Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ РТ (Радиотехнический)_____

КАФЕДРА _____ИУ5 (Системы обработки информации и управления)_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Создание веб-приложения для анализа и визуализации данных с использованием методов машинного обучения

Студентка РТ5-61 _____ Попова Д.А.
(Группа) (Подпись, дата) (Фамилия И.О.)

Руководитель курсового проекта _____ Гапанюк Ю.Е.
(Подпись, дата) (Фамилия И.О.)

Москва, 2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
(Индекс)
_____ _____
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине _____

Студентка группы РТ5-61 Попова Дарья Алексеевна
(Фамилия, имя, отчество)

Тема курсового проекта Создание веб-приложения для анализа и визуализации данных _____с использованием методов машинного обучения _____
_____

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
_____учебная_____
Источник тематики (кафедра, предприятие, НИР) _____кафедра_____

График выполнения работы: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Задание _____провести предобработку выбранного набора данных, обучить модели машинного обучения, оценить метрики качества для каждой модели, подобрать соответствующие гиперпараметры, выбрать лучшие алгоритмы для данного датасета; обернуть всё в интерактивное веб-приложение _____
_____
_____
_____
_____

Оформление курсовой работы:

Расчетно-пояснительная записка на _____ листах формата А4.

_____
_____
_____
_____

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсовой работы _____ _____
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

Введение

В рамках курса «Технологии машинного обучения» текущего семестра мы постепенно наращивали свои знания и проходили все стадии разработки продукта, связанного с анализом данных: разведочный анализ данных, предобработку признаков, обучение моделей, применений метрик качества. Были рассмотрены различные алгоритмы машинного обучения, в том числе и ансамблевые модели. Шаг за шагом мы собирали наработки в рамках лабораторных работ, и вот пришло время аккумулировать всё сделанное за семестр в курсовой работе.

Задание

На выбранном наборе данных:

- произвести разведочный анализ данных, визуализировать признаки (все или некоторые) таким образом, чтобы прояснить структуру данных;
- разделить признаки на независимые фичи и целевую переменную;
- провести всю необходимую предобработку данных (масштабирование числовых признаков, заполнение пропущенных значений, кодирование категориальных признаков);
- провести корреляционный анализ (корреляционная матрица и/или тепловая карта), сделать соответствующие выводы о независимых признаках, сильно коррелирующих между собой или с целевой переменной;
- выбрать на основании предыдущих пунктов те признаки, которые войдут в модели;
- разделить набор данных на обучающую и тестовую выборки;
- определиться с моделями машинного обучения и метриками оценки качества, подходящими для конкретной задачи (классификации или регрессии);

- обучить моделей на данных тестовой выборки без подбора гиперпараметров;
- обучить моделей на всех данных с использованием стратегий кросс-валидации с подбором гиперпараметров и выявлением оптимальных значений;
- сравнить значений метрик качества для двух последних пунктов;
- в веб-приложении сделать возможным задавать гиперпараметры для каждого алгоритма и наблюдать за изменением метрик (в том числе в виде графиков);
- сравнить полученные вручную результаты с pipeline'ом библиотеки AutoML;
- сформулировать выводы о качестве обученных моделей.

Средства реализации

Приложение реализовано на языке программирования Python с использованием веб-фреймворка для задач машинного обучения Streamlit, а также библиотек для работы с данными Pandas, Numpy, Scipy, sklearn.

Выполнение

Будем использовать датасет из 77 колонок с уровнями выделения различных белков у мышей, которые разделены на 2 группы: контрольную и трисомическую (https://www.kaggle.com/ruslankl/mice-protein-expression).

Классификация изначально многоклассовая. Класс состоит из трёх параметров и формируется по следующему принципу:

- c/t - control/trisomic - мышь из контрольной группы или с трисомией (синдромом Дауна)

- CS/SC (control shock/shock control) - поведенческий показатель, отображающий способность мыши к обучению
- m/s (memantine/saline) - некоторым мышкам вводили препарат мемантин для стимуляции способности к обучению, а некоторым - физраствор (saline).

Но стоит обратить внимание на то, что параметр CS/SC формируется полностью на основе столбца Behavior, а m/s - на основе столбца Treatment. В свою очередь то, относится мышь к контрольной группе или к группе с трисомией, определяет признак Genotype.

Я удалю столбцы Genotype, Treatment и Behavior и сделаю целевой признак бинарным. Задача, таким образом, будет сводиться к задаче бинарной классификации и будет состоять в предсказании наличия у мыши трисомии (1 - есть, 0 - нет) на основании 75 колонок с показателями выделения белков корой головного мозга (число независимых переменных уменьшилось в два раза после корреляционного анализа).

Классы поделены почти пополам, поэтому нам не придётся сталкиваться с негативными последствиями несбалансированности исходной выборки.

