California
Science and Technology University

Used Car Price Prediction

A Data-Driven Approach Using Machine Learning

Project report for:
CSE 604 Machine Learning Fundamentals
by Professor Yongchang Feng, Glen Qin

Team 1:
- Cynthia Widjaja - Priyanka Mohapatra
- Justin Chen - Yaoching Chi
- Nathaly Cobo Piza

Objective

Explore both regression and classification techniques in a single dataset and how they can be applied based on specific project goals.

- Model 1 - Price Classification
  - Classify cars into price categories for users to quickly assess the value of cars
  - Provide insights into how features influence car pricing and classification

- Model 2 - Price Prediction with Regression
  - Develop a reliable model to predict car prices based on features
  - Compare the performance of different models for price prediction

Dataset

Dataset Source:

- The dataset was sourced from Used car listings. It contains information on used cars from various manufacturers, collected over several years.

Key Dataset Features:

- Brand: Car manufacturer (Toyota, Honda, Ford, etc.)
- Year: Year of manufacture
- Mileage: Total distance traveled
- Engine Informations:
    a. 6 Cylinder: 6,871 cars
    b. 8 Cylinder: 5,423 cars
    c. 4 Cylinder: 2,760 cars
    d. Other: 2,088 cars
    e. 12 Cylinder: 117 cars
- Fuel Type: Gasoline, Electric, Hybrid, Diesel, etc.
- Transmission: Manual, Automatic
- Price: Target variable
- Accident history
- Clean title

Feature Engineering

To extract the useful features so that we can enhance model performance.

Creating New Features:

- **Age of the car**: Calculated as 2024 - model_year to represent how old the car is.
- **Horsepower, Engine Size Information**:
  - Extracted from the engine description (e.g., 200HP).
  - Indicator variable (has_hp_info) to note if horsepower data is available.
  - Extracted from the engine description (e.g., 2.0L).
  - Indicator variable (has_engine_size_info) to note if engine size information is available.
- **Luxury Status**:
  - Created a new binary feature is_luxury, where luxury brands like BMW, Mercedes-Benz, Audi, Lexus, and Porsche are marked as 1.

Encoding Categorical Variables:

- Using One-Hot encoding to encode the categorical variables.

Data Processing

- **Missing Data Handling**
  Using imputer tool: **SimpleImputer**
  1. Median for numeric features
  2. “Missing” for categorical features.

- **Outlier Detection**
  Used statistical methods like the IQR (Interquartile Range) to detect and remove outliers, ensuring a more robust model.

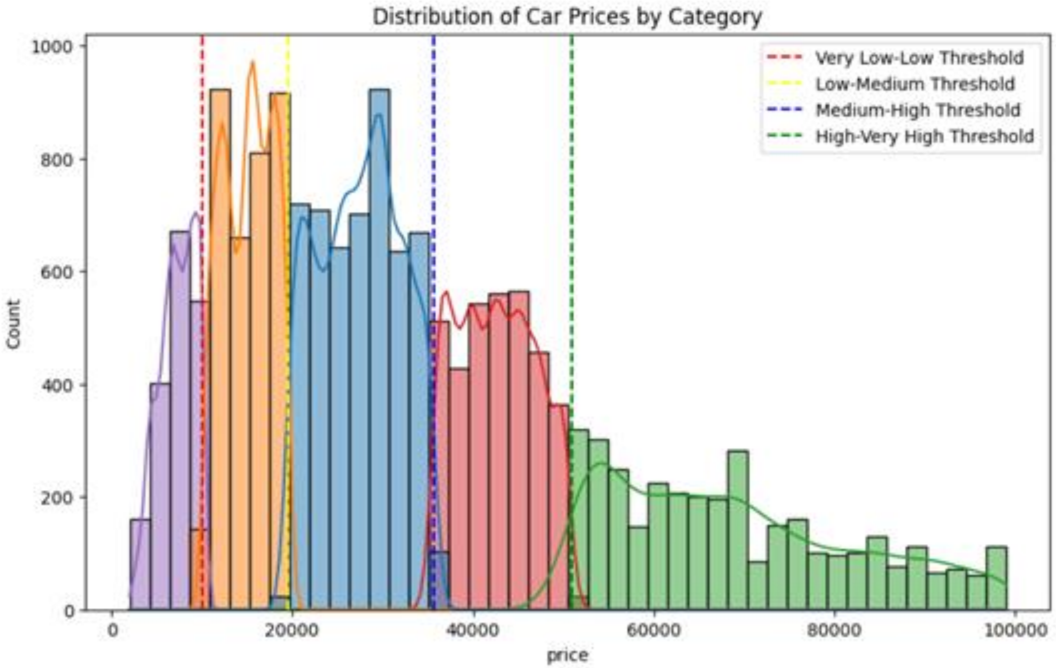
- **Scaling**
  Applied **StandardScaler** and **SMOTE** to balance the dataset.

