МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Кандидат технических наук, доцент		В.Ю. Скобцов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

Классификация табличных данных на основе нейросетевых моделей

по курсу: Интеллектуальный анализ данных на основе методов машинного обучения

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4134к		Столяров Н.С.
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

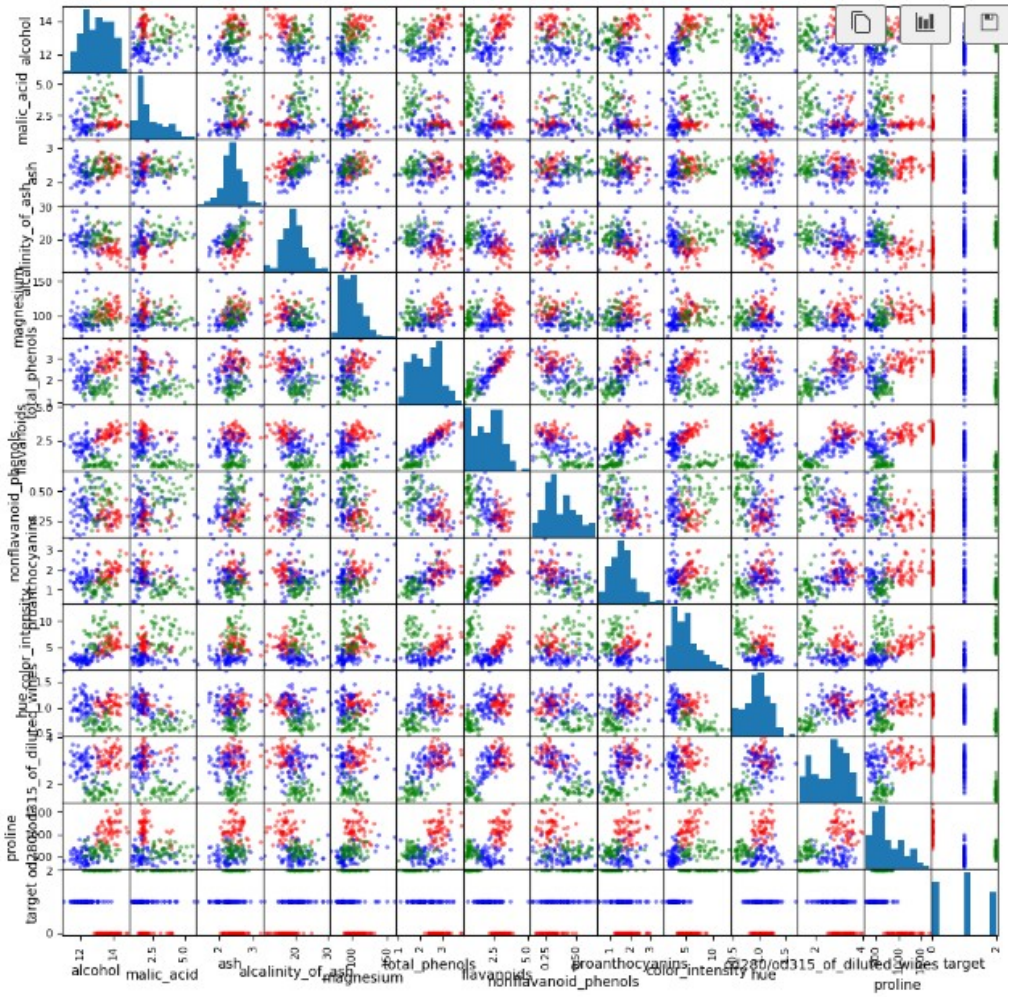
Классификация табличных данных на основе нейросетевых моделей
Дан многомерный табличный размеченный набор данных. Необходимо выполнить классификационный анализ данных по указанному целевому признаку на основе полносвязной нейросетевой модели и нейросетевой модели, указанной в варианте, в соответствии со следующей последовательностью этапов.
1. Загрузить необходимые пакеты и библиотеки.
2. Загрузить данные из указанного источника.
3. Выполнить разведочный анализ данных в соответствии с этапами, описанными в файле Этапы проекта машинного обучения в примерах.pdf:
a. Ознакомление с данными с помощью методов описательной статистики;
b. Выполнить визуализацию данных одномерную для понимания распределения данных и многомерную для выяснения зависимостей между признаками;
c. При необходимости выполнить очистку данных одним из методов.
d. Проанализировать корреляционную зависимость между признаками;
e. Поэкспериментировать с комбинациями атрибутов. При необходимости добавить новые атрибуты в набор данных.
f. Выполнить отбор существенных признаков. Сформировать набор данных из существенных признаков.
g. При необходимости преобразовать текстовые или категориальные признаки одним из методов.
h. Выполнить преобразование данных для обоих наборов (исходного и сформированного) одним из методов по варианту.
4. Анализ выполняется для исходного набора данных, преобразованного исходного набора данных, построенного набора данных и преобразованного построенного набора данных. Во всех наборах данных выделить обучающую, проверочную (валидационную) и тестовую выборки данных.
5. Сравнить качество полносвязной нейросетевой классификационной модели и классификационной нейросетевой модели, указанной в варианте, на обучающей и валидационной выборках для всех наборов данных, включая их преобразованные варианты. Для оценки качества моделей использовать метрики: accuracy, balanced_accuracy (в случае несбалансированности классов – существенное различие численности экземпляров данных в классах), $F1$ метрики (как по всей выборке, так и отдельно по классам).
6. Для лучшей модели на лучшем наборе данных оценить качество на тестовом наборе.
7. Для лучшей классификационной модели на лучшем наборе данных выполнить Grid поиск лучших гиперпараметров классификационной нейросетевой модели на обучающей и валидационной выборках. Определить значения лучших гиперпараметров.
8. Определить показатели качества полученной в результате Grid поиска классификационной нейросетевой модели на тестовом наборе. Сравнить показатели качества лучшей модели на лучшем наборе данных до поиска гиперпараметров и после поиска гиперпараметров.
9. Сделать выводы по проведенному анализу.

