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 _____Информатика и системы управления_____

КАФЕДРА _____Системы обработки информации и управления_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:
Анализ данных

Студент ____ИУ5-63____ _____ __Тарновский Д.Р.__
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ ___Гапанюк Ю.Е.___
(Подпись, дата) (И.О.Фамилия)

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

2022 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 научно-исследовательской работы

по теме _____Анализ данныз_____

Студент группы __ИУ5-63_____

_____Тарновский Даниил Романович_____
(Фамилия, имя, отчество)

Направленность НИР (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 *Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных построить модели машинного обучения для решения задачи классификации*

Оформление научно-исследовательскойработы:

Расчетно-пояснительная записка на 22 листах формата А4.

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсовой работы _____ _____Гапанюк Ю.Е.____
(Подпись, дата) (И.О.Фамилия)

Студент _____ ___Тарновский Д.Р._
(Подпись, дата) (И.О.Фамилия)

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

Оглавление

1.Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.

В качестве основного датафрейма выбрана база данных для идентификации голоса как мужского или женского на основе акустических свойств голоса и речи. Образцы голоса предварительно обрабатываются акустическим анализом в R с использованием пакетов seewave и tuneR с анализируемым частотным диапазоном от 0 до 280 Гц (диапазон человеческого голоса).

Датасет состоит из файла: voice.csv

Основной задачей выбранных данных является классификация голоса по половому признаку.
Для решения задачи классификации в качестве целевого признака будем использовать "label". Поскольку признак содержит только значения 0 и 1, то это задача бинарной классификации.

2.Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных

Импорт библиотек

Ввод [22]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from gmdhpy import gmdh
%matplotlib inline
sns.set(style="ticks")
```

Загрузка данных ¶

Ввод [23]:
```
database = pd.read_csv('voice.csv', sep=",")
```

Основные характеристики датасета

Ввод [24]:
```
# Первые 5 строк датасета
database.head()
```

Out[24]:

Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
0193	0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	0.000000	1
2666	0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	0.052632	1
1908	0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	0.046512	1
7955	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	0.247119	1
6045	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	0.208274	1

Ввод [25]:
```
database.shape, database.shape[0] * database.shape[1]
```

Out[25]:
```
((3168, 21), 66528)
```

Ввод [26]:
```
# Список колонок
database.columns
```

Out[26]:
```
Index(['meanfreq', 'sd', 'median', 'Q25', 'Q75', 'IQR', 'skew', 'kurt',
       'sp.ent', 'sfm', 'mode', 'centroid', 'meanfun', 'minfun', 'maxfun',
       'meandom', 'mindom', 'maxdom', 'dfrange', 'modindx', 'label'],
      dtype='object')
```

Ввод [27]:
```
# Список колонок с типами данных
database.dtypes
```

Out[27]:
```
meanfreq    float64
sd          float64
median      float64
Q25         float64
Q75         float64
IQR         float64
skew        float64
kurt        float64
sp.ent      float64
sfm         float64
mode        float64
centroid    float64
meanfun     float64
minfun      float64
maxfun      float64
meandom     float64
mindom      float64
maxdom      float64
dfrange     float64
modindx     float64
label         int64
dtype: object
```

Ввод [28]:
```
# Проверим наличие пустых значений
database.isnull().sum()
```

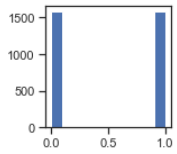
Out[28]:
```
meanfreq    0
sd          0
median      0
Q25         0
Q75         0
IQR         0
skew        0
kurt        0
sp.ent      0
sfm         0
mode        0
centroid    0
meanfun     0
minfun      0
maxfun      0
meandom     0
mindom      0
maxdom      0
dfrange     0
modindx     0
label       0
dtype: int64
```

Построение графиков для понимания структуры данных

Ввод [31]:
```
# Убедимся, что целевой признак
# для задачи бинарной классификации содержит только 0 и 1
database['label'].unique()
```

Out[31]:
```
array([1, 0], dtype=int64)
```

Ввод [32]:
```
# Оценим дисбаланс классов для Occupancy
fig, ax = plt.subplots(figsize=(2,2))
plt.hist(database['label'])
plt.show()
```



Ввод [33]:
```
database['label'].value_counts()
```

Out[33]:
```
0    1584
1    1584
Name: label, dtype: int64
```

Ввод [34]:
```
# посчитаем дисбаланс классов
total = database.shape[0]
class_0, class_1 = database['label'].value_counts()
print('Класс 0 составляет {}%, а класс 1 составляет {}%.'
      .format(round(class_0 / total, 4)*100, round(class_1 / total, 4)*100))
```

```
Класс 0 составляет 50.0%, а класс 1 составляет 50.0%.
```

Вывод. Дисбаланс классов отсутствует.

Ввод [40]:
```
database.columns
```

Out[40]:
```
Index(['meanfreq', 'sd', 'median', 'Q25', 'Q75', 'IQR', 'skew', 'kurt',
       'sp.ent', 'sfm', 'mode', 'centroid', 'meanfun', 'minfun', 'maxfun',
       'meandom', 'mindom', 'maxdom', 'dfrange', 'modindx', 'label'],
      dtype='object')
```

3.Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.

Ввод [44]:
```
database.dtypes
```

Out[44]:
```
meanfreq    float64
sd          float64
median      float64
Q25         float64
Q75         float64
IQR         float64
skew        float64
kurt        float64
sp.ent      float64
sfm         float64
mode        float64
centroid    float64
meanfun     float64
minfun      float64
maxfun      float64
meandom     float64
mindom      float64
maxdom      float64
dfrange     float64
modindx     float64
label         int64
dtype: object
```

Для построения моделей будем использовать все признаки.

Категориальные признаки отсутствуют, их кодирования не требуется.

Выполним масштабирование данных.

Ввод [48]:
```
# Числовые колонки для масштабирования
scale_cols = ['meanfreq', 'sd', 'median', 'Q25', 'Q75', 'IQR', 'skew', 'kurt',
       'sp.ent', 'sfm', 'mode', 'centroid', 'meanfun', 'minfun', 'maxfun',
       'meandom', 'mindom', 'maxdom', 'dfrange', 'modindx']
```

Ввод [49]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(database[scale_cols])
```

