REPORT

(Jiayi_QIAN 87297, Shujuan_ZHU 87313)

1. Introduction

The goal of this project is to develop a machine learning model that predicts car prices based on user input. We utilized a dataset from Kaggle, performed data exploration and preprocessing, trained multiple machine learning models, and deployed an API using FastAPI.

2. Performance Metric

To evaluate the predictive power of our model, we chose Root Mean Squared Error (RMSE) and R^2 Score as the performance metrics:

- RMSE (Root Mean Squared Error): Measures the average error magnitude between the predicted and actual car prices. A lower RMSE indicates better performance.
- R^2 Score: Represents the proportion of variance explained by the model. A higher value (closer to 1) indicates a better fit.

These metrics were chosen because car prices have a continuous distribution, and minimizing the RMSE helps improve prediction accuracy.

3. Performance of algorithm

```
y_pred = baseline_model.predict(X_test)

# evaluate
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

# print
print(f"Baseline Model Performance:")
print(f"RMSE: {rmse:.2f}")
print(f"R²: {r2:.4f}")
```

```
Baseline Model Performance:
RMSE: 11218.57
R²: 0.3970
```

```
Random Forest Model Performance:
RMSE: 7476.36
R²: 0.7322
```

```
# XGBoost Model
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score


scaler = StandardScaler()
X_train[num_features] = scaler.fit_transform(X_train[num_features])
X_test[num_features] = scaler.transform(X_test[num_features])

# XGBoost
xgb_model = XGBRegressor(n_estimators=200, max_depth=10, learning_rate=0.05, random_state=42)
xgb_model.fit(X_train, y_train)

# predict
y_pred = xgb_model.predict(X_test)

# evaluate model
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

# print results
print(f"XGBoost Model Performance:")
print(f"RMSE: {rmse:.2f}")
print(f"R²: {r2:.4f}")
```

```
XGBoost Model Performance:
RMSE: 6753.39
R²: 0.7815
```

- Baseline Model: RMSE: 11218.57 R^2: 0.3970 The baseline model is a simple model (linear regression). RMSE of 11218.57 means the model's predictions deviate from the actual values by about 11218 units on average. R^2 of 0.3970 suggests that the model can explain about 39.7% of the variance in the target variable (Price). This is a relatively low R^2, indicating that the model is not doing a very good job of explaining the target variable's variation, which is typical for a simple baseline model.

- Random Forest Model: RMSE: 7476.36 R^2: 0.7322 The Random Forest model, a more complex ensemble method, performs better: RMSE of 7476.36: The average deviation between predicted and actual values is much lower than the baseline model, meaning the model's predictions are significantly closer to the true values. R^2 of 0.7322: This model explains about 73.22% of the variance in the target variable, which is a substantial improvement over the baseline model. This suggests that Random Forest is capturing much more of the underlying relationships in the data.

- XGBoost Model: RMSE: 6753.39 R^2: 0.7815 The XGBoost model is performing better than Random Forest, but only slightly: RMSE of 6753.39: The deviation between predicted and actual values is lower than both the baseline and Random Forest models, indicating that XGBoost is providing the most accurate predictions of all the models tested. R^2 of 0.7815: This model explains 78.15% of the variance in the target variable, which is the highest R^2 among the three models. XGBoost has clearly learned more from the data and has a stronger predictive power compared to both the baseline and Random Forest models.

4. Hyperparameter Optimization Rounds

```
Best Hyperparameters: {'colsample_bytree': 0.7, 'learning_rate': 0.05, 'max_depth': 10, 'n_estimators': 200, 'subsample': 0.9}
XGBoost Model Performance with Optimized Hyperparameters:
RMSE: 6506.23
R²: 0.7972
```

- First round of tuning：Through GridSearchCV, you optimized parameters such as n_estimators, max_depth, and learning_rate in a smaller search space, which improved the performance of the model.

