

Министерство образования Республики Беларусь
Учреждение образования
«Брестский Государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3
По дисциплине «ОМО»

Выполнил:
Студент 3-го курса
Группы АС-65
Егоренков Н. Д.
Проверил:
Крощенко А.А.

Брест 2025

Цель: На практике сравнить работу нескольких алгоритмов классификации, таких как метод k-ближайших соседей (k-NN), деревья решений и метод опорных векторов (SVM). Научиться подбирать гиперпараметры моделей и оценивать их влияние на результат.

Задачи:

1. Загрузить датасет по варианту;
2. Разделить данные на обучающую и тестовую выборки;
3. Обучить на обучающей выборке три модели: k-NN, Decision Tree и SVM;
4. Для модели k-NN исследовать, как меняется качество при разном количестве соседей (k);
5. Оценить точность каждой модели на тестовой выборке;
6. Сравнить результаты, сделать выводы о применимости каждого метода для данного набора данных.

Вариант 5

- Mushroom Classification
- Определить, является ли гриб ядовитым или съедобным
- **Задания:**
 1. Загрузите данные и преобразуйте все категориальные признаки в числовые (например, с помощью One-Hot Encoding);
 2. Разделите данные на обучающую и тестовую части;
 3. Обучите классификаторы k-NN, Decision Tree и SVM;
 4. Рассчитайте точность и полноту (precision и recall) для класса "ядовитый";
 5. Сделайте вывод о том, какой классификатор лучше всего справляется с этой задачей, где цена ошибки очень высока.

Ход работы

Код программы данной лабораторной работы:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score,
classification_report, confusion_matrix, \
    ConfusionMatrixDisplay
import matplotlib

matplotlib.use('Agg')
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('mushrooms.csv')

X_raw = df.drop('class', axis=1)
```

```

y = df['class'].apply(lambda x: 1 if x == 'p' else 0)

X_train_raw, X_test_raw, y_train, y_test = train_test_split(
    X_raw, y, test_size=0.5, random_state=42, stratify=y
)

print(f"\nРазмеры выборок до кодирования:")
print(f"Обучающая: {X_train_raw.shape}")
print(f"Тестовая: {X_test_raw.shape}")


def encode_categorical(df_train, df_test):
    df_train_encoded = df_train.copy()
    df_test_encoded = df_test.copy()
    encoders = {}

    for column in df_train.columns:
        le = LabelEncoder()
        # Обучаем кодировщик ТОЛЬКО на обучающих данных
        df_train_encoded[column] =
        le.fit_transform(df_train[column].astype(str))
        # Применяем к тестовым данным БЕЗ переобучения
        # Используем try-except для обработки новых категорий в тестовой
        выборке
        try:
            df_test_encoded[column] =
            le.transform(df_test[column].astype(str))
        except ValueError:
            # Если в тестовой выборке есть новые категории, присваиваем -1
            df_test_encoded[column] = -1
        encoders[column] = le

    return df_train_encoded, df_test_encoded, encoders


X_train_encoded, X_test_encoded, feature_encoders =
encode_categorical(X_train_raw, X_test_raw)

print(f"\nРазмеры после кодирования:")
print(f"Обучающая: {X_train_encoded.shape}")
print(f"Тестовая: {X_test_encoded.shape}")

train_counts = np.unique(y_train, return_counts=True)
test_counts = np.unique(y_test, return_counts=True)

knn = KNeighborsClassifier(n_neighbors=5)
dt = DecisionTreeClassifier(random_state=42, max_depth=5)
svm = SVC(random_state=42, probability=True)

knn.fit(X_train_encoded, y_train)
dt.fit(X_train_encoded, y_train)
svm.fit(X_train_encoded, y_train)

knn_pred = knn.predict(X_test_encoded)

```

```

dt_pred = dt.predict(X_test_encoded)
svm_pred = svm.predict(X_test_encoded)

