 3, 5, 9]

for k in k_values:
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(X_train, y_train)
    knn_pred = knn_model.predict(X_test)
    print_classification_results(y_test, knn_pred, f"K-Nearest Neighbors
(k={k})")

# b) Наивный байесовский метод
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_pred = nb_model.predict(X_test)
print_classification_results(y_test, nb_pred, "Naive Bayes")

# c) Случайный лес
n_estimators_values = [5, 10, 15, 20, 50]

for n_estimators in n_estimators_values:
    rf_model = RandomForestClassifier(n_estimators=n_estimators,
random_state=42)
    rf_model.fit(X_train, y_train)
    rf_pred = rf_model.predict(X_test)
    print_classification_results(y_test, rf_pred, f"Random Forest
(n_estimators={n_estimators})")

# Отобразить область принятия решений для k-ближайших соседей с k=5
knn_model_5 = KNeighborsClassifier(n_neighbors=5)
knn_model_5.fit(X_train, y_train)
plot_decision_regions(X, y, knn_model_5, "K-Nearest Neighbors Decision
Regions (k=5)")
```

```
Results for K-Nearest Neighbors (k=1):
Confusion Matrix:
[[10  1]
 [ 2 12]]

Classification Report:
Accuracy: 0.88
Precision: 0.9230769230769231
Recall: 0.8571428571428571
F1 Score: 0.888888888888889
AUC-ROC Score: 0.8831168831168831

Results for K-Nearest Neighbors (k=3):
Confusion Matrix:
[[11  0]
 [ 0 14]]

Classification Report:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
AUC-ROC Score: 1.0

Results for K-Nearest Neighbors (k=5):
Confusion Matrix:
[[11  0]
 [ 2 12]]

Classification Report:
Accuracy: 0.92
Precision: 1.0
Recall: 0.8571428571428571
F1 Score: 0.923076923076923
AUC-ROC Score: 0.9285714285714286

Results for K-Nearest Neighbors (k=9):
Confusion Matrix:
[[11  0]
 [ 2 12]]

Classification Report:
Accuracy: 0.92
Precision: 1.0
Recall: 0.8571428571428571
F1 Score: 0.923076923076923
AUC-ROC Score: 0.9285714285714286

Results for Naive Bayes:
Confusion Matrix:
[[11  0]
 [ 1 13]]

Classification Report:
Accuracy: 0.96
Precision: 1.0
Recall: 0.9285714285714286
F1 Score: 0.962962962962963
AUC-ROC Score: 0.9642857142857143

Results for Random Forest (n_estimators=5):
Confusion Matrix:
[[11  0]
 [ 0 14]]
```

```
Classification Report:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
AUC-ROC Score: 1.0

Results for Random Forest (n_estimators=10):
Confusion Matrix:
[[11  0]
 [ 1 13]]

Classification Report:
Accuracy: 0.96
Precision: 1.0
Recall: 0.9285714285714286
F1 Score: 0.962962962962963
AUC-ROC Score: 0.9642857142857143

Results for Random Forest (n_estimators=15):
Confusion Matrix:
[[11  0]
 [ 0 14]]

Classification Report:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
AUC-ROC Score: 1.0

Results for Random Forest (n_estimators=20):
Confusion Matrix:
[[11  0]
 [ 0 14]]

Classification Report:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
AUC-ROC Score: 1.0

Results for Random Forest (n_estimators=50):
Confusion Matrix:
[[11  0]
 [ 1 13]]

