Car Price Prediction Project Report

1. Introduction

Project Objective

The goal of this project is to develop a **machine learning model** that predicts the price of used cars based on various features such as manufacturer, model, engine volume, mileage, and fuel type. This project follows a structured **data science pipeline** including **data preprocessing, exploratory data analysis (EDA), model training, hyperparameter tuning, API deployment**, and **version control**.

Tools & Technologies Used

- **Programming Language:** Python
- **Libraries:** Pandas, NumPy, Scikit-learn, XGBoost, FastAPI, Uvicorn
- **Model Deployment:** Render
- **Version Control:** GitHub
- **Notebook:** Jupyter Notebook

---

2. Data Preprocessing

Dataset

The dataset contains multiple features relevant to used cars. The key variables include:

- **Levy**: Tax-related attribute
- **Manufacturer**: Car brand
- **Model**: Specific car model
- **Production Year**: Manufacturing year of the vehicle
- **Category**: Type of vehicle (e.g., sedan, SUV)
- **Engine Volume**: Engine capacity in liters
- **Mileage**: Distance driven
- **Cylinders**: Number of engine cylinders
- **Doors**: Number of doors
- **Fuel Type**: Petrol, diesel, hybrid, etc.
- **Gearbox Type**: Automatic, manual, etc.

Data Cleaning & Transformation

- **Handling Missing Values**: Missing values in Levy were replaced with the median.

- **Feature Engineering**:
    - Extracted numerical values from Engine Volume.
    - Converted categorical variables into numerical representations.
    - Standardized text-based features to lowercase.
- **One-Hot Encoding**: Categorical features such as Drive wheels, Gearbox type, and Fuel type were converted into one-hot encoded variables.
- **Label Encoding**: Applied to categorical columns for better model compatibility.

---

3. Exploratory Data Analysis (EDA)

Correlation Heatmap

- A **correlation matrix** was plotted to identify relationships between features and the target variable (Price).
- **Key Findings:**
    - Mileage and Production Year showed strong relationships with car price.
    - Fuel type and gearbox type had moderate impacts.

Feature Distributions

- Histograms and boxplots were used to visualize the distributions of key numerical variables.
- **Outlier Treatment:** Winsorization was applied to Mileage, Engine Volume, and Levy to limit extreme values.

---

4. Model Selection & Training

Models Compared

1. **Linear Regression**
2. **Random Forest Regressor**
3. **XGBoost Regressor**

Performance Metrics

- The models were evaluated using:
    - **Mean Squared Error (MSE)**
    - **Mean Absolute Error (MAE)**
    - **R-Squared Score (R^2)**

Best Performing Model

- **XGBoost Regressor** achieved the highest accuracy and lowest error, making it the final model choice.

---

5. Hyperparameter Tuning

To optimize the XGBoost model, **GridSearchCV** was used to fine-tune hyperparameters:

- n_estimators: [100, 200, 300]
- max_depth: [3, 5, 7]
- learning_rate: [0.01, 0.1, 0.2]

The **best combination** found was:

- n_estimators=200
- max_depth=5
- learning_rate=0.1

---

6. Model Deployment

API Development using FastAPI

A RESTful API was built using **FastAPI** to allow users to input car details and receive a predicted price.

API Endpoints

- GET / → Returns a welcome message.
- POST /predict → Takes car features as input and returns the predicted price.

Example API Request

Request:

```
{
    "Levy": 1500
    "Manufacturer": 3,
    "Model": 25,
    "Prod_year": 2018,
    "Category": 2,
    "Leather_interior": 1,
    "Engine_volume": 2.0,
    "Mileage": 45000,
    "Cylinders": 4,
```

"Doors": 4,

"Wheel": 1,

"Color": 5,

"Airbags": 6,

"Drive_4x4": false,

"Drive_front": true,

"Drive_rear": false,

"Gear_box_automatic": true,

"Gear_box_manual": false,

"Gear_box_tiptronic": false,

"Gear_box_variator": false,

"Fuel_cng": false,

"Fuel_diesel": false,

"Fuel_hybrid": false,

"Fuel_hydrogen": false,

"Fuel_lpg": false,

"Fuel_petrol": true,

"Fuel_plug_in_hybrid": false

}

Response:

{"predicted_price": 8789.5810546875}

---

7. Deployment on Render

- The API was deployed using **Render**, making it accessible online.
- The deployment included:
    1. Connecting the GitHub repository to Render.
    2. Setting up an environment with **Python 3**.
    3. Running uvicorn to start the FastAPI server.

Deployment URLs

- **Public API URL:** https://car-price-api-m4xn.onrender.com/
- **API Docs URL:** https://car-price-api-m4xn.onrender.com/docs
- **GitHub Repository:** https://github.com/Hospitas4u/DS-Project-_-Jinsheng-Yu-Shanyu-Wang

---

8. Version Control & GitHub

- The project was managed using **Git** and stored in **GitHub**.
- .gitignore was set up to exclude environment file.
- **GitHub Actions** can be added for automated testing and deployment.

---

9. Conclusion & Future Work

Summary

- Successfully built a **car price prediction model** with **XGBoost**.
- Deployed the model using **FastAPI** and hosted it on **Render**.
- The API is **live and accessible**.

Future Enhancements

- **Improve Feature Engineering**: Consider additional attributes like accident history.
- **Enhance Model Performance**: Try deep learning approaches.
- **Improve API Security**: Implement authentication and rate limiting.