Vehicle Maintenance Cost Prediction Project Report

For

Machine Learning Course

Saher Mohamed Abdelaziz	202000706
Mayer Mamdouh Noshy	202002127
Mohammed Ashraf Abdulsami	202002700

Under supervision of

Dr. Ghada Ahmed Khoriba
Eng. Amira Tarek

Problem Summary

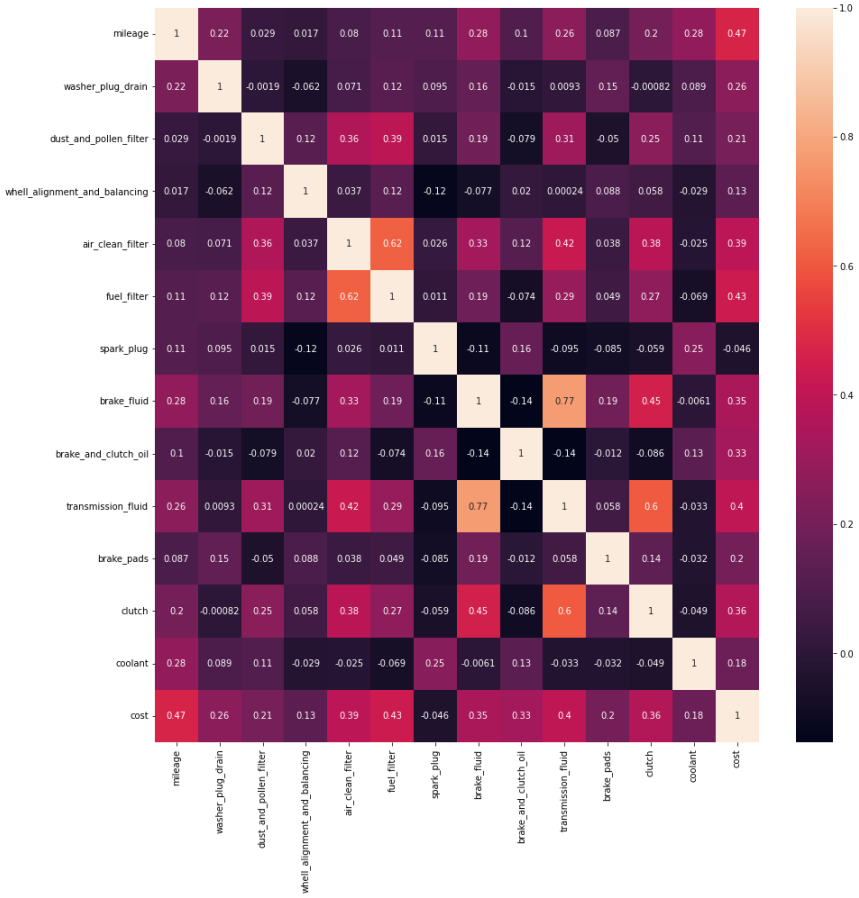
The data we used in the project presents the parts of vehicles that need maintenance, the mileage on each car record, and the total cost of the maintenance process. Inside the data, the vehicle parts that need maintenance are flagged with 1 in each record accordingly. These parts are [oil filter, engine oil, washer plug drain, dust and pollen filter, wheel alignment and balancing, air clean filter, fuel filter, spark plug brake fluid brake and clutch oil, transmission fluid, brake pads, clutch, coolant].

Methodology

Preprocessing

Data cleaning

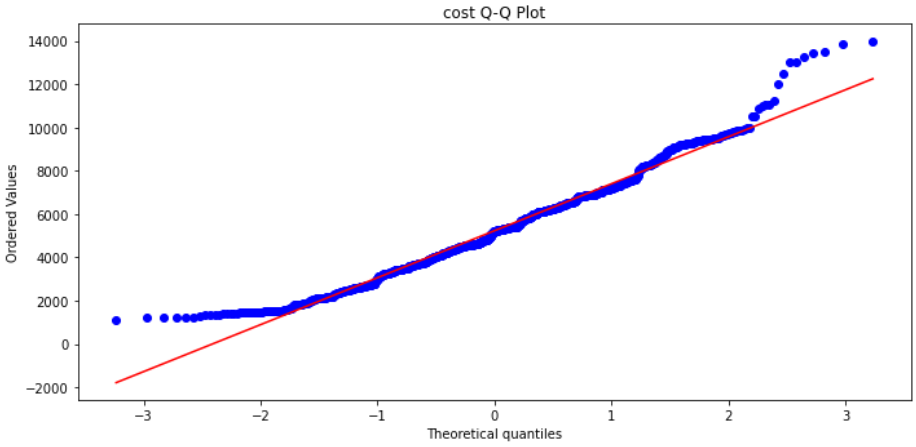
For the most part, the data was clean, except for a couple of columns that had the same values for all the records, so these columns were dropped out of the data frame.
These columns were: [oil filter, engine oil, washer plug drain, vehicle type]. Other columns did not have values at all like the [region].



Data analysis

Various visualization functions were used to analyze the data such as correlation analysis using seaborn's heatmap for the correlation between the dataset features.