Ввод [50]:
```
# Добавим масштабированные данные в набор данных
for i in range(len(scale_cols)):
    col = scale_cols[i]
    new_col_name = col + '_scaled'
    database[new_col_name] = sc1_data[:,i]
```

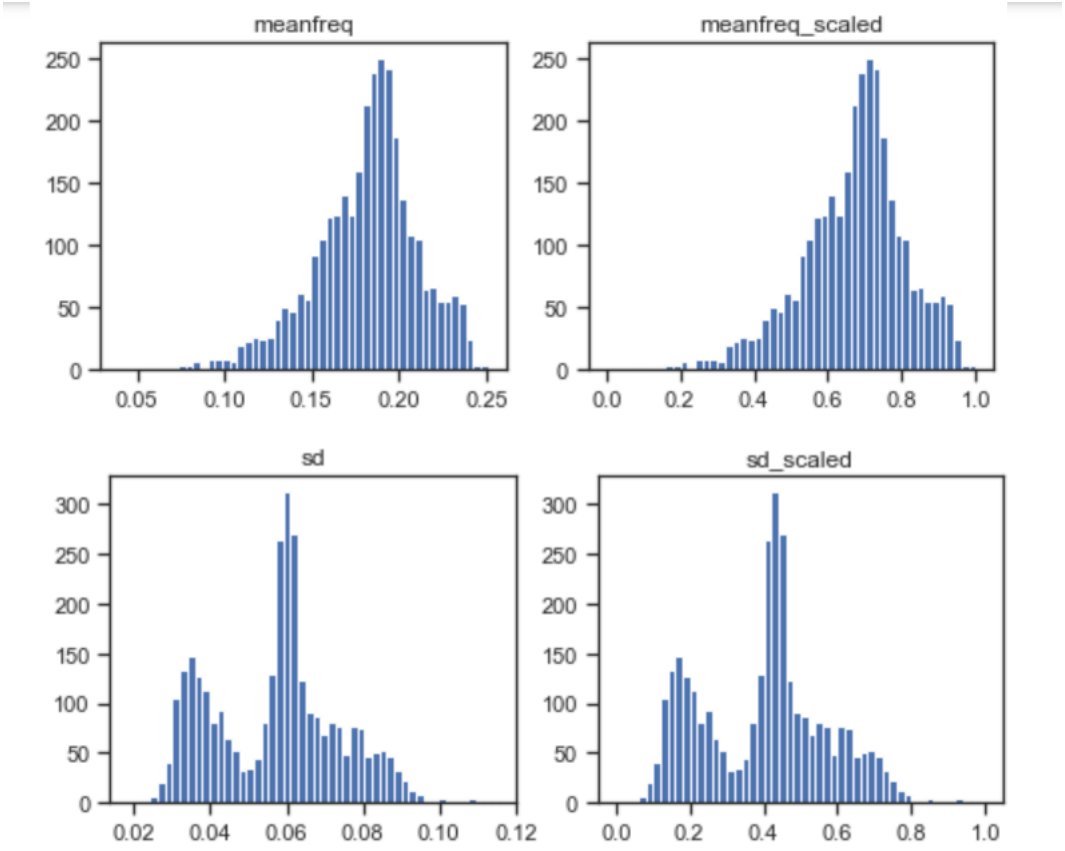
Ввод [51]:
```
database.head()
```

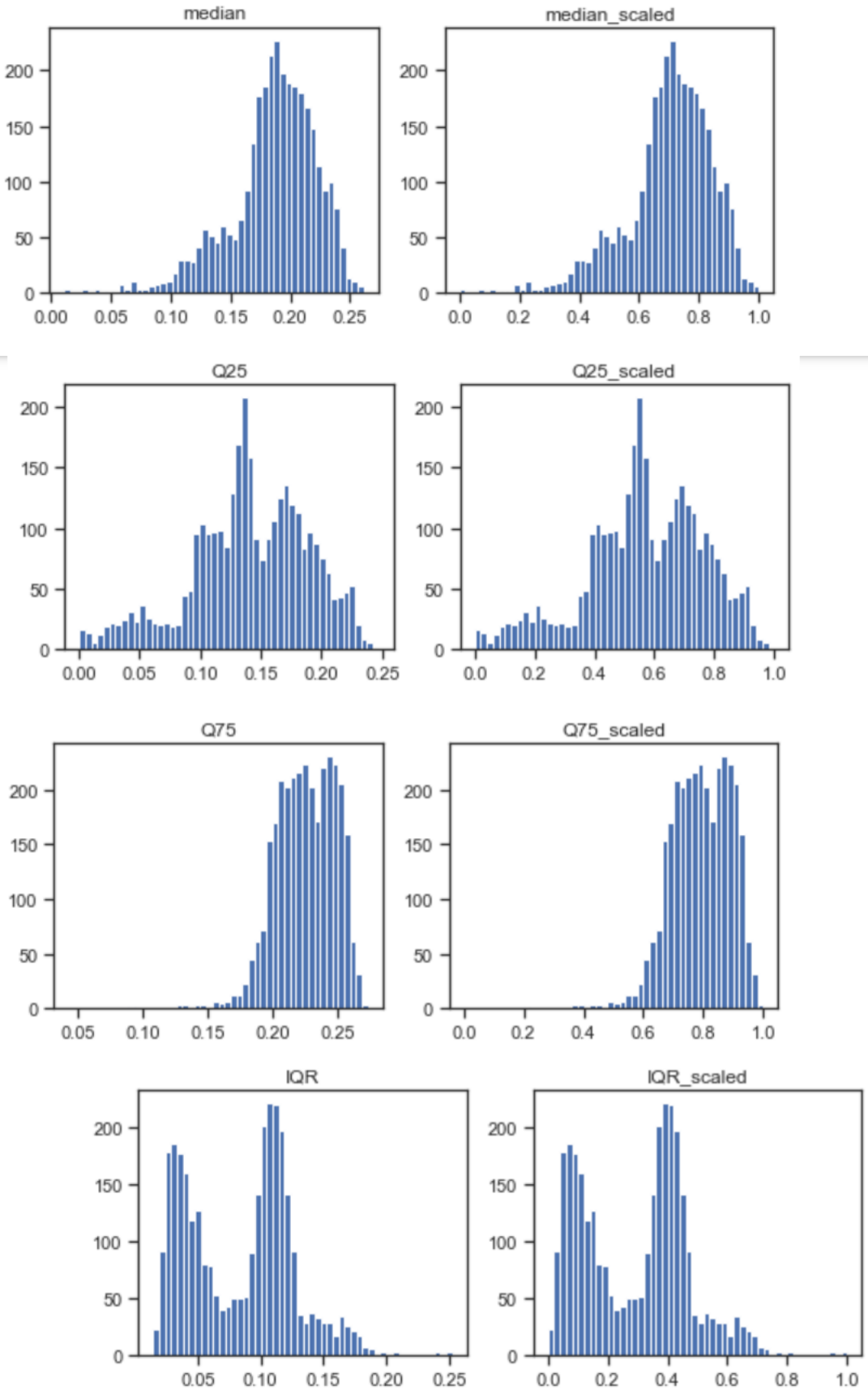
Out[51]:

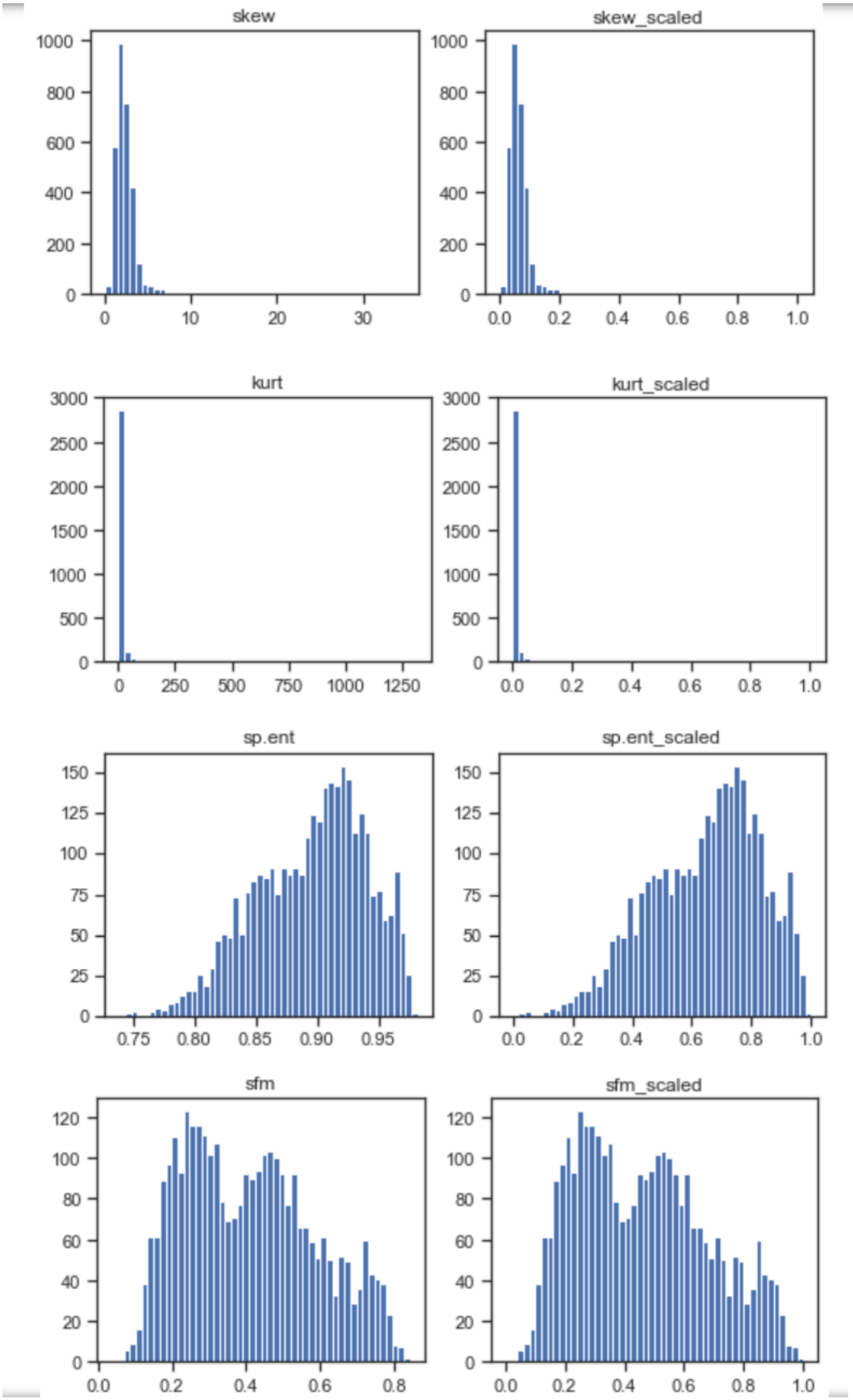
	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	mode_scaled	centroid_scaled	meanfun_scaled
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918	...	0.000000	0.096419	0.157706
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724	...	0.000000	0.125828	0.287642
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.000000	0.179222	0.236945
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.299565	0.528261	0.183442
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.372362	0.452195	0.279190

5 rows × 41 columns

Ввод [52]:
```
# Проверим, что масштабирование не повлияло на распределение данных
for col in scale_cols:
    col_scaled = col + '_scaled'

    fig, ax = plt.subplots(1, 2, figsize=(8,3))
    ax[0].hist(database[col], 50)
    ax[1].hist(database[col_scaled], 50)
    ax[0].title.set_text(col)
    ax[1].title.set_text(col_scaled)
    plt.show()
```




