# F20DL – Data Mining and Machine Learning

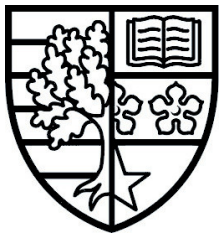
## *Multimodal Car Analytics*

*Car Identification through Images*
*&*
*Resale Price Prediction*

**Group 12**

*by*

Aditya Vikram Singh Dahiya (H00444458)
Joseph Cherian Kariampally (H00417153)
Laiba Shehzad (H00456235)
Mohammed Nibras (H00416887)
Vir Wadwani (H00442389)



**Heriot-Watt University**
**School of Mathematical and Computer Sciences**

**20th November 2025**

**GitHub Repo Link:** https://github.com/JCherry101/DMML-GROUP_12

# 1 Introduction

Determining the fair resale price of a used car is often complex, time-consuming, and highly subjective. Sellers, buyers, and inspection experts must rely on manual evaluation of numerous factors such as the car's conditions, specifications, mileage, etc., making the process labour-intensive and prone to human error.

This project tries to streamline that process, reduce human effort, and improve consistency. It focuses on developing a machine learning system that predicts the resale price of used cars based on tabular data (discussed in section 3), and also classifying features of a car from its images (discussed in section 4).

Using the DVM-CAR dataset [1], we train and compare three regression models: Linear Regression, XGBoost, and Random Forest, to determine which approach has the most accurate price prediction. We use a Convolutional Neural Network (CNN) designed to classify visual features from car images, such as maker, model, colour, and body type, from images of different viewing angles. This combines structured data analysis with computer vision and creates a robust system to assist in predicting a fair resale price, as well as increasing the efficiency of identifying attributes of a car from images.

# 2 Dataset & Data Analysis

For this project, we are using the DVM-CAR (Deep Visual Marketing Car) dataset [1]. It was created by researchers from the University of Southampton as part of the Deep Visual Marketing project. It is a large-scale automotive dataset designed for both visual and market specification research.

The dataset contains six relational CSV tables covering: basic model specification data, UK car sales over time, new car entry-level prices, trim level details, used car advertisement data, and image metadata, along with 1,451,784 JPEG images of cars from 899 UK market car models.

We performed several preprocessing and data cleaning steps on the tabular data. We primarily relied on the `adv_table` and `price_table`. The features in the `adv_table` include `Adv_ID`, `Genmodel_ID`, `Reg_year`, `Cumulative_mileage`, `Selling_price`, `Bodytype`, `Maker`, `Model`, etc.

We applied a normalisation step in which car makers with fewer than 60 listed vehicles were removed to ensure sufficient representation. Additionally, all categories (maker, model, colour, body type, gearbox type, and fuel type) were converted into fully numeric encodings.

# 3 Model Implementation for Tabular Data

To predict the resale price of the car, we processed our multimodal car analytics dataset by implementing and comparing three supervised machine learning algorithms: Linear Regression, XGBoost, and Random Forest.

## 3.1 Linear Regression

The DVM-CAR dataset provided tabular data, which we used to create a Linear Regression model for establishing resale price prediction baselines. The final dataset contained a merged dataset whose size depended on the available rows after merging the Advertisement and Price tables based on Maker and `Genmodel_ID` and `Reg_year`.

The model required all numeric fields, including mileage and engine size and price, to have uniform formatting before starting the modelling process. The model incorporated up to five numerical predictor variables, which included Entry Price and Mileage and Engine Size and Seat Count and Door Count, depending on availability in the merged dataset and uses an 80/20 training-to-testing ratio based on the `'Maker'` of each car, which produced training and testing examples according to the final number of valid rows in the dataset.

Linear Regression has no tunable hyperparameters in its basic scikit-learn implementation as it fits the model using Ordinary Least Squares.