knn_train_pred = knn.predict(X_train_encoded)
dt_train_pred = dt.predict(X_train_encoded)
svm_train_pred = svm.predict(X_train_encoded)

print("\nТОЧНОСТЬ НА ОБУЧАЮЩЕЙ ВЫБОРКЕ:")
print(f"k-NN: {accuracy_score(y_train, knn_train_pred):.4f}")
print(f"Decision Tree: {accuracy_score(y_train, dt_train_pred):.4f}")
print(f"SVM: {accuracy_score(y_train, svm_train_pred):.4f}")

print("\nТОЧНОСТЬ НА ТЕСТОВОЙ ВЫБОРКЕ:")
print(f"k-NN: {accuracy_score(y_test, knn_pred):.4f}")
print(f"Decision Tree: {accuracy_score(y_test, dt_pred):.4f}")
print(f"SVM: {accuracy_score(y_test, svm_pred):.4f}")

k_values = range(1, 21)
knn_accuracies = []
knn_precisions = []
knn_recalls = []

print("\n" + "=" * 60)
print("ИССЛЕДОВАНИЕ k-NN ПРИ РАЗНОМ КОЛИЧЕСТВЕ СОСЕДЕЙ")
print("=" * 60)

for k in k_values:
    knn_temp = KNeighborsClassifier(n_neighbors=k)
    knn_temp.fit(X_train_encoded, y_train)
    pred_temp = knn_temp.predict(X_test_encoded)

    knn_accuracies.append(accuracy_score(y_test, pred_temp))
    knn_precisions.append(precision_score(y_test, pred_temp, pos_label=1))
    knn_recalls.append(recall_score(y_test, pred_temp, pos_label=1))

knn_results_df = pd.DataFrame({
    'k': k_values,
    'Accuracy': knn_accuracies,
    'Precision': knn_precisions,
    'Recall': knn_recalls
})
print(knn_results_df.round(4))

best_k_idx = np.argmax(knn_accuracies)
best_k = k_values[best_k_idx]
print(f"\nОптимальное количество соседей: {best_k}")
print(f"Лучшая точность: {knn_accuracies[best_k_idx]:.4f}")

def plot_confusion_matrix(model_name, y_true, y_pred, filename=None):
    cm = confusion_matrix(y_true, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
        display_labels=['Съедобные', 'Ядовитые'])

```

```

fig, ax = plt.subplots(figsize=(8, 6))
disp.plot(cmap='Blues', ax=ax, values_format='d')
plt.title(f'Матрица ошибок: {model_name}\n', fontsize=14,
fontweight='bold')

plt.text(0.5, -0.2,
         f"TN (True Negative): {cm[0, 0]} - правильно определены
съедобные\n"
         f"FP (False Positive): {cm[0, 1]} - съедобные ошибочно помечены
как ядовитые\n"
         f"FN (False Negative): {cm[1, 0]} - ядовитые ошибочно помечены
как съедобные\n"
         f"TP (True Positive): {cm[1, 1]} - правильно определены
ядовитые",
         transform=ax.transAxes, fontsize=10,
         bbox=dict(boxstyle="round,pad=0.3", facecolor="lightgray"),
         ha='center', va='center')

plt.tight_layout()

if filename:
    plt.savefig(filename, dpi=300, bbox_inches='tight')
    print(f"Матрица ошибок сохранена как: {filename}")
return cm


def evaluate_model(name, y_true, y_pred):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, pos_label=1)
    recall = recall_score(y_true, y_pred, pos_label=1)
    f1 = 2 * (precision * recall) / (precision + recall) if (precision +
recall) > 0 else 0

    print(f"\n{name}:")