median
median_scaled
Q25
Q25_scaled
Q75
Q75_scaled
IQR
IQR_scaled


skew
skew_scaled
kurt
kurt_scaled
sp.ent
sp.ent_scaled
sfm
sfm_scaled

4.Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения

Воспользуемся наличием тестовых выборок,

включив их в корреляционную матрицу

Ввод [54]:
```
corr_cols_1 = scale_cols + ['label']
corr_cols_1
```

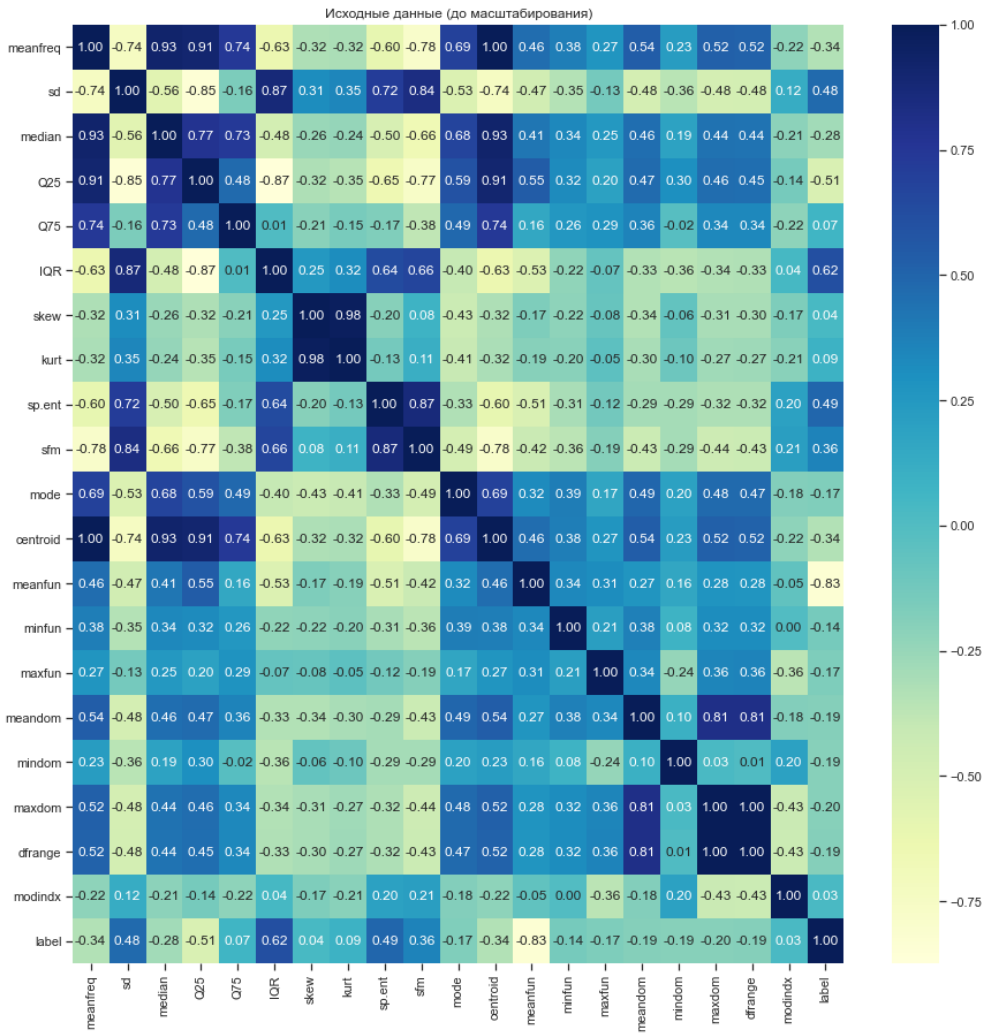
Out[54]:
```
['meanfreq',
 'sd',
 'median',
 'Q25',
 'Q75',
 'IQR',
 'skew',
 'kurt',
 'sp.ent',
 'sfm',
 'mode',
 'centroid',
 'meanfun',
 'minfun',
 'maxfun',
 'meandom',
 'mindom',
 'maxdom',
 'dfrange',
 'modindx',
 'label']
```

Ввод [59]:
```
scale_cols_postfix = [x+'_scaled' for x in scale_cols]
corr_cols_2 = scale_cols_postfix + ['label']
corr_cols_2
```

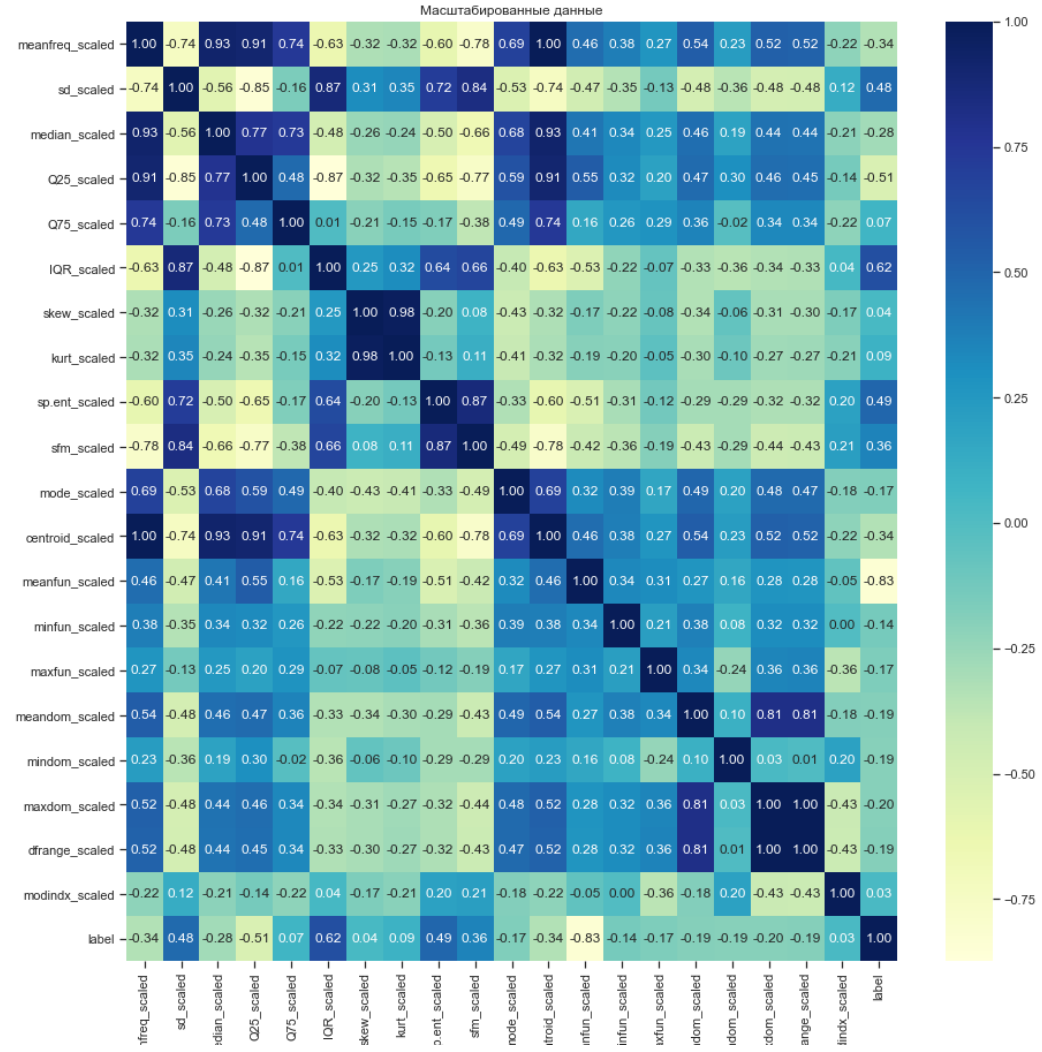
Out[59]:
```
['meanfreq_scaled',
 'sd_scaled',
 'median_scaled',
 'Q25_scaled',
 'Q75_scaled',
 'IQR_scaled',
 'skew_scaled',
 'kurt_scaled',
 'sp.ent_scaled',
 'sfm_scaled',
 'mode_scaled',
 'centroid_scaled',
 'meanfun_scaled',
 'minfun_scaled',
 'maxfun_scaled',
 'meandom_scaled',
 'mindom_scaled',
 'maxdom_scaled',
 'dfrange_scaled',
 'modindx_scaled',
 'label']
```

