Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №7

Выполнил:
Мажитов В.
группа ИУ5-62Б

Проверил:
Гапанюк Ю.Е.

Дата: 24.05.25

Подпись:

Дата:

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'.")
```