Листинг

```
import streamlit as st
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score,
roc_curve
from sklearn.metrics import plot_confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

```
from sklearn.model_selection import GridSearchCV
from tpot import TPOTClassifier


# загрузка датасета
@st.cache
def load_data():
    url =
'https://raw.githubusercontent.com/strangledzelda/ML_coursework/main/Data_Cor
tex_Nuclear.csv'
    data = pd.read_csv(url)
    # data =
pd.read_csv('C:\\Users\\Дасупс\\Downloads\\Data_Cortex_Nuclear.csv')
    return data


# отрисовка ROC-кривой
# функция написана Юрием Евгеньевичем Гапанюком
#
https://nbviewer.jupyter.org/github/ugapanyuk/ml_course_2021/blob/main/common
/notebooks/metrics/metrics.ipynb
def draw_roc_curve(y_true, y_score, ax, pos_label=1, average='micro'):
    fpr, tpr, thresholds = roc_curve(y_true, y_score,
                                     pos_label=pos_label)
    roc_auc_value = roc_auc_score(y_true, y_score, average=average)
    lw = 2
    ax.plot(fpr, tpr, color='magenta',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc_value)
    ax.plot([0, 1], [0, 1], color='darkgreen', lw=lw, linestyle='--')
    ax.set_xlim([0.0, 1.0])
    ax.set_xlim([0.0, 1.05])
    ax.set_xlabel('False Positive Rate')
    ax.set_ylabel('True Positive Rate')
    ax.set_title('Receiver operating characteristic')
    ax.legend(loc="lower right")


# обучение моделей и вывод результатов
def model_results(classifier, search):
    classifier.fit(X_train, y_train)
    predicted_values = classifier.predict(X_test)
    proba = classifier.predict_proba(X_test)
    y_proba = proba[:,1]
    st.write(f'accuracy = {round(accuracy_score(y_test,
predicted_values),3)}')
    st.write(f'f1 = {round(f1_score(y_test, predicted_values),3)}')
    st.write(f'ROC AUC = {round(roc_auc_score(y_test, y_proba),3)}')
    if search is False:
        fig, (ax1,ax2) = plt.subplots(ncols=2, figsize=(10, 5))
        draw_roc_curve(y_test, y_proba, ax1)
        plot_confusion_matrix(classifier, X_test, y_test, ax=ax2,
cmap=plt.cm.Blues)
        st.pyplot(fig)

data = load_data()

# разведочный анализ данных
st.header('Курсовая работа по дисциплине "Технологии машинного обучения"
студентки РТ5-61Б Поповой Дарьи')

st.subheader('Взглянем на данные')
st.write(data.head())
```

```
st.markdown('Датасет состоит из 77 колонок с уровнями выделения различных белков у мышей, которые разделены на '
            '2 группы: контрольную и трисомическую. В описании датасета упомянуто, что для измерений использовали '
            '38 мышей в контрольной группе и 34 мыши в трисомической (таким образом, всего 72 мыши). Однако сказано,'
            ' что каждую строку можно рассматривать как отдельный самостоятельный образец. Мы так и поступим.')
cols = data.columns

rows = data.shape[0]
st.write(f'Число образцов = {rows}')
col_num = rows = data.shape[1]

st.subheader('Визуализация данных')

st.write('Построим гистограммы для некоторых признаков:')
fig, (ax1,ax2) = plt.subplots(ncols=2, figsize=(10, 5))
sns.distplot(data['BDNF_N'], ax=ax1, color='#8b00ff')
sns.distplot(data['pCFOS_N'], ax=ax2, color='#8b00ff')
st.pyplot(fig)

fig, (ax1,ax2) = plt.subplots(ncols=2, figsize=(10, 5))
sns.distplot(data['GFAP_N'], ax=ax1, color='#8b00ff')
sns.distplot(data['P3525_N'], ax=ax2, color='#8b00ff')
st.pyplot(fig)
st.write('Как видно, признаки распределены нормально.')

st.write('Построим диаграммы рассеяния для некоторых пар признаков:')
fig, (ax1,ax2) = plt.subplots(ncols=2, figsize=(10, 5))
sns.scatterplot(ax=ax1, x='CDK5_N', y='EGR1_N', data=data, color='#f64a46')
sns.scatterplot(ax=ax2, x='pAKT_N', y='GFAP_N', data=data, color='#f64a46')
st.pyplot(fig)

fig, (ax1,ax2) = plt.subplots(ncols=2, figsize=(10, 5))
sns.scatterplot(ax=ax1, x='H3AcK18_N', y='BDNF_N', data=data,
color='#f64a46')
sns.scatterplot(ax=ax2, x='NR2A_N', y='SYP_N', data=data, color='#f64a46')
st.pyplot(fig)


st.subheader('Корреляционный анализ данных')
# корреляционная матрица
df_corr = data.corr()
st.write(df_corr)