Ввод [62]:

```
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(database[corr_cols_1].corr(), cmap='YlGnBu', annot=True, fmt='.2f')
ax.set_title('Исходные данные (до масштабирования)')
plt.show()
```



Ввод [65]:
```
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(database[corr_cols_2].corr(), cmap='YlGnBu', annot=True, fmt='.2f')
ax.set_title('Масштабированные данные')
plt.show()
```



На основе корреляционной матрицы можно сделать следующие выводы:

- Корреляционные матрицы для исходных и масштабированных данных совпадают.

- Целевой признак классификации "label" наиболее сильно коррелирует с meanfun: среднее значение основной частоты, измеренное по акустическому сигналу (-0.83), IQR: межквантильный диапазон (0.62) , Q25: первый квантиль (-0.51) и sp.ent: спектральная энтропия (0.49) Эти признаки обязательно следует оставить в модели классификации. ¶

- Большие по модулю значения коэффициентов корреляции свидетельствуют о значимой корреляции между исходными признаками и целевым признаком. На основании корреляционной матрицы можно сделать вывод о том, что данные позволяют построить модель машинного обучения.

5.Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.

Сохранение и визуализация метрик

Ввод [70]:
```
class MetricLogger:

    def (sel__init__f):
        self.df = pd.DataFrame(
            {'metric': pd.Series([], dtype='str'),
             'alg': pd.Series([], dtype='str'),
             'value': pd.Series([], dtype='float')})

    def add(self, metric, alg, value):
        """
        Добавление значения
        """
        # Удаление значения если оно уже было ранее добавлено
        self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)].index, inplace = True)
        # Добавление нового значения
        temp = [{'metric':metric, 'alg':alg, 'value':value}]
        self.df = self.df.append(temp, ignore_index=True)

    def get_data_for_metric(self, metric, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        temp_data = self.df[self.df['metric']==metric]
        temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
        return temp_data_2['alg'].values, temp_data_2['value'].values

    def plot(self, str_header, metric, ascending=True, figsize=(5, 5)):
        """
        Вывод графика
        """
        array_labels, array_metric = self.get_data_for_metric(metric, ascending)
        fig, ax1 = plt.subplots(figsize=figsize)
        pos = np.arange(len(array_metric))
        rects = ax1.barh(pos, array_metric,
                         align='center',
                         height=0.5,
                         tick_label=array_labels)
        ax1.set_title(str_header)
        for a,b in zip(pos, array_metric):
            plt.text(0.5, a-0.05, str(round(b,3)), color='white')
        plt.show()
```

6.Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.

Будем решать задачу классификации и используем следующие модели:

- Логистическая регрессия
- Метод ближайших соседей
- Машина опорных векторов
- Решающее дерево
- Случайный лес
- Градиентный бустинг

7.Формирование обучающей и тестовой выборок на основе исходного набора данных.

Формирование обучающей и тестовой выборок на основе исходного набора данных.

Ввод [72]:
```
# Признаки для задачи классификации
task_clas_cols = ['Q25_scaled', 'IQR_scaled',
                  'meanfun_scaled','sp.ent_scaled']
```

Ввод [74]:
```
clas_X_train, clas_X_test, clas_Y_train, clas_Y_test = train_test_split(database[task_clas_cols], database['label'] , test_size=
```

Ввод [75]:
```
clas_X_train.shape, clas_X_test.shape, clas_Y_train.shape, clas_Y_test.shape
```

Out[75]:
```
((2534, 4), (634, 4), (2534,), (634,))
```

8.Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.

