# EPL448 Data Mining Project: Predicting Prices for Used Cars

Andreas Hadjoullis
Julios Fotiou

Department of Computer Science
University of Cyprus

April 25, 2025

# Outline

1. Introduction
2. Exploratory Data Analysis
3. Data Splitting
4. Data Preprocessing
5. Model Training and Evaluation
6. Classification Problem: Predicting Price Categories

# Introduction

- The used car market is huge and rapidly changing. Pricing a used car correctly is crucial for both sellers and buyers.
- **Dataset:** Our dataset initially contained 371,528 rows and 21 columns, sourced from Kaggle.
- **Problem:** Accurately predict the selling price of a used car based on its features (brand, model, year, gearbox, etc.).
- **Objectives:**
  - Explore and analyze the dataset.
  - Engineer features and preprocess data.
  - Build and evaluate predictive models.
- **Challenges:** Presence of outliers, irrelevant features, and highly skewed price distribution.

# Exploratory Data Analysis

We have the following features:

**Numerical**

- price (**target**)
- powerPS
- kilometer
- monthRegistration
- yearOfRegistration
- nrOfPictures
- postalCode

**Categorical**

- index
- name
- seller
- offerType
- abtest
- vehicleType
- gearbox
- model
- fuelType
- brand
- notRepairedDamage

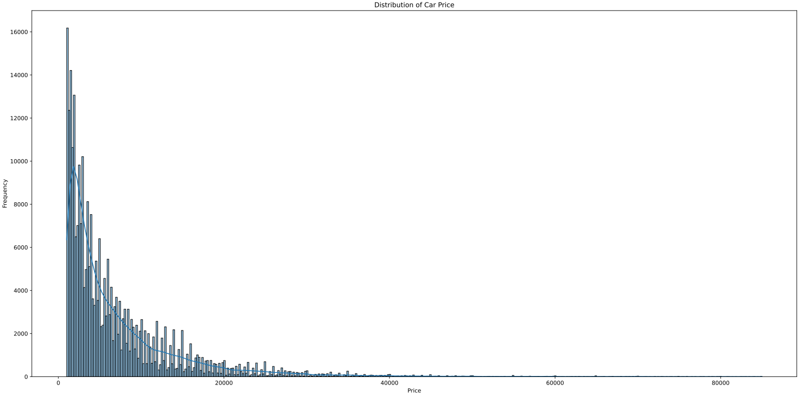
**Date**

- dateCrawled
- dateCreated
- lastSeen

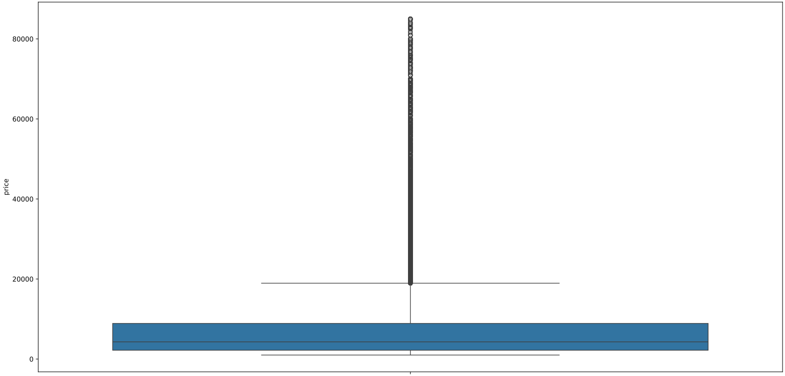
# Exploratory Data Analysis

- Kept only rows with realistic prices (€ 1,000–200,000) for effective modeling. After removal we remain with 288,023 rows.
- `offerType` and `seller` each had two categories, but one category dominated almost entirely (the minority category had only a handful of rows).
- `nrOfPictures` had only a single unique value for all entries.
- These columns were removed as they provided no useful information for prediction.