# тепловая карта
fig, ax = plt.subplots()
sns.heatmap(data.corr(), ax=ax)
st.write(fig)
st.write('На основании кореляционного анализа можно сделать следующие выводы: \n Коэффициент корреляции очень близок '
         'к единице для следующих пар признаков: \n * DYRK1A_N и ITSN1_N \n * DYRK1A_N и pERK_N \n *  DYRK1A_N и BRAF_N'
         '\n * pNR1_N и NR1_N \n *  NR1_N и Bcatenin_N. \n\n Таким образом, можем не включать в модель признаки '
         'DYRK1A_N и NR1_N')
st.header('Предобработка данных')

st.markdown('Как можно заметить, классификация изначально многоклассовая. Класс состоит из трёх параметров и '
            'формируется по следующему принципу: \n* c/t - control/trisomic - мышь из контрольной группы или с '
```

```
            'трисомией (синдромом Дауна) \n* CS/SC (control shock/shock
control) - поведенческий показатель, '
            'отображающий способность мыши к обучению \n* m/s
(memantine/saline) - некоторым мышкам вводили препарат '
            'мемантин для стимуляции способности к обучению, а некоторым -
физраствор (saline). \n\nНо стоит обратить '
            'внимание на то, что параметр CS/SC формируется полностью на
основе столбца Behavior, а m/s - на основе '
            'столбца Treatment. В свою очередь то, относится мышь к
контрольной группе или к группе с трисомией, '
            'определяет признак Genotype. \n\nЯ удалю столбцы Genotype,
Treatment и Behavior и сделаю целевой признак '
            'бинарным. Задача, таким образом, будет сводиться к задаче
бинарной классификации и будет состоять в '
            'предсказании наличия у мыши трисомии (1 - есть, 0 - нет) на
основании 77 колонок с показателями '
            'выделения белков корой головного мозга.')

# делаем классификацию бинарной
data['class'] = data['class'].replace(['c-CS-m', 'c-SC-m', 'c-CS-s', 'c-SC-
s'], 0)
data['class'] = data['class'].replace(['t-CS-m', 't-SC-m', 't-CS-s', 't-SC-
s'], 1)

st.subheader('Распределение классов в целевом признаке')

# посмотрим, сколько образцов каждого класса содержится в наборе данных
labels, counters = np.unique(data['class'], return_counts=True)
labels = labels.tolist()
counters = counters.tolist()
for i in range(data['class'].nunique()):
    st.write('Количество образцов класса {} = {} ({}%)'.format(
    labels[i], counters[i], round(100 * counters[i] / data.shape[0], 2)))

st.write('Классы поделены почти пополам, поэтому нам не придётся сталкиваться
с негативными последствиями '
         'несбалансированности исходной выборки. Разделим данные на
независимые фичи и целевой признак. \n\n Для '
         'столбцов-предсказателей к тому же удалим столбец с ID каждой мыши,
а также столбцы Genotype, Treatment и '
         'Behavior.')
X = data.drop(['MouseID', 'Genotype', 'Treatment', 'Behavior', 'DYRK1A_N',
'NR1_N', 'class'], axis=1)
y = data['class']

st.subheader('Заполнение пропусков в данных')
# убедимся, что в целевой функции у нас нет пропусков
target_na = data['class'].isnull().sum()
st.write(f'В целевой функции {target_na} пропусков.')
# сначала убедимся, что все 77 фичей являются числовыми
num = 0
for column in X.columns:
    if X[column].dtype == 'float64' or X[column].dtype == 'int':
        num += 1
st.write(f'{num} фичей из {len(X.columns)} являются числовыми.')
st.write('Следовательно, кодирование категориальных признаков можно не
проводить.')

na_in_cols = []
na_in_cols_count = []

st.write('Колонки с пропусками:')
```

```
for column in X.columns:
    null_count = X[column].isnull().sum()
    if null_count > 0:
        na_in_cols.append(column)
        na_in_cols_count.append(round((null_count / X.shape[0]) * 100.0, 2))

df = pd.DataFrame(columns=na_in_cols)
df.loc[0] = na_in_cols_count

fig, ax = plt.subplots(figsize=(5, 10))
ax.set_title('Пропуски в данных')
df.loc[0].plot.barh(subplots=True, color='#926eae', legend=False)
st.pyplot(fig)

st.write('Как можно заметить, в большинстве столбцов пропусков меньше одного процента. В некоторых колонках процент '
         'пропущенных значений достигает 25-26%, но это не так критично, поэтому оставим все признаки.')
st.write('Заполняем пропуски...')
# заполним пропуски во всех колонках
for column in X.columns:
    null_count = X[column].isnull().sum()
    if null_count > 0:
        imputer = SimpleImputer(missing_values=np.nan, strategy='median')
        X[column] = imputer.fit_transform(X[[column]])
# убедимся, что пропусков не осталось
null_sum = 0
for column in X.columns:
    null_count = X[column].isnull().sum()
    null_sum += null_count
st.write('Использовалась стратегия заполнения медианой.')
st.write(f'Осталось {null_sum} пропусков')

st.subheader('Масштабирование данных')
st.write('Проверим, нужно ли будет масштабировать признаки.')
fig, ax = plt.subplots()
sns.kdeplot(data=data, legend=False)
st.write(fig)
st.write('Практически все данные распределены в промежутке [0, 3], так что масштабирование можно не проводить.')
st.write('Посмотрим, как теперь выглядят данные')
st.write(X.head())