**Results:** The Linear Regression model achieved the following results, summarised in table 1:

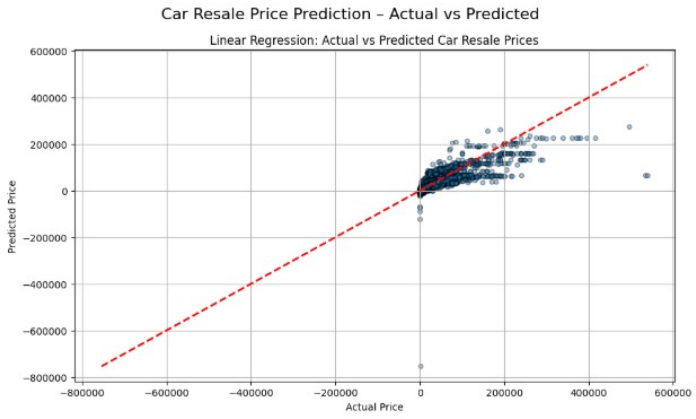



Figure 1: Linear Regression Actual vs Predicted. The loose diagonal trend indicates a partial fit, but widespread points around the $y = x$ line highlight difficulty in capturing complex market trends.

| Metric | Value | Interpretation |
|---|---|---|
| $R^2$ | 0.7182 | 71.82% Var. |
| MAE | 4329.76 | £4,329.76 Error |
| RMSE | 10,050.65 | Std. Dev. |

Table 1: Linear Regression Performance Metrics

| Feature | Value |
|---|---|
| Model | Jaguar XJ |
| Year | 2016 |
| **Actual** | **£40,000** |
| **Predicted** | **£48,126.04** |
| Diff | £1,706.15 |

Table 2: Sample Prediction Analysis

The Actual vs Predicted scatter plot in fig. 1 shows a loose diagonal trend, indicating that the linear regression model partially follows the true pricing behaviour. However, the widespread points around the $y = x$ line highlight that the model has difficulty capturing complex market trends. Hence, we implement more advanced ensemble methods in the following sections.

### 3.2 XGBoost

Extreme Gradient Boosting (XGBoost) is a highly efficient ensemble learning model based on the decision tree model. It uses the tree classifier for better results of prediction and higher operation efficiency [2].

Before training the model, the data was pre-processed as both the advertisement table and price table required cleaning and feature preparation. Both tables were merged using three key identifiers: Maker, `GenModel_ID`, and the car's Registration Year (Year). By merging the tables, we included the `Entry_price` (Manufacturer's Suggested Retail Price – MSRP) into the advertisement table, which served as an important feature for the resale value.

The target variable, Price, and the `Runned_Miles` feature were also cleaned by removing commas, currency symbols, and leading whitespaces, and were then converted to a numerical (float) data type. The feature 'Engin_size' contained values like '2.0L', which was also cleaned to contain only numerical float values in a new column `Engine_size_L`. Missing values in `Runned_Miles`, `Engine_size_L`, `Seat_num`, `Door_Num` and the target feature 'Price' were dropped; however, missing values for `Entry_price` were filled with the median of the existing values in the column.

The model was trained using six numerical and six categorical features:

- **Numerical:** `Runned_Miles`, `Engine_size_L`, `Seat_num`, `Door_num`, `Adv_year`, and `Entry_price`
- **Categorical:** Maker, Genmodel, Colour, Bodytype, Gearbox, and `Fuel_type`

All the categorical features were converted to a numerical value using One-Hot Encoding, as XGBoost can only take numerical inputs.

A very critical step for our model was the custom test-train split, which was implemented to ensure that the distribution of car manufacturers was preserved across the training and testing sets. More specifically, the data was split by 80% for training and 20% for testing on a per-Maker basis. For example, if there were 100 cars with the 'Maker' being Bentley, 80 cars ($0.80 \times 100$) were used for training and 20 cars ($0.20 \times 100$) were used for testing. This ensured that our model is not being tested on a car model that the model has not been trained on. The final test train split was:

- Training Samples: 204,693 (80.0%)
- Testing Samples: 51,205 (20.0%)

The XGBoost Regressor was configured using standard parameters suitable for a regression task: the objective function was set to `reg:squarederror` (minimising Mean Squared Error), with `n_estimators` set to 100, `learning_rate` at 0.1, and a `max_depth` of 6.