# Target Variable Analysis



# Target Variable Analysis

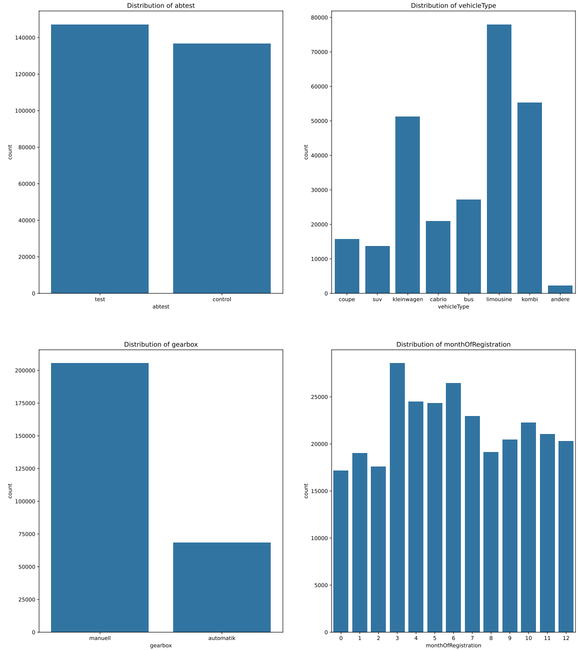


Price boxplot

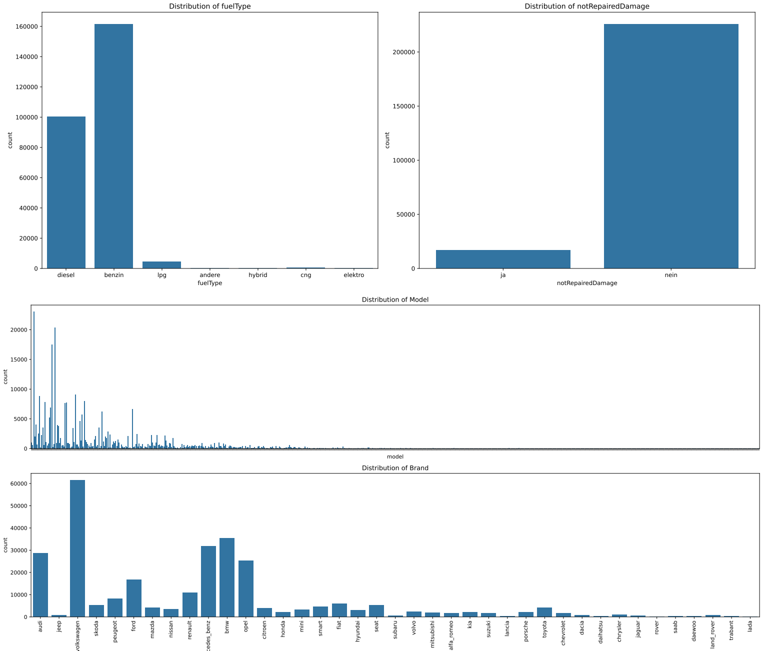
# Target Variable Analysis

- The primary target is the car's selling `price`.
- **Distribution:** Price distribution is highly right-skewed, with many extreme outliers.
- **Normality tests:** All tests (Anderson-Darling, Kolmogorov-Smirnov, D'Agostino-Pearson, Jarque-Bera, Lilliefors) indicate that `price` does **not** follow a normal distribution.
- **Skewness:** 5.14 (very high, confirms extreme right skew).
- **Implications:**
  - Model will perform best on low/average prices, but may struggle with high-price cars due to data imbalance.
  - Considered unskewing techniques such as Box-Cox and Yeo-Johnson.