# разделение датасета на обучающую и тестовую выборку
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)

st.header('*Построение и обучение моделей*')
st.subheader('Выбор подходящих моделей')
st.markdown('Будем обучать модели на основе следующих алгоритмов:  \n* логистическая регрессия  \n* метод k ближайших '
            'соседей  \n* решающее дерево  \n* SVM \n* градиентный бустинг \n* случайный лес')
st.subheader('Выбор подходящих метрик')
st.markdown('В качестве метрик для оценки предсказаний модели задачи классификации будем использовать:  \n* accuracy  '
            '\n* F1-меру (как среднее гармоническое между precision и recall) \n* ROC AUC \n\n Также отметим, '
            'что распределение классов в выборке сбалансировано, поэтому никаких специальных мер при расчёте метрик '
            'предпринимать не требуется.')
st.subheader('Обучение "базового" решения (baseline) без подбора гиперпараметров')
```

```
st.write('Обучение моделей производится на основе обучающей выборки, а оценка качества моделей - на основе тестовой '
         'выборки, то есть разбиение на фолды и кросс-валидация не используется.')

st.subheader('*Логистическая регрессия*')
log_reg_l1 = st.slider('l1-ratio * 10', min_value=0, max_value=10, value=5, step=1)
model_results(LogisticRegression(penalty='elasticnet', l1_ratio=log_reg_l1 * 0.1, solver='saga', max_iter=1000, random_state=42), search=False)
st.write('\n\n _После подбора гиперпараметров:_')
params = {'l1_ratio': np.arange(0, 1, 0.1)}
clf_log = GridSearchCV(LogisticRegression(penalty='elasticnet', solver='saga', max_iter=1000, random_state=42), params, cv=5, n_jobs=-1)
model_results(clf_log, search=True)
best_log = clf_log.best_params_
st.write(f'Лучшие значения гиперпараметров : {best_log}')


st.subheader('*k ближайших соседей*')
knn_slider = st.slider('n_neighbors', min_value=1, max_value=640, value=5, step=1)
model_results(KNeighborsClassifier(n_neighbors=knn_slider), search=False)
st.write('\n\n _После подбора гиперпараметров:_')
params = {'n_neighbors': range(1, 640)}
clf_knn = GridSearchCV(KNeighborsClassifier(), params, cv=5, n_jobs=-1)
model_results(clf_knn, search=True)
best_knn = clf_knn.best_params_
st.write(f'Лучшие значения гиперпараметров : {best_knn}')

st.subheader('*Решающее дерево*')
max_depth = st.slider('max_depth:', min_value=1, max_value=15, value=5, step=1)
model_results(DecisionTreeClassifier(max_depth=max_depth, random_state=42), search=False)
st.write('\n\n _После подбора гиперпараметров:_')
params = {'max_depth': range(1, 15, 1)}
clf_tree = GridSearchCV(DecisionTreeClassifier(), params, cv=5, n_jobs=-1)
model_results(clf_tree, search=True)
best_tree = clf_tree.best_params_
st.write(f'Лучшие значения гиперпараметров : {best_tree}')


st.subheader('*Метод опорных векторов*')
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
c_slider = st.slider('Степень коэффициента регуляризации C:', min_value=-3, max_value=3, value=0, step=1)
c = 10 ** c_slider
if st.checkbox('Описание гиперпараметра'):
    '''Значение слайдера - это степень, в которую будет возведена десятка. 
Т.е. при значении -3 C = 0.001,
    а при значении 2 C = 100. Дефолтная степень - нуль, т.е. C = 1. '''
gammas = [0.01, 0.2, 1, 10, 150]
gamma_value = st.select_slider('Выберите значение параметра гамма', gammas)
kernel_select = st.selectbox('Выберите тип ядра:', kernels)
degree_slider = st.slider('Выберите степень для полиномиального ядра', min_value=1, max_value=15, value=3, step=1)
model_results(SVC(kernel=kernel_select, degree=degree_slider, gamma=gamma_value, C=c, probability=True, random_state=42), search=False)
st.write('\n\n _После подбора гиперпараметров:_')
params = {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
          'C': np.geomspace(0.001, 1000, 7),
          'gamma': [0.01, 0.2, 1, 10, 150]}
```

