House Price Prediction Application

1. Overview

The **House Price Prediction Application** is designed to predict housing prices based on specific input parameters such as the average number of rooms, percentage lower status of the population, pupil-teacher ratio, and proportion of non-retail business acres per town. This application was built using **Streamlit** for the user interface, **FastAPI** for the backend, and **scikit-learn** for the machine learning model. The application is fully deployed and accessible online.

---

2. Project Structure

Below is the structure of the project directory:

```
Machine Learning/
|-- data/
|  |-- HousingData.csv
|-- src/
|  |-- __pycache__/
|  |-- api.py
|  |-- app.py
|  |-- load_data.py
|  |-- model.py
|  |-- preprocess.py
|  |-- tuning.py
|-- best_model.pkl
|-- README.md
|-- requirements.txt
|-- Document.pdf
```

---

3. Technologies Used

- **Frontend**: Streamlit
- **Backend**: FastAPI
- **Machine Learning**: scikit-learn
- **Deployment**: Render (for backend) and local hosting (for Streamlit UI)

4. Application Flow

1. **Input Features**: Users enter housing parameters through the Streamlit interface.
2. **Prediction Request**: The app sends the input data to the FastAPI backend.
3. **Model Prediction**: The backend processes the data and returns a predicted house price.
4. **Output**: The predicted price is displayed on the Streamlit app.

---

5. Code Explanation

5.1 Streamlit Frontend (app.py)

The app.py file creates a user-friendly interface to input housing data and visualize predictions.

```
import streamlit as st

import requests


# Title and Description

st.title("🏡 House Price Prediction App")

st.write("Enter the house features below to predict the price.")

# Input Form

with st.form("prediction_form"):

    rm = st.number_input("Average number of rooms (RM):", min_value=1.0, max_value=10.0, step=0.1)

    lstat = st.number_input("Percentage lower status of the population (LSTAT):", min_value=0.0, max_value=40.0, step=1.0)

    ptratio = st.number_input("Pupil-teacher ratio (PTRATIO):", min_value=10.0, max_value=30.0, step=0.1)

    indus = st.number_input("Proportion of non-retail business acres per town (INDUS):", min_value=0.0, max_value=30.0)


    # Submit Button

    submit_button = st.form_submit_button(label="Predict Price")


# Prediction Logic

if submit_button:

    # Backend API URL (update with your Render-deployed FastAPI URL)

    api_url = "https://house-price-predicting-ai.onrender.com/predict/"
```

```
    # Prepare input data
    input_data = {
        "rm": rm,
        "lstat": lstat,
        "ptratio": ptratio,
        "indus": indus
    }

    # Make POST request to the API
    response = requests.post(api_url, json=input_data)

    if response.status_code == 200:
        prediction = response.json().get("predicted_price", "Error: No prediction returned.")
        st.success(f"Predicted House Price: ${prediction}")
    else:
        st.error("An error occurred while fetching the prediction.")
```

5.2 Backend API (api.py)

The api.py file defines the FastAPI backend for processing prediction requests.

```
from fastapi import FastAPI
from pydantic import BaseModel
import pickle
# Load the pre-trained model
model_path = "best_model.pkl"
with open(model_path, "rb") as f:
    model = pickle.load(f)
# Initialize FastAPI app
app = FastAPI()
# Input Data Schema
class InputData(BaseModel):
    rm: float
    lstat: float
    ptratio: float
```

```
    indus: float

# Prediction Endpoint
@app.post("/predict/")
def predict(data: InputData):
    features = [[data.rm, data.lstat, data.ptratio, data.indus]]
    predicted_price = model.predict(features)[0]
    return {"predicted_price": predicted_price}
```

---

6. Deployment

6.1 Backend Deployment

The FastAPI backend was deployed on Render:

1. Uploaded the api.py and best_model.pkl files.
2. Configured the environment for Python with the required dependencies listed in requirements.txt.

6.2 Streamlit Deployment

The Streamlit app is hosted locally but can also be deployed using services like Streamlit Community Cloud.