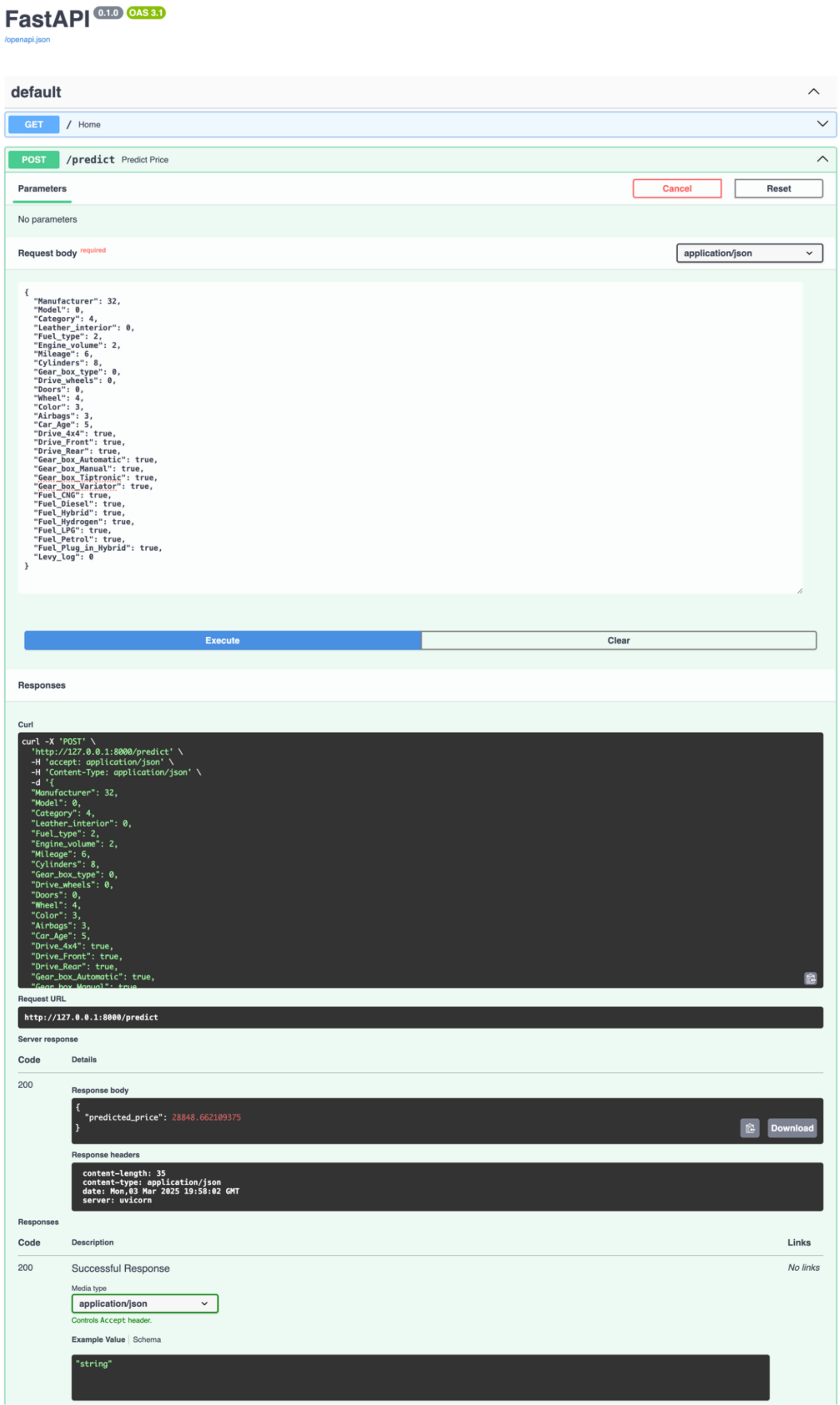
```
Best Hyperparameters: {'colsample_bytree': np.float64(0.7211248392548631), 'learning_rate': np.float64(0.030891871761536023), 'max_depth': 1
1, 'n_estimators': 250, 'subsample': np.float64(0.7420252045709572)}
XGBoost Model Performance with Optimized Hyperparameters:
RMSE: 6431.21
R²: 0.8018
```

- Second round of tuning:A wider search was performed using RandomizedSearchCV, resulting in a lower learning_rate, an additional layer of max_depth, and further tuning of subsample and colsample_bytree to improve generalization.

```
Best Hyperparameters: {'colsample_bytree': np.float64(0.7380330511289547), 'gamma': np.float64(0.3476192144826835), 'learning_rate': np.float
64(0.013180468148516514), 'max_depth': 11, 'min_child_weight': 1, 'n_estimators': 744, 'subsample': np.float64(0.7843115072130903)}
Optimized XGBoost Model Performance:
RMSE: 6429.74
R²: 0.8019
```

- Final optimization results: This search was more refined, introducing gamma and min_child_weight, and significantly improving n_estimators, ultimately achieving the lowest RMSE (6429.74) and highest R^2 (0.8019), indicating that the model has better fitting ability and generalization performance.