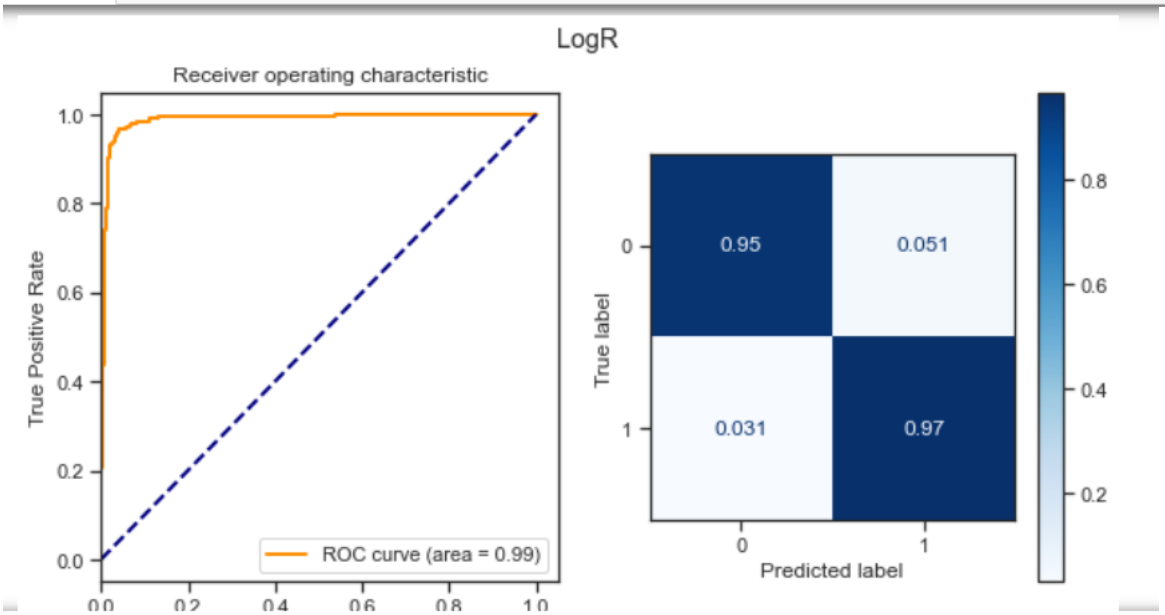
Ввод [77]:
```
# Модели
clas_models = {'LogR': LogisticRegression(),
               'KNN_5':KNeighborsClassifier(n_neighbors=5),
               'SVC':SVC(probability=True),
               'Tree':DecisionTreeClassifier(),
               'RF':RandomForestClassifier(),
               'GB':GradientBoostingClassifier()}
```

Ввод [78]:
```
# Сохранение метрик
clasMetricLogger = MetricLogger()
```

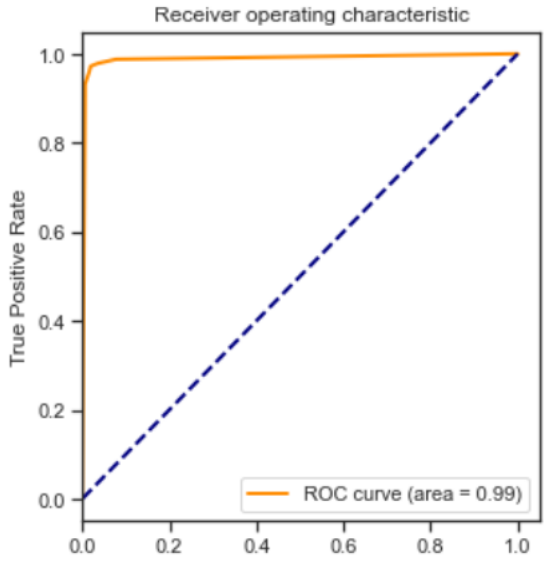
Ввод [79]:
```
# Отрисовка ROC-кривой
def draw_roc_curve(y_true, y_score, ax, pos_label=1, average='micro'):
    fpr, tpr, thresholds = roc_curve(y_true, y_score,
                                     pos_label=pos_label)
    roc_auc_value = roc_auc_score(y_true, y_score, average=average)
    #plt.figure()
    lw = 2
    ax.plot(fpr, tpr, color='darkorange',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc_value)
    ax.plot([0, 1], [0, 1], color='navy', 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")
```

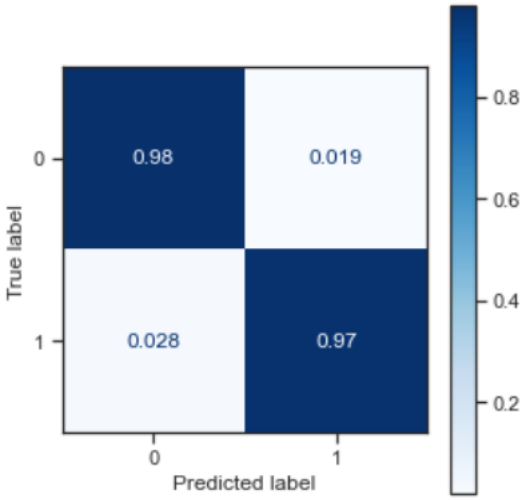
Ввод [80]:
```
def clas_train_model(model_name, model, clasMetricLogger):
    model.fit(clas_X_train, clas_Y_train)
    # Предсказание значений
    Y_pred = model.predict(clas_X_test)
    # Предсказание вероятности класса "1" для roc auc
    Y_pred_proba_temp = model.predict_proba(clas_X_test)
    Y_pred_proba = Y_pred_proba_temp[:,1]

    precision = precision_score(clas_Y_test.values, Y_pred)
    recall = recall_score(clas_Y_test.values, Y_pred)
    f1 = f1_score(clas_Y_test.values, Y_pred)
    roc_auc = roc_auc_score(clas_Y_test.values, Y_pred_proba)

    clasMetricLogger.add('precision', model_name, precision)
    clasMetricLogger.add('recall', model_name, recall)
    clasMetricLogger.add('f1', model_name, f1)
    clasMetricLogger.add('roc_auc', model_name, roc_auc)

    fig, ax = plt.subplots(ncols=2, figsize=(10,5))
    draw_roc_curve(clas_Y_test.values, Y_pred_proba, ax[0])
    plot_confusion_matrix(model, clas_X_test, clas_Y_test.values, ax=ax[1],
                          display_labels=['0','1'],
                          cmap=plt.cm.Blues, normalize='true')
    fig.suptitle(model_name)
    plt.show()
```

Ввод [81]:
```
for model_name, model in clas_models.items():
    clas_train_model(model_name, model, clasMetricLogger)
```