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision (для ядовитых): {precision:.4f}")
    print(f"Recall (для ядовитых): {recall:.4f}")
    print(f"F1-Score: {f1:.4f}")

    return accuracy, precision, recall, f1


results = {}

print("\n" + "=" * 50)
print("РЕЗУЛЬТАТЫ КЛАССИФИКАЦИИ НА ТЕСТОВОЙ ВЫБОРКЕ")
print("=" * 50)

results['k-NN'] = evaluate_model("k-Nearest Neighbors", y_test, knn_pred)
results['Decision Tree'] = evaluate_model("Decision Tree", y_test, dt_pred)
results['SVM'] = evaluate_model("Support Vector Machine", y_test, svm_pred)

print("\n" + "=" * 50)
print("МАТРИЦЫ ОШИБОК")

```

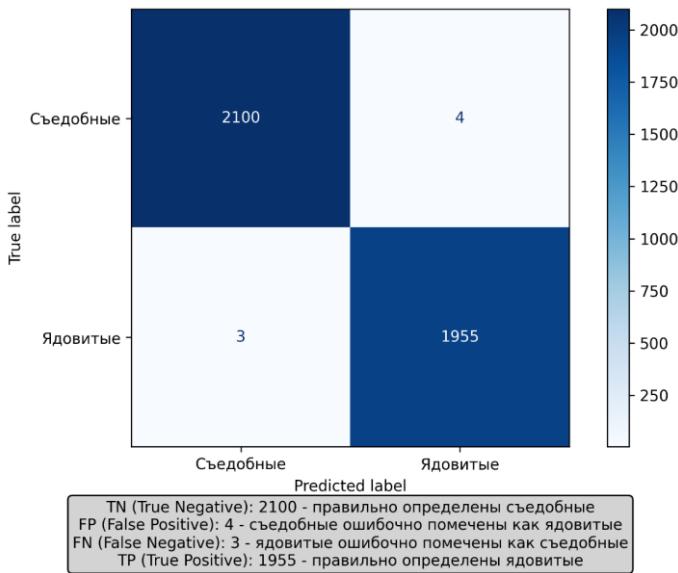
```

print("=" * 50)

cm_knn = plot_confusion_matrix("k-Nearest Neighbors", y_test, knn_pred,
"confusion_matrix_knn.png")
print(f"\nk-NN Матрица ошибок:")
print(f"TN: {cm_knn[0, 0]}, FP: {cm_knn[0, 1]}, FN: {cm_knn[1, 0]}, TP:
{cm_knn[1, 1]}")

```

Матрица ошибок: k-Nearest Neighbors

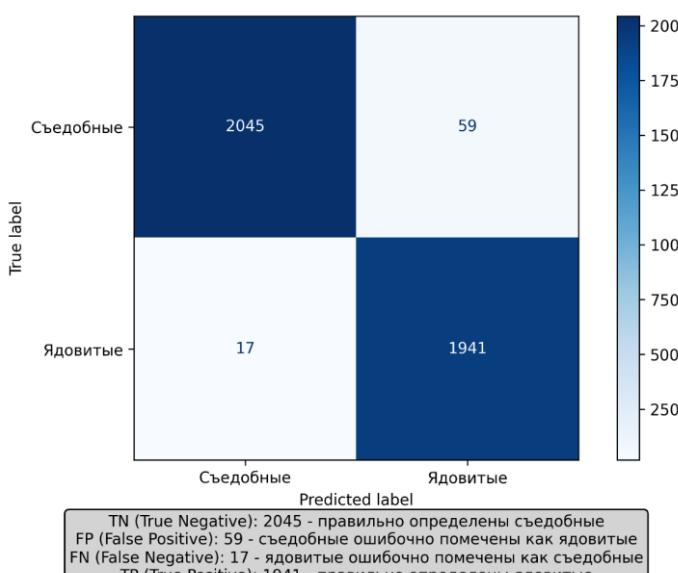


