|  |  |
| --- | --- |
|  | **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.

**ЗАДАНИЕ**

**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 4](#_Toc73563181)

[Задание 4](#_Toc73563182)

[Средства реализации 5](#_Toc73563183)

[Выполнение 5](#_Toc73563184)

[Листинг 6](#_Toc73563185)

[Выводы 20](#_Toc73563186)

[Список источников информации 20](#_Toc73563187)

# 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).

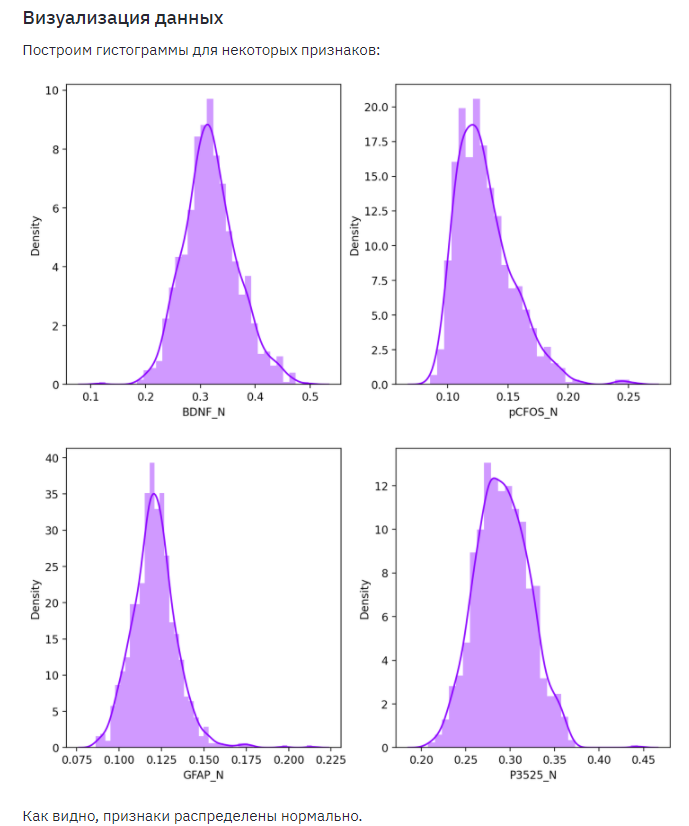
Классы поделены почти пополам, поэтому нам не придётся сталкиваться с негативными последствиями несбалансированности исходной выборки.

