**Московский государственный технический   
университет им. Н. Э. Баумана**

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №7

Выполнил: Проверил:  
Мажитов В. Гапанюк Ю.Е.  
группа ИУ5-62Б

Дата: 24.05.25 Дата:

Подпись: Подпись:

Москва, 2025 г.

**Цель лабораторной работы:** изучение возможностей демонстрации моделей машинного обучения с помощью веб-приложений.

**Задание:**

Разработайте макет веб-приложения, предназначенного для анализа данных.

Вариант 1. Макет должен быть реализован для одной модели машинного обучения. Макет должен позволять:

* задавать гиперпараметры алгоритма,
* производить обучение,
* осуществлять просмотр результатов обучения, в том числе в виде графиков.

**Ход выполнения:**

import streamlit as st

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.ensemble import RandomForestRegressor

from sklearn.pipeline import Pipeline

from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error

import joblib # Для сохранения/загрузки предобработчика

st.set\_page\_config(page\_title="Прогнозирование спроса на велосипеды", layout="wide")

@st.cache\_data

def load\_data():

"""Загружает и минимально подготавливает данные."""

try:

df = pd.read\_csv('SeoulBikeData.csv', encoding='ISO-8859-1')

except FileNotFoundError:

st.error("Файл SeoulBikeData.csv не найден. Убедитесь, что он находится в той же директории, что и app.py.")

st.stop()

df['Date'] = pd.to\_datetime(df['Date'], format='%d/%m/%Y')

df['Year'] = df['Date'].dt.year

df['Month'] = df['Date'].dt.month

df['Day'] = df['Date'].dt.day

df['DayOfWeek'] = df['Date'].dt.dayofweek

# 'Hour' уже есть

return df

@st.cache\_resource

def get\_preprocessor(\_df\_fit):

"""Создает и обучает ColumnTransformer для предобработки данных."""

X\_temp = \_df\_fit.drop(['Rented Bike Count', 'Date'], axis=1, errors='ignore')

categorical\_cols\_for\_ohe = ['Seasons', 'Holiday', 'Functioning Day', 'Month', 'DayOfWeek', 'Hour', 'Year']

numerical\_cols\_for\_scaling = ['Temperature(°C)', 'Humidity(%)', 'Wind speed (m/s)',

'Visibility (10m)', 'Dew point temperature(°C)',

'Solar Radiation (MJ/m2)', 'Rainfall(mm)', 'Snowfall (cm)', 'Day']

missing\_cols\_ohe = [col for col in categorical\_cols\_for\_ohe if col not in X\_temp.columns]

missing\_cols\_scale = [col for col in numerical\_cols\_for\_scaling if col not in X\_temp.columns]

if missing\_cols\_ohe:

st.warning(f"Отсутствуют колонки для OHE: {missing\_cols\_ohe}. Они будут проигнорированы.")

categorical\_cols\_for\_ohe = [col for col in categorical\_cols\_for\_ohe if col in X\_temp.columns]

if missing\_cols\_scale:

st.warning(f"Отсутствуют колонки для масштабирования: {missing\_cols\_scale}. Они будут проигнорированы.")

numerical\_cols\_for\_scaling = [col for col in numerical\_cols\_for\_scaling if col in X\_temp.columns]

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), numerical\_cols\_for\_scaling),

('cat', OneHotEncoder(handle\_unknown='ignore', drop='first'), categorical\_cols\_for\_ohe)

],

remainder='passthrough'

)

preprocessor.fit(X\_temp)

return preprocessor

def train\_model(X\_train, y\_train, preprocessor, n\_estimators\_val, max\_depth\_val, min\_samples\_split\_val):

"""Обучает модель RandomForestRegressor с заданными гиперпараметрами."""

model = RandomForestRegressor(

n\_estimators=n\_estimators\_val,

max\_depth=max\_depth\_val,

min\_samples\_split=min\_samples\_split\_val,

random\_state=42,

n\_jobs=-1

)

pipeline = Pipeline(steps=[('preprocessor', preprocessor),

('regressor', model)])

pipeline.fit(X\_train, y\_train)

return pipeline

df\_source = load\_data()

X = df\_source.drop(['Rented Bike Count', 'Date'], axis=1, errors='ignore')

y = df\_source['Rented Bike Count']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

preprocessor\_fitted = get\_preprocessor(X\_train.copy())

st.title("Демонстрация модели: Прогноз спроса на велосипеды")

st.markdown("""

Это приложение демонстрирует работу модели `RandomForestRegressor` для предсказания количества арендованных велосипедов.

Вы можете изменять некоторые гиперпараметры модели и видеть, как это влияет на ее качество на тестовой выборке.