```

cm_dt = plot_confusion_matrix("Decision Tree", y_test, dt_pred,
"confusion_matrix_decision_tree.png")
print(f"\nDecision Tree Матрица ошибок:")
print(f"TN: {cm_dt[0, 0]}, FP: {cm_dt[0, 1]}, FN: {cm_dt[1, 0]}, TP:
{cm_dt[1, 1]}")

```

Матрица ошибок: Decision Tree

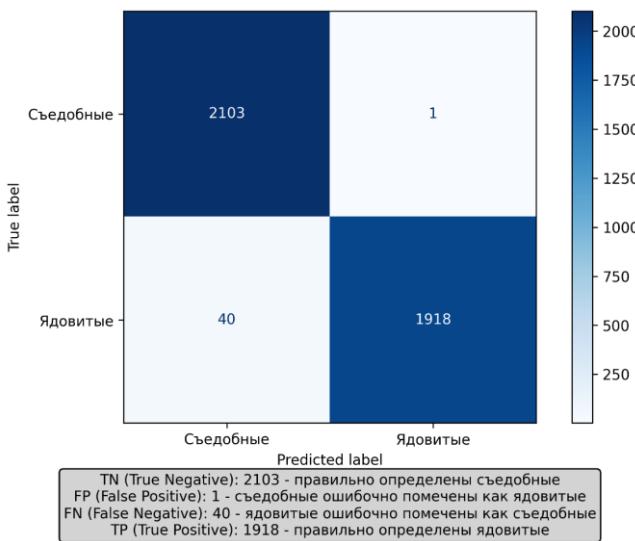


```

cm_svm = plot_confusion_matrix("Support Vector Machine", y_test, svm_pred,
"confusion_matrix_svm.png")
print(f"\nSVM Матрица ошибок:")
print(f"TN: {cm_svm[0, 0]}, FP: {cm_svm[0, 1]}, FN: {cm_svm[1, 0]}, TP:
{cm_svm[1, 1]}")

```

Матрица ошибок: Support Vector Machine



```
print("\n" + "=" * 50)
print("СРАВНИТЕЛЬНАЯ МАТРИЦА ОШИБОК")
print("=" * 50)

fig, axes = plt.subplots(1, 3, figsize=(18, 5))
models = [("k-NN", knn_pred, cm_knn),
          ("Decision Tree", dt_pred, cm_dt),
          ("SVM", svm_pred, cm_svm)]

for idx, (name, pred, cm) in enumerate(models):
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['Съедобные', 'Ядовитые'])
    disp.plot(ax=axes[idx], cmap='Blues', values_format='d')
    axes[idx].set_title(f'{name}\nFalse Negative: {cm[1, 0]}',
fontweight='bold')

    for i in range(2):
        for j in range(2):
            if i == 1 and j == 0: # False Negative - самые опасные
                axes[idx].text(j, i, f'{cm[i, j]}',
                               ha="center", va="center",
                               color="red", fontweight='bold', fontsize=12)
            else:
                axes[idx].text(j, i, f'{cm[i, j]}',
                               ha="center", va="center",
                               color="black", fontsize=11)

plt.suptitle('СРАВНЕНИЕ МАТРИЦ ОШИБОК (False Negative - самые опасные ошибки)',
             fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.savefig('comparison_confusion_matrices.png', dpi=300,
bbox_inches='tight')

print("Сравнительная матрица ошибок сохранена как:
'comparison_confusion_matrices.png'")

print("\n" + "=" * 50)
```

```

print("ДЕТАЛЬНЫЙ ОТЧЕТ ДЛЯ DECISION TREE:")
print("=" * 50)
print(classification_report(y_test, dt_pred,
                            target_names=['Съедобные', 'Ядовитые'],
                            zero_division=0))

comparison_df = pd.DataFrame({
    'Model': ['k-NN', 'Decision Tree', 'SVM'],
    'Accuracy': [results['k-NN'][0], results['Decision Tree'][0],
    results['SVM'][0]],
    'Precision': [results['k-NN'][1], results['Decision Tree'][1],
    results['SVM'][1]],
    'Recall': [results['k-NN'][2], results['Decision Tree'][2],
    results['SVM'][2]],
    'F1-Score': [results['k-NN'][3], results['Decision Tree'][3],
    results['SVM'][3]]
})