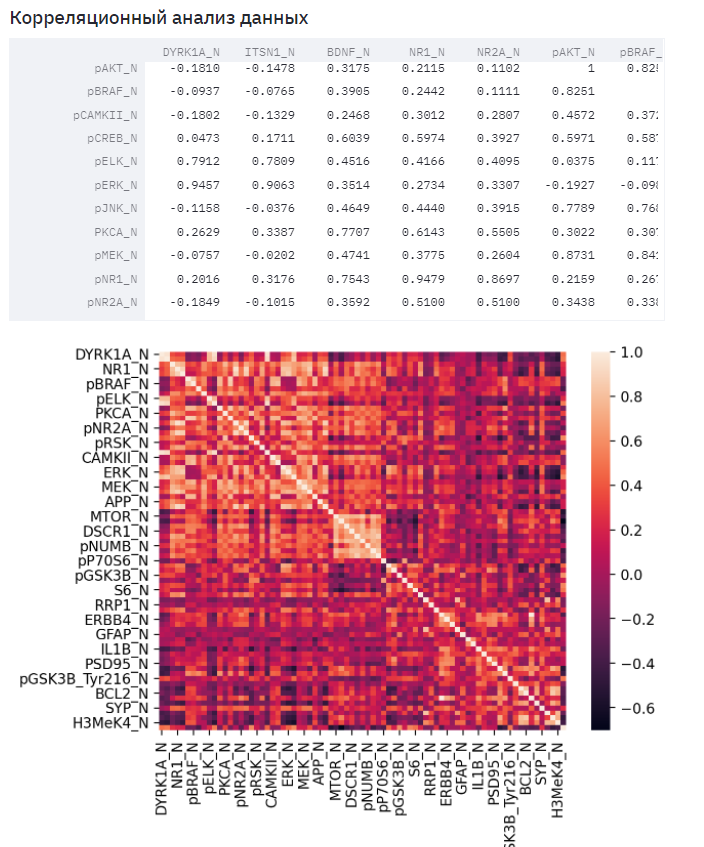
# Листинг

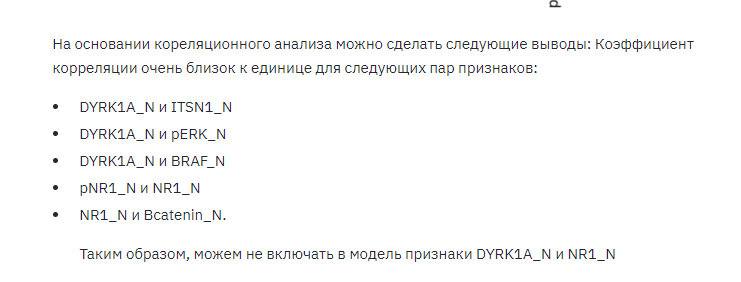
**import** streamlit **as** st  
**import** pandas **as** pd  
**import** numpy **as** np  
**import** seaborn **as** sns  
**import** matplotlib.pyplot **as** plt  
**from** sklearn.impute **import** SimpleImputer  
**from** sklearn.model\_selection **import** train\_test\_split  
**from** sklearn.linear\_model **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\_Cortex\_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 С:'**, 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 С = 0.001,   
 а при значении 2 С = 100. Дефолтная степень - нуль, т.е. С = 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'**)