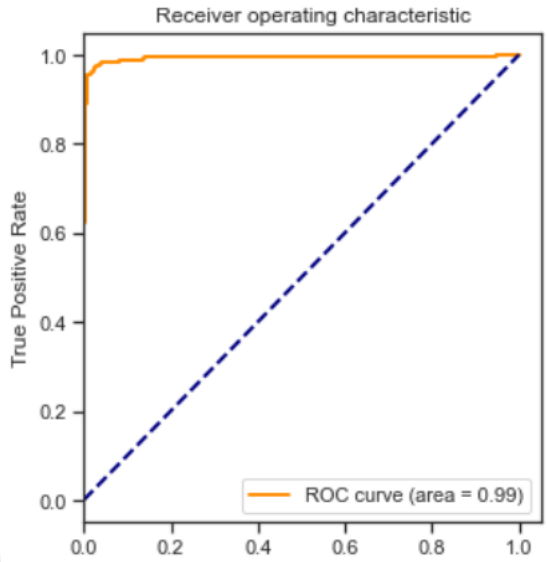


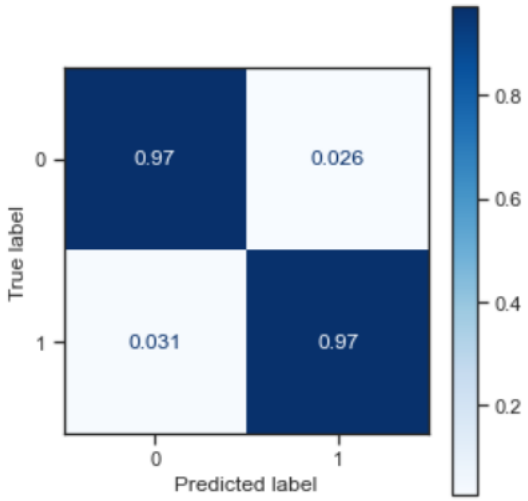
KNN_5



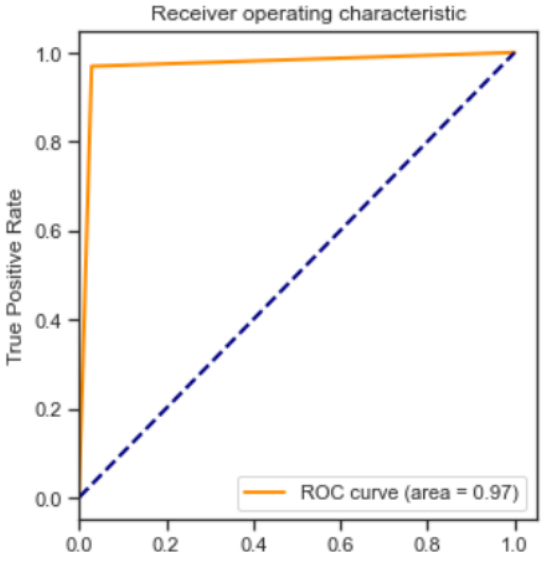


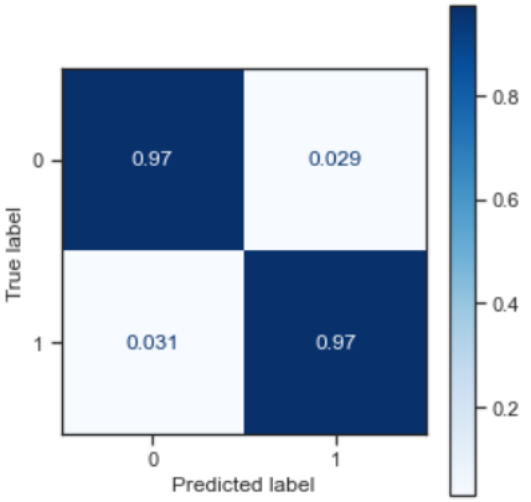
SVC



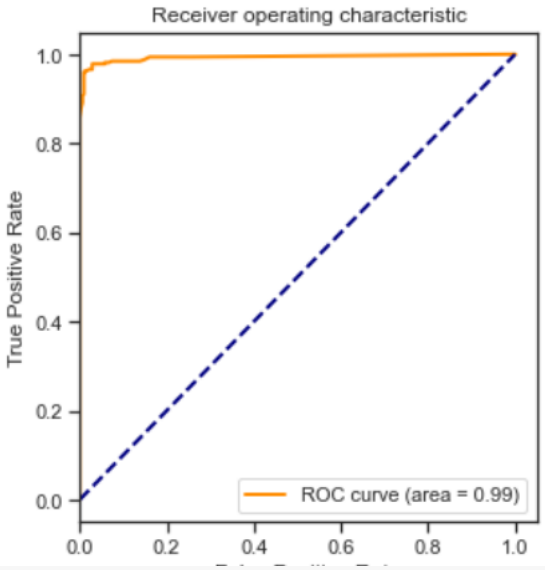


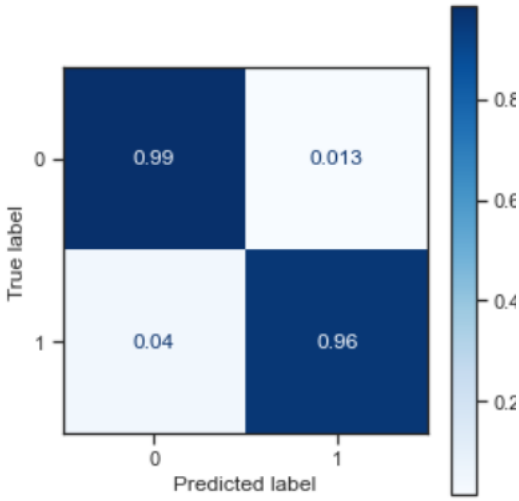
Tree



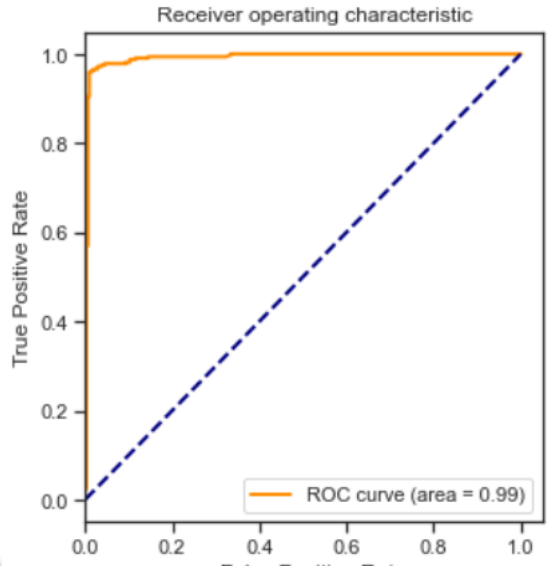


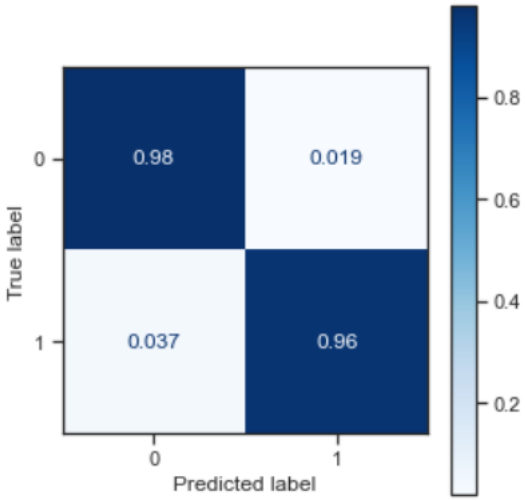
RF


Receiver operating characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.99)
0.0
0.2
0.4
0.6
0.8
1.0


