Car Price Prediction With Machine Learning

Submitted by: Omeerpal

Table of Contents

1. Project Description

This project develops machine learning models to accurately predict car prices based on factors like brand, year, fuel type, transmission, and mileage.

2. Technologies Used

Python 3.x, Pandas, NumPy, Matplotlib, Seaborn, Plotly, Scikit-learn, XGBoost, LightGBM, SHAP.

3. Dataset Description

The dataset contains car attributes including Car_Name, Year, Selling_Price (target), Present_Price, Kms_Driven, Fuel_Type, Seller_Type, Transmission, Owner.

4. Data Preprocessing

Includes handling missing values, feature engineering (car age), dropping unused columns, encoding categorical variables, and normalizing numeric values.

```
import pandas as pd

import numpy as np
```

```
# Load the dataset
df = pd.read_csv("car_data.csv")

# Drop duplicates and handle missing values
df.drop_duplicates(inplace=True)
df.dropna(inplace=True)

# Create age feature
df['Car_Age'] = 2025 - df['Year']

# Drop unused columns
df.drop(['Car_Name', 'Year'], axis=1, inplace=True)

# Convert categorical columns
df = pd.get_dummies(df, drop_first=True)

df.head()
```

5. Exploratory Data Analysis

Visual analysis includes histogram of car prices, correlation heatmap to identify important relationships between features.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Distribution of car prices
plt.figure(figsize=(8,5))
```

```
sns.histplot(df['Selling_Price'], bins=30, kde=True, color='skyblue')

plt.title("Distribution of Selling Price")

plt.show()


# Correlation heatmap

plt.figure(figsize=(10,6))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap")

plt.show()
```

6. Feature Engineering

Applies log transformation to skewed variables, standard scaling, and calculates car age.

```
# Log transformation of skewed variables

df['Kms_Driven'] = np.log1p(df['Kms_Driven'])


# Normalization

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()


X = df.drop('Selling_Price', axis=1)

y = df['Selling_Price']

X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

7. Model Building

Trains Linear Regression, Random Forest, and XGBoost models to learn car price prediction.

```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
```

```
from sklearn.ensemble import RandomForestRegressor

from xgboost import XGBRegressor


# Split data

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)


# Linear Regression

lr = LinearRegression()

lr.fit(X_train, y_train)


# Random Forest

rf = RandomForestRegressor(n_estimators=200, max_depth=8, random_state=42)

rf.fit(X_train, y_train)


# XGBoost

xgb = XGBRegressor(n_estimators=200, learning_rate=0.1, max_depth=5, random_state=42)

xgb.fit(X_train, y_train)
```

8. Hyperparameter Tuning

Uses RandomizedSearchCV for tuning XGBoost parameters like n_estimators, max_depth, learning_rate.

```
from sklearn.model_selection import RandomizedSearchCV


param_dist = {

    'n_estimators': [100, 200, 300],

    'max_depth': [3, 5, 7, 10],

    'min_child_weight': [1, 3, 5],
```

```
    'learning_rate': [0.01, 0.05, 0.1]
}


rs_cv = RandomizedSearchCV(XGBRegressor(), param_dist, n_iter=10, scoring='r2', cv=3,
random_state=42)
rs_cv.fit(X_train, y_train)
best_model = rs_cv.best_estimator_
```

9. Model Evaluation

Compares models using R^2 Score and RMSE. XGBoost performs best after tuning.

```
from sklearn.metrics import r2_score, mean_squared_error


models = {'Linear': lr, 'Random Forest': rf, 'XGBoost': xgb, 'Tuned XGB': best_model}
for name, model in models.items():
    y_pred = model.predict(X_test)
    print(f"{name} - R2 Score: {r2_score(y_test, y_pred):.3f}, RMSE:
{np.sqrt(mean_squared_error(y_test, y_pred)):.2f}")
```

10. SHAP Interpretation (Feature Importance)

```
import shap
explainer = shap.Explainer(best_model)
shap_values = explainer(X_test)


# Summary plot
shap.summary_plot(shap_values, X_test, plot_type='bar')
```

11. Conclusion

- XGBoost with hyperparameter tuning gave the best results.
- Important features: Car Age, Present Price, Fuel Type.
- Can be deployed using Streamlit or Flask for real-world use.