МИНОБРНАУКИ РО ССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Машинное обучение»
Тема: Исследование алгоритмов классификации

Студентка гр. 2373 _____ Шавлохова А.А.

Преподаватель _____ Татчина Я.А.

Санкт-Петербург
2024

ЗАДАНИЕ

НА ИССЛЕДОВАНИЕ АЛГОРИТМОВ КЛАССИФИКАЦИИ

ЛАБОРАТОРНУЮ

Студентка Шавлохова А.А.

Группа 2373

Тема лабораторной: Исследование алгоритмов классификации

Задание на лабораторную:

1. Необходимо оценить и сравнить результаты классификации, используя следующие
алгоритмы классификации:
* kNN
* дерево решений

2. Сравните полученные результаты с помощью различных метрик оценки качества:
Accuracy
Presicion, Recall, F-measure
ROC

3. Объяснить полученные результаты
Отчет должен включать описания выполнения каждой подзадачи.

1 Часть

Для выполнения данной практической работы был выбран датасет с сайта: https://www.kaggle.com/datasets, который был предложен в Задании 1, лабораторной работы №1.

Был выбран датасет «Mushroom Dataset (Binary Classification)» (https://www.kaggle.com/datasets/prishasawhney/mushroom-dataset). Датасет показывает набор данных о грибах (в двоичной классификации). Этот набор данных был очищен с использованием различных методов, таких как модальное вменение, горячее кодирование, нормализация z-показателя и выбор признаков.

Этот набор данных содержит следующие атрибуты:

- Cap Diameter - диаметр шляпки
- Cap Shape - форма шляпки
- Gill Attachment - прикрепление с помощью жабр
- Gill Color - цвет жабр
- Stem Height - длина стебля
- Stem Width - ширина стебля
- Stem Color - цвет стебля
- Season - сезон
- Target Class - Is it edible or not? - целевой класс - съедобен или нет?

2 Часть

1. Необходимо оценить и сравнить результаты классификации, используя следующие алгоритмы классификации:

* kNN
* дерево решений

Были выполнены алгоритмы «kNN» и «дерево решений». Дерево решений было построено (Рис.1.1) с помощью кода на отображенного на

Рис.1.2.

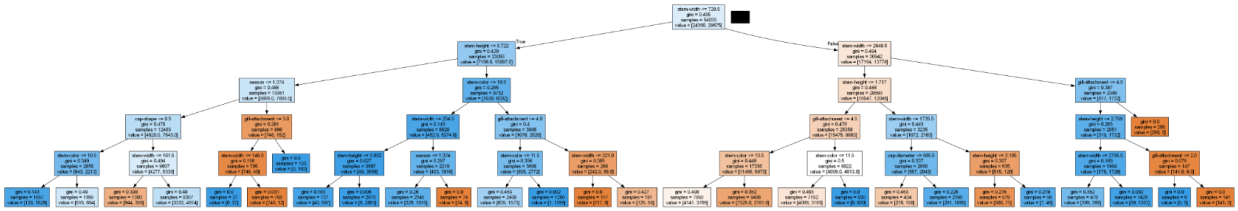

Рис.1.1 – Дерево решений для датасета

```
data = pd.read_csv('mushroom_cleaned.csv')

X = data.drop('class', axis=1)
y = data['class']

tree = DecisionTreeClassifier(max_depth=5, min_samples_split=5, min_samples_leaf=5)
tree.fit(X, y)

dot_data = StringIO()
export_graphviz(tree, feature_names=X.columns, out_file=dot_data, filled=True)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

display(Image(graph.create_png()))
```

Рис.1.2 – Код для построения дерева решений для датасета

Оценивание и сравнение результатов классификации было проведено с помощью представленных на рисунках кодов:

```
tree = DecisionTreeClassifier(random_state=17)
np.mean(cross_val_score(tree, X_train, y_train, cv=5))
```

```
np.float64(0.9746192857739533)
```

```
knn = KNeighborsClassifier()
np.mean(cross_val_score(knn, X_train, y_train, cv=5))
```

```
np.float64(0.7033896500031631)
```

