Министерство науки и высшего образования РФ
федеральное государственное автономное
образовательное учреждение высшего образования
«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Прикладная математика и фундаментальная информатика»

Расчетно-графическая работа

по дисциплине **Машинное обучение и большие данные**

Тема: Разработка Web-приложения (дашборда)

для инференса моделей ML и анализа данных

Студента	Курпенова Куата Ибраимовича
Курс	2 Группа ФИТ-222
Направление	02.03.02 Фундаментальная информатика и информационные технологии
Руководитель	доцент
	Моисеева Наталья Александровна
Выполнил	28.10.2024
Проверил	28.10.2024

Омск 2024

Введение 3
Основная часть 5
Список использованных источников 10
Приложение А 11
Приложение Б 13

Введение

Тема машинного обучения (ML) является крайне актуальной и востребованной в современном мире. Она находит применение во многих областях, включая медицину, финансы, транспорт, маркетинг и другие. Развитие решений на основе ML позволяет автоматизировать процессы, улучшать прогнозирование, оптимизировать принятие решений и создавать инновационные продукты и услуги.

Тема машинного обучения (ML) является крайне актуальной и востребованной в современном мире. Она находит применение во многих областях, включая медицину, финансы, транспорт, маркетинг и другие. Развитие решений на основе ML позволяет автоматизировать процессы, улучшать прогнозирование, оптимизировать принятие решений и создавать инновационные продукты и услуги.

1. Scikit-learn (Sklearn): Это одна из наиболее популярных библиотек машинного обучения для языка Python. Она предоставляет широкий спектр алгоритмов машинного обучения, обработки данных, предобработки, валидации моделей и метрик для оценки результатов. Sklearn обладает простым и интуитивно понятным API, и его можно использовать для классификации, регрессии, кластеризации и других задач.
2. Matplotlib: Это библиотека визуализации данных для языка Python. Она позволяет создавать простые и сложные графики, включая гистограммы, диаграммы рассеяния, линейные и логарифмические графики и др. Matplotlib играет важную роль в анализе и визуализации результатов ML-моделей, представлении данных и понимании моделирования.
3. Pandas: Это библиотека для обработки и анализа данных в Python. Она предоставляет удобные структуры данных, такие как DataFrames, для эффективной работы с табличными данными. pandas позволяет проводить предобработку данных, выполнение операций фильтрации, агрегации и трансформации, а также интеграцию данных с моделями машинного обучения. Основываясь на полученных знаниях и использованных инструментах, мы разработали web-приложение (дашборд) с использованием библиотеки Streamlit. Главная цель приложения состоит в том, чтобы представить модели машинного обучения и результаты анализа данных с возможностью их визуализации в различных форматах. Это позволяет пользователям легко взаимодействовать с моделями и получать наглядные представления результатов анализа данных.
4. Streamlit: Это библиотека для создания интерактивных веб-приложений для машинного обучения и анализа данных на языке Python. Streamlit позволяет разработчикам быстро и просто создавать красивые дашборды, включая визуализацию данных, интерактивные элементы

управления и интеграцию моделей машинного обучения. Он может быть полезным инструментом для демонстрации результатов и предоставления доступа к моделям в режиме реального времени.