- **Train-Test Split**
  Split the dataset into training (80%) and testing (20%) sets to evaluate model performance.

Model 1 - Price Classification

We developed a **classification model** to predict car price categories based on various features such as model year, mileage, brand, fuel type, transmission type, and engine specifications.

The project involves extensive **data cleaning**, **feature engineering**, and the use of a **machine learning model, XGBoost,** to classify car prices into categories like:
"Very Low" "Low"
"Medium" "High"
”Very High”



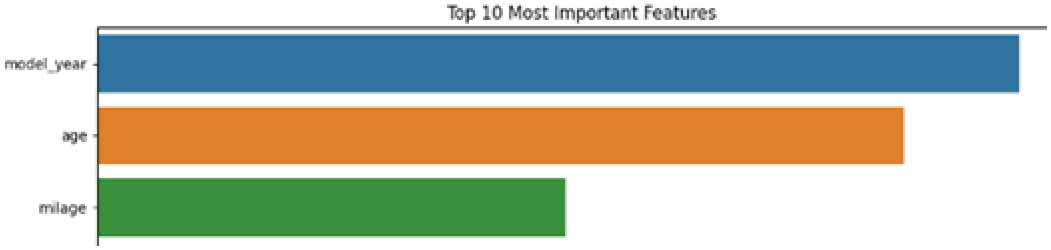
Model 1 Code Snippet

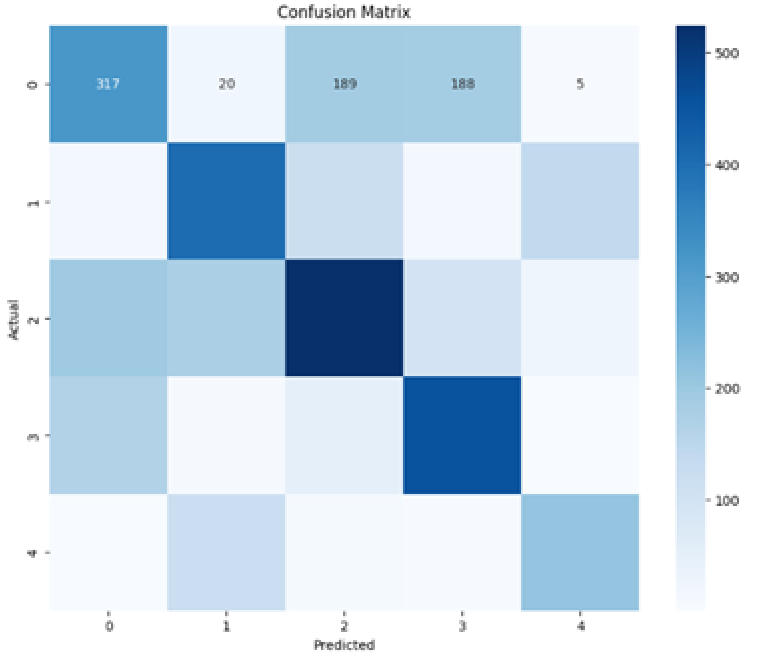
```
# Create a pipeline with SMOTE
xgb_pipeline = ImbPipeline([
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42)),
    ('classifier', XGBClassifier(random_state=42))
])
print(X)
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# Hyperparameter tuning
param_grid = {
    'classifier__n_estimators': [100, 200],
    'classifier__max_depth': [3, 5, 7],
    'classifier__learning_rate': [0.01, 0.1]
}

grid_search = GridSearchCV(xgb_pipeline, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best model
best_model = grid_search.best_estimator_

# Make predictions
y_pred = best_model.predict(X_test)
```

Model 1 - Report

```
Accuracy: 0.5527230590961761

Classification Report:
              precision    recall  f1-score   support

        High       0.46      0.44      0.45       719
         Low       0.55      0.58      0.57       690
      Medium       0.59      0.51      0.55      1018
   Very High       0.60      0.67      0.63       681
    Very Low       0.55      0.61      0.58       344

    accuracy                           0.55      3452
   macro avg       0.55      0.56      0.56      3452
weighted avg       0.55      0.55      0.55      3452
```

The accuracy of the best model was approximately **55%**, indicating room for improvement. This could be due to the complexity of the dataset or the need for further feature engineering.





