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 (ИУ5)_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Решение комплексной задачи машинного обучения

Студент ИУ5-62Б _____ И.А. Гришин
(Группа) (Подпись, дата) (И.О.Фамилия)

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

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

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 ____ИУ5___
(Индекс)
_____ В.М. Черненький
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине Технологии машинного обучения

Студент группы ИУ5-62Б

Гришин Илья Алексеевич
(Фамилия, имя, отчество)

Тема курсового проекта Решение комплексной задачи машинного обучения

Направленность КП (учебный, исследовательский, практический, производственный, др.)

Источник тематики (кафедра, предприятие, НИР) _____

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

Задание

Оформление курсового проекта:

Расчетно-пояснительная записка на 24 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсового проекта _____ И.А. Гришин
(Подпись, дата) (И.О.Фамилия)

Студент _____ Ю.Е. Гапанюк
(Подпись, дата) (И.О.Фамилия)

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

Оглавление

1. Введение

Курсовой проект – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного обучения. Результатом курсового проекта является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

2. Основная часть

2.1. Схема типового исследования

Схема типового исследования содержит выполнение следующих шагов:

1. Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных необходимо построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.
2. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.
3. Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
4. Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения.
5. Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.
6. Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.
7. Формирование обучающей и тестовой выборок на основе исходного набора данных.
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.
9. Подбор гиперпараметров для выбранных моделей с помощью методов кросс-валидации. В зависимости от используемой библиотеки можно

применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.

10. Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.
11. Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества отобразить в виде графиков и сделать выводы в форме текстового описания. Построение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.
12. Разработка макета веб-приложения, предназначенного для анализа данных.
13. Применение к выбранному набору данных произвольной библиотеки AutoML и сравнение качества моделей, полученных вручную и с использованием AutoML.