Основная часть

1. Описание выбранных моделей.
    a. Naive Bayes: простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими (наивными) предположениями о независимости. В зависимости от точной природы вероятностной модели, наивные байесовские классификаторы могут обучаться очень эффективно. Во многих практических приложениях для оценки параметров для наивных байесовых моделей используют метод максимального правдоподобия; другими словами, можно работать с наивной байесовской моделью, не веря в байесовскую вероятность и не используя байесовские методы.Несмотря на наивный вид и, несомненно, очень упрощенные условия, наивные байесовские классификаторы часто работают намного лучше нейронных сетей во многих сложных жизненных ситуациях.
    b. Бэггинг (Bagging): Бэггинг - это метод ансамблевого обучения, в котором создается несколько независимых моделей на основе подвыборки обучающих данных. Затем, для получения окончательного прогноза, результаты этих моделей усредняются (в случае регрессии) или используется голосование (в случае классификации). Бэггинг помогает уменьшить дисперсию модели и повысить её стабильность.
    c. Градиентный бустинг (Gradient Boosting): Градиентный бустинг - это метод ансамблевого обучения, который обучает последовательность слабых моделей (например, деревьев решений) в итеративном режиме. Каждая новая модель приближает остатки предыдущей модели. Затем результаты этих моделей суммируются для получения окончательного прогноза. Градиентный бустинг является мощным методом, позволяющим получить лучшую модель, чем только одиночная модель.
    d. Стекинг (Stacking): Стекинг - это метод ансамблевого обучения, который комбинирует результаты нескольких базовых моделей, обученных на одних и тех же данных. Базовые модели предоставляют свои предсказания, а затем метамодель используется для объединения этих предсказаний и получения окончательного прогноза.
    e. Глубокая полносвязная нейронная сеть: Глубокая полносвязная нейронная сеть - это модель машинного обучения, состоящая из нескольких слоев нейронов, где каждый нейрон из предыдущего слоя соединен с каждым нейроном в следующем слое. Эта модель может иметь множество слоев и обучается на больших объемах данных. Глубокая полносвязная нейронная сеть широко используется для выполнения сложных задач, таких как распознавание образов и обработка естественного языка.
2. Процесс сериализации моделей.
    a. Обученные модели были сериализованы с использованием библиотеки joblib.

3. Процесс разработки дашборда.

a. Web-страница 1: Информация о разработчике. На рисунке 1 приложения А показано как выглядит разработанная web-страничка. Ниже представлен код реализации данной web-страницы.

```python
import streamlit as st

st.set_page_config(page_title="Author")

st.title("ML model inferences demo")

full_name = "Kuat Kurpenov"
st.write(f"Full name: {full_name}")

study_group = "FIT-222"
st.write(f"Study group: {study_group}")
```

b. Web-страница 2: Информация о наборе данных. На рисунке 2 приложения А показано как выглядит разработанная web-страничка. Ниже представлена часть кода реализации данной web-странички.

```python
import pandas as pd
import streamlit as st

st.set_page_config(page_title="Data")

st.title("Information about data")

st.write(
    """
    The dataset is dedicated to monitoring car prices in Moldova.
    Using this dataset, you can solve the regression problem -
    predict car prices depending on the brand, mileage, and so on.
    """
)

st.subheader("Example of preprocessed data")
df = pd.read_csv("../data/preprocessed_moldova_cars_dataset.csv")
df.drop("Unnamed: 0", axis=1, inplace=True)
st.dataframe(df.head())

st.subheader("Description of columns")
st.markdown(
    """
    - `make` &mdash; brand of car
    - `model` &mdash; model of car
```

c. Web-страница 3: Визуализация зависимостей в наборе данных. На рисунках 3-7 приложения А показано как выглядит разработанная web-страничка. Ниже представлена часть кода реализации данной web-страницы.

```python
from collections import Counter

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import streamlit as st

st.set_page_config(page_title="Data visualization")

st.title("Data visualization")

df = pd.read_csv("../data/preprocessed_moldova_cars_dataset.csv")
df.drop("Unnamed: 0", axis=1, inplace=True)

fig, axs = plt.subplots(2, 2, figsize=(20, 16))

unique_makes = list(df["make"].unique())
counts = df["make"].value_counts() / len(df)
axs[0, 0].bar(unique_makes, counts)
axs[0, 0].set_title("Distribution of car brands")
axs[0, 0].set_xlabel("Car brand")
axs[0, 0].set_ylabel("Fraction")

sns.heatmap(df.corr(), ax=axs[0, 1])
axs[0, 1].set_title("Correlations heatmap")

prices_distributions = Counter(df["price"].values)
axs[1, 0].bar(list(prices_distributions.keys()), list(prices_distributions.values()))
axs[1, 0].set_title("Distribution of car prices")
axs[1, 0].set_xlabel("Price")
axs[1, 0].set_ylabel("Count")

st.pyplot(fig)
```