Model 1 - Price Classification Result

Loading the trained model, making **predictions** on new data, and providing **estimated price ranges** for cars based on the predicted categories

```python
# Example usage
sample_car = [{
    # actual price: $9,991, https:/
    'model_year': 2012,
    'milage': 110189,
    'age': 12,
```

```
Predicted price category: Low
Estimated price range: $10000.00 - $19500.00
```

```python
  # actual price: $21,997, https://
  'model_year': 2017,
  'milage': 67752,
  'age': 7,
```

```
Predicted price category: Medium
Estimated price range: $19500.00 - $35690.00
```

- The model's accuracy and precision should be validated further with a broader range of test cases and real-world data to ensure its robustness and reliability.

Model 2 - Price Prediction with Regression

Accurate price prediction can benefit:

- Customers: help buyers find fair prices for vehicles
- Sellers: ability to price vehicles competitively
- Marketplaces: promoting transparency and trust among users

Used car prices are influenced by a variety of features:

- Brand, model year, milage, fuel type, transmission, accident history, clean title

Regression Feature Preparation

Numerically Interpretable:

- Numerical features: model year, mileage
- Boolean features: Had accident? Has clean title?

Not Numerically Interpretable:

- Categorical features: brand, fuel type, transmission

One-Hot Encodings

One-Hot encode categorical features to allow numerical interpretation

```
Data columns (total 1 columns):
 #   Column     Dtype
---  ------     -----
 0   fuel_type  object
dtypes: object(1)
```

```
Data columns (total 4 columns):
 #   Column              Dtype
---  ------              -----
 0   fuel_type_Diesel    bool
 1   fuel_type_Gasoline  bool
 2   fuel_type_Hybrid    bool
 3   fuel_type_Unknown   bool
dtypes: bool(4)
```

Group sparse categories together to reduce dimensions

```
Original fuel_types:
Gasoline          165940
Hybrid              6832
E85 Flex Fuel       5406
NaN                 5083
Diesel              3955
–                    781
Plug-In Hybrid       521
not supported         15
```

```
Updated fuel_types:
Gasoline    171346
Hybrid        7353
Unknown       5879
Diesel        3955
```

Model 2a - Linear Regression

- Assumptions
  - Linear relationship between features and the target variable (Price)
- Strengths:
  - Easy to interpret and understand the relationship between features and the target
  - It’s inexpensive and fast for training even for large datasets
- Limitations:
  - May not capture complex patterns in the data
  - It may lead to biased predictions and underfitting

Linear Regression Evaluation

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, root_mean_squared_error, mean_absolute_error, r2_score

# Create and train the Linear Regression model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = linear_model.predict(X_test)
```

```
LinearRegression results:
Mean Squared Error:        213551520.6136
Root Mean Squared Error:   14613.40209
Mean Absolute Error:       11049.66592
R-squared:                 0.52059952
```

```
Linear regression equation:
price = -3013463.33 + (1518.23 * model_year) + (-0.18 * milage) + (-2275.37 * accident) + (

Top 30 features by absolute coefficient:
                  Feature   Coefficient
0          brand_Plymouth  31659.521969
1           brand_Bugatti  24333.595025
2          brand_Polestar -17894.081574
3       brand_Rolls-Royce  16110.976543
4           brand_Maybach  14705.163572
5           brand_Bentley  10332.083049
6             brand_Mazda  -9877.104158
```

Model 2b - Gradient Boosting Regression

Iteratively builds decision trees minimizing residual error with gradient descent

- Strengths:
    - It’s a flexible, complex model that can capture non-linear relationships
    - Calculates feature importance during training
- Limitations:
    - Requires more computational resources
    - Slow to train because it builds decisions trees sequentially
    - Can easily overfit the training data especially if the model is not regularized properly
    - Hyperparameter tuning can greatly affect performance

Gradient Boosting Regression Evaluation

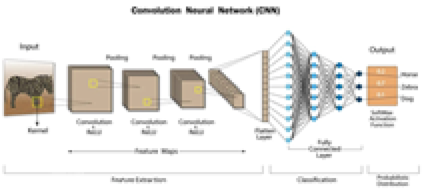
```
from sklearn.ensemble import GradientBoostingRegressor