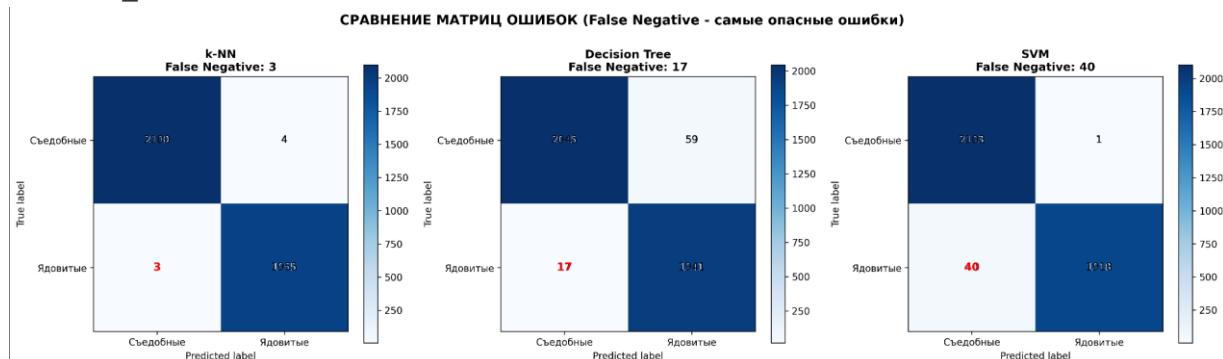
print("\n" + "=" * 60)
print("СРАВНЕНИЕ МОДЕЛЕЙ")
print("=" * 60)
print(comparison_df.round(4))

```

```

feature_importance = pd.DataFrame({
    'feature': X_train_encoded.columns,
    'importance': dt.feature_importances_
}).sort_values('importance', ascending=False)

```