d. Web-страница 4: Предсказания моделей на введенных пользователем данных. На рисунках 9-10 приложения А показано как выглядит разработанная web-страничка. Ниже представлена часть кода реализации данной web-страницы.

```python
st.set_page_config(page_title="Models")

st.title("Machine learning models")

make = st.text_input("Make:")
model = st.text_input("Model:")
year = st.text_input("Year:")
style = st.text_input("Style:")
distance = st.text_input("Distance:")
engine_capacity = st.text_input("Engine capacity (cm3):")
fuel_type = st.text_input("Fuel type:")
transmission = st.text_input("Transmission:")

if st.button("Get price"):
    data = [
        float(make),
        float(model),
        float(year),
        float(style),
        float(distance),
        float(engine_capacity),
        float(fuel_type),
        float(transmission),
    ]

    predicted_prices = predict([data])

    st.success(f"\
    [+] Predicted prices (in euro):\n\
    [+] Linear regression: {int(predicted_prices[0])}\n\
    [+] Catboost: {int(predicted_prices[1])}\n\
    [+] Bagging: {int(predicted_prices[2])}\n\
    [+] Stacking: {int(predicted_prices[3])}\n\
    [+] Neural network: {int(predicted_prices[4])}")
```

f. GitHub-репозиторий и размещение исходных файлов. Все файлы размещены в открытом репозитории GitHub. Подробное описание проекта находится в файле README. Ссылка на репозиторий.

e. Веб-сервис, разработанный с использованием библиотеки streamlit был размещён на streamlit cloud и доступен по следующей ссылке.

Заключение

В рамках данной работы было создано web-приложение (dashboard) с использованием библиотеки Streamlit. Основная цель приложения заключалась в выводе результатов инференса моделей машинного обучения и их визуализации.

В рамках данной работы было создано web-приложение (dashboard) с использованием библиотеки Streamlit. Основная цель приложения заключалась в выводе результатов инференса моделей машинного обучения и их визуализации.

Выбранные модели были сериализованы и сохранены на жесткий диск, используя соответствующие инструменты (joblib).

С помощью библиотеки streamlit было реализовано web-приложение (dashboard), содержащее несколько страниц:

1. Страница с информацией о разработчике моделей машинного обучения.
2. Страница с информацией о выбранном наборе данных.
3. Страница с визуализациями зависимостей в наборе данных, используя библиотеки matplotlib и seaborn.
4. Страница, позволяющая получить предсказание соответствующей модели машинного обучения для данных вводимых пользователем.

Список использованных источников

1. user guide / [Электронный ресурс] // scikit-learn : [сайт]. — URL: https://scikit-learn.ru/user_guide/ (дата обращения: 20.10.2024).
2. API Documentation /[Электронный ресурс] // TensorFlow : [сайт]. — URL: https://www.tensorflow.org/api_docs (дата обращения: 20.10.2024).
3. Streamlit documentation / [Электронный ресурс] // Streamlit : [сайт]. — URL: https://docs.streamlit.io (дата обращения: 25.10.2024).