---

7. Requirements

7.1 Dependencies

Listed in requirements.txt:

```
fastapi
uvicorn
pandas
numpy
scikit-learn
streamlit
requests
```

7.2 Installation Steps

1. Clone the repository.
2. Install dependencies:

3. pip install -r requirements.txt
4. Run the backend:
5. uvicorn src.api:app --reload
6. Run the frontend:
7. streamlit run src/app.py

---

8. Results

Users can input housing data into the Streamlit app and receive predicted house prices instantly. The model uses trained data from the Boston Housing Dataset.

---

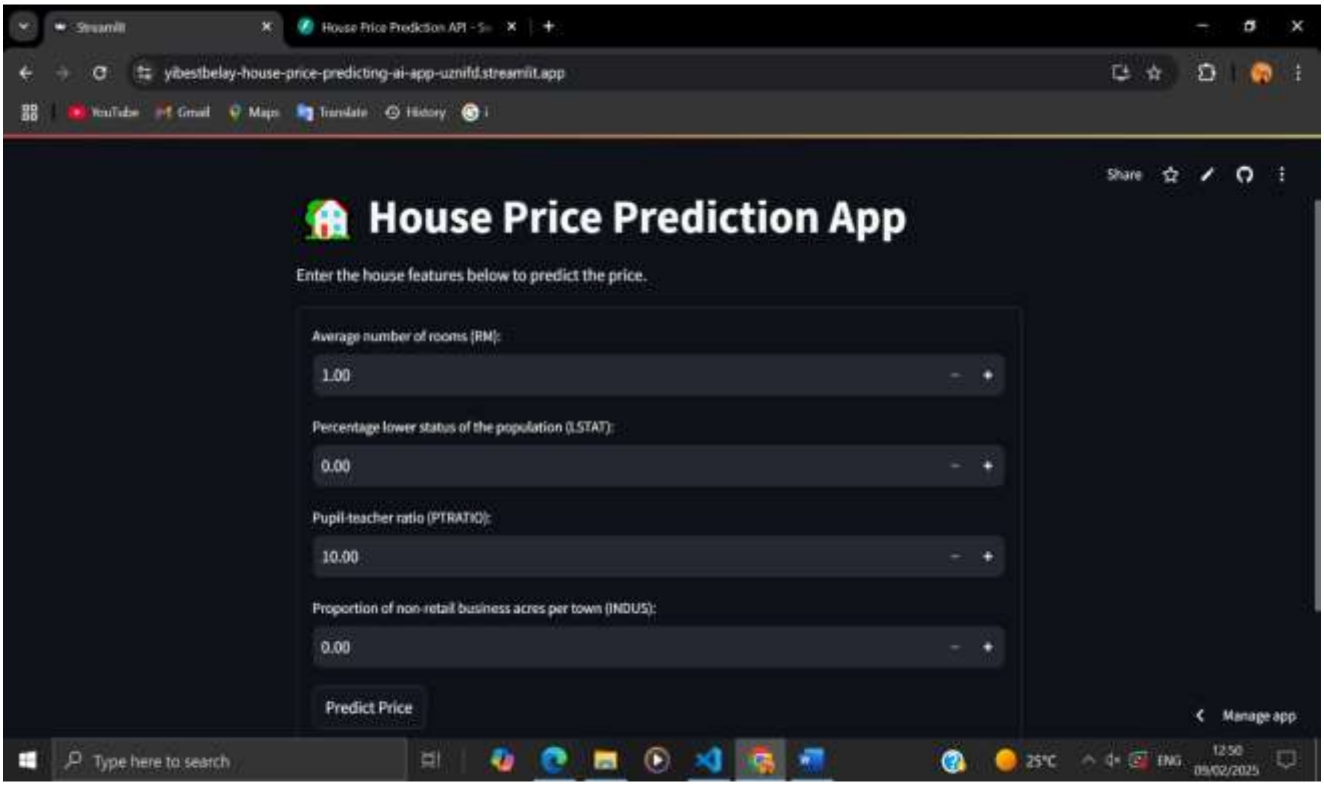
9. Future Improvements

- Add data visualization for user inputs.
- Incorporate additional features for more accurate predictions.
- Deploy the Streamlit app online.