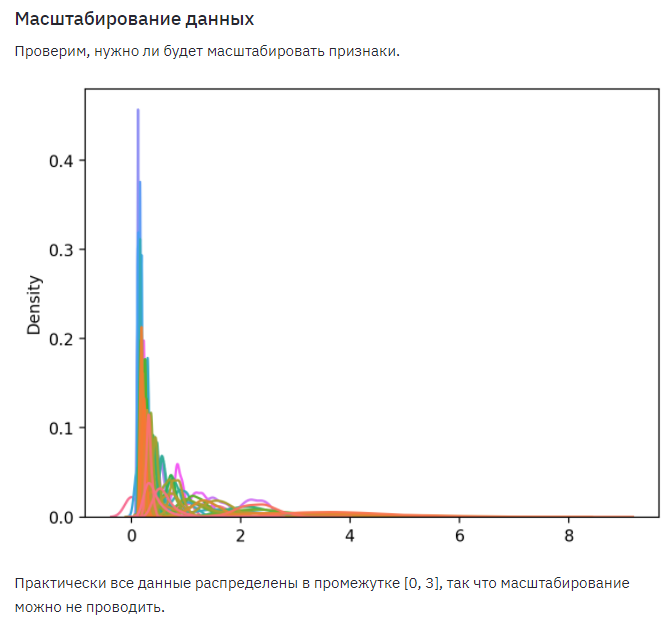
**Результат выполнения**

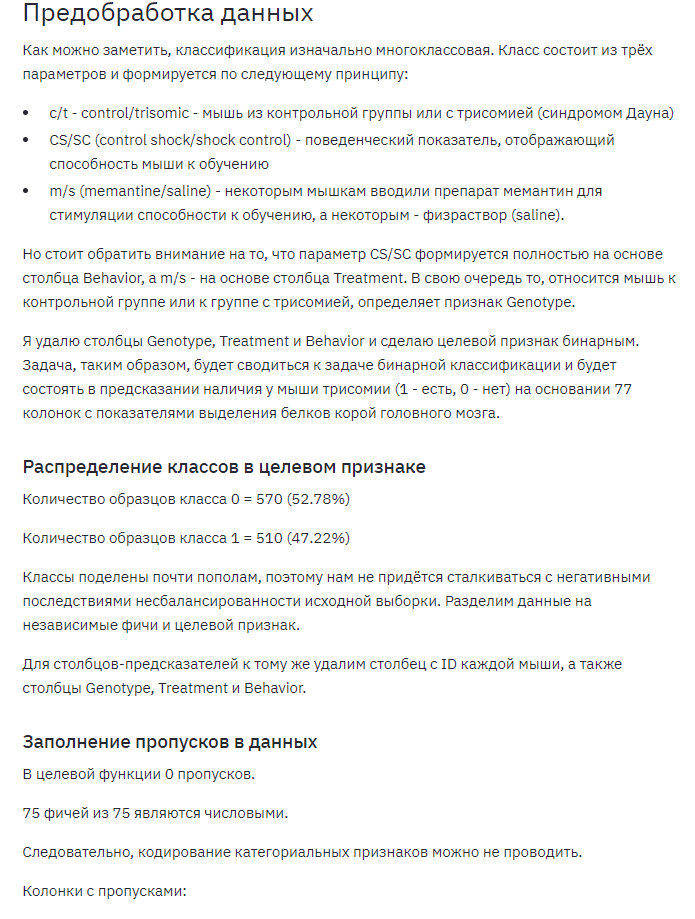


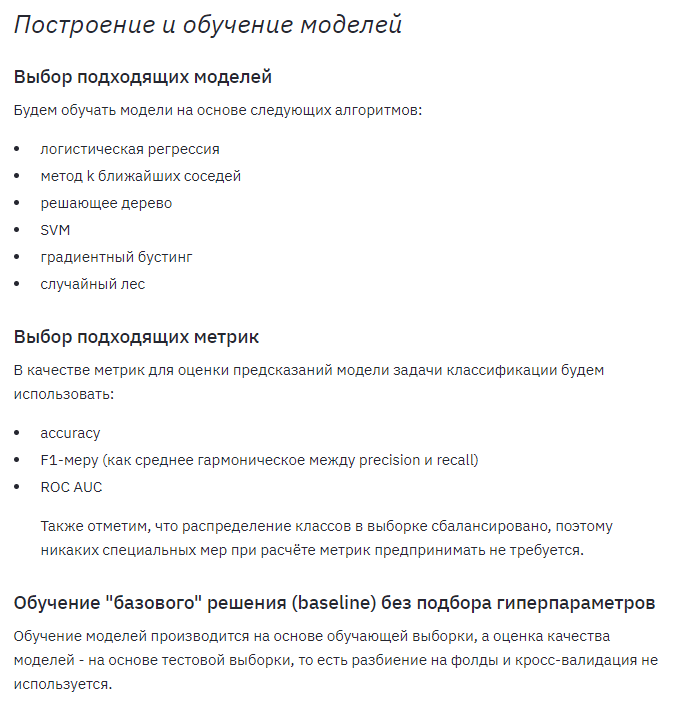


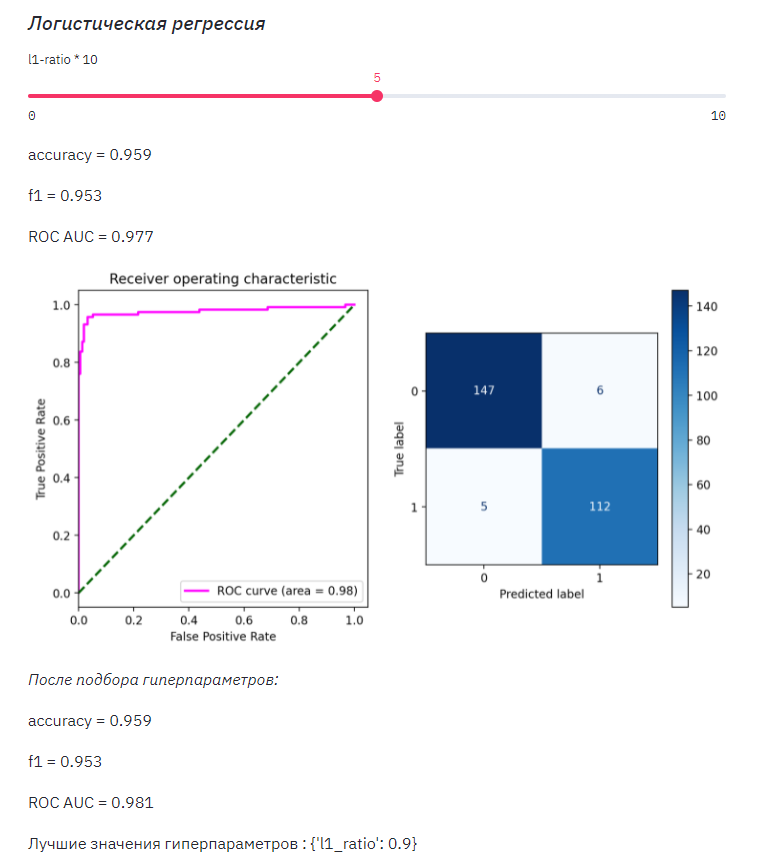