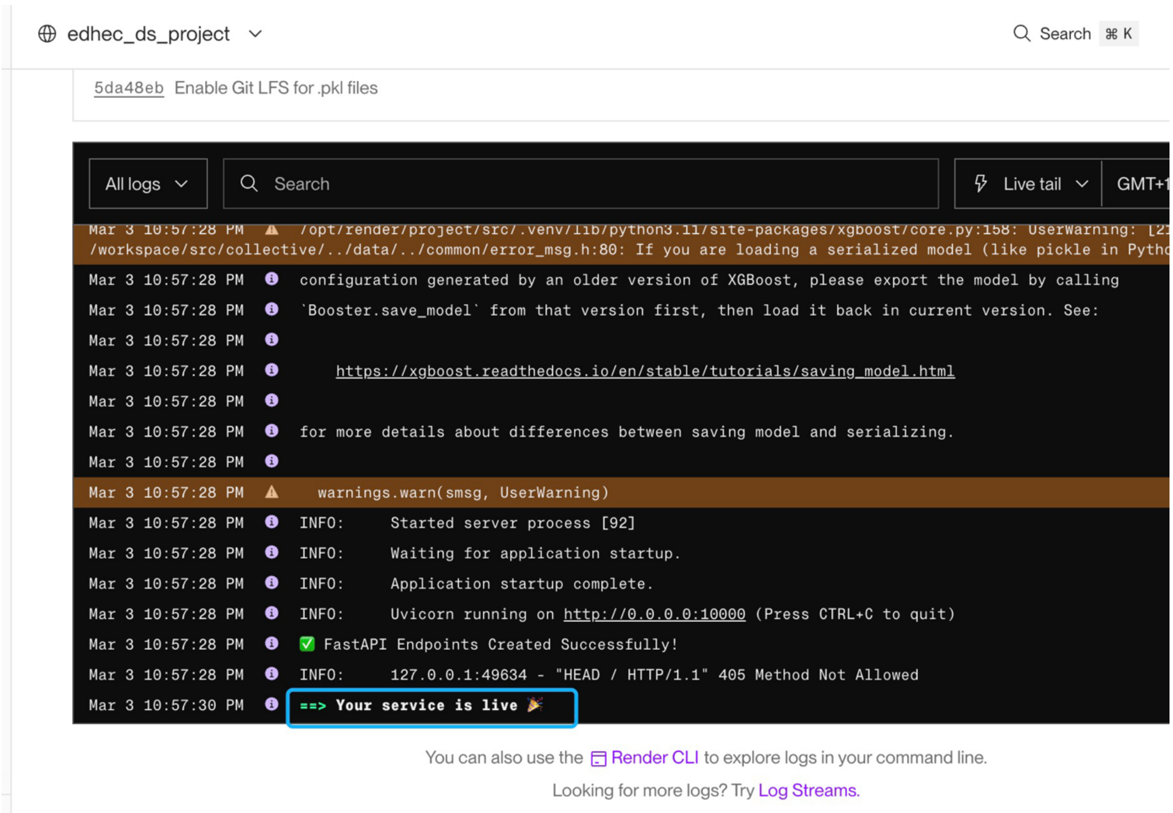
5. API:



Our FastAPI-based car price prediction API enables users to input car features and receive an estimated price. Here is an example of our prediction.

The API has two endpoints:

- GET / – Returns a JSON message confirming that the API is running.
- POST /predict – Accepts car attributes in JSON format and processes them to match the trained model’s expected input.



The steps include:

- Converting the input JSON into a Pandas DataFrame.
- Renaming columns to align with those used during model training.
- Selecting only the necessary features expected by the model.
- Passing the processed data to the XGBoost model (best_xgb_model.pkl) for prediction.

The API returns the predicted car price as a JSON response. This design ensures that the model receives consistent input, minimizing errors and maintaining prediction accuracy. The API is structured for scalability and was deployed on platforms Render for real-time access.

6. The link of the different resources of the project (notebook, github, url of the API on render).

Github：（notebook is also uploaded on github）

https://github.com/shujuan12/Data_science

Render：

https://edhec-ds-project.onrender.com

7. Conclusion of the project.

In this project, we successfully developed a machine learning model to predict car prices based on user-provided features. By leveraging a real-world dataset from Kaggle, we performed extensive data exploration, preprocessing, and model comparison, ultimately selecting XGBoost as the best-performing model. Through multiple rounds of hyperparameter tuning, we improved the model's performance, achieving RMSE of 6429.74 and R^2 of 0.8019, demonstrating strong predictive accuracy.

To make the model accessible, we built a FastAPI-based API that allows users to input car attributes and receive an estimated price. We have successfully presented some examples of prediction in http://127.0.0.1:8080/docs and uploaded the png in github . What's more, the API is designed for scalability, ensuring seamless integration and was deployed in Render for real-world applications.