ML Steps

This is a data science project in which the data scientist were tasked with developing a model that can predict the price of the products been sold by the organisation, such that profit in maximized and dealers are also happy. To complete this task a series of steps were taken which will be surmarized in this report, this steps are as follow:-

(1) Data loading

(2) Data joining

(3) Data cleaning

(4) Data preprocessing

(5) Data splitting

(6) Model training

(7) Model evaluation.

**Data loading:** This process is a two way process, one in which the task was completed both in sql and using a python package called pandas. In this process, 5 datasets were utilized, ‘bodytype.csv’, ‘categories.csv’, ‘condition.csv’, ‘listing.csv’ and ‘trueprices.csv’. In using the pandas library, this datasets were all loaded seperatly and then joined on their ‘id’ column producing a new dataset, To implement this same process using sql, some addition steps had to come into play, first a postgres database was created after whcih different corresponsding tables were also created in the database to mimick the 5 datasets given for the project (command to recreate the same steps are available in the TODOs.docx file that comes with the project).

**Data joining:** After the database creation step was complete, the ‘.csv’ datasets were loaded into their respective tables in the database and a join operation to mimick that which was implemented in pandas was executed (command to recreate this step is also in the ‘TODO.docx’ file that comes with the project).

**Data cleaning:** Once the data joining step was completed, there was a need to clean the new dataset created, Sklearn a python data science library was heavily utilized, Using sklearn, Different data **Transformers** were all created for the purpose of data cleaning, this process was immedietly followed by the creation of a python data pipeline class called **AutochekDataProcessorPipeline**, this data pipeline is responsible for housing all the methods and steps involved in loading, processing and saving dataset for training.

**Data proprocessing:** This step is mainly covered using the class created in step 3 (Data cleaning), going through this step was as easy as running a method ‘.process\_dataset’ after which every other underlining operations were completed.

**Data splitting:** Just like in step 4 (Data preprocessing) this step is mainly covered using the class created in step 3 (Data cleaning), going through this step was also as easy as running a method ‘.split\_data’ after which every other underlining operations are completed.

**Data trainnig:** To train the model to predict the prices of product, Two main python scripts were utilized, ‘model.py’ which houses a python class which is responsible for initializing, loading and training model, second script is a ‘train.py’ file, this file contains codes to specify dataset to train with, model type to use and it also trains the model. Models are saved in a ‘model’ folder after training is complete. To finalize this step 6 different models were also created for this project,

(1) Support vector regression (svr)

(2) Linear regression (lr)

(3) Lasso regression (la)

(4) Ridge regression (rg)

(5) Stochastic gradient descent (sgd)

(6) K-nearest neighbors (kn)

**Data testing:** Testing the model was the easiet part of the project, every method,function,dataset and even trained model were already in place, all that was needed was a script to load and test the model, to do this a ‘test.py’ file was created which utilized all the available tools created for the project. Below are the Root mean square error test results for the 6 different models created.

**Test Results**

**SVR Model**

**Train rmse: 1266052208.7204685**

**Test rmse: 35565281.48782029**

**Validation rmse: 6953501828.898531**

**LR Model**

**Train rmse: 43667074.335155904**

**Test rmse: 53553017.92216383**

**Validation rmse: 6953385169.828215**

**LA Model**

**Train rmse: 43668724.18693882**

**Test rmse: 34553442.1793325**

**Validation rmse: 6953335683.047222**

**RG Model**

**Train rmse: 850810516.4058534**

**Test rmse: 76500747.34652285**

**Validation rmse: 6951859845.2142105**

**SGD Model**

**Train rmse: 1229562771.0914953**

**Test rmse: 68498465.92047666**

**Validation rmse: 6951550214.118547**

**KN Model**

**Train rmse: 1013096329.3713635**

**Test rmse: 34669730.316305675**

**Validation rmse: 6953357277.26849**

**Finally Thought:** From the test results so far, the model that had a high performance on all the datasets is the lasso algorithm.