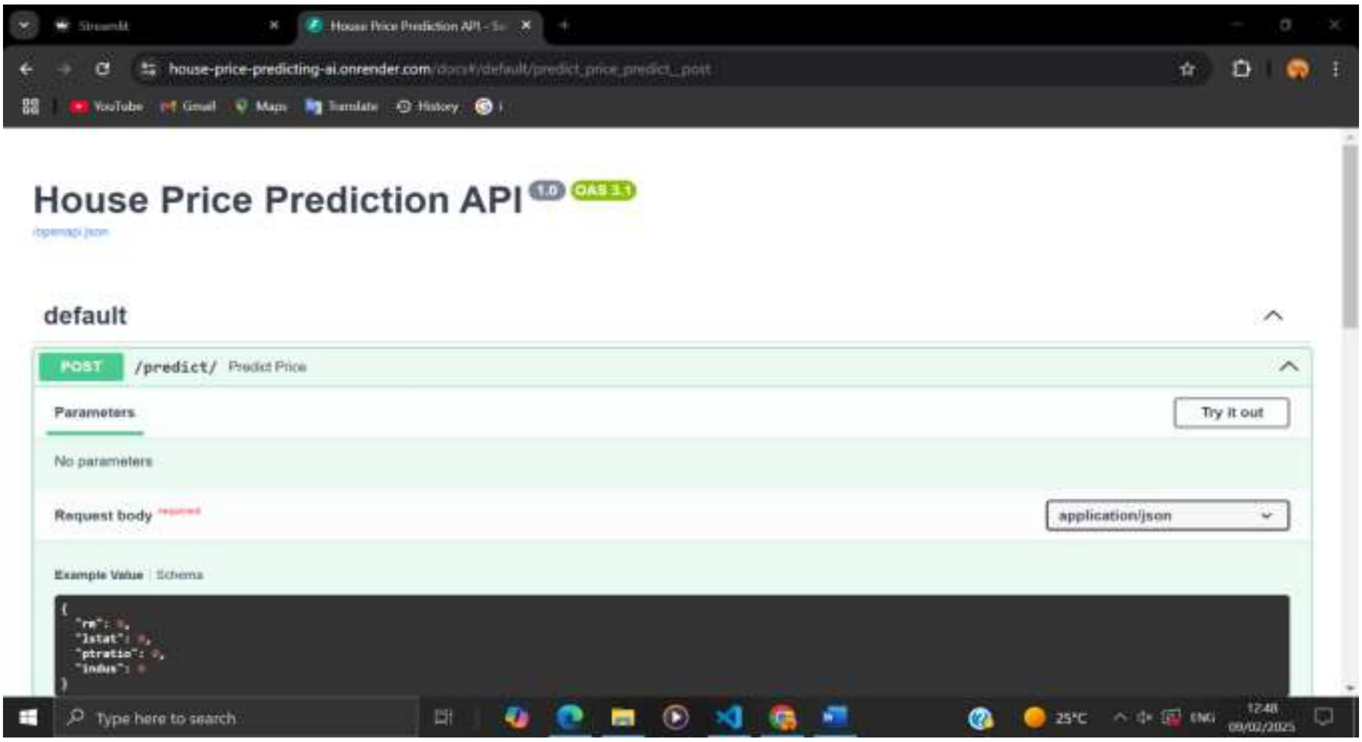
---

10. Screenshots

User Interface



Backend Endpoint



11. Conclusion

The House Price Prediction Application is a robust and user-friendly tool for predicting housing prices using machine learning. It demonstrates the integration of Streamlit, FastAPI, and scikit-learn for seamless ML deployment.

NOTE: ALSO DON’T FORGET TO REFERE THE README FILE