**Results:** The XGBoost Regressor achieved the following results, summarized in table 3:

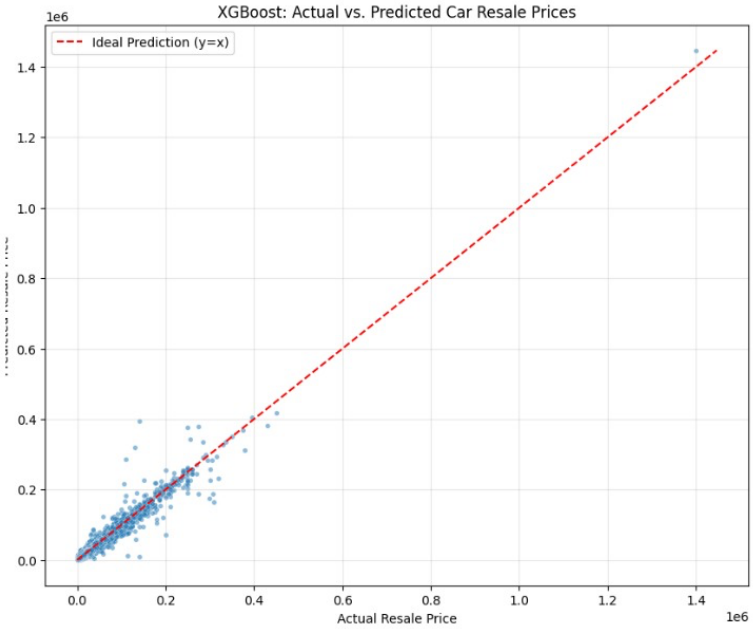



Figure 2: The scatter plot illustrates the strong correlation between actual market values and XGBoost predictions, indicated by the tight clustering along the diagonal ideal prediction line ($y = x$). Minimal deviation in lower price ranges suggests high accuracy.

| Metric | Value | Interpretation |
|---|---|---|
| $R^2$ | 0.9546 | 95.46% Var. |
| MAE | 2030,93 | £2,030.93 Error |
| RMSE | 4247.05 | Std. Dev. |

Table 3: XGBoost Performance Metrics

| Feature | Value |
|---|---|
| Model | Abarth 595 |
| Year | 2018 |
| Mileage | 28,000 |
| **Actual** | **£13,000** |
| **Predicted** | **£12,955.5** |
| Diff | £1,955.5 |

Table 4: Sample Prediction Analysis

The $R^2$ score of 0.9546 is exceptionally high, showing the model learned the complex relationship effectively.

### 3.3 Random Forest

The Random Forest algorithm [3] builds multiple decision trees on random subsets of data and features, and then takes the average of all their predictions for an accurate and stable result. This reduces overfitting and helps the model capture complex, non-linear relationships. Random Forest is well-suited for our project as car resale prices rely on a lot of factors, such as mileage, engine size, brand, and body type. The ability of the model to handle mixed feature types and complex patterns makes it a good choice for predicting car resale prices from tabular data.

The tabular data in the `adv_table` and `price_table` were similarly pre-processed as previously done for the XGBoost model in section 3.2. The model was then run on the same training and testing subset as well to predict the resale price of cars.

The Random Forest model performed very well on our dataset, with the detailed metrics presented in table 5. After training on 204,693 samples and testing with 51,205 samples, the model achieved a Mean Absolute Error (MAE) of 1474. The Root Mean Squared Error (RMSE) of 3683 shows that larger errors are still relatively low compared to typical car prices.