Classification Report:
Accuracy: 0.96
Precision: 1.0
Recall: 0.9285714285714286
F1 Score: 0.962962962962963
AUC-ROC Score: 0.9642857142857143
```



Составим сводную таблицу для отображения результатов векторизации.

	Model	Test Size	Accuracy	Precision	Recall	F1 Score
0	KNN (k=1)	0.10	0.940000	0.892857	1.000000	0.943396
1	KNN (k=3)	0.10	0.920000	0.862069	1.000000	0.925926
2	KNN (k=5)	0.10	0.920000	0.862069	1.000000	0.925926
3	KNN (k=9)	0.10	0.920000	0.862069	1.000000	0.925926
4	Naive Bayes	0.10	0.960000	0.925926	1.000000	0.961538
5	Random Forest (n=5)	0.10	0.960000	0.925926	1.000000	0.961538
6	Random Forest (n=10)	0.10	0.960000	0.925926	1.000000	0.961538
7	Random Forest (n=15)	0.10	0.960000	0.925926	1.000000	0.961538
8	Random Forest (n=20)	0.10	0.960000	0.925926	1.000000	0.961538
9	Random Forest (n=50)	0.10	0.960000	0.925926	1.000000	0.961538
10	KNN (k=1)	0.35	0.942857	0.913043	0.976744	0.943820
11	KNN (k=3)	0.35	0.948571	0.905263	1.000000	0.950276
12	KNN (k=5)	0.35	0.942857	0.895833	1.000000	0.945055
13	KNN (k=9)	0.35	0.942857	0.895833	1.000000	0.945055
14	Naive Bayes	0.35	0.977143	0.965909	0.988372	0.977011
15	Random Forest (n=5)	0.35	0.965714	0.934783	1.000000	0.966292
16	Random Forest (n=10)	0.35	0.977143	0.955556	1.000000	0.977273
17	Random Forest (n=15)	0.35	0.965714	0.934783	1.000000	0.966292
18	Random Forest (n=20)	0.35	0.977143	0.955556	1.000000	0.977273
19	Random Forest (n=50)	0.35	0.971429	0.945055	1.000000	0.971751

	AUC ROC
0	0.940000
1	0.940000
2	0.936800
3	0.930400
4	0.977600
5	0.977600
6	0.976000
7	0.974400
8	0.971200
9	0.969600
10	0.943428
11	0.952704
12	0.954207
13	0.951071
14	0.989940
15	0.979227
16	0.980206
17	0.977724
18	0.977463
19	0.978769

Проведем классификацию с10% выборки и 35% выборки

```
# Разбиение данных на обучающую и тестовую выборки (10% тестовая)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
random_state=42)

# Реализация, обучение и тестирование моделей
def classify_and_evaluate(classifier, model_name, X_train, X_test, y_train,
y_test):
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)

    # Вывод результатов
    print(f"\nResults for {model_name} with 10% test set:")
    print("Confusion Matrix:")
    print(metrics.confusion_matrix(y_test, y_pred))
    print("\nClassification Report:")
    print(metrics.classification_report(y_test, y_pred))
    print(f"AUC ROC: {metrics.roc_auc_score(y_test,
classifier.predict_proba(X_test)[:, 1])}")

# a) k-ближайших соседей (n_neighbors = {1, 3, 5, 9})
for n_neighbors in [1, 3, 5, 9]:
    knn_classifier = KNeighborsClassifier(n_neighbors=n_neighbors)
    classify_and_evaluate(knn_classifier, f"KNN (k={n_neighbors})", X_train,
X_test, y_train, y_test)

# b) Наивный байесовский метод
nb_classifier = GaussianNB()
classify_and_evaluate(nb_classifier, "Naive Bayes", X_train, X_test, y_train,
y_test)

# c) Случайный лес (n_estimators = {5, 10, 15, 20, 50})
for n_estimators in [5, 10, 15, 20, 50]:
    rf_classifier = RandomForestClassifier(n_estimators=n_estimators,
random_state=42)
    classify_and_evaluate(rf_classifier, f"Random Forest (n={n_estimators})",
X_train, X_test, y_train, y_test)
```

```
Results for KNN (k=1) with 10% test set:
Confusion Matrix:
[[81  8]
 [ 2 84]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.91      0.94        89
           1       0.91      0.98      0.94        86

    accuracy                           0.94       175
   macro avg       0.94      0.94      0.94       175
weighted avg       0.95      0.94      0.94       175