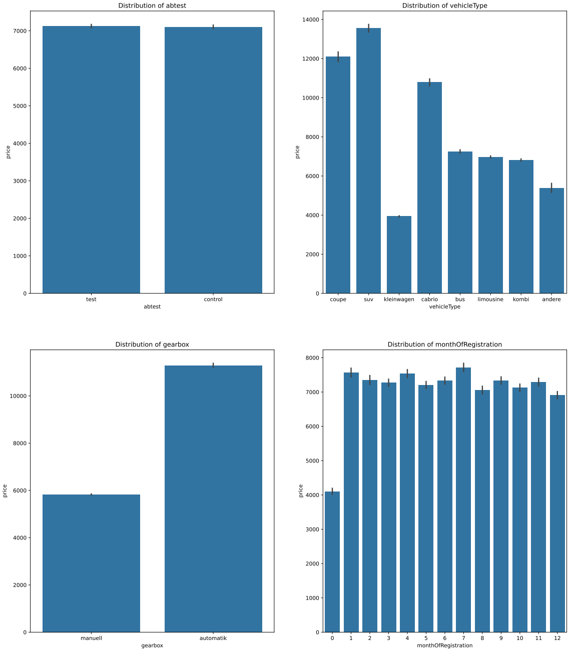
# Categorical Features Analysis



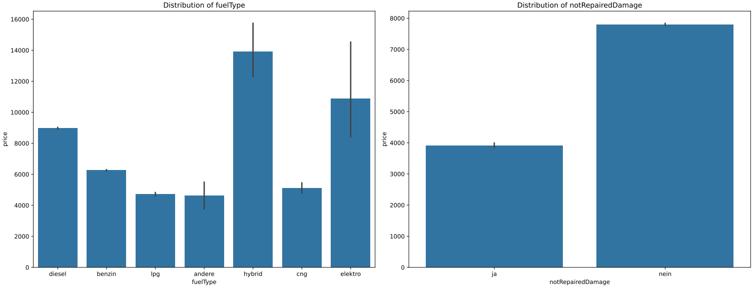
# Categorical Features Analysis



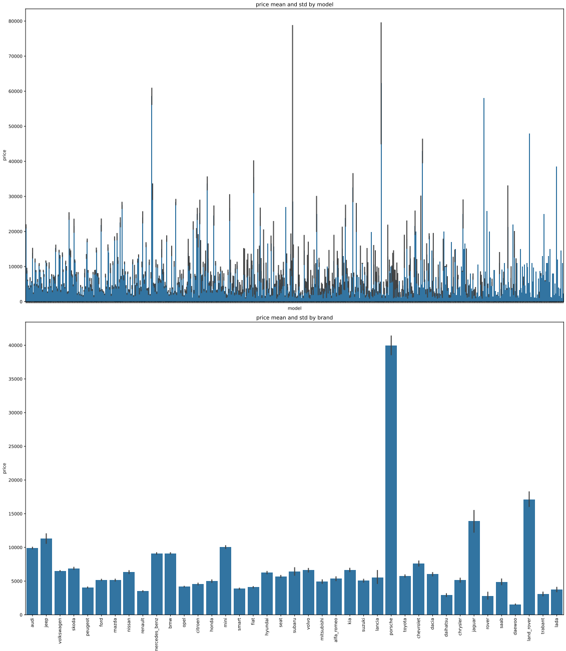
# Categorical Features Analysis



# Categorical Features Analysis



# Categorical Features Analysis



price mean and std by model

price mean and std by brand

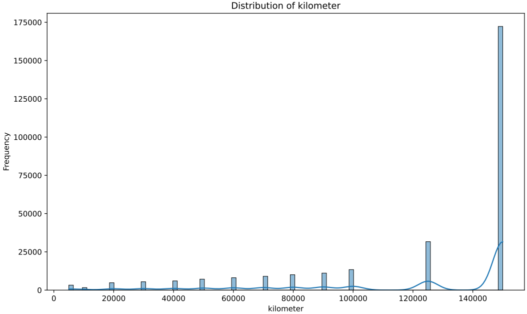
# Categorical Features Analysis

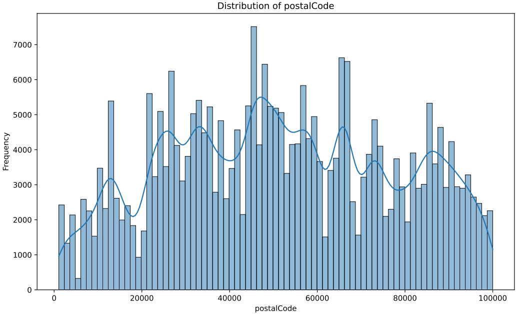
- Explored distributions of key categorical variables using histograms.
- **ABtest (`abtest`):** Both categories (”test” and ”control”) are similarly distributed, but show no influence on price.
- **Vehicle Type (`vehicleType`):** Distribution is dominated by a few types (kleinwagen, limousine, kombi), which strongly influence price.
- **Gearbox (`gearbox`):** Majority are manual, but automatic is also significant. Gearbox type shows high correlation with price.