boosting_model = GradientBoostingRegressor(
    loss='squared_error',
    n_estimators=200,
    learning_rate=0.2,
    max_depth=3,
    random_state=33,
    verbose=1,
)
boosting_model.fit(X_train, y_train)

# Predictions
y_pred = boosting_model.predict(X_test)
```

```
GradientBoostingRegressor results:
Mean Squared Error:          187662300.0360
Root Mean Squared Error:     13698.98902
Mean Absolute Error:         9875.92732
R-squared:                   0.57871807
```

Top 12 Feature importances in Gradient Boosting Model:

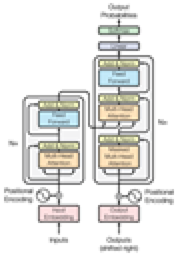
	Feature	Importance
0	milage	0.642035
1	model_year	0.292553
2	brand_Porsche	0.009180
3	clean_title	0.006047
4	brand_Mercedes-Benz	0.005727
5	transmission_Transmission w/Dual Shift Mode	0.004869
6	brand_Mazda	0.004508
7	fuel_type_Diesel	0.004200
8	accident	0.003828
9	brand_Rolls-Royce	0.002579

Use deep learning models on homogenous features



Training CNN for image processing

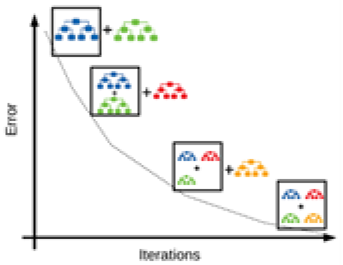
- Data: labeled images
- Features: all numerical pixel data



Training transformer model for natural language processing

- Data: written articles
- Features: all text data

Use gradient boosting models on heterogeneous features



Training gradient boosting regressor for used car price predictions

- Data: used car information
- Features: some numerical (year/mileage), some text (brand/model), some boolean (has accident/clean title)

Model Comparison

Metric	Linear Regression	Gradient Boosting
Mean Absolute Error	213551520.6136	187662300.0360
Root Mean Squared Error	14613.40209	13698.98902
Mean Squared Error	11049.66592	9875.92732
R-Squared (R^2)	0.52059952	0.57871807

Gradient Boosting has a lower error (MAE and MSE), and a higher R^2 value, indicating that it predicts prices more accurately than Linear Regression.

Model 2 - Price Prediction Result

Car Details: 2020 Mercedes-Benz, 19,300 miles, Gasoline, Automatic, no accident, clean title		
Actual Price	Linear Regression	Gradient Boosting Regression
$59,900	$50,371.50	$55,478.15
Car Details: 2009 Jeep, 130,000 miles, Gasoline, Automatic, had accident, clean title		
Actual Price	Linear Regression	Gradient Boosting Regression
$12,500	$7,492.55	$10,550.92
Result: Gradient Boosting price predictions are far more accurate than Linear regression		

Improvement Suggestions

- Balance class distributions more effectively.
- Add more relevant features for better model precision.
- Explore different techniques for handling imbalanced data to optimize performance.
- Perform other feature engineering method to convert categorical features a score.
- Adjust hyperparameter to optimize performance and finding balance between speed and accuracy.

Key Takeaways

Model 1: XGBoost Classification for Car Price Categories

- The XGBoost model demonstrates a basic capability in categorizing car prices, but further refinement is required for improved accuracy.
- Addressing class imbalance with SMOTE (Synthetic Minority Over-sampling Technique) to ensure the model is not biased toward the most frequent price categories.

Model 2: Price Prediction with Regression

- Linear Regression is faster for training but may not be as effective as Gradient Boosting for capturing complex relationships in the data.
- Gradient Boosting is the recommended model for predicting used car prices, providing more reliable and precise results.

Thank you!

Cynthia Widjaja	Justin Chen	Nathaly Cobo Piza	Priyanka Mohapatra	Yaoching Chi