Рис.1.3 – Код для вычисления средней оценки точности модели дерева решений и KNN для датасета

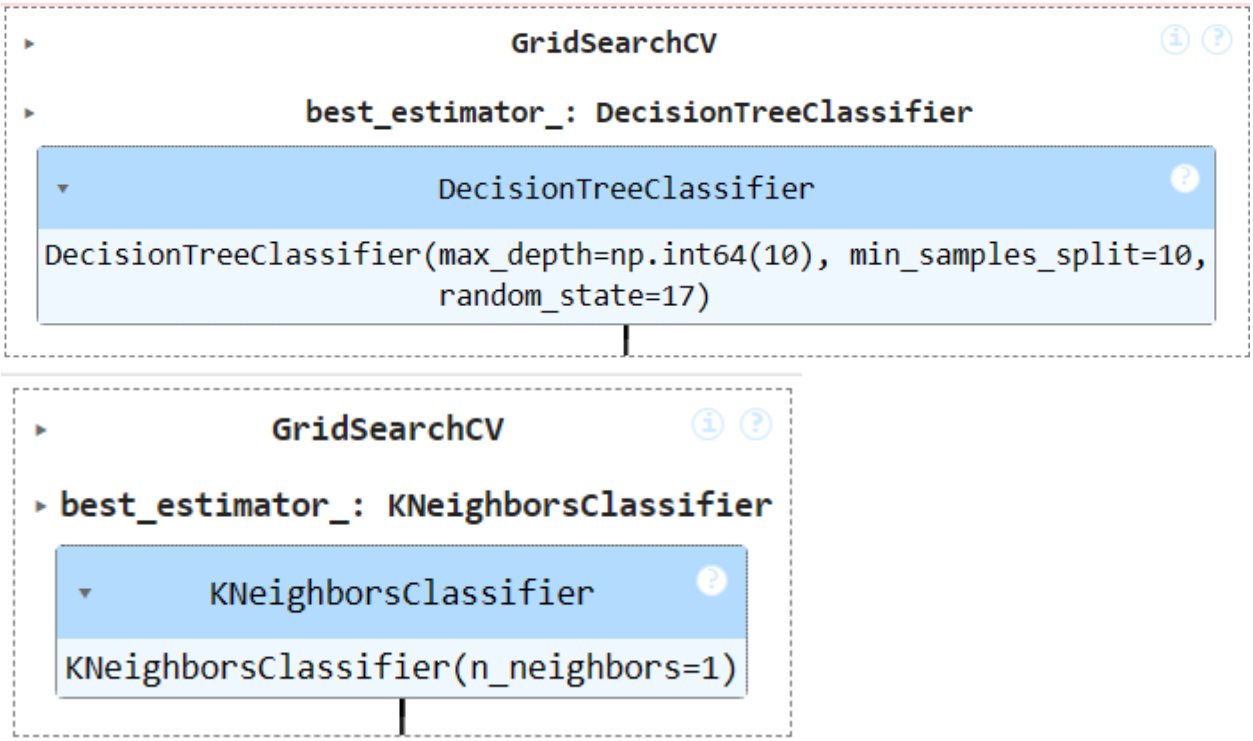

Рис.1.4-5 – Вычисление значений для оптимального построение дерева решений и KNN для датасета

По рисункам видно, что средняя оценка точности модели KNN равна 70%, а средняя оценка точности модели дерева решений равна 97%. Что показывает: насколько хорошо модели будут работать на новых данных.

Также можно заметить оптимальное количество значений для дерева решений: максимальная глубина 10 и минимальное количество образцов 10; для KNN: оптимальное число ближайших соседей 1.