- **Month of Registration (`monthOfRegistration`):** Mostly balanced, except for ”0” (likely missing values).

# Categorical Features Analysis

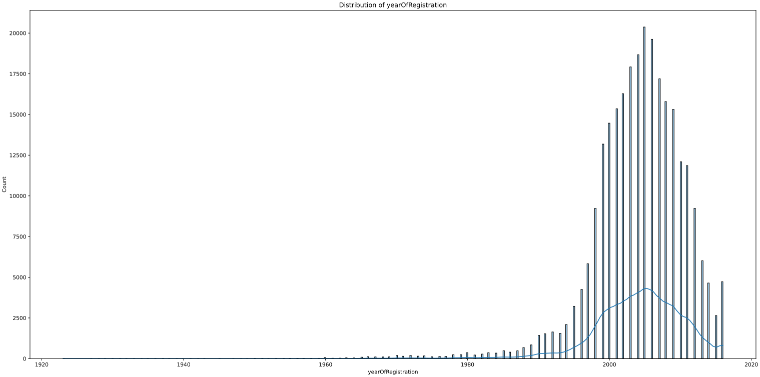
- **Fuel Type (`fuelType`):** Benzin dominates, diesel also frequent. Rare types (lpg, andere, cng) can be grouped as "other", but "hybrid" and "elektro" show distinct price patterns.
- **Not Repaired Damage (`notRepairedDamage`):** "Nein" dominates, but "ja" shows much lower prices, indicating a strong correlation with the target.
- **Model & Brand:** Both have high correlation with price and large within-category deviations, especially for brands with many models (e.g., Porsche 911 vs Cayenne).
- **General observations:**
  - Many categorical features are unbalanced; stratified sampling may be needed for fair modeling.
  - Outliers are present in every category.
  - Grouping or discarding rare categories considered to improve model stability.

