MARMARA UNIVERSITY
FACULTY OF ENGINEERING

CSE4062 – S25

Data Science Project Delivery #3
Predictive Analytics

Car Price Prediction System

Group 8:

Muhammed Furkan Kahyaoğlu (ME) – 150420058,
furkankahyaoglu@marun.edu.tr

Özlem Demirtaş (IE) – 150320006, demirtasozlem444@gmail.com

Niyazi Ozan Ateş (CSE) – 150121991, niyaziozanates@gmail.com

Doğukan Onmaz (CSE) – 150120071, dogukanonmaz@marun.edu.tr

Şükrü Can Mayda (CSE) – 150120031, canmayda@marun.edu.tr

Instructor: Dr. Murat Can Ganiz

1. Feature Selection

Table 1: Future Selection Table

#	Feature Name	Description	Type	Mutual Information	Lasso	Information Gain
1	year	The manufacturing year of the vehicle	float64	0.5214	0.0325	0
2	price_in_euro	The price of the car in euros	float64	-	-	0.0964
3	power_kw	Engine power measured in kilowatts	float64	0.5944	0.1327	0.1404
4	power_ps	Engine power measured in horsepower	float64	0.5913	0.4306	0.0161
5	fuel_ consumption_l_100 km	Fuel consumption in liters per 100 km	float64	0.2521	0.0042	0.4075
6	fuel_ consumption_g_km	CO_2 emissions measured in g/km	float64	0.3364	0	0.0954
7	mileage_ in_km	Total distance the car travelled in km	float64	0.4123	-0.7816	0.2199
8	color_ encoded	Numerical enconding shows vehicle's color	int64	0.0447	0.0043	0
9	brand_ encoded	Numerical enconding shows vehicle's brand	int64	0.0826	0	0.0006
10	model_encoded	Numerical enconding shows vehicle's specific model	int64	0.04	0.068	0
11	fuel_type_ encoded	Numerical enconding shows type of fuel used	int64	0.0809	-0.2642	-
12	transmission_type_ encoded	Numerical enconding shows transmission type	int64	0.179	-0.2953	0.0238

In this section, feature importance was evaluated using three methods: Mutual Information, Lasso Regression, and Information Gain. These techniques helped identify which features were most relevant for predicting car prices. Features like power_kw, mileage_in_km, and fuel_consumption_l_100km consistently showed high importance. The results guided the selection of features used in regression and classification models, improving overall model performance.

2. Classification/Regression Experiments

Regression Results

In order to build an effective regression model for predicting the price_in_euro attribute, we employed both Artificial Neural Networks (ANN) and Support Vector Regression (SVR). Since the original dataset was large, we randomly sampled 20,000 instances to reduce training time and computational cost while maintaining representative data diversity.

1) Artificial Neural Network (ANN) Report

In this experiment, a Multi-Layer Perceptron (MLPRegressor) was used to predict car prices. The network consisted of two hidden layers with 128 and 64 neurons respectively. The ReLU activation function and Adam optimizer were selected to enhance training performance and convergence speed.

Data Preprocessing

Logarithmic Transformation: The target variable price_in_euro exhibited a highly skewed distribution with significant outliers. To stabilize the variance and reduce the impact of extreme values, a logarithmic transformation was applied. This approach enables the model to better learn relative changes in price rather than being dominated by large absolute differences.

Dataset Sampling: Training deep learning models such as MLP on large datasets can be computationally intensive and time-consuming. To mitigate this, a sample of approximately 20000 instances from the dataset was used. This reduced the training time significantly and allowed for more efficient cross-validation and model tuning.

Model Performance

Fold	MAE (EUR)	MSE	MAPE (%)	R^2
Fold 1	5,573.27	472,265,695.91	23.29	0.5824
Fold 2	5,041.16	200,030,655.42	18.94	0.7136
Fold 3	8,511.94	18,136,295,589.53	32.49	-18.6107
Fold 4	5,343.15	233,675,050.26	19.67	0.7384
Fold 5	5,593.38	333,170,020.30	22.25	0.6657
Fold 6	5,413.65	174,991,126.19	21.73	0.8170
Fold 7	5,866.15	367,202,949.15	23.24	0.7388
Fold 8	6,004.98	428,694,715.70	19.66	0.6816
Fold 9	7,258.86	6,084,609,544.80	28.17	-4.8285
Fold 10	5,269.67	189,381,649.27	21.30	0.7871

The ANN model exhibited generally good performance in most folds, with MAE values typically ranging from €5,000 to €6,000 and R^2 scores above 0.65, suggesting solid predictive ability. The best performance was seen in Fold 6 ($R^2 = 0.8170$), while Folds 3 and 9 significantly underperformed, with very high MSE values and negative R^2 scores. These outliers likely stem from the presence of extreme values or variance in the validation sets.