Q-Q plot was used to check the normal distribution of the data for both the “cost” and the “mileage”.



Data processing

Data Shape and Feature Selection

In order to make the data usable by our models, columns with text values had to be removed and replaced instead by columns to represent each of the values that existed inside the column with a binary value 1 meaning that the record is flagged for having such property or not. One hot encoding was performed on each of the columns: [brand, model, engine_type] as each of them did not have numerical data.

The selection of features was simple as the data had only one column that can be considered as the target and all the other columns as features. The target column in this case was the “cost” column.

Data scaling

Since the data had a wide range of values for the “mileage” column for example, the data had to be scaled in order for the models to better work with it and to minimize the error resulting during the training process for each model.

The model used in the scaling process was the sicketlearn’s “`StandardScaler`” class which scales by removing the mean and scales to the unit variance. The method used was “`fit_transform`”. The scaling process is divided into 2 parts. First, the function finds the mean of each column/feature and the standard deviation of each in a process known as fitting. It then starts subtracting the values of each column from the mean and then divides it by the standard deviation from before in a process known as the transformation. The data then will start from 0 as an initial value, and have a total standard deviation of 1 in each column.

ML Models

Lasso

Lasso is a regression model but it uses L1 Regularization. The model has many hyperparameters but the 2 primary hyperparameters are *alpha* and *max_iteration*. *alpha*, which is a hyperparameter , controls the strengthening of the regularization . *max_iteration* is the max number of iterations needed in the model. There many ways to calculate the optimal *alpha* one of them is cross validation technique which uses cv to calculate the *alpha*. Also the cv is a hyperparameter and can be calculated using the "LeaveOneOut" function. The main purpose of the lasso model is to fix overfitting so it uses regularization and alpha to minimize the error.

Xgboost

(extreme gradient boosting) this model utilizes decision tree ensembles in an iterative manner so that many trees work to find the best weights for the model in many iterations in order to produce a correct prediction. The trees combined work to correct each other.

The parameters of the model are:

booster which have the values *gbtree*, *dart*, and *gblinear*. The one that worked best with the data was the *dart* booster as it produced the best accuracy compared to the other two boosters.

max_leaves, *max_depth*, *learning_rate* are hyper parameters for the model. They were changed multiple times to produce the best accuracy.

alpha which is the regularization function L1 "Lasso regularization". Can be considered as another hyper parameter, as it was changed to multiple times to produce the best accuracy of the model.

Decision Tree

A common machine learning approach for classification and regression tasks is decision trees. They can handle categorical and numerical data and are simple to understand and visualize. The three major hyperparameters of a decision tree are max_depth, min_samples_leaf, and min_samples_split. The setting of Max_depth regulates the tree's maximum depth, which might limit model complexity and aid avoid overfitting. The minimal number of samples needed to construct an internal node and a leaf node, respectively, are controlled by the variables min_samples_leaf and min_samples_split. These hyperparameters can aid in preventing the model from memorizing the training data and enhance its ability to generalize to new data. To establish a fair balance between model complexity and performance, it is essential to select optimal values for these hyperparameters.

Random Forest

An ensemble learning technique called random forest makes predictions by using several different decision trees. A sophisticated algorithm noted for its excellent accuracy and resistance to noise and outliers in the data, random forest is capable of handling both regression and classification problems. A random forest tree's primary hyperparameters are the number of trees in the forest (n_estimators), the maximum depth of each tree (max_depth), and the bare minimum amount of samples needed to split an internal node (min_samples_split). The lowest number of samples needed to generate a leaf node (min_samples_leaf), the maximum number of features to take into account when looking for the optimum split (max_features), and the criterion used to assess the quality of a split (criterion) are other crucial hyperparameters.

Grid Search

GridSearchCV is a function from the scikit-learn library that conducts an exhaustive search over a designated hyperparameter grid to identify the ideal set of hyperparameters for a particular machine learning algorithm. A model's performance can be considerably impacted by hyperparameters, which are movable parameters that are established before a model is trained. By comparing the model's performance across all conceivable grid-specified combinations of hyperparameter values and utilizing cross-validation to prevent overfitting, GridSearchCV automates the process of fine-tuning these hyperparameters. In the end, a set of ideal hyperparameters that may be used to train the model on the entire dataset are produced. With its widespread use in both academia and business, GridSearchCV is a potent tool for enhancing the performance of machine learning models.