2.2. Текст программы

```
import streamlit as st
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_l
og_error, median_absolute_error, r2_score
from sklearn.svm import SVR, NuSVR, LinearSVR
from sklearn.tree import DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
import time
from supervised.automl import AutoML # mljar-supervised

@st.cache(allow_output_mutation=True)
def load_data():
    '''
    Загрузка данных
    '''
    data = pd.read_csv('bike-hour.csv', sep=",")
    return data
```

```
@st.cache(allow_output_mutation=True)
def preprocess_data(data_in):
    '''
    Масштабирование признаков, функция возвращает X и y для кросс-валидации
    '''
    data_out = data_in.copy()
    # Числовые колонки для масштабирования
    scale_cols = ['season', 'hr', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'hum', 'windspeed',
       'casual']
    new_cols = []
    sc1 = MinMaxScaler()
    sc1_data = sc1.fit_transform(data_out[scale_cols])
    for i in range(len(scale_cols)):
        col = scale_cols[i]
        new_col_name = col + '_scaled'
        new_cols.append(new_col_name)
        data_out[new_col_name] = sc1_data[:,i]

    temp_X = data_out[new_cols]
    temp_y = data_out['cnt']
    data_X_train, data_X_test, data_y_train, data_y_test  = train_test_split(temp_X
, temp_y, test_size=0.25, random_state=1)
    return data_X_train, data_X_test, data_y_train, data_y_test
class MetricLogger:

    def __init__(self):
        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)].ind
ex, 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='black')
        st.pyplot(fig)

models_list = ['LR', 'KNN_5', 'SVR', 'Tree', 'RF', 'GB', 'AutoML']

st.header('Курсовой проект')

st.sidebar.header('Выбор наиболее подходящих моделей для решения задачи регрессии')
st.sidebar.write("""
           Для задачи регрессии будем использовать следующие модели:
            - Линейная регрессия
            - Метод ближайших соседей
            - Машина опорных векторов
            - Решающее дерево
            - Случайный лес
            - Градиентный бустинг
        """)

st.sidebar.header('Модели машинного обучения')
models_select = st.sidebar.multiselect('Выберите модели машинного обучения:', model
s_list)

data_load_state = st.text('Загрузка данных...')
data = load_data()
data_X_train, data_X_test, data_y_train, data_y_test = preprocess_data(data)
data_load_state.text('Данные загружены!')
st.write('''В качестве [набора данных](https://archive.ics.uci.edu/ml/datasets/Bike
+Sharing+Dataset) мы будем использовать набор данных почасового количества велосипе
дов,
взятых напрокат с соответствующей погодой и сезонной информацией''')
with st.beta_expander("Информация о наборе данных"):
    st.write("""
        Системы проката велосипедов - это новое поколение традиционных прокатов вел
осипедов, в которых весь процесс от членства до аренды и возврата стал
        автоматическим. С помощью этих систем пользователь может легко взять напрок
ат велосипед из определенного места и вернуться обратно в другом месте.
        В настоящее время в мире существует около 500 программ обмена велосипедами,
 которые включают более 500 тысяч велосипедов. Сегодня к этим системам
```

```
        проявляется большой интерес из-
за их важной роли в вопросах дорожного движения, окружающей среды и здоровья.""")
    st.write("""
        Помимо интересных реальных приложений систем проката велосипедов, характери
стики данных, генерируемых этими системами, делают их привлекательными
        для исследования. В отличие от других транспортных услуг, таких как автобус
 или метро, в этих системах четко записывается продолжительность поездки,
        место отправления и прибытия. Эта функция превращает систему проката велоси
педов в виртуальную сенсорную сеть, которую можно использовать для
        определения мобильности в городе. Следовательно, ожидается, что большинство
 важных событий в городе можно будет обнаружить с помощью мониторинга
        этих данных.
    """)

with st.beta_expander("Информация об атрибутах"):
    st.write("""
        - instant : индекс записи
        - dteday: дата
        - season: Сезон (1: зима, 2: весна, 3: лето, 4 : осень)
        - mnth: месяц (от 1 до 12)
        - hour: час (от 0 до 23)
        - holiday: выходной или нет
        - weekday: день недели
        - workingday: если день не является ни выходным, ни праздничным - 1, в прот
ивном случае - 0.
        + weathersit:
            - 1: Ясно, Небольшая облачность, Небольшая облачность, Небольшая облачн
ость
            - 2: Туман + Облачно, Туман + Разбитые облака, Туман + Несколько облако
в, Туман
            - 3: слабый снег, легкий дождь + гроза + рассеянные облака, легкий дожд
ь + рассеянные облака
            - 4: сильный дождь + ледяные поддоны + гроза + туман, снег + туман
        - temp: нормализованная температура в градусах Цельсия
        - atemp: нормализованная температура ощущения в градусах Цельсия
        - hum: нормализованная влажность
        - windspeed: нормализованная скорость ветра
        - casula: количество случайных прохожих
        - cnt: общее количество взятых напрокат велосипедов
    """)
#Количество записей
data_len = data.shape[0]
st.write('Количество строк в наборе данных - {}'.format(data_len))

if st.checkbox('Показать первые 5 значений'):
    st.subheader('Первые 5 значений')
    st.write(data.head())

if st.checkbox('Показать все данные'):
    start_time = time.time()
    st.subheader('Данные')
```

```
    st.write(data)
    st.write("--- %s seconds ---" % (time.time() - start_time))

if st.checkbox('Показать типы колонок'):
    st.subheader('Типы колонок')
    st.write(data.dtypes)

if st.checkbox('Показать пропущенные значения'):
    st.subheader('Пропущенные значения')
    st.write(data.isnull().sum())
    with st.beta_expander("Вывод"):
        st.write("""
            Представленный набор данных не содержит пропусков.
        """)

if st.checkbox('Показать парные диаграммы'):
    st.subheader('Парные диаграммы')
    start_time = time.time()
    fig, ax = plt.subplots(figsize=(20,20))
    st.pyplot(sns.pairplot(data))
    st.write("--- %s seconds ---" % (time.time() - start_time))

if st.checkbox('Показать скрипичные диаграммы для числовых колонок'):
    st.subheader('Скрипичные диаграммы для числовых колонок')
    start_time = time.time()
    col_ch=['instant', 'season', 'mnth', 'hr', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual']
    for col in col_ch:
        fig, ax = plt.subplots(figsize=(5,5))
        sns.violinplot(x=data[col])
        st.pyplot(fig)
    st.write("--- %s seconds ---" % (time.time() - start_time))

scale_cols = ['season', 'hr', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'hum', 'windspeed',
       'casual']
new_scale_cols = []
for i in range(len(scale_cols)):
    col = scale_cols[i]
    new_col_name = col + '_scaled'
    new_scale_cols.append(new_col_name)

corr_cols_1 = scale_cols + ['cnt']

scale_cols_postfix = [x+'_scaled' for x in scale_cols]
corr_cols_2 = scale_cols_postfix + ['cnt']
data_1 = data

if st.checkbox('Проверим, что масштабирование не повлияло на распределение данных')
:
```

```
    with st.beta_expander("Подробнее"):
        st.write("""
        Для построения моделей будем использовать все признаки кроме признаков dted
ay и instant, потому что мы не рассматриваем наши данные как временной ряд.

        Категориальные признаки отсутствуют, их кодирования не требуется.

        Вспомогательные признаки для улучшения качества моделей в данном примере мы
 строить не будем.

        Выполним масштабирование данных.
        """)
    sc1 = MinMaxScaler()
    sc1_data_1 = sc1.fit_transform(data_1[scale_cols])
    for i in range(len(scale_cols)):
        data_1 = data_1.drop(scale_cols[i], 1)
        data_1[new_scale_cols[i]] = sc1_data_1[:,i]
    for i in range(len(scale_cols)):
        fig, ax = plt.subplots(1, 2, figsize=(8,3))
        ax[0].hist(data[scale_cols[i]], 50)
        ax[1].hist(data_1[new_scale_cols[i]], 50)
        ax[0].title.set_text(scale_cols[i])
        ax[1].title.set_text(new_scale_cols[i])
        st.pyplot(fig)

if st.checkbox('Показать корреляционные матрицы'):
    col_ch=['instant', 'dteday', 'season', 'mnth', 'hr', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual','cnt']
    fig, ax = plt.subplots(figsize=(15,15))
    sns.heatmap(data[col_ch].corr(), annot=True, fmt='.4f', cmap="YlGnBu")
    ax.set_title('Корреляционная матрица для всех колонок')
    st.pyplot(fig)

    with st.beta_expander("Выводы"):
        st.write("""
            Для построения моделей не будем использовать признаки dteday и instant,
 потому что мы не рассматриваем наши данные как временной ряд.

            На основе нашей корреляционной матрице, визуализированной с помощью теп
ловой карты, определим признаки которые коррелируют с нашим целевым признаком.

            На основе корреляционной матрицы можно сделать следующие выводы:

            - Целевой признак регрессии "общее количество взятых напрокат велосипед
ов" ("cnt") наиболее сильно коррелирует
            с количеством случайных прохожих ("casual") (0.71247), температурой ("t
emp") (0.4512), температурой ощущения ("atemp") (0.4470),
            часом ("hr") (0.4075). Поэтому эти признаки следует оставить в модели к
лассификации.
```

```
            - Признаки сезон ("season") и месяц ("mnth") имеют корреляцию, близкую к 1, поэтому оба признака не следуют включать в модели.
            Будем использовать признак "season", так как он лучше чем "mnth" коррелирует с целевым признаком регрессии.
            - Признаки температура ("temp") и температура ощущения ("atemp") имеют корреляцию, близкую к 1, поэтому оба признака не следуют включать в модели.
            Будем использовать признак "temp", так как он лучше чем "atemp" коррелирует с целевым признаком регрессии.