Despite the two poor-performing folds, the overall results indicate that the ANN model is effective for car price prediction, especially with additional data cleaning or robust training techniques to mitigate outlier effects.

2. Support Vector Regression (SVR) Report

Model Description

Support Vector Regression (SVR) with a Radial Basis Function (RBF) kernel was employed to predict car prices. The model was configured with a penalty parameter C=100 and an epsilon-insensitive tube of epsilon=0.1. SVR is known for its robustness in handling non-linear regression tasks, especially with smaller or moderately sized datasets.

Data Preprocessing

- **Logarithmic Transformation:** As with the ANN model, the target variable price_in_euro was transformed using the natural logarithm to reduce skewness and stabilize variance. This preprocessing step allowed the model to better generalize across different price ranges.
- **Dataset Sampling:** Due to the computational cost of training SVR on large datasets, the same sample of approximately 20000 instances used in the ANN model was also used here. This ensured fair comparison while maintaining manageable training times.

Model Performance

Fold	MAE (EUR)	MSE	MAPE (%)	R^2
Fold 1	5,149.45	437,033,862.14	22.77	0.6136
Fold 2	5,387.02	276,578,977.98	20.33	0.6041
Fold 3	5,113.87	299,758,512.47	19.60	0.6759
Fold 4	5,066.59	367,001,266.81	18.86	0.5892
Fold 5	5,109.73	237,807,494.38	20.21	0.7614
Fold 6	5,092.61	162,083,205.46	19.49	0.8305
Fold 7	5,187.10	325,472,701.76	20.77	0.7685
Fold 8	5,053.72	167,762,093.93	19.53	0.8754
Fold 9	4,929.36	179,570,140.36	20.02	0.8280
Fold 10	5,343.33	285,210,242.85	20.65	0.6794

The SVR model demonstrated consistent and reliable performance across all folds, with MAE values tightly clustered around €5,000 and R^2 scores mostly above 0.65. The model showed especially strong results in Folds 6, 8, and 9, achieving R^2 scores above 0.82, indicating a high level of explained variance.

Unlike the ANN model, SVR did not suffer from extreme outliers or negative R^2 values, suggesting better generalization and robustness. This consistent behavior indicates that SVR is well-suited for the car price prediction task, especially when stability across folds is a priority.

Comparison of ANN and SVR Models

To build a regression model for predicting car prices, we employed two approaches: Artificial Neural Networks (ANN) and Support Vector Regression (SVR). Due to the large size of the original dataset, a sample of 20,000 instances was used to reduce training time. Both models were evaluated using 10-fold cross-validation, and their performance metrics—MAE, MSE, MAPE, and R^2—were compared.

Metric	ANN (Mean)	SVR (Mean)
MAE	5,587.82	5,134.88
MSE	2.71e+09	2.74e+08
MAPE	23.00%	20.02%
R^2	0.58	0.72

SVR consistently outperformed ANN with lower errors (MAE, MSE, MAPE) and higher R^2 values, indicating better predictive accuracy and model stability.

Statistical Significance Test

A paired t-test was conducted on the MAE values across the 10 folds to determine if the performance difference between ANN and SVR is statistically significant.

- t-statistic: 2.3193
- p-value: 0.0455

Since the p-value is less than 0.05, we reject the null hypothesis and conclude that the difference in MAE between ANN and SVR is statistically significant. This confirms that SVR performs significantly better than ANN on this dataset.

Classification Results

We have made for **3 models**, Naïve Bayes, K-NN, and Random Forest Tree, **3 predictions**, making it a total of **9 experiments**.