Вариант 2
Данные – результаты химического анализа вин, выращенных и произведенных в одном и том же регионе Италии тремя разными производителями. Для разных компонентов, обнаруженных в трех типах вина, проведено тринадцать различных измерений. Построить классификационную модель для целевого

признака «target» - признак производства вина одним из производителей.
a. Пункт 5 – простая рекуррентная сеть
b. Пункт 3.h – Стандартизация

Листинг

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

# Загрузка данных
sheet = pd.read_csv('V2_classification_lr3.csv')

# Получение значений целевой переменной
target = sheet['target']

# Создание цветовой карты на основе значений target
colors = target.map({0: 'red', 1: 'blue', 2:'green'})  # Замените 0 и 1 на ваши значения целевой переменной и соответствующие цвета

# Построение матрицы рассеяния
scatter_matrix(sheet, figsize=(12, 12), c=colors, alpha=0.5, diagonal='hist')

# Показ графика
plt.show()
```



```
print(sheet.shape)
print(sheet.dtypes)
print(sheet.describe())
print(sheet.info())
```

```
(178, 14)
alcohol                         float64
malic_acid                      float64
ash                             float64
alcalinity_of_ash               float64
magnesium                       float64
total_phenols                   float64
flavanoids                      float64
nonflavanoid_phenols            float64
proanthocyanins                 float64
color_intensity                 float64
hue                             float64
od280/od315_of_diluted_wines    float64
proline                         float64
target                            int64
dtype: object
          alcohol  malic_acid         ash  alcalinity_of_ash   magnesium  \
count  178.000000  178.000000  178.000000         178.000000  178.000000
mean    13.000618    2.336348    2.366517          19.494944   99.741573
std      0.811827    1.117146    0.274344           3.339564   14.282484
min     11.030000    0.740000    1.360000          10.600000   70.000000
25%     12.362500    1.602500    2.210000          17.200000   88.000000
50%     13.050000    1.865000    2.360000          19.500000   98.000000
75%     13.677500    3.082500    2.557500          21.500000  107.000000
max     14.830000    5.800000    3.230000          30.000000  162.000000
...
 13  target                       178 non-null    int64
dtypes: float64(13), int64(1)
memory usage: 19.6 KB
None
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Проверка на сбалансированность датасета

```
class_distribution = sheet['target'].value_counts()
print(class_distribution)
```

```
target
1    71
0    59
2    48
Name: count, dtype: int64
```