```
clf_svc = GridSearchCV(SVC(probability=True), params, cv=5, n_jobs=-1)
model_results(clf_svc, search=True)
best_svc = clf_svc.best_params_
st.write(f'Лучшие значения гиперпараметров : {best_svc}')

st.subheader('*Случайный лес*')
if st.checkbox('Описание метода:'):
    '''
    Концепция случайного леса состоит в том, что для каждой отдельной
случайной выборки строится решающее дерево
    на основании случайного набора признаков. Объём этого набора можно
регулировать гиперпараметром max_features.
    Случайный лес ориентирован на борьбу с переобучением и, соответственно,
нацелен (вместе с бэггингом) на уменьшение
    дисперсии. Случайный лес хорошо работает на данных модели, склонных к
переобучению, в которых нет сложных
    зависимостей.
    '''
num_features_slider = st.slider('Выберите размер подмножества признаков',
min_value=1, max_value=75, value=6, step=1)
max_depth_forest = st.slider('max_depth_forest:', min_value=1, max_value=15,
value=5, step=1)
model_results(RandomForestClassifier(criterion='entropy',
max_features=num_features_slider, max_depth=max_depth_forest,
random_state=42), search=False)
st.write('\n\n _После подбора гиперпараметров:_')
params = {'max_features': range(1, 20),
          'max_depth': range(1, 10)}
clf_forest = GridSearchCV(RandomForestClassifier(), params, cv=5, n_jobs=-1)
model_results(clf_forest, search=True)
best_forest = clf_forest.best_params_
st.write(f'Лучшие значения гиперпараметров : {best_forest}')

st.subheader('*Градиентный бустинг*')
if st.checkbox('Описание метода'):
    '''
    Идея заключается в следующем: градиентный бустинг обучает первую модель
на целевом признаке.
    Вторую модель - на разнице между предсказаниями первой модели и целевого
признака.
    Третью - на разнице между предсказаниями второй и целевым признаком и так
далее.
    Каждая модель пытается скомпенсировать ошибку, каждый раз уменьшая её
степень.
    Таким образом, радиентный бустинг борется со смещением и строит модель на
основании более сложных зависимостей.
    '''
num_estimators = [10, 100, 1000]
estimator_slider = st.select_slider('Выберите число моделей', num_estimators,
value=100)
rates = [0.01, 0.1, 0.5]
learning_rate_slider = st.select_slider('Назначьте learning rate', rates,
value=0.1)
model_results(GradientBoostingClassifier(n_estimators=estimator_slider,
learning_rate=learning_rate_slider, random_state=42), search=False)
st.write('\n\n _После подбора гиперпараметров:_')
params = {'n_estimators': [10, 100, 1000],
          'learning_rate': [0.01, 0.1, 0.5]}
clf_gb = GridSearchCV(GradientBoostingClassifier(), params, cv=5, n_jobs=-1)
model_results(clf_gb, search=True)
best_gb = clf_gb.best_params_
st.write(f'Лучшие значения гиперпараметров : {best_gb}')
```

```
st.header('Использование AutoML')
tpot = TPOTClassifier(generations=5, population_size=20, cv=5,
random_state=42, verbosity=2)
tpot.fit(X_train, y_train)
automl_score = tpot.score(X_test, y_test)
st.write(f'Результат, полученный с помощью библиотеки TPOT: {automl_score}')
st.markdown(f'Лучшая модель (скопировано из терминала) -
MLPClassifier(RobustScaler(input_matrix), alpha=0.0001, '
            f'learning_rate_init=0.01). Multi-layer Perceptron classifier
находится в разделе scikit-learn с '
            f'говорящим о многом названии neural_networks...')

tpot.export('exported_pipeline.py')