Preprocessing

For preprocessing we converted all objects to numeric values and filled in all NaN values by their mean or by their median. After we accomplished that, we have reduced the number of unique values so that the prediction that can be made would be easier and more precise.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 251079 entries, 0 to 251078
Data columns (total 15 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   Unnamed: 0               251079 non-null  int64
 1   brand                    251079 non-null  object
 2   model                    251079 non-null  object
 3   color                    250913 non-null  object
 4   registration_date        251075 non-null  object
 5   year                     251079 non-null  object
 6   price_in_euro            251079 non-null  object
 7   power_kw                 250945 non-null  object
 8   power_ps                 250950 non-null  object
 9   transmission_type        251079 non-null  object
 10  fuel_type                251079 non-null  object
 11  fuel_consumption_l_100km 224206 non-null  object
 12  fuel_consumption_g_km    251079 non-null  object
 13  mileage_in_km            250927 non-null  float64
 14  offer_description        251078 non-null  object
dtypes: float64(1), int64(1), object(13)
memory usage: 28.7+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 251079 entries, 0 to 251078
Data columns (total 12 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   year                     251079 non-null  float64
 1   price_in_euro            251079 non-null  float64
 2   power_kw                 251079 non-null  float64
 3   power_ps                 251079 non-null  float64
 4   fuel_consumption_l_100km 251079 non-null  float64
 5   fuel_consumption_g_km    251079 non-null  float64
 6   mileage_in_km            251079 non-null  float64
 7   color_encoded            251079 non-null  int64
 8   brand_encoded            251079 non-null  int64
 9   model_encoded            251079 non-null  int64
 10  fuel_type_encoded        251079 non-null  int64
 11  transmission_type_encoded 251079 non-null  int64
dtypes: float64(7), int64(5)
memory usage: 23.0 MB
```

```
Unique values
brand: 6
model: 6
color: 6
year: 33
price_in_euro: 18081
power_kw: 512
power_ps: 521
transmission_type: 4
fuel_type: 6
fuel_consumption_l_100km: 268
fuel_consumption_g_km: 492
mileage_in_km: 71767
```

The code performs binning operations on three numerical features — price_in_euro, mileage_in_km, and year — to convert them into categorical variables suitable for classification tasks. Here's a breakdown of the steps:

1. **Price Binning**:

   The price_in_euro column is divided into five ranges: Very Low, Low, Medium, High, and Very High, using specified thresholds. This creates a new categorical column called price_category. Any rows with missing values in this new category are dropped to ensure data integrity.

2. **Mileage Binning**:

   The mileage_in_km column is similarly binned into three categories: Low, Medium, and High, based on its value range. The resulting column, mileage_in_km_category, replaces the numerical feature for classification purposes. Again, rows with missing values in this new column are removed.

3. **Year Binning**:

   The year column is binned into four historical ranges: Before 2000, 2000-2010, 2010-2020, and After 2020, and stored in the year_category column. Rows with missing values in the original year column are dropped.

Feature Selection Using Multiple Methods

The goal is to determine which features are most informative for building a classification model
Feature Selection table:

```
Feature Selection Table:
                      Feature                                Description  \
0                        year                     Description of year
1                   power_ps                  Description of power_ps
2                   power_kw                  Description of power_kw
3               mileage_in_km              Description of mileage_in_km
4   transmission_type_encoded  Description of transmission_type_encoded
5       fuel_consumption_g_km      Description of fuel_consumption_g_km
6    fuel_consumption_l_100km   Description of fuel_consumption_l_100km
7           fuel_type_encoded          Description of fuel_type_encoded
8               model_encoded              Description of model_encoded
9               brand_encoded              Description of brand_encoded
10              color_encoded              Description of color_encoded
```

Mutual Information

The mutual_info_classif function calculates the **mutual information** between each feature and the target variable. Mutual information measures the amount of shared information between the feature and the target. Higher values indicate a stronger relationship. It works well with both linear and non-linear relationships.

	Mutual Information
0	0.348771
1	0.296521
2	0.296448
3	0.244958
4	0.136824
5	0.110378
6	0.091430
7	0.047939
8	0.039531
9	0.037592
10	0.018749

Price

	Mutual Information
0	0.290181
1	0.218967
2	0.086794
3	0.072355
4	0.072323
5	0.063662
6	0.049960
7	0.031209
8	0.030748
9	0.025290
10	0.015965

Mileage

Mutual Information
0.458572
0.349177
0.147406
0.145580
0.100761
0.084577
0.070170
0.058187
0.046793
0.031922
0.027305

Year

ANOVA F-value

The f_classif function performs an ANOVA F-test for each feature to check whether the mean of the feature differs significantly across the classes in the target variable. The result is a list of F-values, where a higher F-value means the feature is more discriminative with respect to the classes.

ANOVA F-value
294.257926
42077.457508
38203.913159
33506.477296
17715.514839
3608.738339
51.916958
955.926735
1421.407696
176.954692
154.782685

Price

ANOVA F-value
490.596829
8374.486860
9088.312706
528.435289
517.418479
3758.705879
2223.369261
5.474472
134.101373
795.271099
2052.252255

Mileage

ANOVA F-value
63016.796503
6648.341386
683.429678
653.629825
3710.367359
24.190354
1684.039055
1859.163819
658.838411
198.527889
3205.222883

Year

Random Forest Feature Importance

A Random Forest Classifier is trained using the feature matrix X and encoded labels y_encoded. After training, the feature_importances_ attribute gives the relative importance of each feature based on how useful they were in the decision trees of the ensemble.

RF Importance	RF Importance	RF Importance
0.255503	0.248417	0.381035
0.113186	0.314409	0.270563
0.121756	0.059885	0.054557
0.225946	0.052132	0.055062
0.061629	0.054235	0.064196
0.066416	0.038124	0.078244
0.063269	0.083149	0.019856
0.015629	0.075526	0.031742
0.010386	0.007681	0.027904
0.031050	0.051630	0.005456
0.035229	0.014812	0.011386
Price	Mileage	Year

Model Definitions

Three classification models are defined:

- **Naive Bayes** (GaussianNB()): A simple probabilistic classifier based on Bayes' theorem.
- **k-Nearest Neighbors (k=5)** (KNeighborsClassifier): A distance-based classifier where each sample is assigned the majority label of its 5 nearest neighbors.
- **Random Forest** (RandomForestClassifier): An ensemble of decision trees trained with bagging and random feature selection for robustness and performance.

Feature Subsets

Feature sets used for training are:

- **All Features**: The full list of features.
- **Top 5 MI Features**: The top 5 features with the highest Mutual Information scores.
- **Top 8 MI Features**: The top 8 based on Mutual Information as well.

This allows testing how models perform when using only the most informative features vs. all features.

Cross-Validation Setup

- **StratifiedKFold (10 splits)**: Ensures that each fold maintains the class distribution (stratification), preventing class imbalance issues in train-test splits.

Evaluation Loop

For **each combination of model and feature subset**:

- The data is split into training and testing sets based on cross-validation folds.
- The model is trained on the training data.
- Predictions are made on the test data.
- The following metrics are computed for each fold:
    - **Accuracy**
    - **F1 Score (Macro)**: Treats all classes equally.

- **F1 Score (Micro)**: Aggregates contributions of all classes for overall performance.
- **AUC (Area Under Curve)**:
  - For multi-class, One-vs-Rest (OvR) strategy is used.
  - If predict_proba is not available, or the calculation fails, AUC is set to NaN.

Each metric is averaged over all 10 folds to get a robust performance estimate. All 9 results are show below:

```
Classification Experiments Results:
                          Experiment  Accuracy  F1-macro  F1-micro  \
0         Naive Bayes with All Features  0.498556  0.452167  0.498556
1    Naive Bayes with Top 5 MI Features  0.537982  0.485920  0.537982
2    Naive Bayes with Top 8 MI Features  0.498409  0.453210  0.498409
3          k-NN (k=5) with All Features  0.758817  0.634051  0.758817
4     k-NN (k=5) with Top 5 MI Features  0.764210  0.635357  0.764210
5     k-NN (k=5) with Top 8 MI Features  0.758379  0.633816  0.758379
6       Random Forest with All Features  0.882623  0.814855  0.882623
7  Random Forest with Top 5 MI Features  0.814580  0.759090  0.814580
8  Random Forest with Top 8 MI Features  0.867815  0.796301  0.867815

        AUC
0  0.860711
1  0.884247
2  0.860338
3  0.867596
4  0.876645
5  0.867493
6  0.971636
7  0.946391
8  0.964765
```

Price Results