            На основании корреляционной матрицы можно сделать вывод о том, что данные позволяют построить модель машинного обучения.

            Отрицательный коэффициент корреляции показывает, что две переменные могут быть связаны таким образом, что при возрастании значений одной из них значения другой убывают.
        """)
    fig, ax = plt.subplots(figsize=(15,15))
    sns.heatmap(data[corr_cols_1].corr(), annot=True, fmt='.4f', cmap="YlGnBu")
    ax.set_title('Исходные данные (до масштабирования)')
    st.pyplot(fig)

    fig, ax = plt.subplots(figsize=(15,15))
    sns.heatmap(data_1[corr_cols_2].corr(), annot=True, fmt='.4f', cmap="YlGnBu")
    ax.set_title('Масштабированные данные')
    st.pyplot(fig)
    with st.beta_expander("Выводы"):
        st.write("Корреляционные матрицы для исходных и масштабированных данных совпадают.")

st.header('Выбор метрик для последующей оценки качества моделей')
with st.beta_expander("Подробнее"):
        st.write("""
            ### В качестве метрик для решения задачи регрессии будем использовать:
        #### [Mean absolute error](https://en.wikipedia.org/wiki/Mean_absolute_error) - средняя абсолютная ошибка
        """)
        st.latex(r'''MAE(y,\hat{y}) = \frac{1}{N} \cdot \sum\limits_{i=1}^N \lvert y_i - \hat{y_i} \rvert ''')
        st.write("""
        где:
        - $y$ - истинное значение целевого признака
        - $\hat{y}$ - предсказанное значение целевого признака
        - $N$ - размер тестовой выборки

        Чем ближе значение к нулю, тем лучше качество регрессии.

        Основная проблема метрики состоит в том, что она не нормирована.

        Вычисляется с помощью функции [mean_absolute_error.](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html#sklearn.metrics.mean_absolute_error)
```

```
        #### [Mean squared error](https://en.wikipedia.org/wiki/Mean_squared_error)
 - средняя квадратичная ошибка
        """)

        st.latex(r'''MSE(y,\hat{y}) = \frac{1}{N} \cdot \sum\limits_{i=1}^N ( y_i -
 \hat{y_i} )^2''')
        st.write("""
        где:
        - $y$ - истинное значение целевого признака
        - $\hat{y}$ - предсказанное значение целевого признака
        - $N$ - размер тестовой выборки

        Вычисляется с помощью функции [mean_squared_error.](https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html#sklearn.
metrics.mean_squared_error)

        #### [Метрика $R^2$ или коэффициент детерминации](https://ru.wikipedia.org/
wiki/%D0%9A%D0%BE%D1%8D%D1%84%D1%84%D0%B8%D1%86%D0%B8%D0%B5%D0%BD%D1%82_%D0%B4%D0%B
5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B0%D1%86%D0%B8%D0%B8)
        """)
        st.latex(r'''R^2(y,\hat{y}) = 1 - \frac{\sum\limits_{i=1}^N ( y_i - \hat{y_
i} )^2}{\sum\limits_{i=1}^N ( y_i - \overline{y_i} )^2}''')
        st.write("""
        где:
        - $y$ - истинное значение целевого признака
        - $\hat{y}$ - предсказанное значение целевого признака
        - $N$ - размер тестовой выборки
        - $\\overline{y_i}$ - вычисляется по формуле:""")
        st.latex(r'''\overline{y_i} = \frac{1}{N} \cdot \sum\limits_{i=1}^N y_i''')
        st.write("""
        Вычисляется с помощью функции [r2_score.](https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2
_score)
                """)

regr_models = {'LR': LinearRegression(),
               'KNN_5':KNeighborsRegressor(n_neighbors=5),
               'SVR':SVR(),
               'Tree':DecisionTreeRegressor(),
               'RF':RandomForestRegressor(),
               'GB':GradientBoostingRegressor(),
               'AutoML':AutoML(mode="Explain")}
# Сохранение метрик
regrMetricLogger = MetricLogger()

def train_param(regr_gs, label, mae_list, mse_list, r2_list):
    regr_gs.fit(data_X_train, data_y_train)
    regr_gs_best_params_txt = str(regr_gs.best_params_)
    st.write("Лучшее значение параметров: "+regr_gs_best_params_txt)
```

```
    if (label == 'KNN_best_GSCV'):
            fig, ax = plt.subplots(figsize=(5,5))
            plt.plot(n_range, regr_gs.cv_results_['mean_test_score'])
            ax.set_title("Изменение качества на тестовой выборке в зависимости от К
-соседей")
            st.pyplot(fig)

    regr_models_grid = {label:regr_gs.best_estimator_}
    for model_name, model in regr_models_grid.items():
        mae, mse, r2 = regr_train_model(model_name, model, regrMetricLogger)
        current_models_list.append(model_name)
        models_select.append(model_name)
        mae_list.append(round(mae, 3))
        mse_list.append(round(mse, 3))
        r2_list.append(round(r2, 3))
    return mae_list, mse_list, r2_list


def regr_train_model(model_name, model, regrMetricLogger):
    model.fit(data_X_train, data_y_train)
    Y_pred = model.predict(data_X_test)

    mae = mean_absolute_error(data_y_test, Y_pred)
    mse = mean_squared_error(data_y_test, Y_pred)
    r2 = r2_score(data_y_test, Y_pred)

    regrMetricLogger.add('MAE', model_name, mae)
    regrMetricLogger.add('MSE', model_name, mse)
    regrMetricLogger.add('R2', model_name, r2)

    print('{} \t MAE={}, MSE={}, R2={}'.format(
        model_name, round(mae, 3), round(mse, 3), round(r2, 3)))
    return mae, mse, r2

if len(models_select)>0:
    st.header('Оценка качества моделей')
    current_models_list = []
    mae_list = []
    mse_list = []
    r2_list = []

    for model_name in models_select:
        model = regr_models[model_name]
        mae, mse, r2 = regr_train_model(model_name, model, regrMetricLogger)
        current_models_list.append(model_name)
        mae_list.append(round(mae, 3))
        mse_list.append(round(mse, 3))
        r2_list.append(round(r2, 3))


    if "KNN_5" in current_models_list:
```

```
        if st.button('Подбор гиперпараметров для KNN'):
            n_range = np.array(range(1,50,1))
            tuned_parameters = [{'n_neighbors': n_range}]
            regr_gs = GridSearchCV(KNeighborsRegressor(), tuned_parameters, cv=5, s
coring='neg_mean_squared_error', n_jobs = -1, verbose = 2)
            label = 'KNN_best_GSCV'
            mae_list, mse_list, r2_list = train_param(regr_gs, label, mae_list, mse
_list, r2_list)


    if "RF" in current_models_list:
        if st.button('Подбор гиперпараметров для RF'):
            random_grid = {'bootstrap': [True, False],
               'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
               'max_features': ['auto', 'sqrt'],
               'min_samples_leaf': [1, 2, 4],
               'min_samples_split': [2, 5, 10],
               'n_estimators': [130, 180, 230]}
            clf = RandomForestRegressor()
            regr_gs = RandomizedSearchCV(clf, random_grid, cv=5, n_jobs = -
1, verbose = 2)
            label = 'RF_best_RSCV'
            mae_list, mse_list, r2_list = train_param(regr_gs, label, mae_list, mse
_list, r2_list)

    if "Tree" in current_models_list:
        if st.button('Подбор гиперпараметров для Tree'):
            random_grid = {
               'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
               'max_features': ['auto', 'sqrt'],
               'min_samples_leaf': [1, 2, 4],
               'min_samples_split': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110,
None]}
            clf = DecisionTreeRegressor()
            regr_gs = RandomizedSearchCV(clf, random_grid, cv=5, n_jobs = -
1, verbose = 2)
            label = 'Tree_best_RSCV'
            mae_list, mse_list, r2_list = train_param(regr_gs, label, mae_list, mse
_list, r2_list)
    if "LR" in current_models_list:
        if st.button('Подбор гиперпараметров для LR'):
            random_grid = {
               'fit_intercept': [True, False],
               'normalize': [True, False],
               'copy_X': [True, False]}
            clf = LinearRegression()
            regr_gs = GridSearchCV(clf, random_grid, cv=5, n_jobs = -
1, verbose = 2)
            label = 'LR_best_GSCV'
            mae_list, mse_list, r2_list = train_param(regr_gs, label, mae_list, mse
_list, r2_list)
```

```
    if "SVR" in current_models_list:
        if st.button('Подбор гиперпараметров для SVR'):
            random_grid = {
                'kernel' : ('linear', 'poly', 'rbf', 'sigmoid'),
                'C' : [1,5,10],
                'degree' : [3,8],
                'coef0' : [0.01,10,0.5],
                'gamma' : ('auto','scale')}
            clf = SVR()
            regr_gs = RandomizedSearchCV(clf, random_grid, cv=5, n_jobs = -
1, verbose = 2)
            label = 'SVR_best_RSCV'
            mae_list, mse_list, r2_list = train_param(regr_gs, label, mae_list, mse
_list, r2_list)

    if "GB" in current_models_list:
        if st.button('Подбор гиперпараметров для GB'):
            random_grid = {
                'n_estimators':[100,500],
                'learning_rate': [0.1,0.05,0.02],
                'max_depth':[4],
                'min_samples_leaf':[3],
                'max_features':[1.0] }
            clf = GradientBoostingRegressor()
            regr_gs = RandomizedSearchCV(clf, random_grid, cv=5, n_jobs = -
1, verbose = 2)
            label = 'GB_best_RSCV'
            mae_list, mse_list, r2_list = train_param(regr_gs, label, mae_list, mse
_list, r2_list)


    if len(current_models_list)>0:
        st.subheader('Таблица метрик')
        temp_d = {'mean_absolute_error': mae_list, 'mean_squared_error': mse_list,
'r2_score': r2_list}
        temp_df = pd.DataFrame(data=temp_d, index=current_models_list)
        st.table(temp_df)
        regrMetricLogger.plot('Метрика: ' + 'MAE', 'MAE', ascending=False, figsize=
(7, 6))
        regrMetricLogger.plot('Метрика: ' + 'MSE', 'MSE', ascending=False, figsize=
(7, 6))
        regrMetricLogger.plot('Метрика: ' + 'R2', 'R2', ascending=True, figsize=(7,
 6))

    if len(models_select)==7:
        with st.beta_expander("Выводы"):
            st.write("""
            - Модель машинного обучения RandomForestRegressor (случайный лес) показ
ала наилучшие результаты среди остальных сравниваемых моделей по всем выбранным нам
и метрикам.
```