st.header('Выводы')
st.markdown('* Лучше всего себя показала модель опорных векторов, все метрики
которой были равны единичкам \n\n * '
            'Однако у других моделей показатели качества несильно отличаются
от этих результатов \n\n * Разница в '
            'результатах, найденных вручную и с помощью AutoML TPOT,
очевидно, незначительна')
```

Результат выполнения

Курсовая работа по дисциплине "Технологии машинного обучения" студентки РТ5-61Б Поповой Дарьи

Взглянем на данные

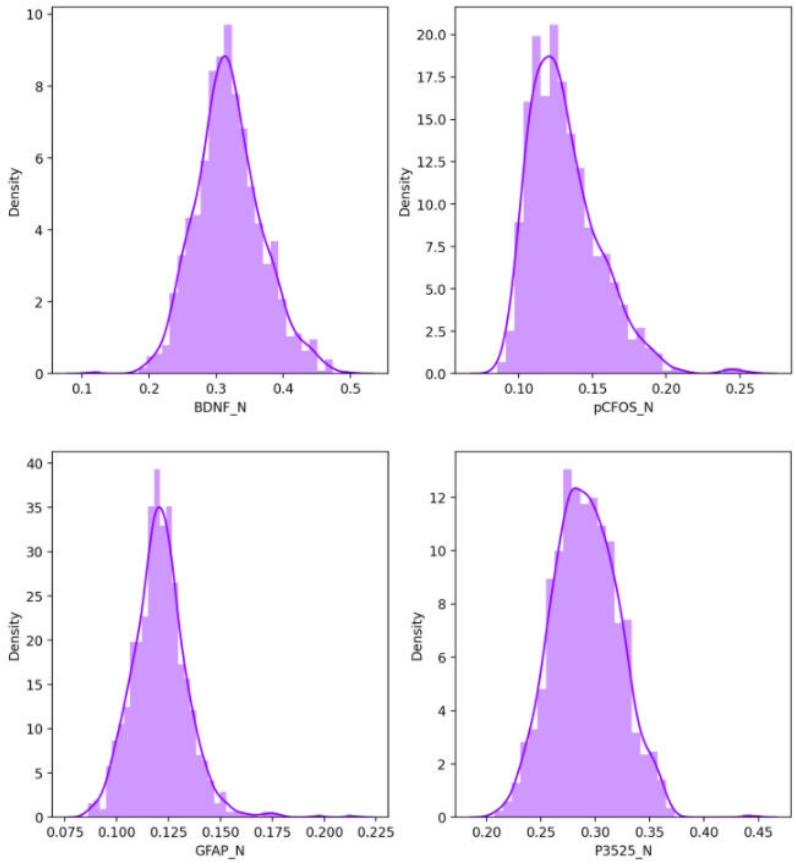
	MouseID	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_N	pCA
0	309_1	0.5036	0.7472	0.4302	2.8163	5.9902	0.2188	0.1776	
1	309_2	0.5146	0.6891	0.4118	2.7895	5.6850	0.2116	0.1728	
2	309_3	0.5092	0.7302	0.4183	2.6872	5.6221	0.2090	0.1757	
3	309_4	0.4421	0.6171	0.3586	2.4669	4.9795	0.2229	0.1765	
4	309_5	0.4349	0.6174	0.3588	2.3658	4.7187	0.2131	0.1736	

Датасет состоит из 77 колонок с уровнями выделения различных белков у мышей, которые разделены на 2 группы: контрольную и трисомическую. В описании датасета упомянуто, что для измерений использовали 38 мышей в контрольной группе и 34 мыши в трисомической (таким образом, всего 72 мыши). Однако сказано, что каждую строку можно рассматривать как отдельный самостоятельный образец. Мы так и поступим.

Число образцов = 1080

Визуализация данных

Построим гистограммы для некоторых признаков:



Как видно, признаки распределены нормально.

Корреляционный анализ данных

	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_
pAKT_N	-0.1810	-0.1478	0.3175	0.2115	0.1102	1	0.82
pBRAF_N	-0.0937	-0.0765	0.3905	0.2442	0.1111	0.8251	
pCAMKII_N	-0.1802	-0.1329	0.2468	0.3012	0.2807	0.4572	0.37
pCREB_N	0.0473	0.1711	0.6039	0.5974	0.3927	0.5971	0.58
pELK_N	0.7912	0.7809	0.4516	0.4166	0.4095	0.0375	0.11
pERK_N	0.9457	0.9063	0.3514	0.2734	0.3307	-0.1927	-0.09
pJNK_N	-0.1158	-0.0376	0.4649	0.4440	0.3915	0.7789	0.76
PKCA_N	0.2629	0.3387	0.7707	0.6143	0.5505	0.3022	0.30
pMEK_N	-0.0757	-0.0202	0.4741	0.3775	0.2604	0.8731	0.84
pNR1_N	0.2016	0.3176	0.7543	0.9479	0.8697	0.2159	0.26
pNR2A_N	-0.1849	-0.1015	0.3592	0.5100	0.5100	0.3438	0.33

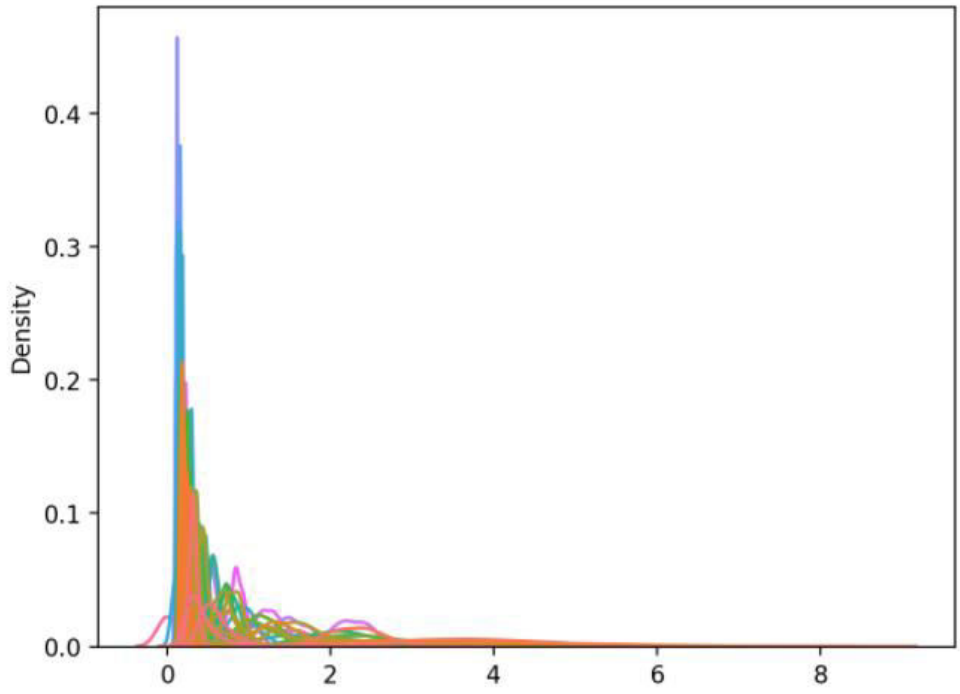
На основании кореляционного анализа можно сделать следующие выводы: Коэффициент корреляции очень близок к единице для следующих пар признаков:

- DYRK1A_N и ITSN1_N
- DYRK1A_N и pERK_N
- DYRK1A_N и BRAF_N
- pNR1_N и NR1_N
- NR1_N и Bcatenin_N.

Таким образом, можем не включать в модель признаки DYRK1A_N и NR1_N

Масштабирование данных

Проверим, нужно ли будет масштабировать признаки.



Практически все данные распределены в промежутке [0, 3], так что масштабирование можно не проводить.

Предобработка данных

Как можно заметить, классификация изначально многоклассовая. Класс состоит из трёх параметров и формируется по следующему принципу:

- c/t - control/trisomic - мышь из контрольной группы или с трисомией (синдромом Дауна)
- CS/SC (control shock/shock control) - поведенческий показатель, отображающий способность мыши к обучению
- m/s (memantine/saline) - некоторым мышкам вводили препарат мемантин для стимуляции способности к обучению, а некоторым - физраствор (saline).

Но стоит обратить внимание на то, что параметр CS/SC формируется полностью на основе столбца Behavior, а m/s - на основе столбца Treatment. В свою очередь то, относится мышь к контрольной группе или к группе с трисомией, определяет признак Genotype.

Я удалю столбцы Genotype, Treatment и Behavior и сделаю целевой признак бинарным. Задача, таким образом, будет сводиться к задаче бинарной классификации и будет состоять в предсказании наличия у мыши трисомии (1 - есть, 0 - нет) на основании 77 колонок с показателями выделения белков корой головного мозга.

Распределение классов в целевом признаке

Количество образцов класса 0 = 570 (52.78%)

Количество образцов класса 1 = 510 (47.22%)

Классы поделены почти пополам, поэтому нам не придётся сталкиваться с негативными последствиями несбалансированности исходной выборки. Разделим данные на независимые фичи и целевой признак.

Для столбцов-предсказателей к тому же удалим столбец с ID каждой мыши, а также столбцы Genotype, Treatment и Behavior.

Заполнение пропусков в данных

В целевой функции 0 пропусков.

75 фичей из 75 являются числовыми.

Следовательно, кодирование категориальных признаков можно не проводить.

Колонки с пропусками:

Построение и обучение моделей

Выбор подходящих моделей

Будем обучать модели на основе следующих алгоритмов:

- логистическая регрессия
- метод k ближайших соседей
- решающее дерево
- SVM
- градиентный бустинг
- случайный лес

Выбор подходящих метрик