```
Classification Experiments Results:
                          Experiment  Accuracy  F1-macro  F1-micro  \
0         Naive Bayes with All Features  0.345732  0.379833  0.345732
1    Naive Bayes with Top 5 MI Features  0.395899  0.425202  0.395899
2    Naive Bayes with Top 8 MI Features  0.344891  0.377827  0.344891
3          k-NN (k=5) with All Features  0.774103  0.646679  0.774103
4     k-NN (k=5) with Top 5 MI Features  0.782479  0.661056  0.782479
5     k-NN (k=5) with Top 8 MI Features  0.773745  0.645318  0.773745
6       Random Forest with All Features  0.833367  0.744605  0.833367
7  Random Forest with Top 5 MI Features  0.793794  0.689565  0.793794
8  Random Forest with Top 8 MI Features  0.825402  0.733597  0.825402

        AUC
0  0.688875
1  0.727826
2  0.687567
3  0.853536
4  0.862116
5  0.853350
6  0.937034
7  0.905133
8  0.932035
```

Mileage Results

```
Classification Experiments Results:
                              Experiment  Accuracy  F1-macro  F1-micro  \
0         Naive Bayes with All Features  0.493518  0.381785  0.493518
1    Naive Bayes with Top 5 MI Features  0.780240  0.550361  0.780240
2    Naive Bayes with Top 8 MI Features  0.492407  0.380811  0.492407
3          k-NN (k=5) with All Features  0.832336  0.601099  0.832336
4     k-NN (k=5) with Top 5 MI Features  0.832272  0.601037  0.832272
5     k-NN (k=5) with Top 8 MI Features  0.832332  0.601096  0.832332
6       Random Forest with All Features  0.918575  0.820048  0.918575
7  Random Forest with Top 5 MI Features  0.895495  0.741932  0.895495
8  Random Forest with Top 8 MI Features  0.916469  0.816174  0.916469