# Numerical Features Analysis for All Values

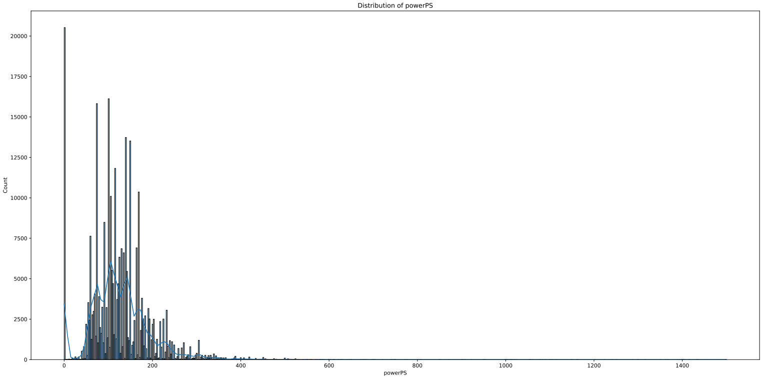




# Numerical Features Analysis for All Values



# Numerical Features Analysis for All Values



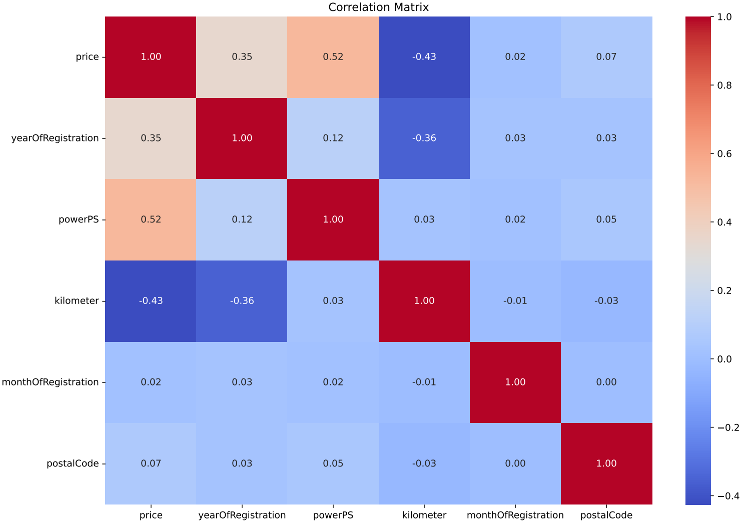
# Numerical Features Analysis for All Values

- Initial analysis revealed extreme skewness and non-normality in several numerical features:
  - `yearOfRegistration`: Skewness 106.79
  - `powerPS`: Skewness 61.90
  - `kilometer`: Skewness -1.33
  - `postalCode`: Skewness $\approx$ 0.02, no strong skew
- **Normality tests:** All tests (Anderson-Darling, Kolmogorov-Smirnov, etc.) strongly rejected the hypothesis of normal distribution for all features.

# Numerical Features Analysis for Valid Values

- After our initial analysis, we carefully checked each feature for valid value ranges:
  - For `yearOfRegistration`, we kept only values from 1920 to 2016, since `DateCreated` was never after 2016 and earlier years were invalid.
  - For `powerPS`, we kept only values less than or equal to 1500, as it is not realistic for a car under €200,000 to have more power.
  - For `kilometer` and `postalCode`, we confirmed all values were within realistic ranges.
- This cleaning significantly reduced skewness:
  - `yearOfRegistration`: -1.65
  - `powerPS`: 2.79
  - `kilometer`: -1.27
  - `postalCode`: $\approx 0$

# Correlation Analysis



Correlation Matrix

| | price | yearOfRegistration | powerPS | kilometer | monthOfRegistration | postalCode |
|---|---|---|---|---|---|---|
| price | 1.00 | 0.35 | 0.52 | -0.43 | 0.02 | 0.07 |
| yearOfRegistration | 0.35 | 1.00 | 0.12 | -0.36 | 0.03 | 0.03 |
| powerPS | 0.52 | 0.12 | 1.00 | 0.03 | 0.02 | 0.05 |
| kilometer | -0.43 | -0.36 | 0.03 | 1.00 | -0.01 | -0.03 |
| monthOfRegistration | 0.02 | 0.03 | 0.02 | -0.01 | 1.00 | 0.00 |
| postalCode | 0.07 | 0.03 | 0.05 | -0.03 | 0.00 | 1.00 |

# Correlation Analysis

- **kilometer** and **price** have a moderate negative correlation: cars with more kilometers tend to be less expensive.
- **powerPS** and **price** have a moderate positive correlation: higher power typically means a higher price.
- **yearOfRegistration** and **price** have a moderate positive correlation: newer cars are generally more expensive.
- **yearOfRegistration** is negatively correlated with **kilometer**: newer cars generally have lower mileage.
- **postalCode** shows little or no correlation with price, indicating it may not be a useful predictor.
- These correlations align with expectations and support feature selection for modeling.