При таком большом количестве значений в датасете была необходимость уменьшить глубину до подходящих значений, что ускорило работу алгоритмов и увеличило качество результатов. (На Рис.1.1 представлено дерево с глубиной 5, из-за неудобства отображения и трудноразличимости на скриншоте)

2. Сравните полученные результаты с помощью различных метрик оценки качества: Accuracy, Presicion, Recall, F-measure, ROC

Первоначально были проведены проверки:

Для поиска выбивающихся значений был выбран ящичный метод. Кривых данных не обнаружено:

```
def detect_and_visualize_outliers_horizontal(df):
    # Выбор числовых столбцов
    numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns

    if len(numeric_cols) == 0:
        print("Нет числовых столбцов для визуализации.")
        return

    num_cols = len(numeric_cols)
    num_rows = (num_cols // 4) + (num_cols % 4 > 0)
    plt.figure(figsize=(15, num_rows * 4))

    for i, col in enumerate(numeric_cols):
        plt.subplot(num_rows, 4, i + 1)  # Определяем строки и столбцы
        sns.boxplot(y=df[col], width=0.3)
        plt.title(f'Ящичный график:\n{col}')

    plt.tight_layout()
    plt.show()
```

```
detect_and_visualize_outliers_horizontal(df)
```

Рис. 2.5 – Пример кода построения ящиков

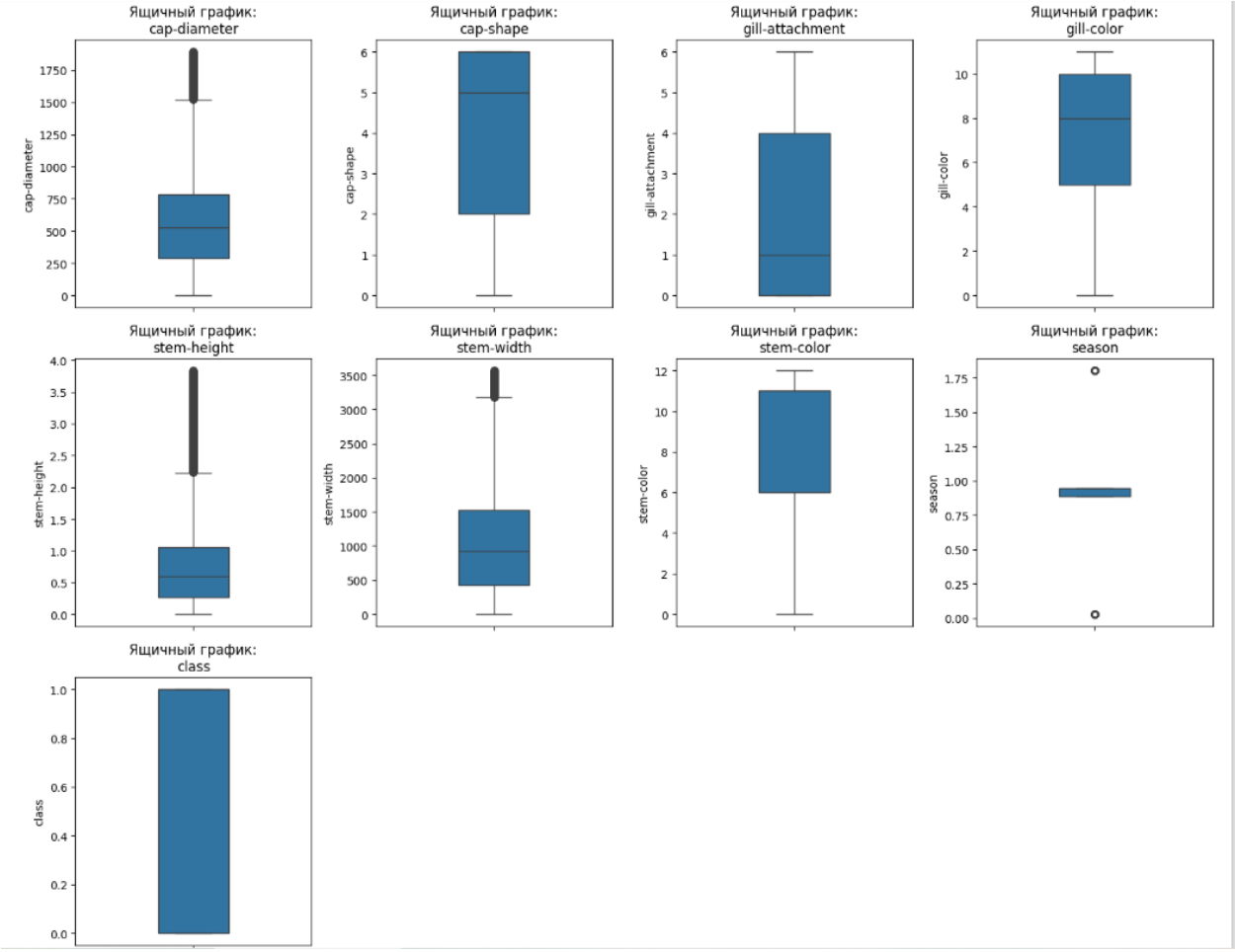

Рис. 2.6 – Выполнение кода для построения ящиков

Были выяснены метрики оценки качества:

Accuracy

KNN: 0.7232126333970761

дерево решений: 0.9072234902226883

Следовательно, дерево предсказывает верный ответ в 91% случаев, а kNN – в 72%.

Presicion

KNN: 0.7501406232422094

дерево решений: 0.9418411037107517

Следовательно, для дерева доля правильных ответов в пределах класса составляет 94%, а для kNN – 75%.

Recall

KNN: 0.7463622117752406

дерево решений: 0.8863890754421312

Следовательно, для дерева доля предсказанных объектов, относящихся к положительному классу, составила 89%, а у kNN – 75%.

F-measure

KNN: 0.7482466475901924

дерево решений: 0.9132741321646869

Следовательно, для дерева точность и полнота составили 91%, а у kNN – 75%.

ROC

KNN:

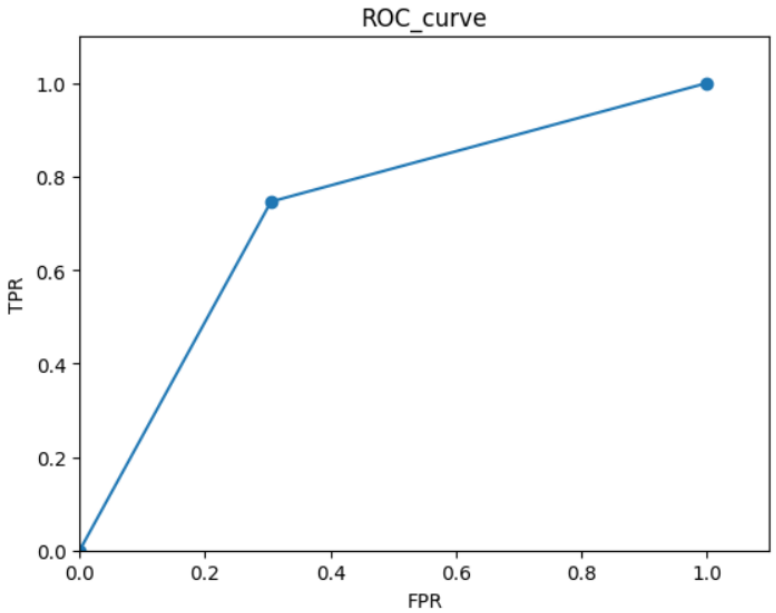

Рис. 2.7 – Кривая ROC KNN

дерево решений:

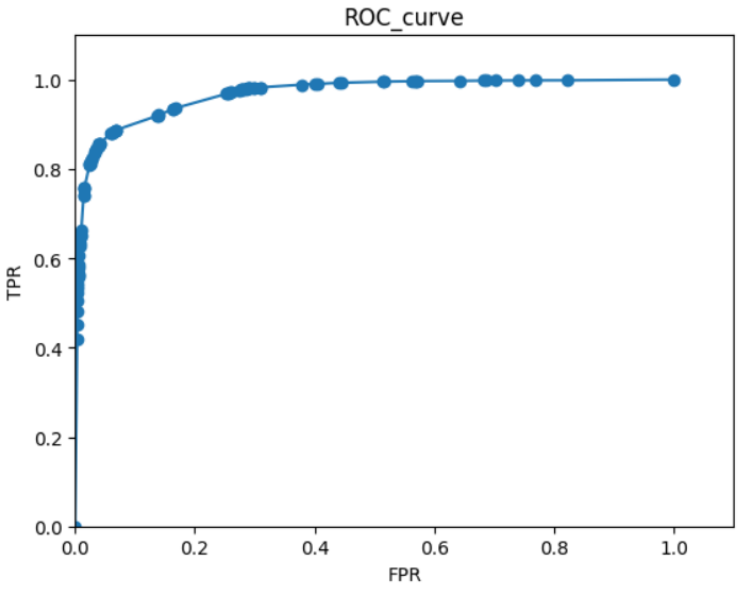

Рис. 2.8 – Кривая ROC дерева решений

Следовательно, для датасета дерево решений работает лучше, так как значения на графике выше, чем у kNN.

ЗАКЛЮЧЕНИЕ

В данной работе были изучены:

- Наборы данных;
- Jupyter Notebook;
- Поиск и обработка пропущенных значений на Python;
- Поиск выбросов на Python;
- алгоритм классификации: kNN;
- алгоритм классификации: дерево решений;
- различные метрики оценки качества (Accuracy, Presicion, Recall, F-measure, ROC).

Были построены:

- Построение графиков на Python;
- Кривая ROC для дерева решений;
- Кривая ROC для kNN;
- Дерево решений;
- Точечные графики для классов.

По построенным графикам, были сделаны и записаны соответствующие выводы.

С помощью написания лабораторной работы стало ясно: как подобрать датасет с атрибут в роли целевого класса (метки) и как оценить сбалансированность классов. Были изучены, оценены и сравнены результаты классификации, с помощью следующих алгоритмов классификации: kNN, дерево решений; и с помощью различных метрик оценки качества: Accuracy, Presicion, Recall, F-measure, ROC. Проанализировав все результаты, стало понятно, что для выбранного датасета лучше подходит метод «дерево решений», так как он показал более верный и точный показатель чем kNN. Также для более корректных результатов для работы следует использовать датасеты меньшего размера,

что улучшит как производительность так и качество алгоритмов, и данные (атрибуты) датасета должны иметь определенную степень зависимости друг от друга, иначе результаты будут разрозненными.

Для выполнения практической работы была использована программа Jupyter-ноутбук; язык программирования Python; библиотеки: matplotlib, numpy, pandas, sklearn, pydotplus, io, IPython.display.

Были изучены ссылки, представленные в методических указаниях к лабораторной работе.

Написание практической работы помогло: усвоить информацию о наборах данных, алгоритмах классификации, метриках оценки качества; разобраться с необходимыми библиотеками.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://www.kaggle.com/datasets/prishasawhney/mushroom-dataset
2. http://neerc.ifmo.ru/wiki/index.php?title=%D0%9E%D1%86%D0%B5%D0%BD%D0%BA%D0%B0_%D0%BA%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%B0_%D0%B2_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0%D1%85_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8_%D0%B8_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%B8
3. https://vk.com/video-158557357_456239019
4. https://habr.com/ru/company/ods/blog/322534/
5. https://github.com/Yorko/mlcourse.ai/blob/master/jupyter_russian/topic03_decision_trees_knn/topic3_trees_knn.ipynb