Алгоритм "случайного леса" активно применяется на практике и часто оказывается лучшим при сравнении метрик качества, как и в этом случае.

- Модель DecisionTreeRegressor (решающее дерево) показала отличные результаты, уступающие только результатам "случайного леса" по метрике MAE и результатам "случайного леса"
и "градиентного бустинга" по метрикам MSE и R2
- Алгоритм GradientBoostingRegressor (градиентный бустинг) делит 2 и 3 место с моделью "решающее дерево" попеременно в разных метриках
- Метод KNeighborsRegressor(n=5) (K-ближайших соседей) показывает неплохие результаты, но заметно отстающие от более лучших моделей по всем метрикам
- Модель LinearRegression (линейная регрессия) показывает нелучшие результаты, уступающие методу ближайших соседей
- Машина опорных векторов (SVR) показывает наихудшие показатели метрик среди остальных моделей, однако, плохие результаты были получены без подбора гиперпараметров
(baseline решение), а для SVR необходимо подбирать гиперпараметры, что является ресурсоемкой задачей, так же метод чувствителен к обучающей выборке (в том числе к выбросам в данных).

После подбора гиперпараметров метод показывает хорошие результаты.

Все вышеперечисленные модели были построены в рамках базового решения (baseline) без подбора гиперпараметров. Производилось обучение моделей на основе обучающей выборки
и оценка качества моделей на основе тестовой выборки. Для выбранных моделей возможен подбор гиперпараметров для лучших результатов моделей.

К выбранному набору данных было применено автоматическое машинное обучение с помощью библиотеки mljar AutoML.

Для наглядности и быстроты происходящего был выбран Explain Mode, который позволяет получить результаты за считанные минуты.