# Data Cleaning

- **Set invalid values to missing:**
  - `yearOfRegistration`: Years $< 1920$ and $> 2016$ are not realistic for used cars in our dataset. Setting these to NaN ensures we only use plausible registration years for modeling.
  - `powerPS`: Values below 4.5 or above 1500 are outside the range for real vehicles (especially considering our price limits). Marking these as missing, removes obvious errors and extreme outliers.
  - `monthOfRegistration`: A value of 0 is not a valid month and likely indicates missing or unknown data, so we treat it as NaN.
- **Grouped rare `fuelType` categories:**
  - Categories like ”lpg” and ”cng” had very few entries, making it hard for the model to learn about them.
  - We combined these into ”andere” (”other”) to improve statistical reliability.
  - ”hybrid” and ”elektro” were kept separate despite being rare, as they showed distinct price patterns and might be important for predictions.

# Feature Engineering: Age Columns

- Added two new features:
  - `adLifespan`: Time (in days) between when an ad was first seen and last seen. We now know how long it took for the car to be sold. It's likely that its higher priced counterparts tend to remain listed longer than cheaper listings which quickly disappear after being seen often enough by members in related markets searching those platforms for new vehicles up for sale at any given time within certain parameters established such as location or age amongst others.
  - `carAge`: Age of the car in days at the time the ad was created.

# Feature Engineering: Age Columns

- Replaced date columns with these new features for modeling convenience.
- Checked and corrected for negative or zero values in `carAge`.
- Dropped columns: `registrationDate`, `dateCrawled`, `dateCreated`, `lastSeen`.
- `adLifespan` remained slightly right-skewed (skewness: 0.96) and did not follow a normal distribution.
- `carAge` and `yearOfRegistration` had a perfect negative correlation ($r = -1.00$), so we dropped `yearOfRegistration`.

# Train/Test Split and Handling Imbalance

- Split the dataset into train and test sets (80/20 split).
- Rare categories in `fuelType` (”hybrid” and ”elektro”) were oversampled in the training set (20x) to address class imbalance and ensure the model learns from these cases (done after data splitting to avoid leakage).
- After oversampling, shuffled the training set to mix synthetic and real rows.

# Scaling and Target Transformation

- Applied robust scaling to numerical features and checked their standard deviation:
  - `carAge`: 0.864884
  - `powerPS`: 0.886468
  - `kilometer`: 0.809288
  - `adLifespan`: 0.760899
- They have high `std` which is what we wanted.
- Transformed the target variable (`price`) using Box-Cox, as all values were positive, to reduce skewness and improve model performance.

# Handling Missing Models Values

| name | model | brand |
|---|---|---|
| Golf_3_1.6 | golf | volkswagen |
| A5_Sportback_2.7_Tdi | NaN | audi |
| Jeep_Grand_Cherokee_"Overland" | grand | jeep |
| GOLF_4_1_4__3TÜRER | golf | volkswagen |
| Skoda_Fabia_1.4_TDI_PD_Classic | fabia | skoda |
| BMW_316i___e36_Limousine___Bastlerfahrzeug__Export | 3er | bmw |
| Peugeot_206_CC_110_Platinum | 2_reihe | peugeot |
| VW_Derby_Bj_80__Scheunenfund | andere | volkswagen |
| Ford_C___Max_Titanium_1_0_L_EcoBoost | c_max | ford |
| VW_Golf_4_5_tuerig_zu_verkaufen_mit_Anhaengerkupplung | golf | volkswagen |

Table: Sample of car name, model, and brand from dataset

# Handling Missing Models Values

- The `model` column contained many missing values, making it unreliable for use directly.
- The `name` column, though messy and user-entered, always existed and typically included the brand, model, and extra info.
- **Solution:** We extracted the car model from the `name` field using the following steps:
    - Collected a comprehensive list of car brands and their possible models from the web.
    - Cleaned and tokenized the name, then generated all 1–3 word n-grams.
    - Matched these n-grams against known models for that brand using fuzzy matching.
    - If a strong match was found (above a similarity threshold), we filled in the missing `model`.