        AUC
0  0.885176
1  0.870265
2  0.884861
3  0.839185
4  0.839114
5  0.839181
6  0.985632
7  0.965456
8  0.984123
```

Year Results

3. ROC Curves

To visualize model performance across different classes, we plotted ROC (Receiver Operating Characteristic) curves for each classification model (Naive Bayes, k-NN, and Random Forest) based on a one-vs-rest multi-class strategy.

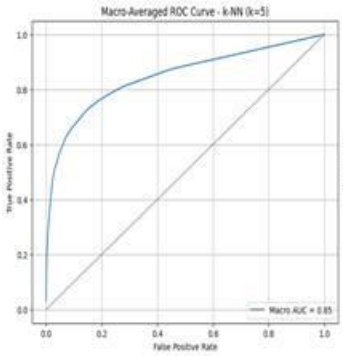
- Only one-fold (the first split) from the cross-validation is used to ensure a fair and consistent comparison.
- The target labels are binarized using the one-vs-rest approach for multi-class ROC analysis. This means for each class, it's treated as the positive class while the rest are negative.
- The predicted probabilities from each model are used to calculate:
    - False Positive Rate (FPR)
    - True Positive Rate (TPR) for each class individually.
- Then, the macro-average ROC curve is computed by averaging the TPR values across all classes for each unique FPR value.
- The macro AUC (Area Under the Curve) is also calculated to represent the overall discriminative ability of the model.

Purpose:

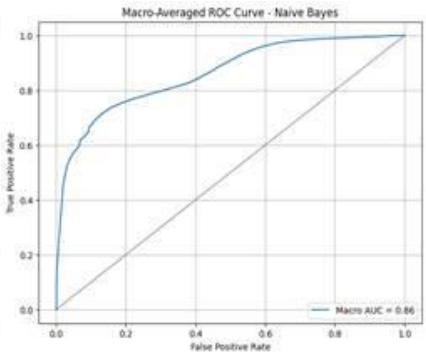
- The goal is to show how well each model can distinguish between classes across all categories, not just the most frequent ones.
- A curve that bows closer to the top-left corner indicates better performance.
- AUC values closer to 1.0 represent high performance, while values closer to 0.5 indicate poor performance (random guessing level).

What You See in the Plot:

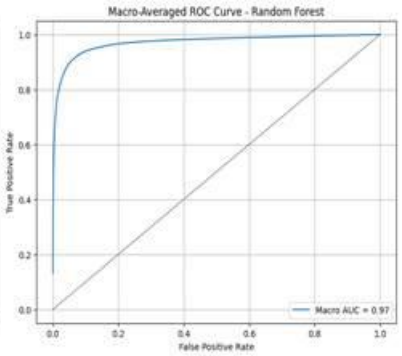
- A line for the macro-averaged ROC curve of the model.
- A diagonal dashed line which represents random classification (baseline).
- AUC score shown in the legend to compare models visually and numerically.



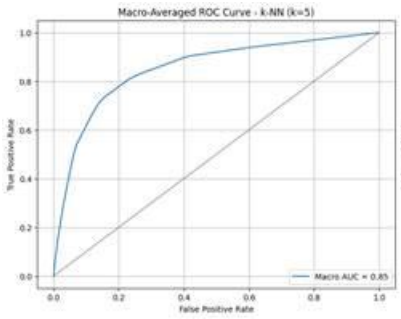
ROC Curve for KNN (Price Category)



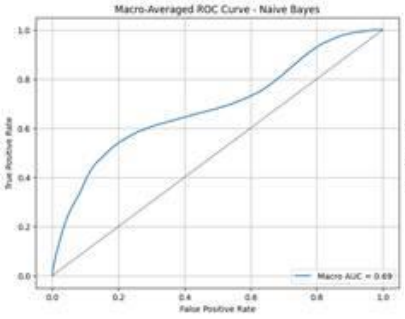
ROC Curve for Naive Bayes (Price Category)



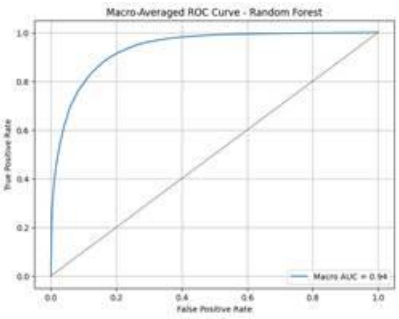
ROC Curve for Random Forest (Price Category)



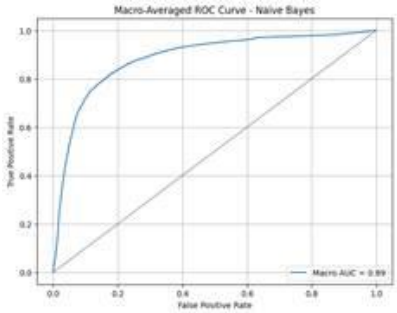
ROC Curve for KNN (Mileage Category)



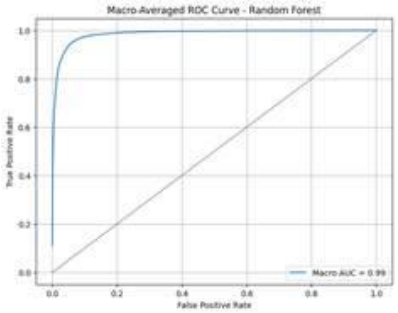
ROC Curve for Naive Bayes (Mileage Category)



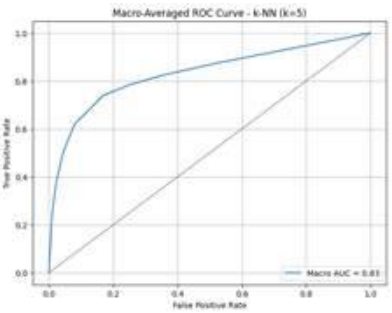
ROC Curve for Random Forest (Mileage Category)



ROC Curve for KNN (Year Category)



ROC Curve for Naive Bayes (Year Category)



ROC Curve for Random Forest (Year Category)

4. Confusion Matrices

To further analyze the classification performance of our models, we utilized confusion matrices for each task (Price, Mileage, Year) and for each model (Naive Bayes, k-NN, Random Forest) using the best-performing feature subset from the feature selection phase.

Purpose of Confusion Matrix:

A confusion matrix provides a detailed breakdown of model predictions by showing:

- How many instances of each class were correctly classified (true positives on the diagonal).
- How many were misclassified, and to which other classes (off-diagonal elements).

This is extremely useful in understanding:

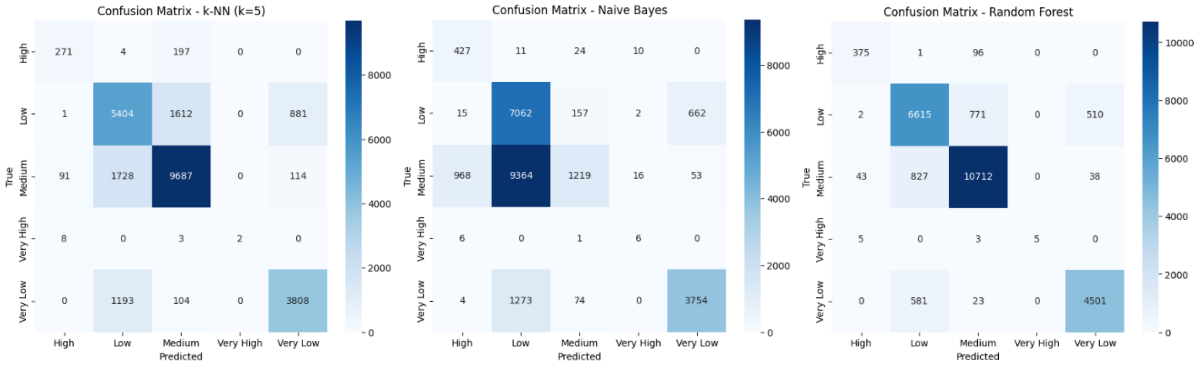