```

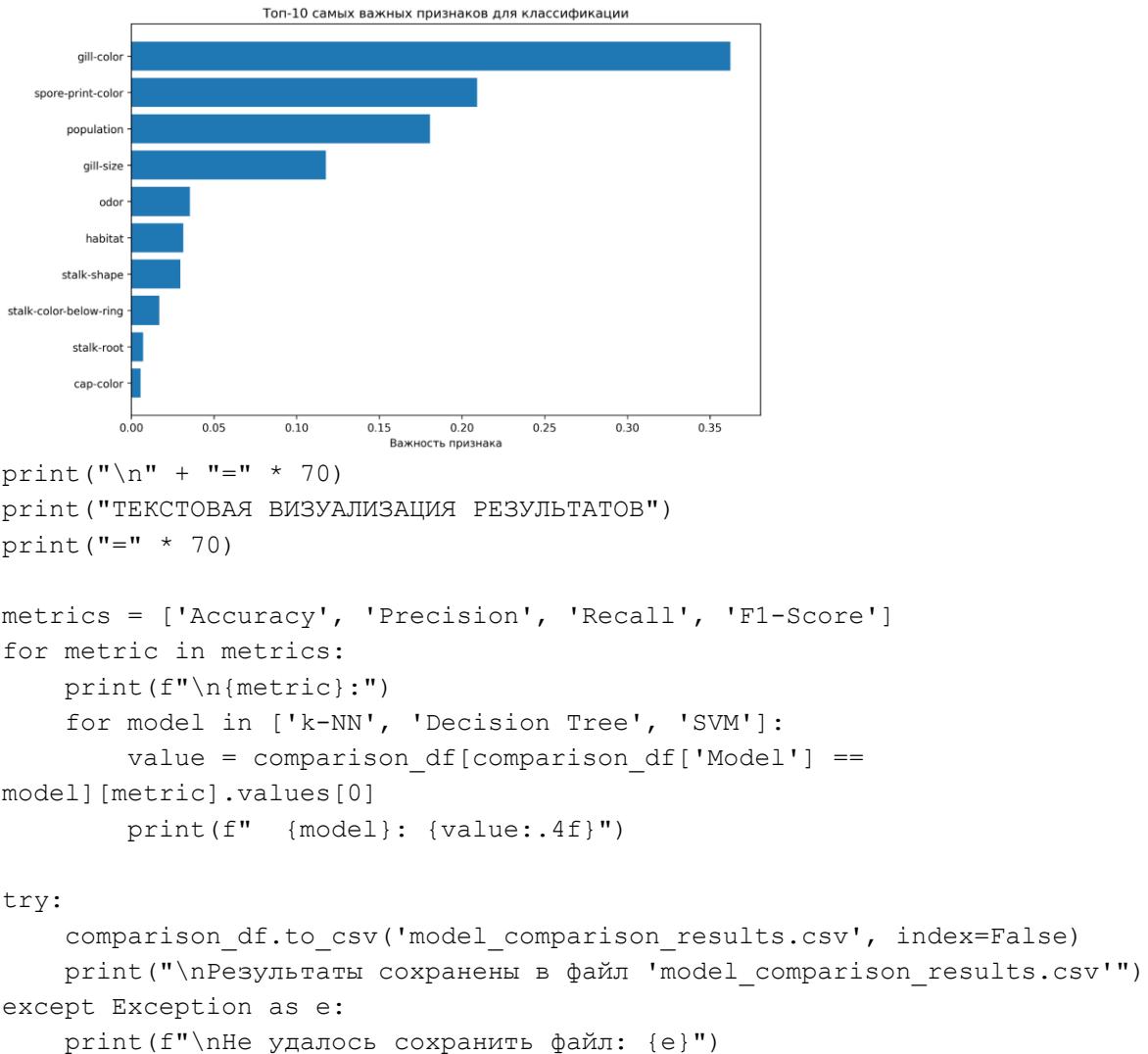
print("\n" + "=" * 50)
print("ТОП-10 ВАЖНЕЙШИХ ПРИЗНАКОВ (Decision Tree)")
print("=" * 50)
print(feature_importance.head(10).round(4))

```

```

plt.figure(figsize=(10, 6))
top_features = feature_importance.head(10)
plt.barh(top_features['feature'], top_features['importance'])
plt.xlabel('Важность признака')
plt.title('Топ-10 самых важных признаков для классификации')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.savefig('feature_importance.png', dpi=300, bbox_inches='tight')

```



Вывод программы:

```

Размеры выборок до кодирования:
Обучающая: (4062, 22)
Тестовая: (4062, 22)

Размеры после кодирования:
Обучающая: (4062, 22)
Тестовая: (4062, 22)