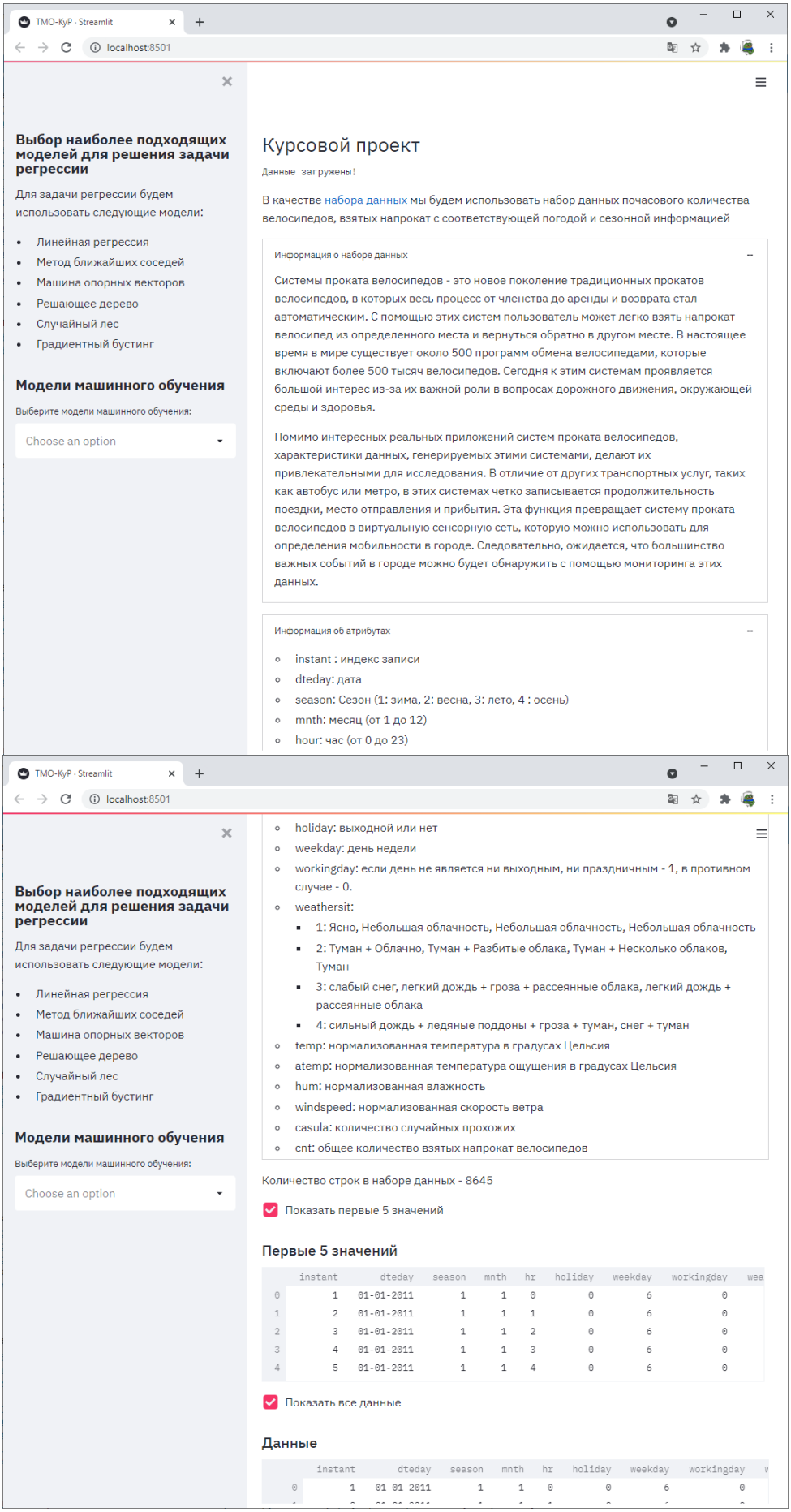
Для алгоритмов используются гиперпараметры по умолчанию. В режиме Explain используются следующие алгоритмы машинного обучения:

Baseline, Linear Model, Decision Tree, Random Forest, Xgboost, Neural Network.

Однако, даже с помощью этого режима были достигнуты наилучшие результаты среди всех выбранных нами моделей baseline решения.

""")

2.3. Экранные формы с примерами выполнения программы



TMO-KyP · Streamlit

localhost:8501

Выбор наиболее подходящих моделей для решения задачи регрессии

Для задачи регрессии будем использовать следующие модели:

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

Модели машинного обучения

Выберите модели машинного обучения:

Choose an option

1	2	01-01-2011	1	1	1	0	6	0
2	3	01-01-2011	1	1	2	0	6	0
3	4	01-01-2011	1	1	3	0	6	0
4	5	01-01-2011	1	1	4	0	6	0
5	6	01-01-2011	1	1	5	0	6	0
6	7	01-01-2011	1	1	6	0	6	0
7	8	01-01-2011	1	1	7	0	6	0
8	9	01-01-2011	1	1	8	0	6	0
9	10	01-01-2011	1	1	9	0	6	0
10	11	01-01-2011	1	1	10	0	6	0

--- 1.7138609886169434 seconds ---

☑ Показать типы колонок

Типы колонок

	0
instant	int64
dteday	object
season	int64
mnth	int64
hr	int64
holiday	int64
weekday	int64
workingday	int64
weathersit	int64
temp	float64
atemp	float64

☑ Показать пропущенные значения

Пропущенные значения

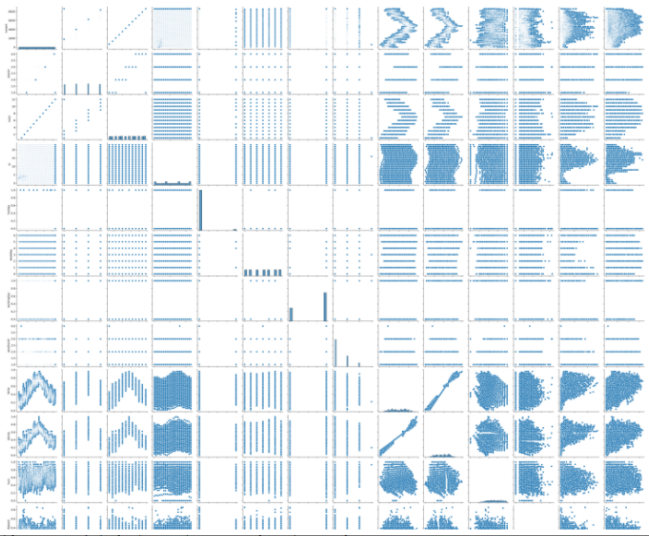
	0
instant	0
dteday	0
season	0
mnth	0
hr	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0