Датасет несбалансированный

RNN на стандартизированных данных

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import SimpleRNN, Dense
from keras.utils import to_categorical
from sklearn.metrics import f1_score


# Функция для получения и стандартизации данных
def get_train_test_and_val_data():
    sheet = pd.read_csv('V2_classification_lr3.csv')

    sheet_x = sheet.iloc[:, :13]
    sheet_y = sheet['target']

    # Преобразование меток в формат one-hot
    sheet_y = to_categorical(sheet_y, num_classes=3)  # Предполагается, что классы 0, 1, 2

    scaler = StandardScaler()
    sheet_x = scaler.fit_transform(sheet_x)

    seed = 7
    test_size = 0.2
    val_size = 0.25

    x_train, x_test, y_train, y_test = train_test_split(sheet_x, sheet_y, test_size=test_size, random_state=seed)
    x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=val_size, random_state=seed)

    return x_train, y_train, x_test, y_test, x_val, y_val

x_train, y_train, x_test, y_test, x_val, y_val = get_train_test_and_val_data()
```

```
print(x_train.shape)

# Изменение формы входных данных для RNN
x_train = x_train.reshape((x_train.shape[0], 1, x_train.shape[1]))
x_val = x_val.reshape((x_val.shape[0], 1, x_val.shape[1]))
x_test = x_test.reshape((x_test.shape[0], 1, x_test.shape[1]))

# Создание модели
model = Sequential()
model.add(SimpleRNN(200, activation='relu', input_shape=(x_train.shape[1],
x_train.shape[2])))
model.add(Dense(3, activation='softmax'))  # Для многоклассовой
классификации

# Компиляция модели
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Обучение модели
history = model.fit(x_train, y_train, epochs=100, batch_size=16,
validation_data=(x_val, y_val))

# Оценка модели
y_train_pred = model.predict(x_train)
y_test_pred = model.predict(x_test)
y_val_pred = model.predict(x_val)


y_train_pred_labels = np.argmax(y_train_pred, axis=1)
y_test_pred_labels = np.argmax(y_test_pred, axis=1)
y_val_pred_labels = np.argmax(y_val_pred, axis=1)


y_train_labels = np.argmax(y_train, axis=1)
y_test_labels = np.argmax(y_test, axis=1)
y_val_labels = np.argmax(y_val, axis=1)


f1_micro_train = f1_score(y_train_labels, y_train_pred_labels,
average='micro')
f1_macro_train = f1_score(y_train_labels, y_train_pred_labels,
average='macro')
f1_weighted_train = f1_score(y_train_labels, y_train_pred_labels,
average='weighted')

f1_micro_test = f1_score(y_test_labels, y_test_pred_labels,
average='micro')
f1_macro_test = f1_score(y_test_labels, y_test_pred_labels,
average='macro')
f1_weighted_test = f1_score(y_test_labels, y_test_pred_labels,
```

```
average='weighted')

f1_micro_val = f1_score(y_val_labels, y_val_pred_labels, average='micro')
f1_macro_val = f1_score(y_val_labels, y_val_pred_labels, average='macro')
f1_weighted_val = f1_score(y_val_labels, y_val_pred_labels,
average='weighted')

# Вывод результатов
print(f'F1 Micro Train: {f1_micro_train}')
print(f'F1 Macro Train: {f1_macro_train}')
print(f'F1 Weighted Train: {f1_weighted_train}')
print()
print(f'F1 Micro Test: {f1_micro_test}')
print(f'F1 Macro Test: {f1_macro_test}')
print(f'F1 Weighted Test: {f1_weighted_test}')
print()
print(f'F1 Micro Val: {f1_micro_val}')
print(f'F1 Macro Val: {f1_macro_val}')
print(f'F1 Weighted Val: {f1_weighted_val}')
```

F1 Micro Train: 1.0
F1 Macro Train: 1.0
F1 Weighted Train: 1.0
F1 Micro Test: 0.9444444444444444
F1 Macro Test: 0.9526143790849674
F1 Weighted Test: 0.9444444444444444
F1 Micro Val: 0.9722222222222222
F1 Macro Val: 0.9740129935032483
F1 Weighted Val: 0.972097284690988

RNN на исходных данных

