FINAL PROJECT MACHINE LEARNING (CS 6375.003)

Submitted By:

Punit Bhalla (pxb173830)

Spandan Dey (sxd174230)

Rajat Koti (rxk174330)

THE UNIVERSITY OF TEXAS AT DALLAS

Mercedes-Benz Greener Manufacturing

Can you cut the time a Mercedes-Benz spends on the test bench?

Introduction

Mercedes-Benz is known for its innovation and quality. Daimler’s Mercedes-Benz cars are leaders in the premium car industry. With so many features and options available for customers, they need to ensure the quality and safety for every car configuration before it is available for sale.

Daimler developed a robust and sophisticated testing system for their cars but due to so many sophisticated configurations available, speed of testing cannot be optimized without a powerful algorithmic approach. Because of the complex and time-consuming process of testing, carbon emission is a problem which Mercedez is facing currently.

The challenge in this Kaggle competition was to tackle the curse of dimensionality of number of features and reduce the time that a car spends on test bench, hence reducing the carbon emission. We need to work with a dataset with different permutation of car features and predict the time it takes to pass testing. This will lead to speedier testing hence less carbon emission without reducing Daimler’s standards

Dataset and Features

The data set provided on Kaggle consisted of two comma separated files. One is for training of models and other is for testing and predicting the output testing time of cars. Dataset is made of anonymized set of variables where each data point represents a custom configuration for a car. The ground truth is labelled as y which represents the time taken for that car to pass the testing for each variable.

Data Set	Multivariate
Default Task	Regression
Attributes Task	Categorical, Real, Integer
# Instances	4209
# Attributes	377
Missing Values	No

Table 1:Dataset Description

Below are the steps followed for data analysis, preprocessing and feature engineering: -

a. We observed that dataset contains no missing or null (?) values.
b. 8 features out of 377 features contains categorical data, therefore we encoded those categorical data to numerical values so that we can apply models to our dataset.
c. We observed that for some of the categorical features, categories given in training and testing were different hence in order to encode the data properly, we first merged test and training dataset together, encoded the categorical data into numerical values and then split the dataset back to training and test dataset.
d. 13 features out of the 377 had same value for all instances provided in the training dataset. We considered that such features will have no contribution towards learning hence, we eliminated these features from training dataset.
e. We observed that one data point is an outlier which may cause significant error during fitting the data on certain models.

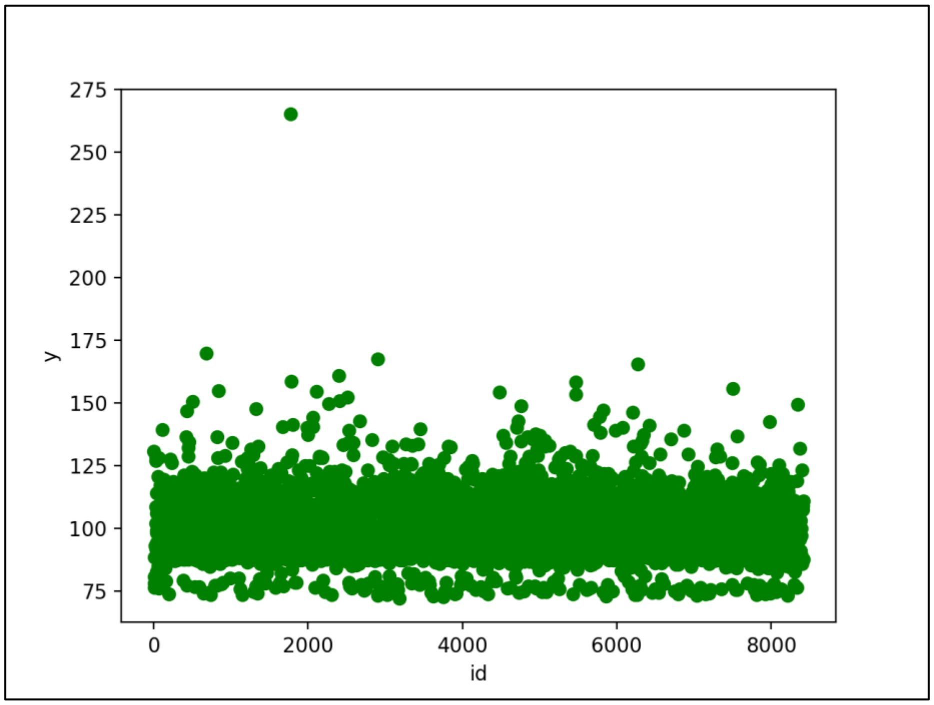

Figure 1: Outlier Detection in the Dataset

f. Therefore, we removed that outlier from the training dataset as shown in the figure below:

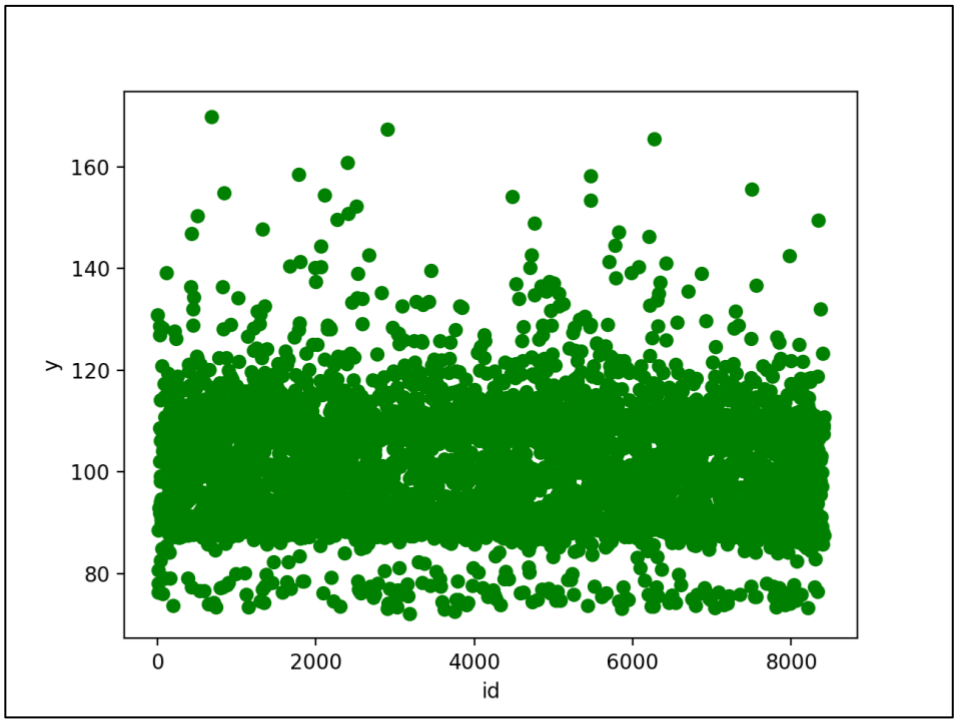

Figure 2: After removal of Outlier from Dataset

g. We have used SciKit Learn's Standard Scaler method to scale the dataset before running regression techniques which involved usage of Support Vectors and Multi Layered Perceptrons.
h. Due to such a high number of features, we considered doing **principal component analysis** to reduce the curse of dimensionality. However, we didn't get sufficient variance within each particular feature.

Models Training and Evaluation

To predict the time to be spent by a car with each type of configuration, we needed to apply regression techniques by fitting the most appropriate curve on the data points provided to us.

Although there were many regression models which we can apply on our dataset, we chose the below mentioned models. We also applied Grid Search with cross validation to choose the best set of hyper parameters we can apply to train and test the models.