- Which classes are more easily confused with each other.
- Whether the model is biased toward certain classes.

How It Was Done:

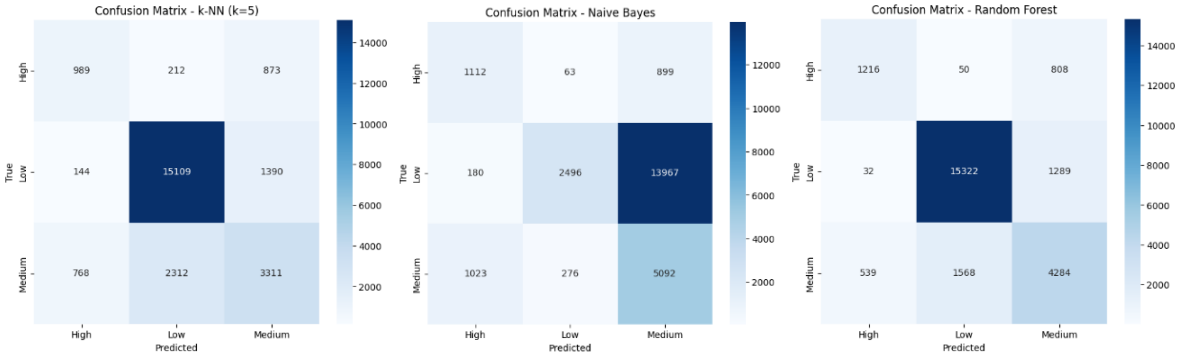
- For each model (Naive Bayes, k-NN, Random Forest), the best feature subset is used (based on earlier feature selection).
- The model is trained on the selected features and then tested on a consistent test set.
- The confusion matrix is computed by comparing the predicted labels to the true labels.
- A heatmap is used to visualize this matrix clearly.

What You See in the Plot:

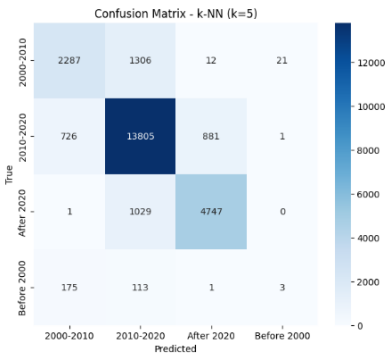
- Rows represent the true class labels.
- Columns represent the predicted class labels.
- Darker or brighter colors indicate higher numbers of instances.
- Diagonal values show the correct classifications.
- Off-diagonal values show which classes were misclassified as which, highlighting model weaknesses.



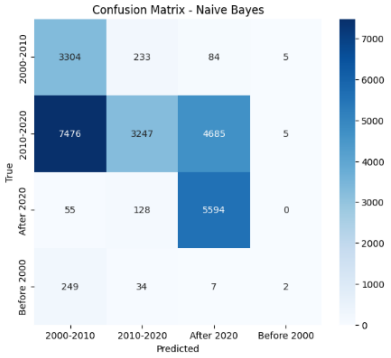
KNN (Price) Naive Bayes (Price) Random Forest (Price)



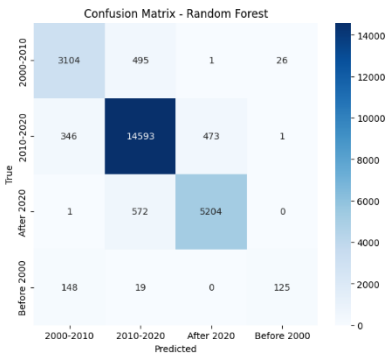
KNN (Mileage) Naive Bayes (Mileage) Random Forest (Mileage)



KNN (Year)



Naive Bayes (Year)



Random Forest(Year)

5. Statistical Significance Analysis

After running all classification experiments, this section checks whether the performance difference between the two best models is statistically significant. Here the goal is to see if the top 2 models (based on F1-macro score) are truly different in performance, or if their performance differences are just due to random chance in cross-validation folds.

Statistic Scores for Price Category:

• T-test for Accuracy: t-statistic = 16.005

• T-test for F1-macro: t-statistic = 2.112

• T-test for F1-micro: t-statistic = 16.005

• T-test for AUC: t-statistic = 2.055

Statistic Scores for Mileage Category:

• T-test for Accuracy: t-statistic = 6.127

• T-test for F1-macro: t-statistic = 5.371

• T-test for F1-micro: t-statistic = 6.127

• T-test for AUC: t-statistic = 7.979

Statistic Scores for Year Category:

• T-test for Accuracy: t-statistic = 2.876

• T-test for F1-macro: t-statistic = 0.747

• T-test for F1-micro: t-statistic = 2.876

• T-test for AUC: t-statistic = 2.527

6. Conclusion

The predictive analytics project presented in this report demonstrates a systematic approach to understanding and modeling car price and category predictions using a range of feature selection techniques, classification algorithms and regression models. Initially, feature selection methods Mutual Information, Lasso coefficients, and Information Gain were applied to determine the most informative attributes for both regression and classification tasks. Attributes such as engine power (kW and PS), mileage, and fuel consumption consistently emerged as highly informative, as evidenced by their high mutual information scores and substantial Information Gain values. These results guided the dimensionality reduction for downstream modeling, ensuring that the most predictive features were emphasized while reducing noisy data.

