**Answer 1. (USE CASES)**

1. Determining price of a car
2. Get a list of (few) relevant features for initial prediction of price
3. Build a better back-end model to more accurately determine the correct price of the car.
4. Design model for predicting risky cars apart from price. If any of the features available in the dataset (apart from price which is already being used) also predict the risk value well, then they too can be used for (more optimal) initial risk setting.
5. Design model for predicting the normalized losses for assisting insurance related decisions for various cars.
6. This data could be used (until the time there is no clicks information) for SEM (search engine marketing) by firms selling used cars. Based on the popularity (say ratio of price/mpg as one example) of a car, the ad could highlight that car when someone searches for some keywords related to Used Cars, e.g., “used BMW 2012 318d” (and similar keywords)

1. Predict City MPG (or mpg in general) using other features in the dataset. Can be used in cases where this info is not already present or needs to be predicted.
2. Matching buyers and sellers (including recommendations) based on specific criteria such as engine strength, car make and type, etc.

If a date of measurement value was present in this dataset, then several other use cases could have been thought of, including a proper depreciation model of car prices

**Answer 2**

I believe that for Auto1, the most useful case could be use case 1 (both 1(a) and 1 (b)) because this is a core feature of Auto1’s business.

**1(a)** is important because it holds value at the front-end for potential customers. If we give an accurate price to end-users up-front, then it is good for the business in the long run. This is because an initial accurate price would mean less disgruntled customers (in case of sharp reduction of price upon further evaluation) later on.

**1 (b)** is obviously important because Auto1 wants to predict the price as well as possible for setting the right profit margins and thus maximizing profits

**Answer 3**

Code.py

**Answer 4**

**PREPROCESSING**

* First of all, even without exploring the data and effect on results, I was tempted to drop the “normalized losses” column, almost as a matter of principle. Alas, I cannot do that. If normalized losses are available to the potential buyers, then Auto1 also has to consider them. Because even though it’s probably a classic case of ‘correlation does not imply causation’, we still have to consider it because potential buyers would do the same.
* On the other hand, from what I understand, column 1 “symbolling”, can be ignored, as that comes post the price.
* Also, I delete all rows which have a “?” in them. My reason for doing that is because I do not believe that these values could be imputed. They occur most frequently for normalized losses. Still, we are left with a healthy enough data set size. Of course, an alternative could have been to simply drop the “normalized losses” column. That would leave us with almost the entire data set (minus only a few rows)
* I one hot encode all the categorical variables apart from **num-of-cylinders.** This is because it can be seen that the number of cylinders is an ordinal variable and should not be turned into a nominal variable. Indeed on checking later on, my hunch was proved right, as number of cylinders with ordinal values gives better prediction results than as a nominal variable.
* I did not scale the data yet, but the data is scaled later on. This was done to ensure that the varying scales in the different features are not impacting the prediction models.

**MODEL for 1(a)**

*Note for both models 1(a) and 1(b), I have also included the results in a file called Results. Alternatively, you may also run the code.*

* For 1(a) use case, for important features to provide to end-users at the start, I applied Recursive feature elimination (RFE) on a subset of features. I did not choose the features that Auto1 is already using (obtained from site: <https://www.wirkaufendeinauto.de> provided by Ms. Bianca Alves during HR interview). I instead focused on features that a user can potentially get from her car dashboard and/or a quick Google search (in case reading technical manual is not possible due to laziness or because its not available).

The features that I used were: **mpg (both highway and city), peak-rpm, horsepower, compression-ratio, stroke and bore.**

After RFE, the important features turned out to be: MPG, horsepower, stroke and compression-ratio. For starters I can recommend MPG to be asked of users, as in most cars they constantly see it on the dashboard.

Instead of Recursive Feature elimination (RFE), I could have used PCA. From the obtained components, if some features were loading the component more than others, then those could be candidate features. In my experience, however, I haven’t had much luck with this technique in the past. Even though this is mentioned a lot by people.