a. **Linear Support Vector Regressor:** A support vector regressor with linear kernel, we used it to plot a line among the data points. After applying Grid Search, below are the parameters which were giving best results.
   i. *C*: 3.0
   ii. *Epsilon*: 0.001
   iii. *Loss*: epsilon_insensitive
   iv. *Random_state*:0

b. **Support Vector Regressor with Gaussian Kernel:** Using a Gaussian kernel in support vector regressor, we tried to scale the data to a higher dimension to see if we can separate the data linearly. Below is the tuned hyper parameter list:
   i. C: 5.0
   ii. Degree: 2
   iii. *Epsilon*: 0.001
   iv. *Kernel*: rbf
   v. *Max_iter*: 400
   vi. *Tol*:0.001

c. **Adaboost Regressor:** Adaboost regressor is an ensemble technique we used to train multiple weak models to give us a cumulative r2_score which should be better than individual models. We applied a Grid Search on the same and received below list of tuned hyper parameter:
   i. *Learning_rate***:**1
   ii. *Loss***:** linear
   iii. *N_estimators***:** 20
   iv. *Random_state***:**0

d. **Bagging Regressor:** It's an ensemble model which gave very good results when used with decision tree regressor as the base model. After performing grid search, we received below parameters
   i. *Max_features***:** 350
   ii. *Max_samples***:** 100
   iii. *N_estimators***:** 100
   iv. *N_jobs***:** 1
   v. *Random_state***:** 0

e. **Gradient Boosting Regressor:** One of the best ensemble regressor models we have observed while fitting the data with Decision Tree as base model. Below are the hyper parameters with their values:
   i. *N_estimators*:200
   ii. *Max_depth*: 5
   iii. *Min_samples_split***:** 3
   iv. *Learning_rate***:** 0.01
   v. *Loss***:** huber
   vi. *Warm_start***:** True
   vii. *Random_state***:** 0

f. **MLPRegressor:** Neural nets are considered to be one of the most powerful models for classification. When used without any activation function for the perceptrons involved, the same can be used for regression as well. We used MLP regressor for our data set which gave better results than any other single model. Only Bagging and Gradient Boosting performed better than MLPRegressor. The parameters are mentioned below:
   i. *Hidden_layer_sizes*: (5,5)
   ii. *Activation*: logistic
   iii. *Solver*: adam
   iv. *Alpha*: 0.001

v. *Learning_rate*: constant
vi. *Learning_rate_init*: 0.01
vii. *Max_iter*: 1000
viii. *Random_state*: 0
ix. *Tol*: 0.0001
x. *Early_stopping*: False
xi. *Epsilon*: 1e-08

Results and Analysis

We have considered R^2, also called the coefficient of determination as the evaluation metric. The coefficient of determination tells us about how much of the variance in y is described by the variance in x, where y is a dependent variable and x is an independent variable.

The coefficient of determination ranges from 0 to 1. R^2 of 1 indicates that the regression predictions perfectly fit the data. Values of R^2 outside the range 0 to 1 can occur when the model fits the data worse than a horizontal hyperplane. This would occur when the wrong model was chosen.

We have also used Mean Squared Error as our evaluation metric. The mean_squared_error function computes mean square error, a risk metric corresponding to the expected value of the squared (quadratic) error or loss. The goal of experimental design is to construct experiments in such a way that when the observations are analyzed, the MSE is close to zero relative to the magnitude of at least one of the estimated treatment effects.

If $\hat{y}_i$ is the predicted value of the i-th sample, and y_i is the corresponding true value, then the mean squared error (MSE) estimated over n_{samples} is defined as

$$\boxed{\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.}$$

Model	Coefficient of Determination (R^2)	Mean Squared Error (MSE)
Linear SVR	-0.241375795413	186.192095597
SVR (Gaussian Kernel)	0.367674495262	94.8415550407
MLP Regressor	0.439536220987	84.0631224699
Adaboost Regressor	0.475767115417	78.6289048991
Bagging Regressor	0.544725272766	68.2859741985
Gradient Boosting Regressor	0.552862061438	67.0655494569

Table 2: Model Comparisons

Conclusion: -

After thoroughly analyzing, cleaning, preprocessing and fitting the data on various models, we concluded that the best estimator in our case is **Gradient Boosting Regressor** which gives us a r2_score of **0.552862061438** and mean squared error equals **67.0655494569**. The best parameters are:

i. *N_estimators*:200
ii. *Max_depth*: 5
iii. *Min_samples_split*: 3
iv. *Learning_rate*: 0.01
v. *Loss*: huber
vi. *Warm_start*: True
vii. *Random_state*: 0