For the regression task of predicting the continuous variable price_in_euro, two distinct models an Artificial Neural Network (ANN) and Support Vector Regression (SVR) were trained and evaluated using 10-fold cross-validation on a sampled subset of 20,000 instances. The ANN model, which incorporated two hidden layers (128 and 64 neurons) with ReLU activation and the Adam optimizer, achieved mean absolute error (MAE) values around €5,587.82 and a mean of 0.58. Although its performance was solid in most folds, it suffered from significant variability, including two folds with negative R^2 scores due to extreme validation set variances. In contrast, the SVR model with an RBF kernel ($C = 100$, $\varepsilon = 0.1$) demonstrated more stable performance, achieving a mean MAE of €5,134.88 and a mean R^2 of 0.72, without encountering negative values in any fold. A paired t-test on the MAE values confirmed that SVR's superior performance was statistically significant ( ), validating that SVR is better suited for car price prediction in this dataset.

In the classification experiments, three algorithms Naive Bayes, k-Nearest Neighbors (k-NN, k=5), and Random Forest were applied to classify binned versions of price, mileage, and year categories. After preprocessing (including binning and handling of missing values), feature selection was again employed to identify top subsets for training. Evaluation metrics across 10-fold cross-validation included accuracy, macro-averaged F1, micro-averaged F1, and ROC AUC. The Random Forest classifier generally outperformed the other models, achieving higher macro F1 scores and ROC AUC values across all three target categories. This superior performance was further supported by confusion matrix analyses, which showed that Random Forest had fewer misclassifications and clearer separation among categorical classes. Statistical significance testing (paired t-tests) between the two best-performing models for each category confirmed that, in most cases, Random Forest's improvements were significant. These findings suggest that ensemble-based methods like Random Forest provide robust and reliable classification for categorical predictions in this domain

In the end, the comprehensive evaluation illustrates that carefully selected features, combined with robust modeling techniques, lead to effective predictive performance. For regression, SVR is recommended due to its consistent error metrics and higher explained variance, while for classification tasks, Random Forest stands out for its ability to accurately distinguish between categories. Future work could involve expanding feature engineering such as incorporating interaction terms or additional external data and exploring hyperparameter

optimization techniques to further enhance model performance. Additionally, integrating ensemble methods that combine SVR and neural networks or stacking multiple classifiers could be investigated to potentially yield incremental gains. By prioritizing both statistical rigor and practical interpretability, this project lays a solid foundation for deploying predictive models in real-world car pricing and classification applications.