# Results of Model Extraction

- **Before extraction:** 12,148 missing values in the `model` column.
- Using n-gram and fuzzy matching based on the car brand and name, we reduced this to 4,114 missing values.
- **Limitations:**
  - 2,857 entries could not be filled because the brand was `sonstige_autos` ("other cars").
  - 1,257 entries had insufficient information in the `name` field.
- **Final step:** To ensure data quality, we dropped all rows where the `model` value remained missing.
- This resulted in a clean and consistent dataset for further analysis and modeling.

# Feature Selection and Dimensionality Reduction

- **Sequential Forward Selection (SFS):**
  - Used SFS with XGBoost to select the most predictive features.
  - Best feature set: vehicleType, gearbox, powerPS, model, kilometer, fuelType, brand, notRepairedDamage, adLifespan, carAge.
  - Features `postalCode` and `monthOfRegistration` were not selected, confirming their low predictive value.
- **Extra Trees Classifier:**
  - Attempted to compute feature importances.
  - Kernel repeatedly crashed with larger `n_estimators`, so results are unstable and not reliable for ranking.
- **Dimensionality Reduction:**
  - **PCA** not used: only suitable for continuous numerical data, but our dataset contains many categorical variables.
  - **SVD** not used: most effective for large, sparse matrices, which does not apply here.
  - Not a limitation, as we do not have too many features.

# Dataset Versions and Preprocessing Strategies

- Created multiple dataset versions (V1–V8) to explore the impact of different preprocessing strategies:
  - Imputation strategies: mean, most frequent, iterative.
  - Encoding methods: label encoding, one-hot encoding.
  - Skewness correction: Box-Cox, Yeo-Johnson.
  - Scaling: RobustScaler, MinMaxScaler.
  - Normalization: normalized whole rows with `12`
  - Compared full feature sets and subsets selected by SFS.
- These experiments allowed us to identify the best preprocessing pipeline for optimal model performance.

# Model Training and Evaluation

- Trained and compared six regression models (Random Forest, AdaBoost, XGBoost, CatBoost, KNN, Decision Tree) using 10-fold cross-validation.
- Compared each model across all dataset and target versions (original and unskewed).
- **Top-performing algorithms:**
  - CatBoost (**original target**): $R^2 = 0.856$
  - RandomForest (**unskewed target**): $R^2 = 0.853$
- **Best dataset versions:**
  - V7 (unskewed target): $R^2 = 0.777$
  - V8 (unskewed target): $R^2 = 0.776$

# Best Dataset Versions

- **Version V7:**
  - Used the optimal feature subset selected by Sequential Forward Selection (SFS).
  - Applied mean imputation for numerical features and most frequent imputation for categorical features.
  - Encoded categorical features with label encoding.
  - Scaled numerical features using RobustScaler.
- **Version V8:**
  - Built on V7 preprocessing steps.
  - Additionally applied Yeo-Johnson power transformation to unskew key numerical features: `kilometer`, `carAge`, `powerPS`, `adLifespan`.

# Hyperparameter Tuning

- After identifying the best-performing algorithms and dataset version, we applied **GridSearchCV** to tune hyperparameters and evaluate model performance using the $R^2$ score.
- However for values `cv` $> 2$ the OS killed our process for using too much memory (`google collab` also killed our process for taking way too long both for CPU and GPU).
- **Final results:**
  - The best overall model was **CatBoost** on dataset **V8**, unskewed, which achieved an $R^2$ score of **0.853** on the test set and **0.876** on the training set.
  - The best version of **RandomForest** on dataset **V8**, unskewed, achieved an $R^2$ score of **0.835** on the test set and **0.976** on the training set.
- CatBoost showed strong generalization with no significant overfitting. However RandomForest seems to have been overtrained.
- CatBoost slightly outperformed RandomForest in the final evaluation.