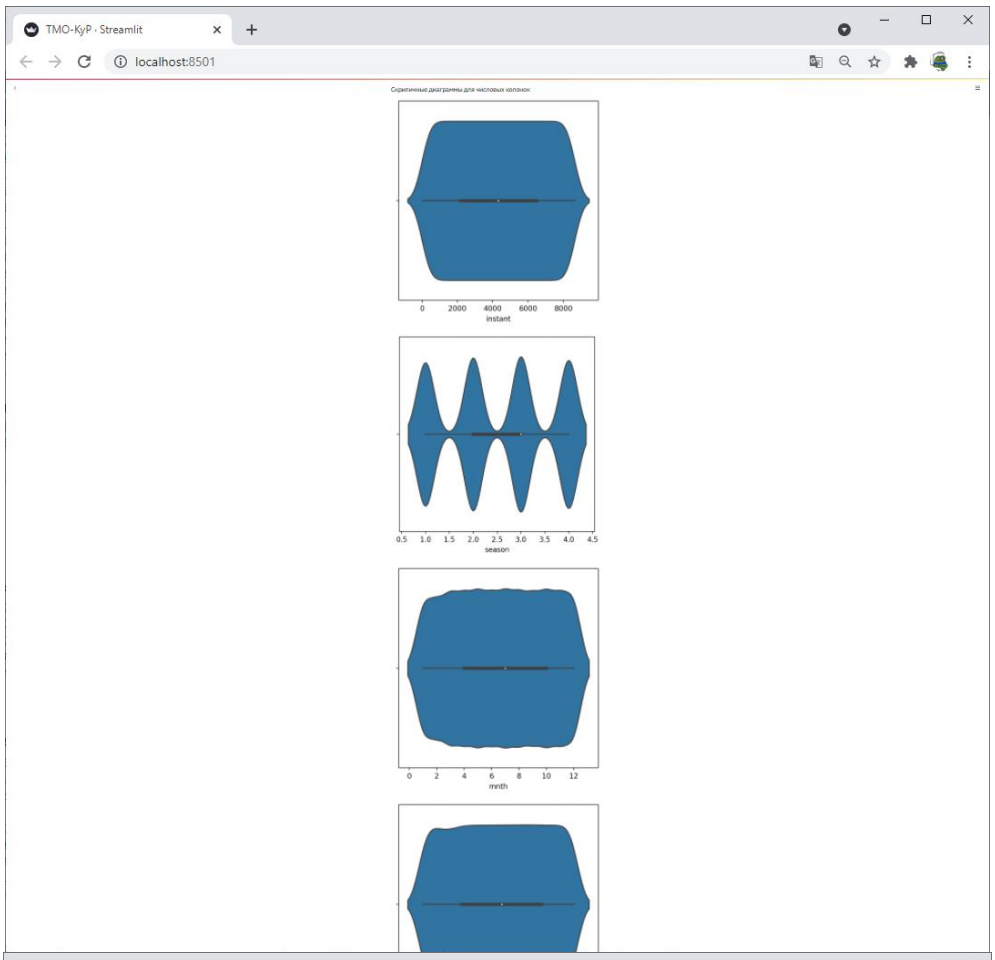
Вывод

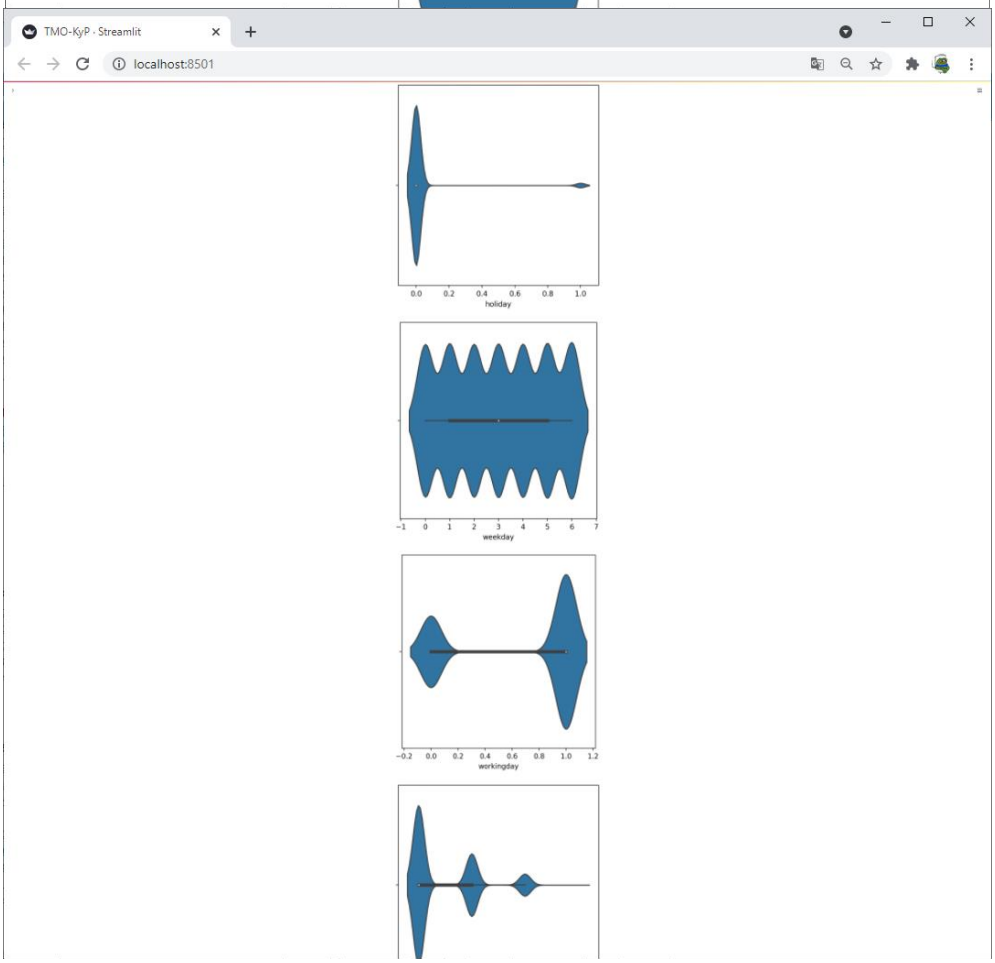
Представленный набор данных не содержит пропусков.