Приложение А

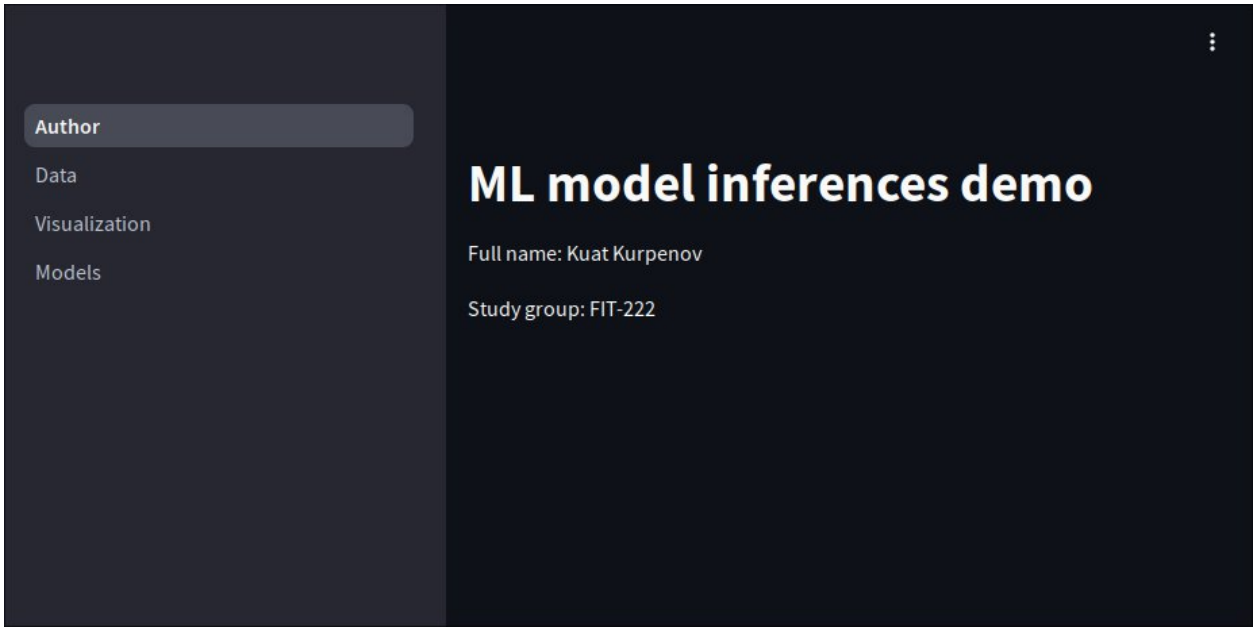

Рисунок 1 — Информация о разработчике моделей ML

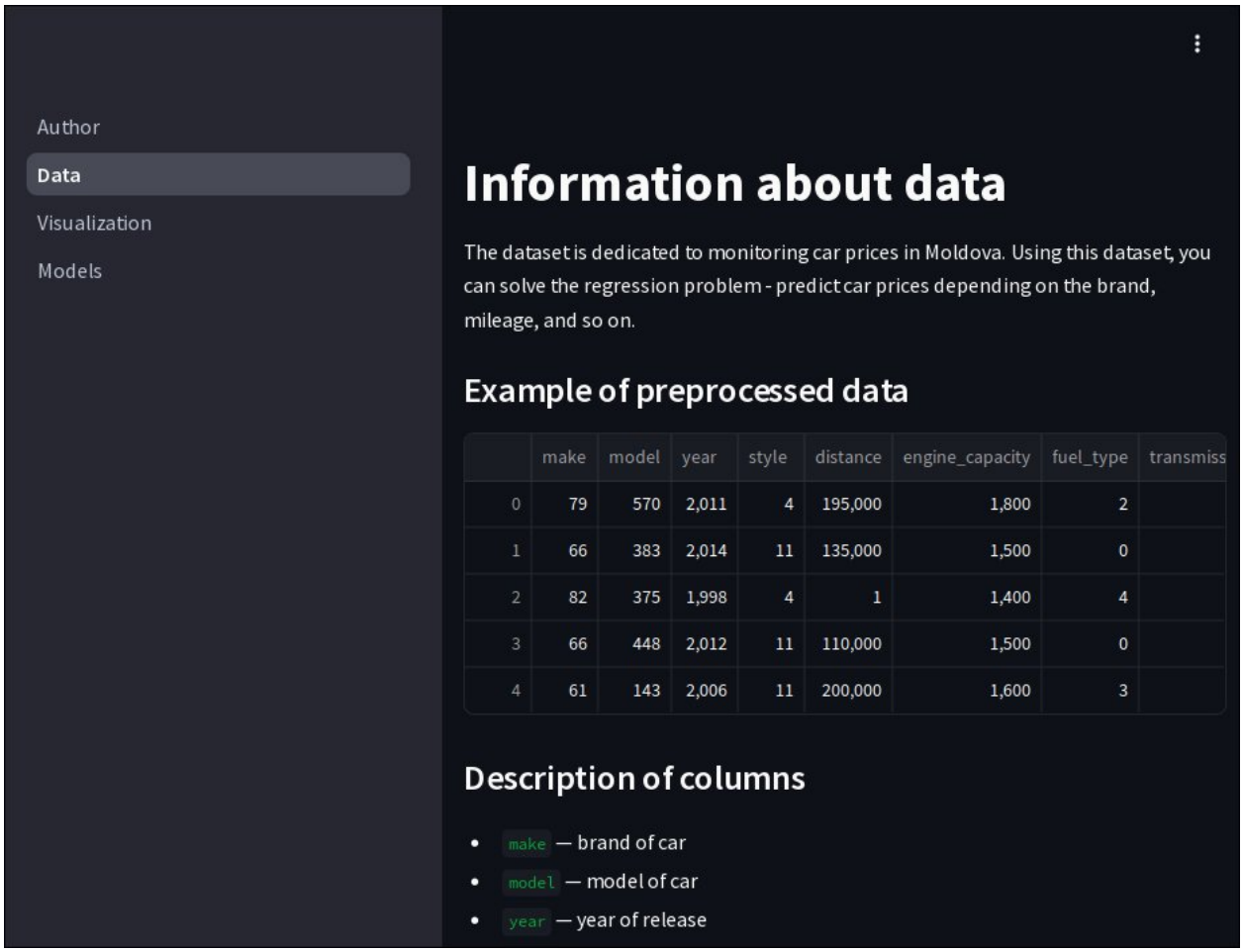

Рисунок 2 — Информация о наборе данных

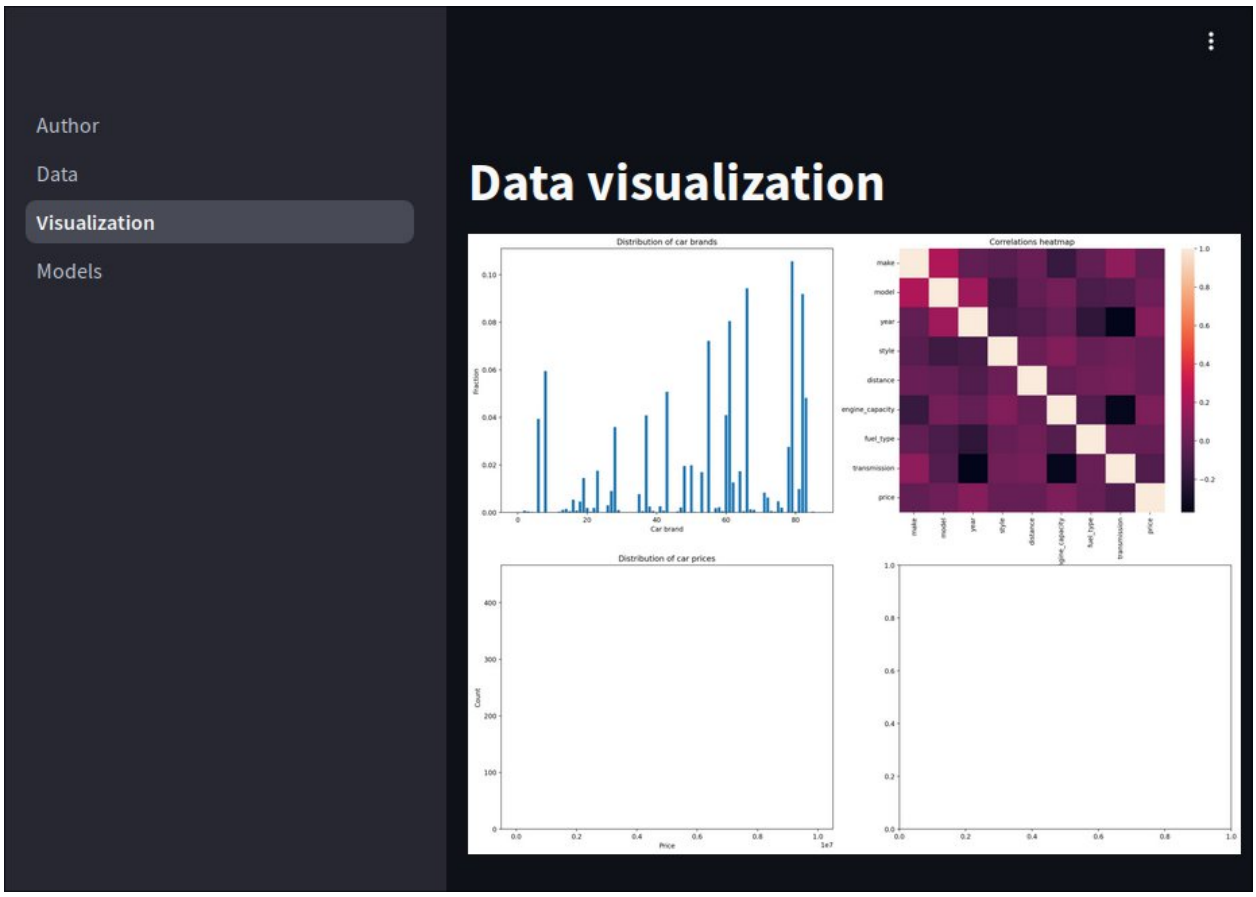

Рисунок 3 — Визуализация зависимостей в наборе данных

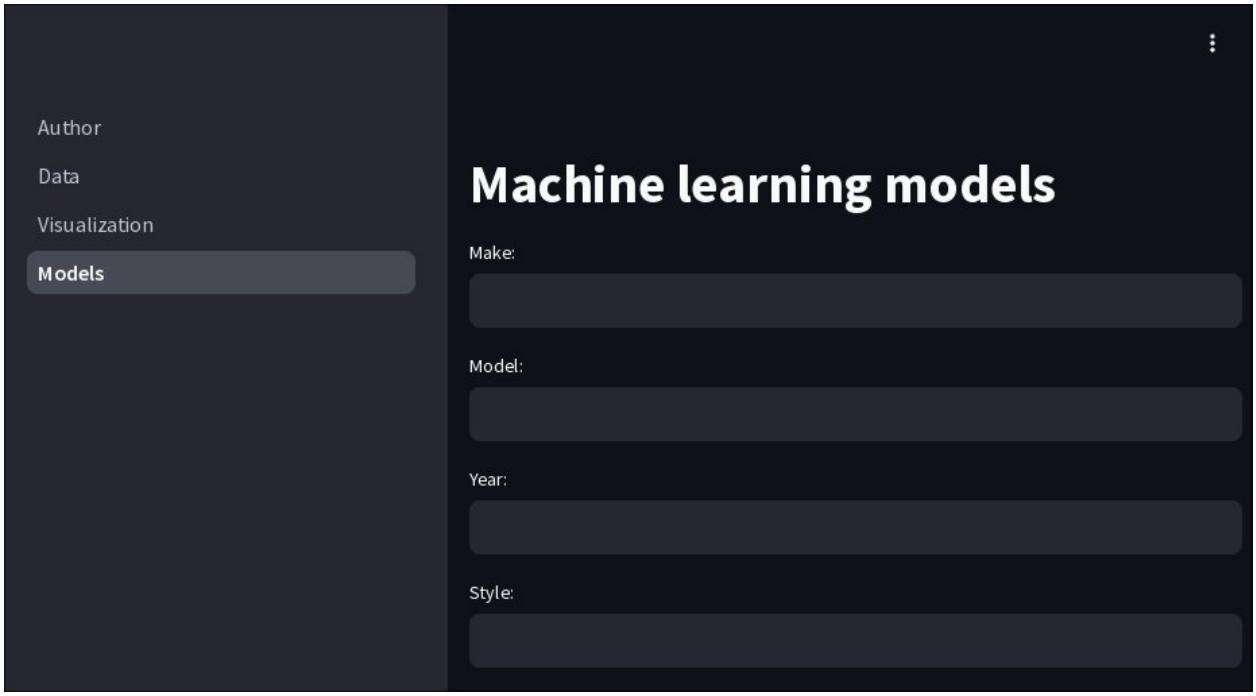

Рисунок 4 — Предсказания моделей на введённых пользователем данных

Приложение Б

```python
def save_model(model, model_name: str) -> None:
    joblib.dump(model, model_name)


def load_model(model_dump_path: str):
    return joblib.load(model_dump_path)


def fit_models(X_train, y_train, X_test, y_test) -> list[float]:
    scaler = StandardScaler().fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)
    save_model(scaler, "dumps/scaler.joblib")

    lr = LinearRegression().fit(X_train, y_train)
    lr_pred = lr.predict(X_test)
    lr_mape = mean_absolute_percentage_error(y_test, lr_pred)
