Contents

1 Problem Statement

We are given a small dataset in the form of a **csv** file, with 10k records. There are 8 attributes in total, both categorical and numerical. The goal of this analysis is to build an initial ML model to predict the Price.

The mainly used libraries are: **sklearn, pandas, numpy, xgboost** and **seaborn**

2 Exploratory Data Analysis

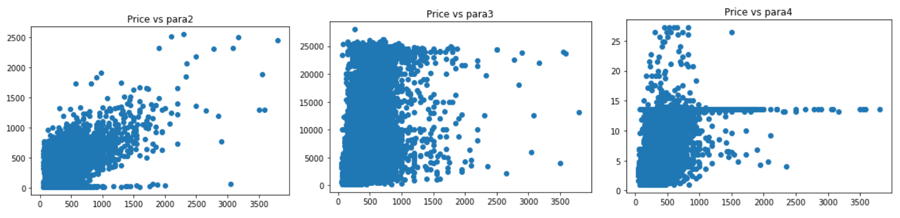
In this section we aim to get a better understanding of the data by looking at each feature and how they interact with each other. We will also explore the presence of missing values and outliers/noise, which will be handled in the next sections. Statistical data about the attributes is summarized in the table below:

	loc1	loc2	dow	para1	para2	para3	para4	price
unique	12	107	7	13	1016	4359	243	932
mean	3.79	42.46	-	1.37	447.38	9547.98	8.45	433.73
mode	2	21	Wed	1	16	24000	13.6	400
range	9	99	-	337	2538	34582	26.2	5649.27

We have 4 categorical features: $loc1, loc2, dow, para1$. By looking at statistics and number of values for each feature, we learn the following:

- dow contains the days of the week.
- $loc1$ contains 12 unique characters, 10 of which are digits and 2 literals ('S' and 'T').
- $loc2$ has the most unique values (107) but we will consider it categorical as there is a direct correlation with $loc1$: the first character of $loc2$ always corresponds to $loc1$. All the values are numerical except: '0C', '0T', '0B', '0N', '0L', 'TS', 'S6'.
- $para1$ has the biggest range (337) which leads us to think there might be some outliers/wrong values in this column, as we have a total of 13 unique values and the mean is really small (1.38)

We have 4 numerical attributes (including the target variable): $para2, para3, para4, price$. $para2$ and $para3$ and integers, while $para4$ and $price$ and real numbers. We employ visual methods for exploring these features, specifically scatter plots and histograms. In **Figure 1** we create scatter plots with the dependent and independent attributes. Based on these alone, it is hard to tell if there is a linear relationship between each attribute and price as there is a lot of noise present.



(a) Price vs. para2 (b) Price vs. para3 (c) Price vs. para4

Figure 1: Plotting the price against the independent features.

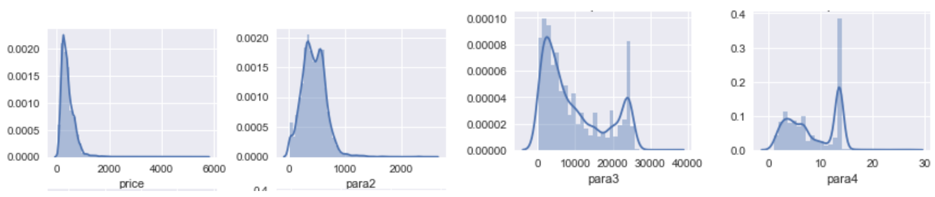


Figure 2: Distribution of numerical attributes.

We are also interested in the distribution of the numerical attributes. In **Figure 2** we see that $price$ and $para2$ have their distributions skewed to the left, while $para3$ and $para4$ display bimodal distributions.

3 Data Preprocessing & Transformation

In this step we handle the data that is not relevant, the outliers and encode the categorical data. We also prepare the data by applying transformations so that it is more suitable for the chosen estimators.

We learned in the previous step about the relationship between $loc1$ and $loc2$. As we lack knowledge about the meaning of these attributes, we will drop the $loc1$ column as the values are already encoded in $loc2$ and the data seems redundant.

We have in total 7 rows which contain literals in $loc2$. We will drop these rows as they represent a small portion ($< 1\%$) and the we again lack the domain knowledge.

In 13 cases out of 10000, data entries occured during the weekend (10 on Sat and 3 on Sun). These also seem like exceptions so we will remove them.

In **Section 2** we disscussed the presence of outliers, which will be further explored here. For outlier detection we cannot use parametric methods (such as z-score) as the distribution of the numerical variables are not Gaussian. We will need to apply non-parametric methods (like DBSCAN or Isolation Forest). In this case, **Isolation Forest** is preferred because of the following reasons:

- It is an effective method when value distributions can not be assumed
- It has few parameters, which makes this method fairly robust and easy to optimize.

This method identified 997 outliers. We remove the rows which were marked as containing outliers, as the plot of the cleaned data looks reasonable.

$loc2, dow$ and $para1$ are categorical variables, so we will use OneHotEncoder on them. We cannot use LabelEncoder as it encodes the items with integers which might afterwards display linear correlation.

MinMaxScaler is applied on the dataset as it essentially shrinks the range such that the range is now between 0 and 1. This is done after removing the outliers (as the MinMAxScaler displays sensitivity to noise).

4 Training and Testing the ML Estimators

The dataset is split into a train and test set, which will be used for training, respectively, evaluating the estimator. The test set (which consists of 33% of the data) will only be used at the end, to check how well the estimator generalizes. A test harness with 10-fold cross validation is used and root mean squared error measures (RMSE) are used to indicate algorithm performance. RMSE is preferred because it shows how close the observed data points are to the models predicted values.

The following models are fitted:

Linear Regression. Experiments were done with Linear Regression, but the results were inconclusive. Initially, $price$ was skewed to the left, so log_{10} was applied to normalize it. StandardScaler was used for the numerical features (even if this scaler assumes that the data is normally distributed). The model generalized well, but the residual plot displays severe heterodasticity, which implies this model is not suited for this dataset. As the data is non-linear, **ElasticNet, Ridge and Lasso** regressions were not used.

Kernel Ridge Regression. It has the ability to learn a non-linear function by employing the kernel trick. It uses squared error loss, combined with l2 regularization and it is also robust to noise. In terms of performance,it generalizes quite well, having a RMSE of 126 for the test set.

Support Vector Regressor. Similar to Kernel Ridge Regression, it can learn a non-linear function, but instead uses ϵ-insensitive loss function. It also generalizes well, but has a poorer performance than Kernel Ridge Regression, reaching approximatively 148 in RMSE for the test set.

XGBRegressor. This estimator has a proven track of dealing with irregularities of data as it is highly sophisticated and powerful. It has an efficient way of dealing with categorical data, which in our case is necessary. GridSearch is used for tuning the parameters. As a result it also has the best performance among the fitted estimators, with a good generalization and 111 of RMSE for the test set.

The performance of all the estimators is summarized in the table below (the values may vary as the train-test split is done with shuffle):

	Linear	Kernel Ridge	Support Vector	XGB
Training	121	122	143	110
Test	128	126	148	111

Overall, I believe these estimators need more tuning and the features more cleaning and engineering, as even the smallest RMSE (given by XGBRegressor) is quite high and in a real-life scenario it would be misleading if the predicted values are over- or under- the actual values by at least 111.

5 Future Work

A number of improvements can be brought to this approach:

- more feature engineering, especially focused on the interaction between attributes
- identify outliers with DBSCAN and Isolation Forest and remove them based on both approaches
- tune all estimators and try more models
- experiment with combinations of features, instead of considering all or feature reduction.