""")

st.sidebar.header("Гиперпараметры RandomForestRegressor")

n\_estimators\_user = st.sidebar.slider(

"Количество деревьев (n\_estimators):",

min\_value=10,

max\_value=300,

value=100,

step=10

)

max\_depth\_user = st.sidebar.select\_slider(

"Максимальная глубина дерева (max\_depth):",

options=[5, 10, 15, 20, 25, 30, None],

value=20

)

min\_samples\_split\_user = st.sidebar.slider(

"Минимальное количество образцов для разделения узла (min\_samples\_split):",

min\_value=2,

max\_value=20,

value=2,

step=1

)

if st.sidebar.button("Обучить модель и оценить"):

with st.spinner("Обучение модели... Пожалуйста, подождите."):

model\_pipeline = train\_model(X\_train, y\_train, preprocessor\_fitted,

n\_estimators\_user, max\_depth\_user, min\_samples\_split\_user)

st.success("Модель успешно обучена!")

y\_pred = model\_pipeline.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

st.subheader("Метрики качества модели на тестовой выборке:")

col1, col2, col3 = st.columns(3)

col1.metric("R² (Коэфф. детерминации)", f"{r2:.4f}")

col2.metric("MAE (Сред. абс. ошибка)", f"{mae:.2f} велосипедов")

col3.metric("RMSE (Корень из ср.кв. ошибки)", f"{rmse:.2f} велосипедов")

st.subheader("Сравнение предсказанных и реальных значений (первые 200 точек)")

results\_df = pd.DataFrame({'Actual': y\_test, 'Predicted': y\_pred}).reset\_index(drop=True)

sample\_size = min(200, len(results\_df))

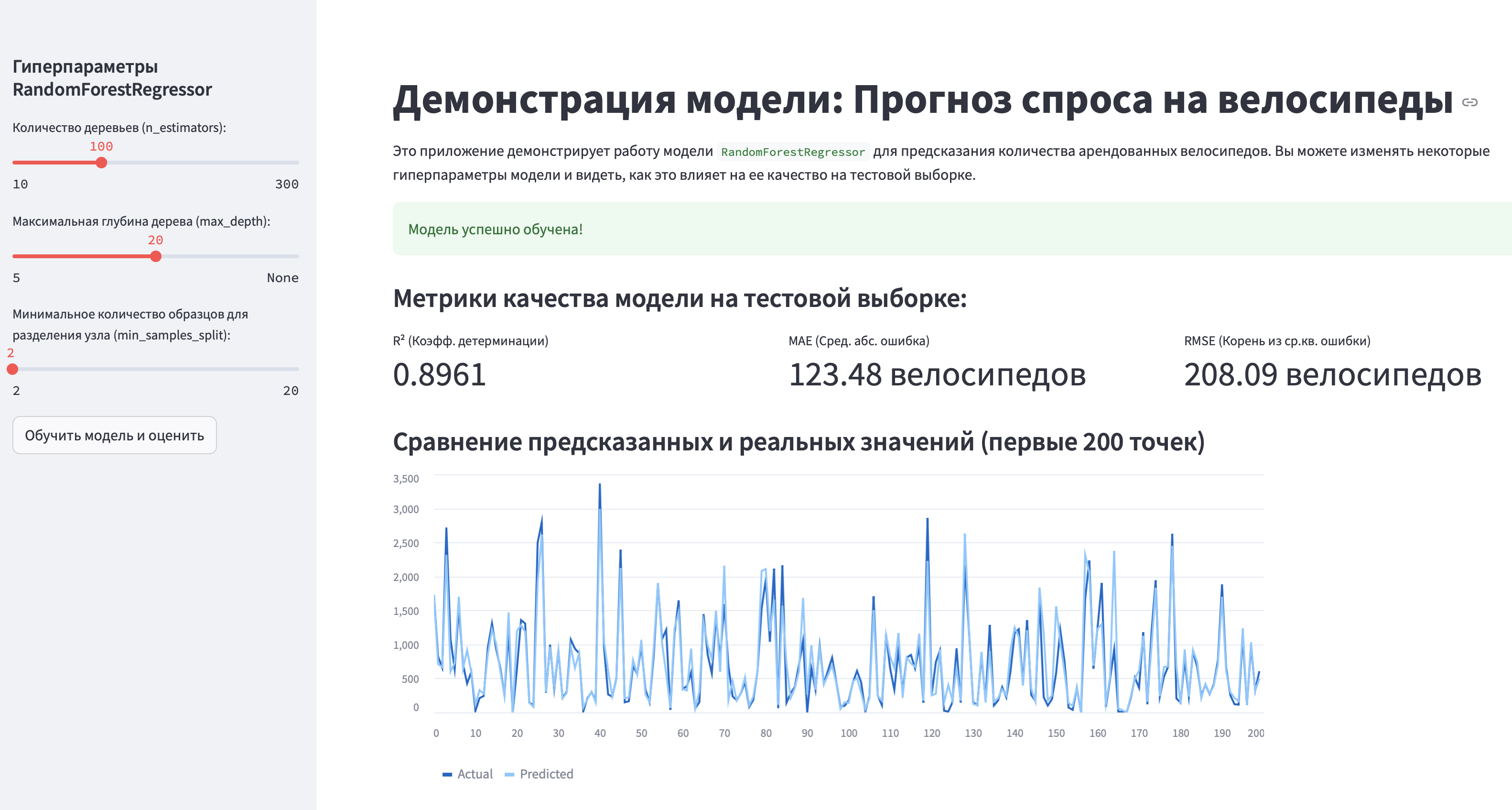
fig\_col, \_ = st.columns([3,1])

with fig\_col:

st.line\_chart(results\_df.head(sample\_size))

else:

st.info("Настройте гиперпараметры в боковой панели и нажмите 'Обучить модель и оценить'.")

****