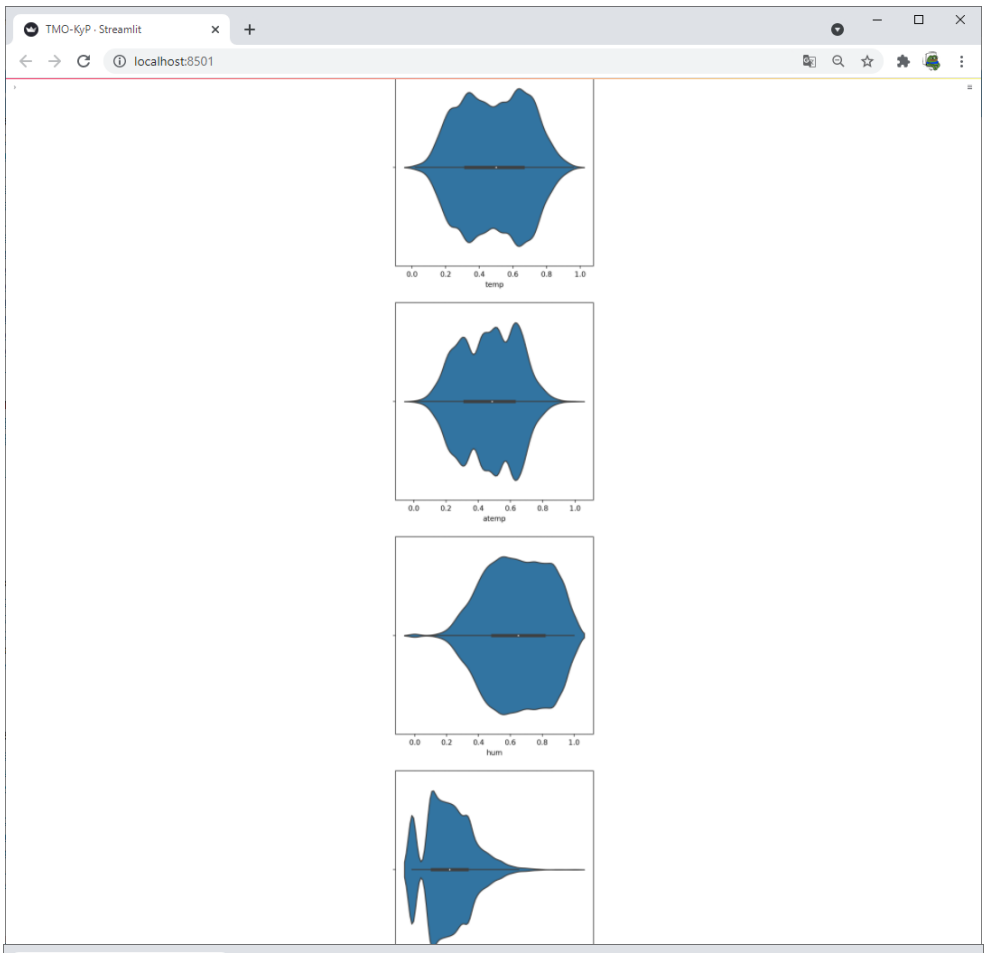
☑ Показать парные диаграммы

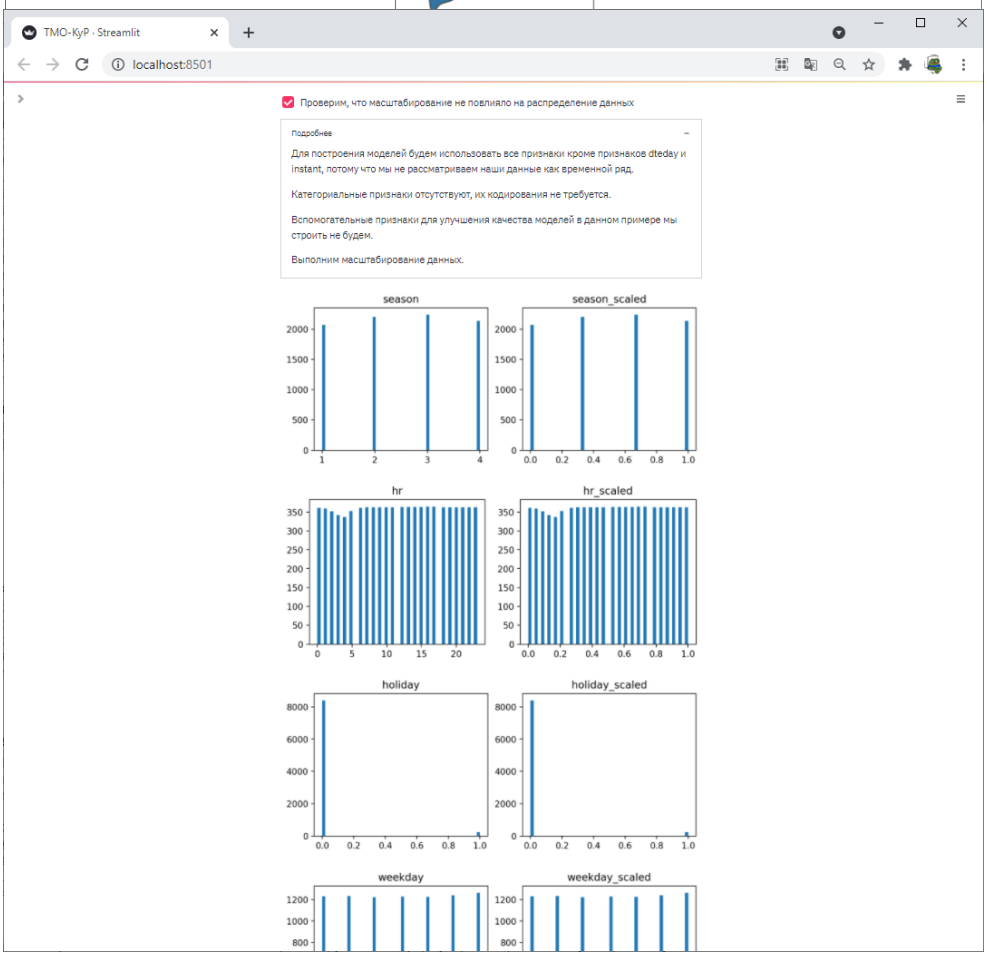
Парные диаграммы

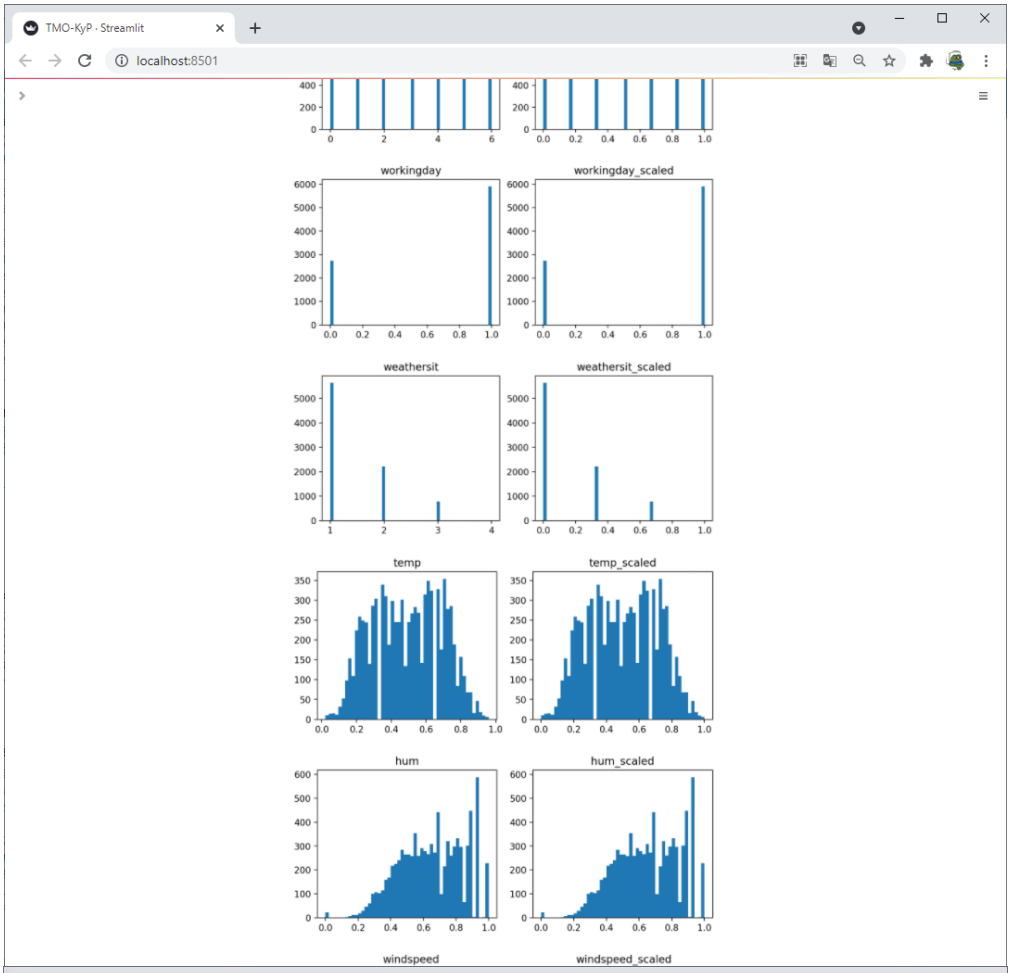


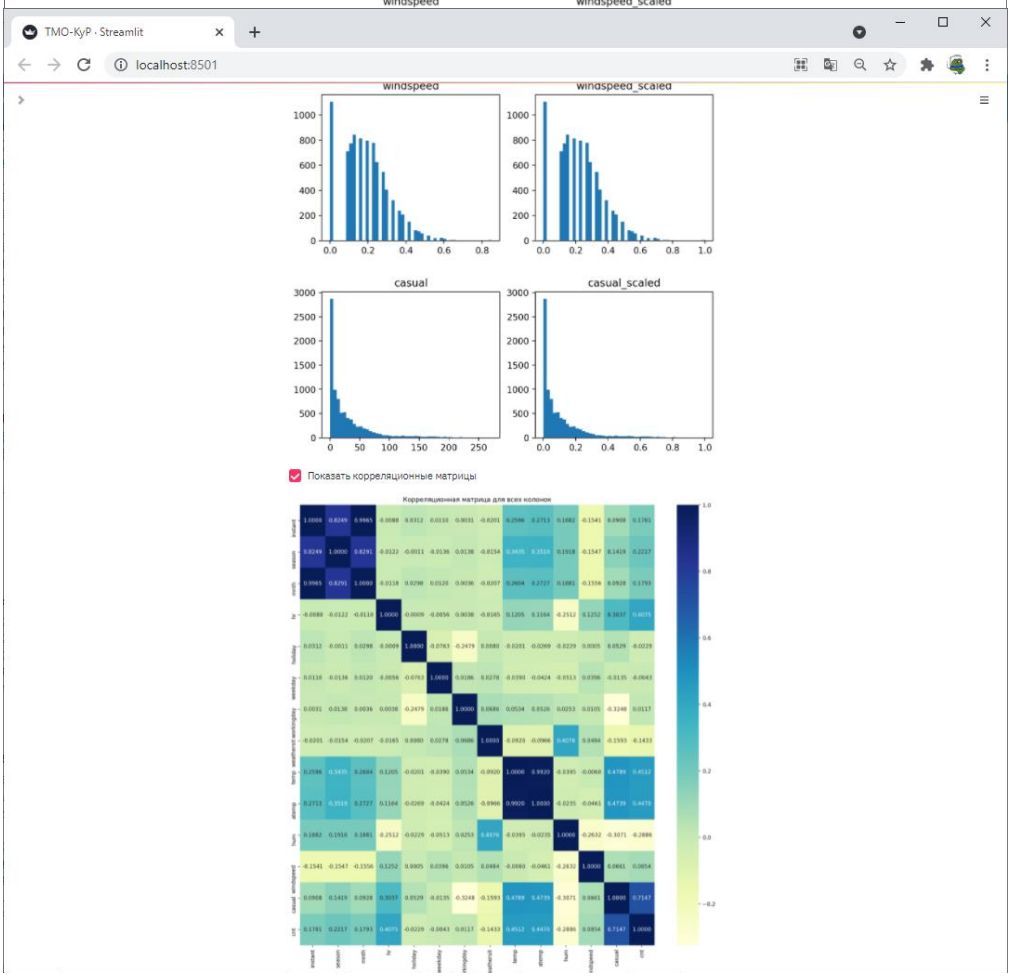

TMO-KyP · Streamlit
localhost:8501
Скрипичные диаграммы для числовых колонок
0 2000 4000 6000 8000
instant
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5
season
0 2 4 6 8 10 12
mnth


TMO-KyP · Streamlit
localhost:8501
0.0 0.2 0.4 0.6 0.8 1.0
holiday
−1 0 1 2 3 4 5 6 7
weekday
−0.2 0.0 0.2 0.4 0.6 0.8 1.0 1.2
workingday


ТМО-КуР · Streamlit
localhost:8501
temp
atemp
hum


ТМО-КуР · Streamlit
localhost:8501
Проверим, что масштабирование не повлияло на распределение данных
Подробнее
Для построения моделей будем использовать все признаки кроме признаков dteday и instant, потому что мы не рассматриваем наши данные как временной ряд.
Категориальные признаки отсутствуют, их кодирования не требуется.
Вспомогательные признаки для улучшения качества моделей в данном примере мы строить не будем.
Выполним масштабирование данных.
season
season_scaled
hr
hr_scaled
holiday
holiday_scaled
weekday
weekday_scaled


TMO-KyP · Streamlit
localhost:8501
workingday
workingday_scaled
weathersit
weathersit_scaled
temp
temp_scaled
hum
hum_scaled
windspeed
windspeed_scaled


TMO-KyP · Streamlit
localhost:8501
windspeed
windspeed_scaled
casual
casual_scaled
Показать корреляционные матрицы
Корреляционная матрица для всех колонок

TMO-KyP · Streamlit

localhost:8501

Выводы

Для построения моделей не будем использовать признаки dteday и instant, потому что мы не рассматриваем наши данные как временной ряд.

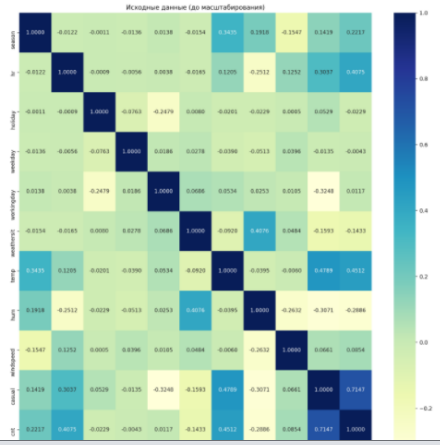
На основе нашей корреляционной матрице, визуализированной с помощью тепловой карты, определим признаки которые коррелируют с нашим целевым признаком.

На основе корреляционной матрицы можно сделать следующие выводы:

- Целевой признак регрессии "общее количество взятых напрокат велосипедов" ("cnt") наиболее сильно коррелирует с количеством случайных прохожих ("casual") (0.71247), температурой ("temp") (0.4512), температурой ощущения ("atemp") (0.4470), часом ("hr") (0.4075). Поэтому эти признаки следует оставить в модели классификации.
- Признаки сезон ("season") и месяц ("mnth") имеют корреляцию, близкую к 1, поэтому оба признака не следуют включать в модели. Будем использовать признак "season", так как он лучше чем "mnth" коррелирует с целевым признаком регрессии.
- Признаки температура ("temp") и температура ощущения ("atemp") имеют корреляцию, близкую к 1, поэтому оба признака не следуют включать в модели. Будем использовать признак "temp", так как он лучше чем "atemp" коррелирует с целевым признаком регрессии.

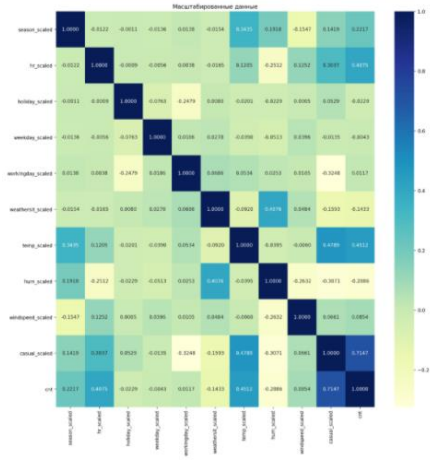
На основании корреляционной матрицы можно сделать вывод о том, что данные позволяют построить модель машинного обучения.

Отрицательный коэффициент корреляции показывает, что две переменные могут быть связаны таким образом, что при возрастании значений одной из них значения другой убывают.