**Results:** The Random Forest model achieved the following results, summarised in table 5:

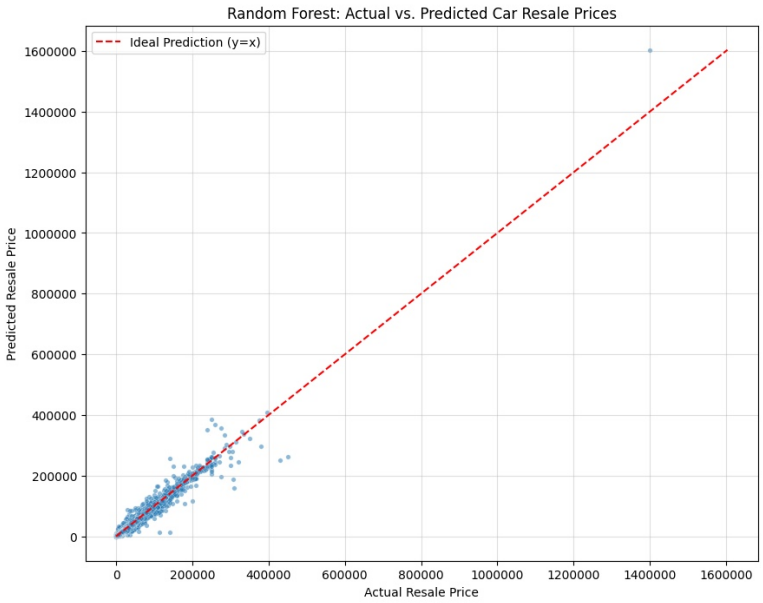



Figure 3: Actual vs. Predicted Car Resale Prices. The Random Forest model exhibits superior precision compared to XGBoost, with tighter clustering around the $y = x$ line.

| Metric | Value | Interpretation |
|---|---|---|
| $R^2$ | 0.966 | 96.6% Var. |
| MAE | 1474 | £1,474 Error |
| RMSE | 3683 | Std. Dev. |

Table 5: Random Forest Performance Metrics

| Feature | Value |
|---|---|
| Model | Abarth 595 |
| Year | 2018 |
| Mileage | 31,886 |
| **Actual** | **£13,000** |
| **Predicted** | **£13,128** |
| Diff | £128 |

Table 6: Sample Prediction Analysis

The model obtained an $R^2$ score of 0.966, indicating that it explains nearly all the variance in car prices and is highly effective at capturing factors that influence resale value. The visual correlation between actual and predicted values is illustrated in fig. 3. These results show that Random Forest is a strong and reliable model for predicting used car resale prices from our dataset.

# 4 Image Classification using Neural Networks (CNN)

## 4.1 Model Architecture and Rationale

To address the challenge of classifying car attributes from images, we implemented a **Dual-Head Convolutional Neural Network (CNN)** named `DualHeadCarNet`. We selected **EfficientNet-B0** as the backbone architecture. EfficientNet was chosen over older architectures like ResNet or VGG because it achieves higher accuracy with significantly fewer parameters and faster inference speeds, which is crucial for processing large datasets like DVM-CAR.

The architecture was modified to support multi-task learning. Instead of training four separate models for Maker, Body Type, Colour, and Model, we utilised a single backbone with four distinct fully connected linear "heads" attached to the feature extraction layer:

- **Backbone:** EfficientNet-B0 (Pre-trained on ImageNet to leverage transfer learning).
- **Heads:**
  - `maker_head`: Classifies the manufacturer (e.g., Bentley, Ford).
  - `body_head`: Classifies the body shape (e.g., SUV, Saloon).
  - `color_head`: Classifies exterior color.
  - `model_head`: Classifies the specific model variant.

## 4.2 Training Methodology and Optimisation

The training process was implemented in PyTorch using a custom `CarDataset` class. We employed a **two-phase training strategy** to maximise performance:

1. **Phase 1: Frozen Backbone (Transfer Learning)** – 2 Epochs. The weights of the EfficientNet backbone were "frozen." Gradients were only calculated for the new classification heads to allow random initialisation to converge without destroying pre-trained feature maps.

2. **Phase 2: Unfrozen Backbone (Fine-Tuning)** – 8 Epochs. The entire network was unfrozen with a lower learning rate ($5e^{-4}$) to allow the backbone to adapt specifically to car features.