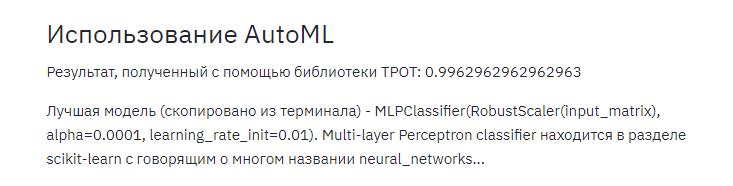


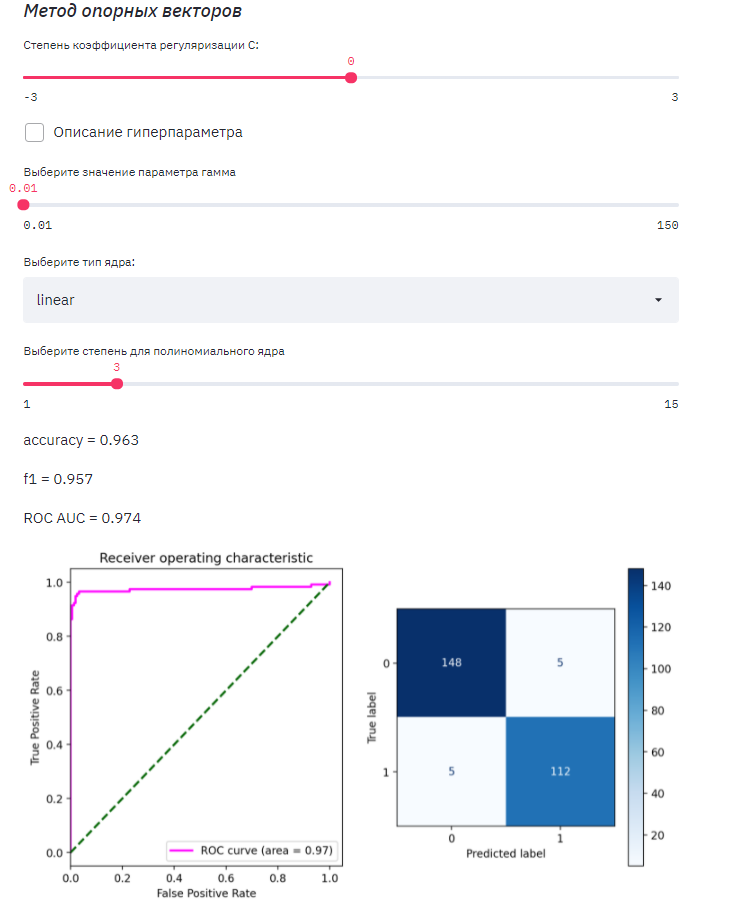




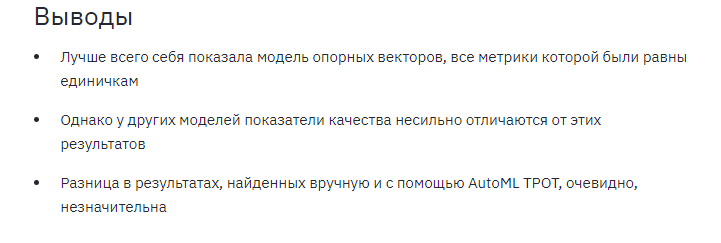








# Выводы



# 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/