В качестве метрик для оценки предсказаний модели задачи классификации будем использовать:

- accuracy
- F1-меру (как среднее гармоническое между precision и recall)
- ROC AUC

  Также отметим, что распределение классов в выборке сбалансировано, поэтому никаких специальных мер при расчёте метрик предпринимать не требуется.

Обучение "базового" решения (baseline) без подбора гиперпараметров

Обучение моделей производится на основе обучающей выборки, а оценка качества моделей - на основе тестовой выборки, то есть разбиение на фолды и кросс-валидация не используется.

Логистическая регрессия

l1-ratio * 10

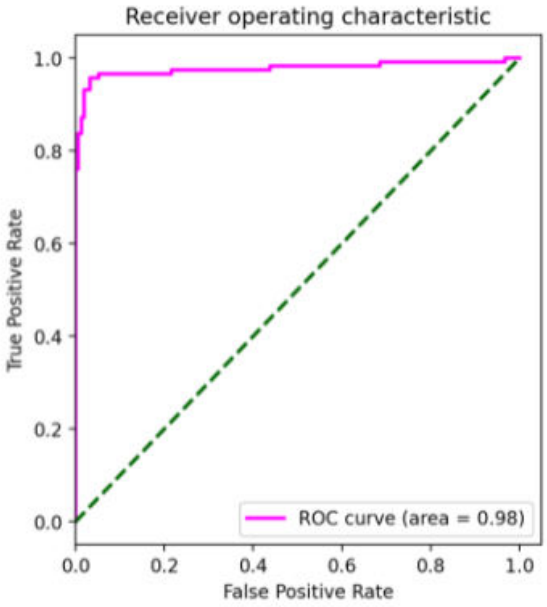
5

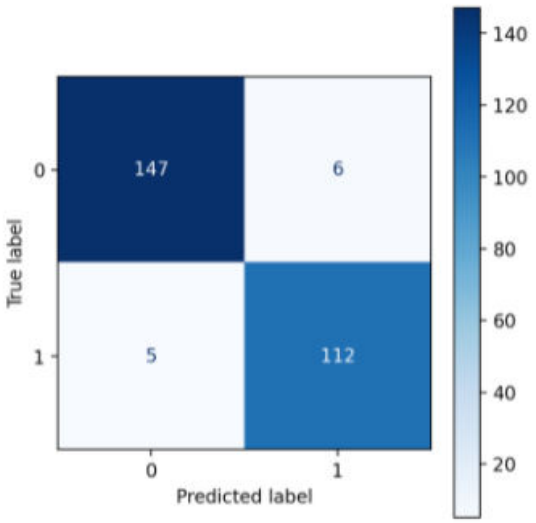
0 10

accuracy = 0.959

f1 = 0.953

ROC AUC = 0.977





После подбора гиперпараметров:

accuracy = 0.959

f1 = 0.953

ROC AUC = 0.981

Лучшие значения гиперпараметров : {'l1_ratio': 0.9}

Использование AutoML

Результат, полученный с помощью библиотеки TPOT: 0.9962962962962963

Лучшая модель (скопировано из терминала) - MLPClassifier(RobustScaler(input_matrix), alpha=0.0001, learning_rate_init=0.01). Multi-layer Perceptron classifier находится в разделе scikit-learn с говорящим о многом названии neural_networks...

Метод опорных векторов

Степень коэффициента регуляризации C:

0

-3 3

Описание гиперпараметра

Выберите значение параметра гамма

0.01

0.01 150

Выберите тип ядра:

linear

Выберите степень для полиномиального ядра

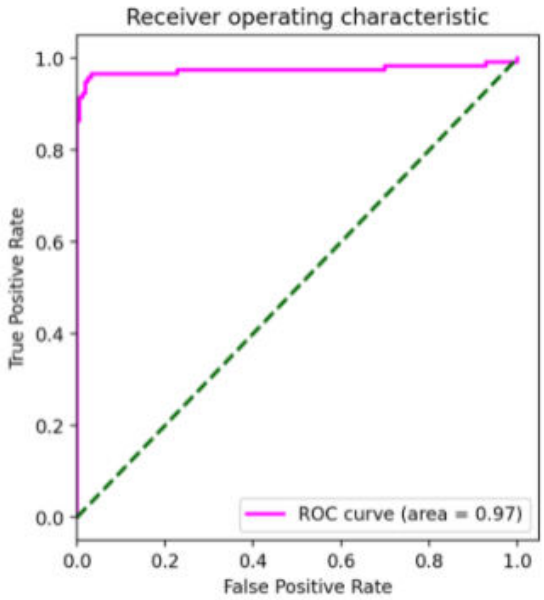
3

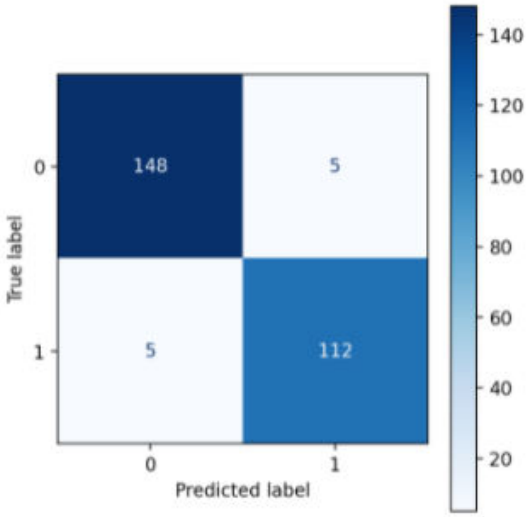
1 15

accuracy = 0.963

f1 = 0.957

ROC AUC = 0.974





Выводы

- Лучше всего себя показала модель опорных векторов, все метрики которой были равны единичкам
- Однако у других моделей показатели качества несильно отличаются от этих результатов
- Разница в результатах, найденных вручную и с помощью AutoML TPOT, очевидно, незначительна

Список источников информации

1. Репозиторий курса по Технологиям машинного обучения [Электронный ресурс] https://github.com/ugapanyuk/ml_course_2021/wiki/COURSE_TMO
2. Документация библиотеки AutoML TPOT [Электронный ресурс] https://github.com/EpistasisLab/tpot
3. Блог Александра Дьяконова. Ансамбли в машинном обучении [Электронный ресурс] https://dyakonov.org/2019/04/19/
4. Блог Александра Дьяконова. Градиентный бустинг [Электронный ресурс] https://dyakonov.org/2017/06/09/