Alternatively, I could have used a simple correlation coefficient value between the target variable and the feature of interest. But often, that turns out to be inadequate and a combination of features is required, which is what RFE does.

**MODEL for 1(b)**

* My first instinct was to use the KNearestNeighbourRegressor (KNNR), but even without evaluating it, upon reflection I realized that it is the wrong model for this problem. Here each feature (especially the categorical ones) have great importance of their own, and a difference of one feature is enough to change the results all together. Therefore, instance similarity which is what KNNR aspires to, does not make sense here. I ended up using Kernel Ridge initially. After evaluating it, I ended up trying several different algorithms. See below:

**EVALUATION**

* On first sight, the results of Kernel Ridge according to the R squared measure, are promising (0.92). However, for our business case (Auto1), with price of used cars being the target variable, this can be a **misleading** metric of evaluation. This is because even a slight difference (say of a few hundred Euros) is quite important and can lead to the difference between profit and loss.

Therefore, I opted for the mean absolute error. It comes to 1097 (Euros). That is not such a small amount. An error of a thousand Euros (either positive or negative) is not such a promising result. I could also use Mean squared error (MSE) but I believe in this case, it is not necessary since MSE penalizes larger errors whereas a difference of 250 euros and that of 500 euros in the predicted vs actual price of a car, is just that. There is no need to penalize 500 exponentially as compared to 250. Of course, MSE can also help if the model is mis-predicting by huge amounts, but is not the case here. See attached graph called PredictedvsActual.

Also, MAE is easier to interpret. The bottomline through MAE is: We are missing by around 1100 Euros on each car. End of story.

Anyhow, just to confirm, I ran K-fold cross validation on the data, using many models, to see which one performs the best over many runs. I also scaled the data in this step, just to be sure that varying scales in the data are not having an adverse effect on the prediction models. At first I chose the kfolds number as 3 because the data is 159 rows. This would have meant that nearly one-third of the data could be used as “test data” in each validation run. However, while that maybe true, that also means that K-fold validation is performed only 3 times. I finally settled on 10 which seemed to give enough validation runs, while still having some rows in the “test” data.

It can be seen from the results that scaling clearly helps. It can also be seen that ensemble methods outperform other (non-ensemble) models.

**HYPER PARAMETER SEARCH**

Out of all the ensemble methods, Gradient Boosting Regressor (GBR) and Extra Trees Regressor are the highest performing (lowest MAE). Therefore, I wanted to fine tune one of them further, by searching for better hyper parameters. According to Scikit documentation, the parameter n\_estimators is the number of trees in the forests. This is the easiest, most non-complicated parameter to tune as up-to a number, the more the trees, the more the marginal performance gain.

**Model Improvements If I had more time**

1. Do some research and try to find out the general car prices. Purpose being that I would like to turn Car + model + body type into an ordinal variable rather than nominal variable that they currently are.

Meaning if a car is a BMW + Sedan then its higher in the hierarchy than a Renault + Sedan (for instance). If the end price contradicts this, then its because of other reasons. But at least for different combinations we can place an ordinal value. I strongly believe having this as ordinal rather than nominal would give better results.

One reason for the above also is that when I applied RFE to the entire space of features, the categorical features did not emerge as important ones. I do believe, in their current form, they are not as important as they could become (if turned into ordinal values)

1. If I had more time, then I would have tried to fine tune the GBM and the Extra Trees regressor and also other ensemble methods, searching for different values for various hyper parameters, individually and in combination. I would ideally like to do a parameter sweep of all tunable parameters for multiple candidate algorithms. Though this seems like a brute force approach (it is), and endangers over-fitting, but such detailed fine-tuning can come in handy especially when done in the presence of appropriate validation techniques such as cross-validation. Of course, if there is more data, then this approach makes more sense. It can lead to more computational time in running the algorithms, but is worth it.
2. Would ask for more data including date of registration + kilometers covered. I think that’s the most important feature which is currently missing from this data set, and that leads to mis-predictions.