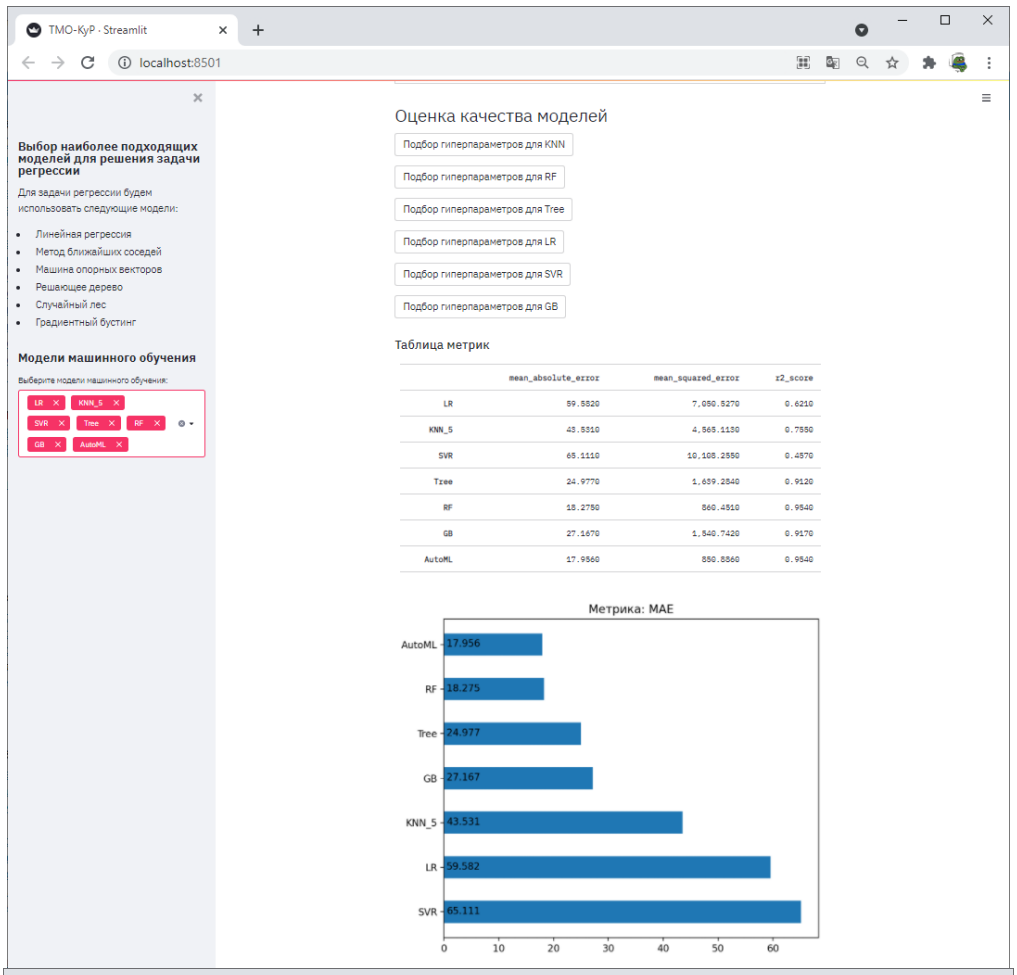
TMO-KyP · Streamlit

localhost:8501

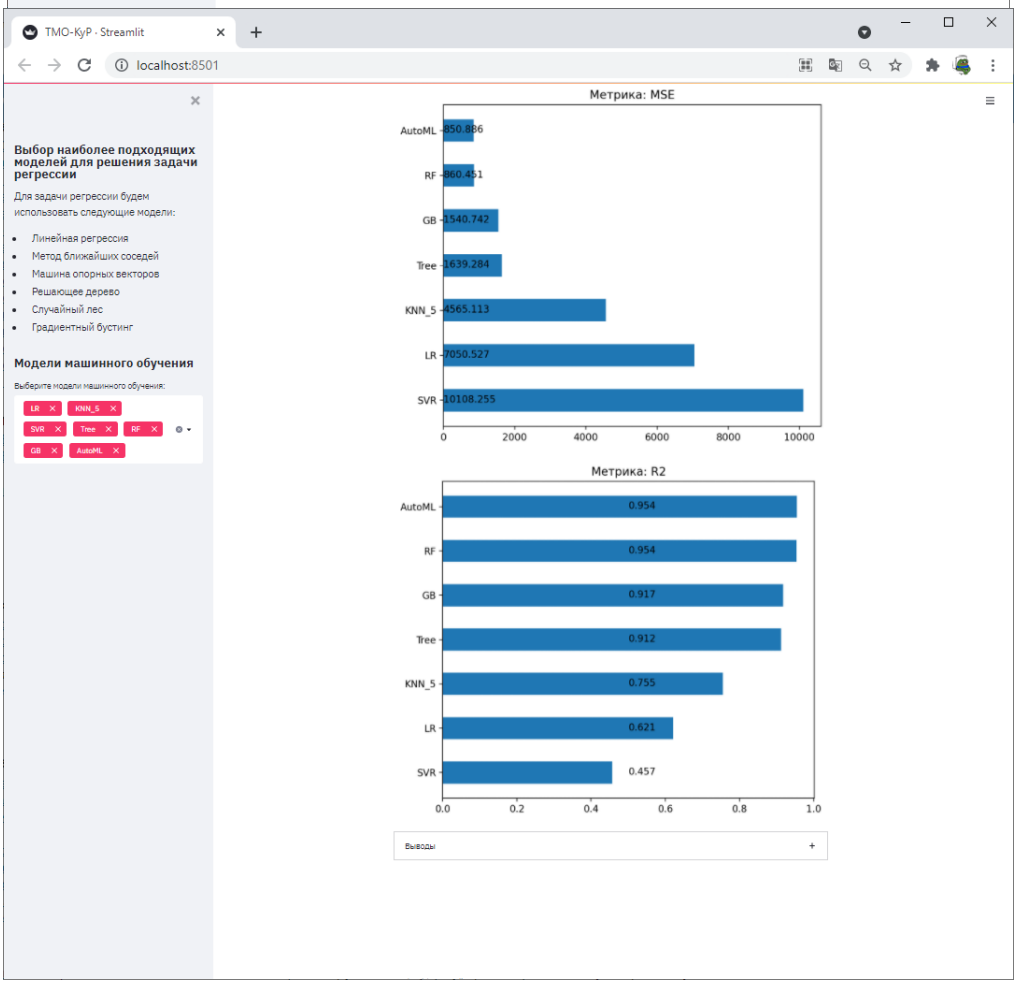


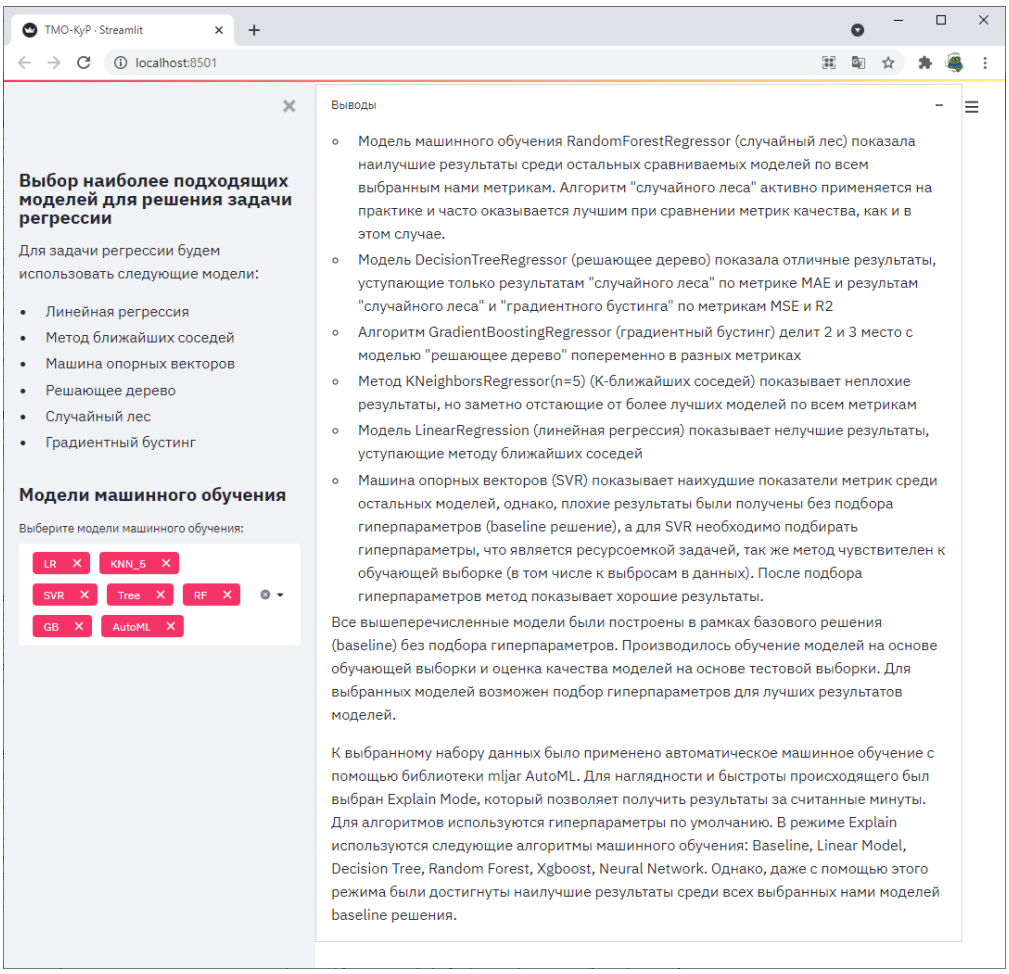
Выводы

Корреляционные матрицы для исходных и масштабированных данных совпадают.



	mean_absolute_error	mean_squared_error	r2_score
LR	59.5820	7,050.5270	0.6210
KNN_5	43.5310	4,565.1130	0.7550
SVR	65.1110	10,108.2550	0.4570
Tree	24.9770	1,639.2840	0.9120
RF	18.2750	860.4510	0.9540
GB	27.1670	1,540.7420	0.9170
AutoML	17.9560	850.8860	0.9540





3. Заключение

В результате курсового проекта была решена комплексной задача машинного обучения. Разработан макет веб-приложения, предназначенного для анализа данных

4. Список использованных источников информации

1) Ю.Е. Гапанюк Лекции по курсу «Технологии машинного обучения», 2020-2021 учебный год.

2) Ю.Е. Гапанюк Репозиторий курса "Технологии машинного обучения", бакалавриат, 6 семестр

3) Орельен Жерон, Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow, 2018. – 688 с.