0.99
0.013
0.04
0.96
True label
Predicted label
0
1

GB


Receiver operating characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.99)
0.0
0.2
0.4
0.6
0.8
1.0


0.98
0.019
0.037
0.96
True label
Predicted label
0
1

9.Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.

Подбор гиперпараметров для выбранных моделей.

Ввод [84]:
```
clas_X_train.shape
```

Out[84]:
```
(2534, 4)
```

Ввод [87]:
```
n_range_list = list(range(0,1000,1))
n_range_list[0] = 1
```

Ввод [88]:
```
n_range = np.array(n_range_list)
tuned_parameters = [{'n_neighbors': n_range}]
tuned_parameters
```

Out[88]:
```
[{'n_neighbors': array([  1,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
                  13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
                  26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
                  39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
                  52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
                  65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
                  78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
                  91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
                 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
                 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
                 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
                 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
                 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
                 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
                 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
                 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
                 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
                 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
                 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
                 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259,
                 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
                 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285,
                 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298,
                 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311,
                 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324,
                 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337,
                 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350,
                 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363,
                 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376,
                 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389,
                 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402,
                 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415,
                 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428,
                 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441,
                 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454,
                 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467,
                 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480,
                 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493,
                 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506,
                 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519,
                 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532,
                 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545,
                 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558,
                 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571,
                 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584,
                 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597,
                 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610,
                 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623,
                 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636,
                 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649,
                 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662,
                 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675,
                 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688,
                 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701,
                 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714,
                 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727,
                 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740,
                 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753,
                 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766,
                 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779,
                 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792,
                 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805,
                 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818,
                 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831,
                 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844,
                 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857,
                 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870,
                 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883,
                 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896,
                 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909,
                 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922,
                 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935,
                 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948,
                 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961,
                 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974,
                 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987,
                 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999])}]
```

Ввод [90]:
```
%%time
clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5, scoring='roc_auc')
clf_gs.fit(clas_X_train, clas_Y_train)
```

```
Wall time: 4min 55s
```

Out[90]:
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid=[{'n_neighbors': array([  1,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91...
       910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922,
       923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935,
       936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948,
       949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961,
       962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974,
       975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987,
       988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999])}],
             scoring='roc_auc')
```

Ввод [91]:
```
# Лучшая модель
clf_gs.best_estimator_
```

Out[91]:
```
KNeighborsClassifier(n_neighbors=13)
```

Ввод [92]:
```
# Лучшее значение параметров
clf_gs.best_params_
```

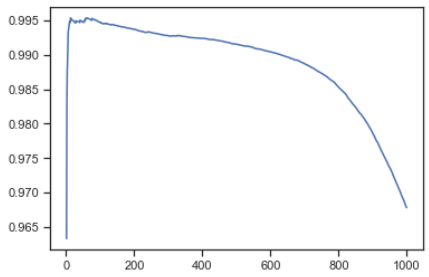
Out[92]:
```
{'n_neighbors': 13}
```

Ввод [93]:
```
clf_gs_best_params_txt = str(clf_gs.best_params_['n_neighbors'])
clf_gs_best_params_txt
```

Out[93]:
```
'13'
```

Ввод [94]:
```
# Изменение качества на тестовой выборке в зависимости от K-соседей
plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

Out[94]:
```
[<matplotlib.lines.Line2D at 0x2a7e37c15e0>]
```



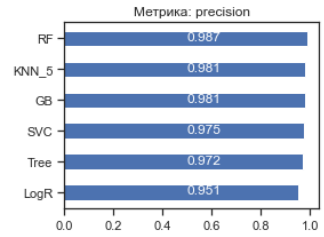
11.Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется построение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

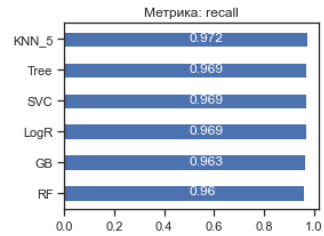
Формирование выводов о качестве построенных моделей на основе выбранных метрик.

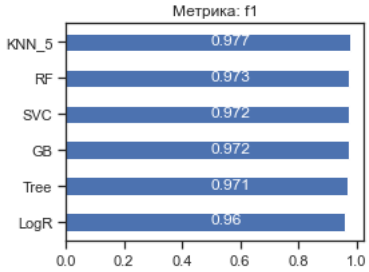
Ввод [97]:
```
# Метрики качества модели
clas_metrics = clasMetricLogger.df['metric'].unique()
clas_metrics
```

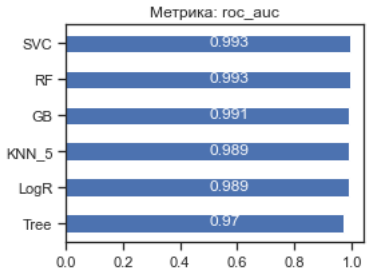
Out[97]:
```
array(['precision', 'recall', 'f1', 'roc_auc'], dtype=object)
```

Ввод [98]:
```
# Построим графики метрик качества модели
for metric in clas_metrics:
    clasMetricLogger.plot('Метрика: ' + metric, metric, figsize=(4, 3))
```









Источники:

1. https://nbviewer.org/github/ugapanyuk/ml_course_2022/blob/main/common/notebooks/ml_project_example/project_classification_regression.ipynb
2. https://github.com/ugapanyuk/ml_course_2022/wiki/COURSE_TMO