**Apple Silicon (M5) Optimization:** The training infrastructure was explicitly optimised for the MacBook Pro M5 architecture - the computer used for training and running the model. The code was modified to utilise the **Metal Performance Shaders (MPS)** backend (`torch.backends.mps`), offloading matrix operations to the M5 GPU, as shown in listing 1.

```
def resolve_device(explicit: Optional[str] = None) -> str:
    if explicit: return explicit
    if torch.backends.mps.is_available():  # Apple Silicon GPU
        return 'mps'
    return 'cpu'
```

Listing 1: Optimizing device selection for M5 chips

### 4.3 Experimental Results

The model was evaluated on the held-out test subset (20% of data). The training logs demonstrate a clear improvement trajectory between the frozen and unfrozen stages, detailed in table 7.

| Metric | Phase 1 (Frozen) | Phase 2 (Unfrozen) | Improvement |
|---|---|---|---|
| Loss | 2.6811 | 0.7975 | -70.2% |
| Maker Accuracy | 97.44% | 97.44% | Stable |
| Body Accuracy | 97.02% | 97.21% | +0.19% |
| Model Accuracy | 88.34% | 93.84% | +5.5% |
| Color Accuracy | 26.26% | 86.39% | +60.13% |

Table 7: CNN Performance Progression (Epoch 2 vs Epoch 8)

The "Frozen" phase achieved high accuracy on broad features (Maker/Body) immediately. However, subtle features like Colour and specific Model variants required the deep feature extraction capabilities of the "Unfrozen" phase, where Colour accuracy spiked from ∼26% to ∼86%.

### 4.4 Inference Case Study

To validate the model's performance on unseen real-world data, we ran inference on selected test images using the final model checkpoint. Figure 4 below illustrates a successful prediction for a white Porsche 918 Spyder, with the corresponding probabilities detailed in table 8. This specific test case highlights the robustness of the Multi-Head architecture in handling specific automotive nuances.

Figure 4: Input Test Image: Porsche 918

| Prediction Class | Probability |
|---|---|
| *Manufacturer Head* | |
| **Porsche** | **0.998** |
| BMW | 0.002 |
| *Body Type Head* | |
| **Convertible** | **0.552** |
| Coupe | 0.445 |
| *Color Head* | |
| **White** | **0.904** |
| Silver | 0.068 |
| *Model Head* | |
| **Porsche::918** | **0.988** |
| Porsche::Boxster | 0.004 |

Table 8: Generated Prediction Probabilities

The inference results in table 8 demonstrate high precision, identifying the **Manufacturer** (**Porsche**, with 99.8% accuracy) and **Model** (918 spyder, with 98.8% accuracy) with near certainty while successfully distinguishing the rare hypercar from the visually similar Boxster and 911. The **Colour** head robustly classified the vehicle as **White** (90.4%), overcoming potential confusion with Silver caused by lighting reflections. Notably, the **Body Type** prediction accurately reflected the 918 Spyder's "targa top" ambiguity with a tight split between **Convertible** (55.2%) and **Coupe** (44.5%); this prioritizes the correct classification while acknowledging the vehicle's dual visual characteristics, confirming the model actively interprets features rather than relying on label memorization.

# 5 Discussion

To optimise the XGBoost Regressor, we conducted tuning experiments, finding that reducing the **learning rate** to 0.05 slightly decreased the R-squared score, and setting `max_depth` to 3 significantly lowered the R-squared to 88%. Based on these results, we chose a setting of `learning_rate=0.1` and `max_depth=10` to maximise performance while ensuring the model captures complexity. This final R-squared of the XGBoost Model was 0.9546 (see section 3.2). By tuning the hyperparameters of the Random Forest model, such as the number of trees, maximum depth and minimum samples per slit, the model's ability to capture complex patterns improved and reduced overfitting. For Linear Regression, we used the baseline model without much experimentation with the hyperparameters, as our focus was on comparing its performance to the other tree-based models.