AUC ROC: 0.9434282727985367
```

```
Results for KNN (k=3) with 10% test set:
Confusion Matrix:
[[80  9]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.90      0.95        89
           1       0.91      1.00      0.95        86

    accuracy                           0.95       175
   macro avg       0.95      0.95      0.95       175
weighted avg       0.95      0.95      0.95       175

AUC ROC: 0.9527044682518944

Results for KNN (k=5) with 10% test set:
Confusion Matrix:
[[79 10]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        89
           1       0.90      1.00      0.95        86

    accuracy                           0.94       175
   macro avg       0.95      0.94      0.94       175
weighted avg       0.95      0.94      0.94       175

AUC ROC: 0.9542069506140579

Results for KNN (k=9) with 10% test set:
Confusion Matrix:
[[79 10]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        89
           1       0.90      1.00      0.95        86

    accuracy                           0.94       175
   macro avg       0.95      0.94      0.94       175
weighted avg       0.95      0.94      0.94       175

AUC ROC: 0.9510713352495427

Results for Naive Bayes with 10% test set:
Confusion Matrix:
[[86  3]
 [ 1 85]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98        89
           1       0.97      0.99      0.98        86

    accuracy                           0.98       175
   macro avg       0.98      0.98      0.98       175
weighted avg       0.98      0.98      0.98       175
```

```
AUC ROC: 0.9899399007055134

Results for Random Forest (n=5) with 10% test set:
Confusion Matrix:
[[83  6]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.93      0.97        89
           1       0.93      1.00      0.97        86

    accuracy                           0.97       175
   macro avg       0.97      0.97      0.97       175
weighted avg       0.97      0.97      0.97       175

AUC ROC: 0.9792265482100861

Results for Random Forest (n=10) with 10% test set:
Confusion Matrix:
[[85  4]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        89
           1       0.96      1.00      0.98        86

    accuracy                           0.98       175
   macro avg       0.98      0.98      0.98       175
weighted avg       0.98      0.98      0.98       175

AUC ROC: 0.9802064280114973

Results for Random Forest (n=15) with 10% test set:
Confusion Matrix:
[[83  6]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.93      0.97        89
           1       0.93      1.00      0.97        86

    accuracy                           0.97       175
   macro avg       0.97      0.97      0.97       175
weighted avg       0.97      0.97      0.97       175

AUC ROC: 0.9777240658479226

Results for Random Forest (n=20) with 10% test set:
Confusion Matrix:
[[85  4]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        89
           1       0.96      1.00      0.98        86
```

```
    accuracy                           0.98       175
   macro avg       0.98      0.98      0.98       175
weighted avg       0.98      0.98      0.98       175

AUC ROC: 0.9774627645675464

Results for Random Forest (n=50) with 10% test set:
Confusion Matrix:
[[84  5]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.94      0.97        89
           1       0.95      1.00      0.97        86

    accuracy                           0.97       175
   macro avg       0.97      0.97      0.97       175
weighted avg       0.97      0.97      0.97       175

AUC ROC: 0.9787692709694278
```

```python
# Разбиение данных на обучающую и тестовую выборки (35% тестовая)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35,
random_state=42)

# Повторение процесса с новыми тестовыми данными
# (можете скопировать и вставить код выше, изменив размер тестовой выборки)
# a) k-ближайших соседей (n_neighbors = {1, 3, 5, 9})
for n_neighbors in [1, 3, 5, 9]:
    knn_classifier = KNeighborsClassifier(n_neighbors=n_neighbors)
    classify_and_evaluate(knn_classifier, f"KNN (k={n_neighbors})", X_train,
X_test, y_train, y_test)

# b) Наивный байесовский метод
nb_classifier = GaussianNB()
classify_and_evaluate(nb_classifier, "Naive Bayes", X_train, X_test, y_train,
y_test)