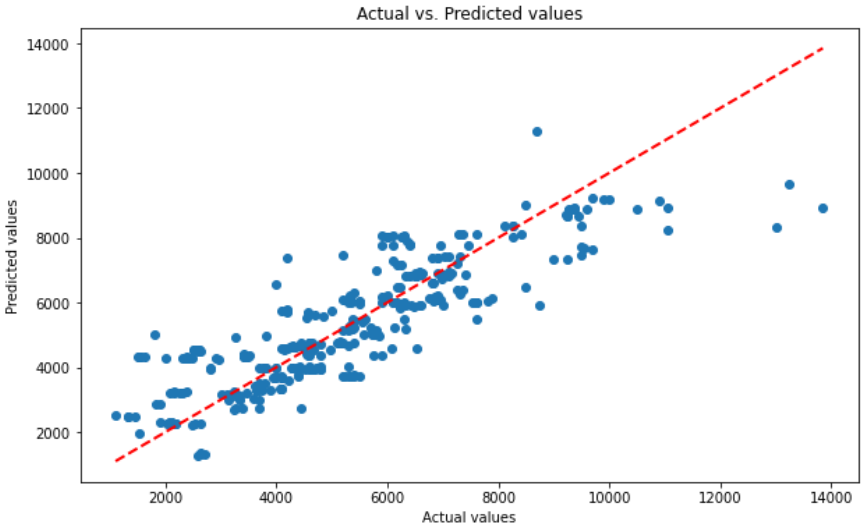
Achievements and skills

- Practical experience with machine learning models
- Understanding the models which we used in the projects and how they handle different kinds of inputs
- Knowing what models best work with our data

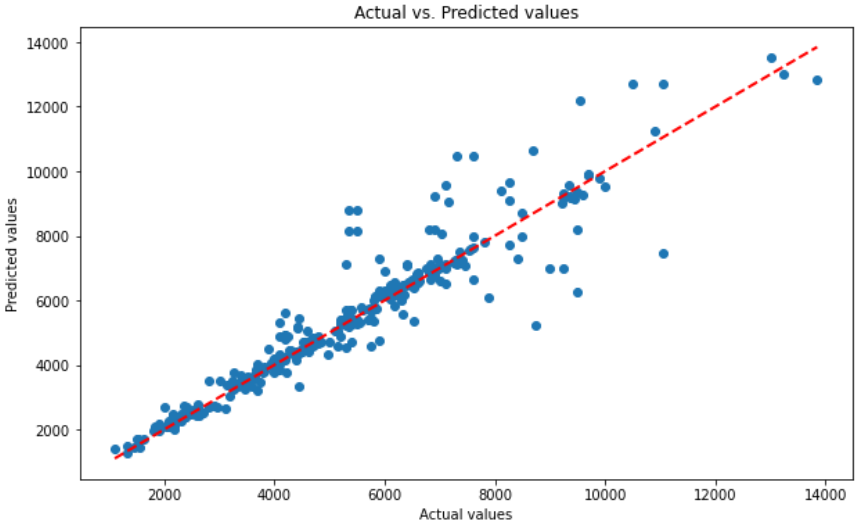
Main Results

The models we used had a different accuracy each. The following is the final accuracy of each model and the way its error was calculated.

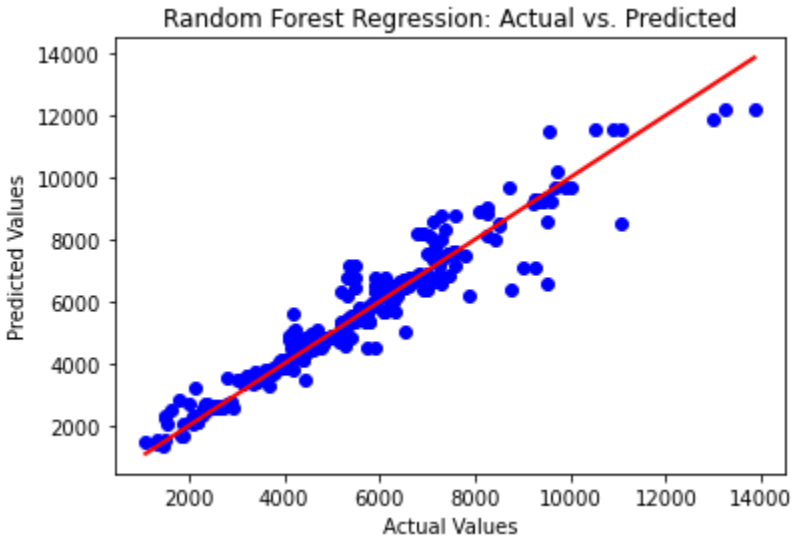
1. Lasso model: 73.5%, R2 score



2. Xboost model: 90.61%, R2 score



3. Decision Tree Regressor model: 93.2%, R2score
4. Random Forest Regression model: 93.6%, RFR.score. While after gridsearch the accuracy became 93.7%



Conclusion & Discussion

This project has given us the chance to learn more about the machine learning work environment, where to find data, what models to use and how. We are not yet perfect in using the models and we will never be, we will always have to learn and work on ourselves to become better at machine learning as a technology and how we can use the many tools machine learning and AI provides for us.

References

Dataset

https://www.kaggle.com/datasets/navins7/vehicle-maintenance-record

Libs

https://pandas.pydata.org/docs/user_guide/index.html#user-guide

https://numpy.org/doc/stable/user/index.html#user

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler.fit_transform

https://scikit-learn.org/stable/

https://seaborn.pydata.org/

Models

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoCV.html#sklearn.linear_model.LassoCV

https://xgboost.readthedocs.io/en/stable/tutorials/index.html

https://www.datacamp.com/tutorial/decision-tree-classification-python

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html