```
def get_raw_train_test_and_val_data():
  sheet = pd.read_csv('V2_classification_lr3.csv')

  sheet_x = sheet.iloc[:, :13]

  sheet_y = sheet['target']

  # Преобразование меток в формат one-hot
  sheet_y = to_categorical(sheet_y, num_classes=3)  # Предполагается, что
классы 0, 1, 2


  seed = 7
  test_size = 0.2
  val_size = 0.25

  x_train, x_test, y_train, y_test = train_test_split(sheet_x, sheet_y,
test_size=test_size, random_state=seed)
```

```
    x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
test_size=val_size, random_state=seed)

    return x_train, y_train, x_test, y_test, x_val, y_val


x_train, y_train, x_test, y_test, x_val, y_val =
get_raw_train_test_and_val_data()
print(x_train.shape)


# Создание модели
model = Sequential()
model.add(SimpleRNN(200, activation='relu', input_shape=(x_train.shape[1],
1)))
model.add(Dense(3, activation='softmax'))  # Для многоклассовой
классификации

# Компиляция модели
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Обучение модели
history = model.fit(x_train, y_train, epochs=100, batch_size=32,
validation_data=(x_val, y_val))

# Оценка модели
y_train_pred = model.predict(x_train)
y_test_pred = model.predict(x_test)
y_val_pred = model.predict(x_val)


y_train_pred_labels = np.argmax(y_train_pred, axis=1)
y_test_pred_labels = np.argmax(y_test_pred, axis=1)
y_val_pred_labels = np.argmax(y_val_pred, axis=1)


y_train_labels = np.argmax(y_train, axis=1)
y_test_labels = np.argmax(y_test, axis=1)
y_val_labels = np.argmax(y_val, axis=1)


f1_micro_train = f1_score(y_train_labels, y_train_pred_labels,
average='micro')
f1_macro_train = f1_score(y_train_labels, y_train_pred_labels,
average='macro')
f1_weighted_train = f1_score(y_train_labels, y_train_pred_labels,
average='weighted')

f1_micro_test = f1_score(y_test_labels, y_test_pred_labels,
```

```
average='micro')
f1_macro_test = f1_score(y_test_labels, y_test_pred_labels,
average='macro')
f1_weighted_test = f1_score(y_test_labels, y_test_pred_labels,
average='weighted')

f1_micro_val = f1_score(y_val_labels, y_val_pred_labels, average='micro')
f1_macro_val = f1_score(y_val_labels, y_val_pred_labels, average='macro')
f1_weighted_val = f1_score(y_val_labels, y_val_pred_labels,
average='weighted')

# Вывод результатов
print(f'F1 Micro Train: {f1_micro_train}')
print(f'F1 Macro Train: {f1_macro_train}')
print(f'F1 Weighted Train: {f1_weighted_train}')
print()
print(f'F1 Micro Test: {f1_micro_test}')
print(f'F1 Macro Test: {f1_macro_test}')
print(f'F1 Weighted Test: {f1_weighted_test}')
print()
print(f'F1 Micro Val: {f1_micro_val}')
print(f'F1 Macro Val: {f1_macro_val}')
print(f'F1 Weighted Val: {f1_weighted_val}')
```

F1 Micro Train: 1.0
F1 Macro Train: 1.0
F1 Weighted Train: 1.0
F1 Micro Test: 0.9722222222222222
F1 Macro Test: 0.975983436853002
F1 Weighted Test: 0.9720151828847481
F1 Micro Val: 0.9722222222222222
F1 Macro Val: 0.9740129935032483
F1 Weighted Val: 0.972097284690988