# c) Случайный лес (n_estimators = {5, 10, 15, 20, 50})
for n_estimators in [5, 10, 15, 20, 50]:
    rf_classifier = RandomForestClassifier(n_estimators=n_estimators,
random_state=42)
    classify_and_evaluate(rf_classifier, f"Random Forest (n={n_estimators})",
X_train, X_test, y_train, y_test)
```

```
Results for KNN (k=1) with 10% test set:
Confusion Matrix:
[[81  8]
 [ 2 84]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.91      0.94        89
           1       0.91      0.98      0.94        86

    accuracy                           0.94       175
```

```
   macro avg       0.94      0.94      0.94       175
weighted avg       0.95      0.94      0.94       175

AUC ROC: 0.9434282727985367

Results for KNN (k=3) with 10% test set:
Confusion Matrix:
[[80  9]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.90      0.95        89
           1       0.91      1.00      0.95        86

    accuracy                           0.95       175
   macro avg       0.95      0.95      0.95       175
weighted avg       0.95      0.95      0.95       175

AUC ROC: 0.9527044682518944

Results for KNN (k=5) with 10% test set:
Confusion Matrix:
[[79 10]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        89
           1       0.90      1.00      0.95        86

    accuracy                           0.94       175
   macro avg       0.95      0.94      0.94       175
weighted avg       0.95      0.94      0.94       175

AUC ROC: 0.9542069506140579

Results for KNN (k=9) with 10% test set:
Confusion Matrix:
[[79 10]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        89
           1       0.90      1.00      0.95        86

    accuracy                           0.94       175
   macro avg       0.95      0.94      0.94       175
weighted avg       0.95      0.94      0.94       175

AUC ROC: 0.9510713352495427

Results for Naive Bayes with 10% test set:
Confusion Matrix:
[[86  3]
 [ 1 85]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98        89
```

```
           1       0.97      0.99      0.98        86

    accuracy                           0.98       175
   macro avg       0.98      0.98      0.98       175
weighted avg       0.98      0.98      0.98       175

AUC ROC: 0.9899399007055134

Results for Random Forest (n=5) with 10% test set:
Confusion Matrix:
[[83  6]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.93      0.97        89
           1       0.93      1.00      0.97        86

    accuracy                           0.97       175
   macro avg       0.97      0.97      0.97       175
weighted avg       0.97      0.97      0.97       175

AUC ROC: 0.9792265482100861

Results for Random Forest (n=10) with 10% test set:
Confusion Matrix:
[[85  4]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        89
           1       0.96      1.00      0.98        86

    accuracy                           0.98       175
   macro avg       0.98      0.98      0.98       175
weighted avg       0.98      0.98      0.98       175

AUC ROC: 0.9802064280114973

Results for Random Forest (n=15) with 10% test set:
Confusion Matrix:
[[83  6]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.93      0.97        89
           1       0.93      1.00      0.97        86

    accuracy                           0.97       175
   macro avg       0.97      0.97      0.97       175
weighted avg       0.97      0.97      0.97       175

AUC ROC: 0.9777240658479226

Results for Random Forest (n=20) with 10% test set:
Confusion Matrix:
[[85  4]
 [ 0 86]]

Classification Report:
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        89
           1       0.96      1.00      0.98        86

    accuracy                           0.98       175
   macro avg       0.98      0.98      0.98       175
weighted avg       0.98      0.98      0.98       175

AUC ROC: 0.9774627645675464

Results for Random Forest (n=50) with 10% test set:
Confusion Matrix:
[[84  5]
 [ 0 86]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.94      0.97        89
           1       0.95      1.00      0.97        86

    accuracy                           0.97       175
   macro avg       0.97      0.97      0.97       175
weighted avg       0.97      0.97      0.97       175

AUC ROC: 0.9787692709694278
```

Вывод

В ходе выполнения данной лабораторной работы я получил базовые навыки работы с языком python и набором функций для анализа данных.