# 6 Conclusions

The project was set out to address the labour-intensive and subjective nature of manual vehicle value prediction by developing a robust, multi-modal machine learning pipeline. We successfully demonstrated that automating the valuation process is not only feasible but highly accurate as well.

Our comparative analysis identified XGBoost (see section 3.2) and Random Forest (see section 3.3) as the better models. Random Forest achieved an impressive $R^2$ of 0.9658, outperforming the Linear Regression baseline and the XGBoost model, and effectively captured non-linear relationships between mileage, engine size, and market value. Linear Regression assumes the relationship between features and price is linear, which is not the case, and they are influenced by conditional patterns. Hence, the Linear Regression model has performed very poorly in our testing. XGBoost had also performed well, but slightly below the Random Forest. This is due to XGBoost is sensitive to noisy data, outliers, and imperfect preprocessing. The dataset contains a wide range of models and uneven brand frequencies, which can introduce noise that weakens XGBoost's performance if not perfectly tuned. XGBoost would require extensive tuning of its hyperparameters to make it perform better. Overall, Random Forest is resistant to noise, has easier training, and the ability to model complex relationships makes it a strong choice for our use case.

For the computer vision, the implementation of a Multi-Head EfficientNet-B0 CNN (see section 4) proved that a single backbone could efficiently learn diverse tasks simultaneously. By utilising a two-phase training strategy—transfer learning followed by fine-tuning—we mitigated the initial poor performance of the colour head (improving from ∼26% to ∼86%) while maintaining high accuracy for manufacturer (¿97%) and model identification ($>$ 93%).

However, our work is not without limitations. As observed in the XGBoost and Random Forest error analysis, prediction variance increases in high-value luxury segments, likely due to the scarcity of training data for rare vehicles. To mitigate this, we applied a normalisation step, filtering out manufacturers with fewer than 60 listings, though this naturally limits the model's applicability to mass-market vehicles. Additionally, visual classification faces challenges with ambiguous body types, such as the Porsche 918 Spyder (convertible vs. coupe). Our mitigation strategy involved analysing softmax probability distributions rather than relying solely on top-1 predictions, providing a more nuanced and interpretable output for human users.

Future iterations of this work would benefit from a unified multimodal architecture, where the visual features extracted by the CNN are fed directly as inputs into the price prediction model, creating a truly holistic valuation system for the automotive industry.

# Appendix

## A Member Contributions

- **Aditya Dahiya:** Pre-processed image dataset, developed, trained, and tested the CNN for image classification.
- **Joseph Cherian:** Pre-processed image dataset, developed, trained, and tested the CNN for image classification.
- **Laiba Shehzad:** Pre-processed dataset; developed and tested XGBoost model.
- **Mohammed Nibras:** Developed and tested Linear Regression model.
- **Vir Wadwani:** Pre-processed dataset; Developed and tested Random Forest model.

## B List of Figures



## C List of Tables



## D List of Code Listings



## E References


[1] J. Huang, B. Chen, L. Luo, S. Yue, and I. Ounis. Dvm-car: A large-scale automotive dataset for visual marketing research and applications, 2022. Accessed: 18-11-2025.

[2] H. Li, Y. Cao, S. Li, J. Zhao, and Y. Sun. Xgboost model and its application to personal credit evaluation. *IEEE Intelligent Systems*, 35(3):52–61, 2020.

[3] H.A. Salman, A. Kalakech, and A. Steiti. Random forest algorithm overview. *Babylonian Journal of Machine Learning*, pages 69–79, 2024.


## F Generative AI

In line with Heriot-Watt University's GenAI policy, this section transparently discloses all uses of generative AI in this project.

**Tool Used:** Microsoft Copilot
**Use Cases:** Copilot assisted with clarifying technical concepts, debugging code snippets, and providing writing and clarity suggestions in the report.. All critical research decisions, code development, analysis, experimental design, and result interpretation were independently performed by the group and its members.