Introduction

Exploratory Data Analysis

Data Splitting

Data Preprocessing

Model Training and Evaluation

Classification Problem: Predicting Price Categories

# Turning Price Prediction Into a Classification Problem

# From Regression to Classification

- In addition to regression, we transformed the problem into classification by categorizing car prices.
- Used `pd.cut` to bin the continuous price into four categories:
  - low ($<$ € 5,000)
  - low average (€ 5,000–10,000)
  - middle average (€ 10,000–50,000)
  - high ($>$ € 50,000)
- This allows us to analyze model performance for distinct price segments and understand price ranges better.

# Data Preparation for Classification

- Dropped the original price column, used the new price category as target.
- Performed an 80/20 train-test split before scaling or encoding to avoid data leakage.
- Oversampled rare fuel types (”hybrid” and ”elektro”) in the training set (20×) to improve learning for underrepresented categories.
- Created new dataset versions (V9, V10) using:
  - Feature selection (from regression task).
  - Imputation for missing values.
  - One-hot encoding and dimensionality reduction (LDA) for categorical features.
  - Scaling and unskewing for numerical features.

# Classification Models and Evaluation

- Evaluated several classifiers with 10-fold cross-validation:
  - RandomForest, XGBoost, CatBoost, DecisionTree, AdaBoost, KNeighbors, Logistic Regression, GaussianNB
- Used weighted F1-score as the main metric due to class imbalance.
- Hyperparameter tuning performed for the best pipelines (RandomForest, XGBoost) with GridSearchCV.
- **Best performing algorithms:**
  - RandomForest: F1-score = **0.873**
  - XGBoost: F1-score = **0.864**
- **Best dataset versions:**
  - V10: F1-score = **0.817**
  - V8: F1-score = **0.811**

# Best Dataset Versions

- **Version V8:**
  - Used the optimal feature subset from Sequential Forward Selection (SFS).
  - Applied mean imputation to numerical features and most frequent imputation to categorical features.
  - Encoded categorical features with label (ordinal) encoding.
  - Scaled numerical features using RobustScaler.
  - Applied Yeo-Johnson power transformation to unskew `kilometer`, `carAge`, `powerPS`, `adLifespan`.
- **Version V10:**
  - Built on V9, which used the same feature selection, imputation, and robust scaling.
  - Applied one-hot encoding to categorical features, followed by LDA for dimensionality reduction.
  - Combined numerical features with LDA components.
  - Also applied Yeo-Johnson power transformation to unskew key numerical features.

# Hyperparameter Tuning

- After identifying the best-performing classification algorithms and dataset version, we applied **GridSearchCV** to tune hyperparameters and evaluated model performance using the weighted F1-score.
- However for values `cv` $> 2$ the OS killed our process for using too much memory (`google collab` also killed our process for taking way too long both for CPU and GPU).
- **Final results:**
    - The best overall model was **XGBoostClassifier** on dataset V8, which achieved a weighted F1-score of **0.875** on the test set and **0.928** on the training set.
    - The best version of **RandomForestClassifier** on dataset V8 achieved a weighted F1-score of **0.874** on the test set and **0.976** on the training set.
    - XGBoost showed strong generalization with no significant overfitting. However RandomForest seems to have been overtrained.
    - XGBoost slightly outperformed RandomForest in the final evaluation.

# Key Takeaways: Classification Task

- Transforming price prediction to classification allowed us to focus on distinct market segments.
- Proper binning, feature selection, dimensionality reduction, and robust preprocessing are essential for strong performance.
- RandomForest and XGBoost achieved the highest F1-scores, showing strong capability for price category prediction.

Thank you!
Questions?