ТОЧНОСТЬ НА ОБУЧАЮЩЕЙ ВЫБОРКЕ:
k-NN: 0.9993
Decision Tree: 0.9774
SVM: 0.9877

ТОЧНОСТЬ НА ТЕСТОВОЙ ВЫБОРКЕ:
k-NN: 0.9983
Decision Tree: 0.9813
SVM: 0.9899

```

=====

ИССЛЕДОВАНИЕ k-NN ПРИ РАЗНОМ КОЛИЧЕСТВЕ СОСЕДЕЙ

=====

	k	Accuracy	Precision	Recall
0	1	0.9998	0.9995	1.0000
1	2	0.9998	1.0000	0.9995
2	3	0.9988	0.9975	1.0000
3	4	0.9983	0.9985	0.9980
4	5	0.9983	0.9980	0.9985
5	6	0.9973	0.9985	0.9959
6	7	0.9970	0.9974	0.9964
7	8	0.9956	0.9985	0.9923
8	9	0.9953	0.9974	0.9928
9	10	0.9943	0.9990	0.9893
10	11	0.9943	0.9974	0.9908
11	12	0.9934	0.9985	0.9877
12	13	0.9929	0.9964	0.9888
13	14	0.9921	0.9984	0.9852
14	15	0.9924	0.9979	0.9862
15	16	0.9899	0.9984	0.9806
16	17	0.9919	0.9984	0.9847
17	18	0.9882	0.9984	0.9770
18	19	0.9892	0.9974	0.9801
19	20	0.9870	0.9974	0.9755

Оптимальное количество соседей: 1

Лучшая точность: 0.9998

=====

РЕЗУЛЬТАТЫ КЛАССИФИКАЦИИ НА ТЕСТОВОЙ ВЫБОРКЕ

=====

k-Nearest Neighbors:

Accuracy: 0.9983

Precision (для ядовитых): 0.9980

Recall (для ядовитых): 0.9985

F1-Score: 0.9982

Decision Tree:

Accuracy: 0.9813

Precision (для ядовитых): 0.9705

Recall (для ядовитых): 0.9913

F1-Score: 0.9808

Support Vector Machine:

Accuracy: 0.9899

Precision (для ядовитых): 0.9995

Recall (для ядовитых): 0.9796

F1-Score: 0.9894

```
=====
МАТРИЦЫ ОШИБОК
=====
Матрица ошибок сохранена как: confusion_matrix_knn.png

k-NN Матрица ошибок:
TN: 2100, FP: 4, FN: 3, TP: 1955
Матрица ошибок сохранена как: confusion_matrix_decision_tree.png

Decision Tree Матрица ошибок:
TN: 2045, FP: 59, FN: 17, TP: 1941
Матрица ошибок сохранена как: confusion_matrix_svm.png

SVM Матрица ошибок:
TN: 2103, FP: 1, FN: 40, TP: 1918

=====
СРАВНИТЕЛЬНАЯ МАТРИЦА ОШИБОК
=====
Сравнительная матрица ошибок сохранена как: 'comparison_confusion_matrices.png'

=====
ДЕТАЛЬНЫЙ ОТЧЕТ ДЛЯ DECISION TREE:
=====

      precision    recall   f1-score   support

Съедобные       0.99     0.97     0.98     2104
Ядовитые       0.97     0.99     0.98     1958

accuracy          0.98           0.98     0.98     4062
macro avg       0.98     0.98     0.98     4062
weighted avg     0.98     0.98     0.98     4062

=====
СРАВНЕНИЕ МОДЕЛЕЙ
=====

      Model  Accuracy  Precision  Recall  F1-Score
0      k-NN    0.9983    0.9980  0.9985  0.9982
1  Decision Tree  0.9813    0.9705  0.9913  0.9808
2        SVM    0.9899    0.9995  0.9796  0.9894

=====
ТОП-10 ВАЖНЕЙШИХ ПРИЗНАКОВ (Decision Tree)
=====

      feature  importance
8          gill-color      0.3624
19         spore-print-color    0.2091
20          population      0.1806
7          gill-size        0.1177
4             odor          0.0355
21          habitat          0.0314
9          stalk-shape      0.0297
14 stalk-color-below-ring    0.0170
10         stalk-root        0.0072
2          cap-color        0.0057
```

```
=====
ТЕКСТОВАЯ ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ
=====

Accuracy:
  k-NN: 0.9983
  Decision Tree: 0.9813
  SVM: 0.9899

Precision:
  k-NN: 0.9980
  Decision Tree: 0.9705
  SVM: 0.9995

Recall:
  k-NN: 0.9985
  Decision Tree: 0.9913
  SVM: 0.9796

F1-Score:
  k-NN: 0.9982
  Decision Tree: 0.9808
  SVM: 0.9894

Результаты сохранены в файл 'model_comparison_results.csv'

Process finished with exit code 0
```

Выход: мы научились решать задачи классификации многими методами, строить матрицы ошибок, а также укрепили свои знания в области машинного обучения.