    save_model(lr, "dumps/lr.joblib")

    dbscan = DBSCAN(eps=0.5, min_samples=5).fit(X_train)
    dbscan_ss = silhouette_score(X_train, dbscan.labels_)
    save_model(dbscan, "dumps/dbscan.joblib")

    cb = CatBoostRegressor(verbose=False).fit(X_train, y_train)
    cb_pred = cb.predict(X_test)
    cb_mape = mean_absolute_percentage_error(y_test, cb_pred)
    save_model(cb, "dumps/catboost.joblib")

    br = BaggingRegressor().fit(X_train, y_train)
    br_pred = br.predict(X_test)
    br_mape = mean_absolute_percentage_error(y_test, br_pred)
    save_model(br, "dumps/bagging.joblib")

    estimators = [
        ("lr", RidgeCV()),
        ("svr", LinearSVR()),
    ]
    sr = StackingRegressor(estimators).fit(X_train, y_train)
    sr_pred = sr.predict(X_test)
    sr_mape = mean_absolute_percentage_error(y_test, sr_pred)
    save_model(sr, "dumps/stacking.joblib")

    nn = MLPRegressor().fit(X_train, y_train)
    nn_pred = nn.predict(X_test)
    nn_mape = mean_absolute_percentage_error(y_test, nn_pred)
    save_model(nn, "dumps/nn.joblib")

    return [lr_mape, dbscan_ss, cb_mape, br_mape, sr_mape, nn_mape]
```

```python
def predict(data: list[list[float]]) -> list[float]:
    scaler = joblib.load("dumps/scaler.joblib")
    data = scaler.transform(data)

    lr = joblib.load("dumps/lr.joblib")
    lr_pred = lr.predict(data)

    # dbscan = joblib.load("dumps/dbscan.joblib")
    # dbscan_pred = dbscan.predict(data)

    cb = joblib.load("dumps/catboost.joblib")
    cb_pred = cb.predict(data)

    br = joblib.load("dumps/bagging.joblib")
    br_pred = br.predict(data)

    sr = joblib.load("dumps/stacking.joblib")
    sr_pred = sr.predict(data)

    nn = joblib.load("dumps/nn.joblib")
    nn_pred = nn.predict(data)

    return [lr_pred, cb_pred, br_pred, sr_pred, nn_pred]
```

```python
st.set_page_config(page_title="Models")

st.title("Machine learning models")

make = st.text_input("Make:")
model = st.text_input("Model:")
year = st.text_input("Year:")
style = st.text_input("Style:")
distance = st.text_input("Distance:")
engine_capacity = st.text_input("Engine capacity (cm3):")
fuel_type = st.text_input("Fuel type:")
transmission = st.text_input("Transmission:")

if st.button("Get price"):
    data = [
        float(make),
        float(model),
        float(year),
        float(style),
        float(distance),
        float(engine_capacity),
        float(fuel_type),
        float(transmission),
    ]

    predicted_prices = predict([data])
```

```
116
117     st.success(f"\
118     [+] Predicted prices (in euro):\n\
119     [+] Linear regression: {int(predicted_prices[0])}\n\
120     [+] Catboost: {int(predicted_prices[1])}\n\
121     [+] Bagging: {int(predicted_prices[2])}\n\
122     [+] Stacking: {int(predicted_prices[3])}\n\
123     [+] Neural network: {int(predicted_prices[4])}")
124
```