CS 6375
Final Project Report

SUBMITTED

BY

Sai Kiran Taduri
(sxt161730)

1. Introduction:

Testing has always been an imported criterion for any product to reach the end user. Thus, every electronic device, automobile, software etcetera needs to pass the testing phase to be a successful product. The time a device takes for passing the testing phase plays a vital role when the number of products to be tested is huge. Having a knowledge of the time the product takes to pass that phase is vital in many cases.

The same case applies to automobiles also. To ensure the safety and reliability of each car configuration before they hit the road, a testing system is a must to test the product. Optimizing the speed of testing for so many feature combinations is complex without an accurate algorithmic approach. The Daimler Mercedes-Benz Greener Manufacturing challenge in Kaggle is based on this prediction of time[1]. We are trying to tackle the curse of dimensionality and reduce the time a car takes to pass testing by making predictions from only important features of the car. We worked with a dataset representing different permutations of Mercedes-Benz car features to predict the time it takes to pass testing.

2. Problem Definition and Algorithm :

2.1. Task definition :

The data provided consists of several features of a car like air suspension, head up display, 4WD etcetera. There were 376 number of attributes and 4209 instances in training and test data.

Each instance is made of different dimensions and they have been rescaled to 20 principal attributes using preprocessing techniques. This was quite challenging because the value to be predicted is the testing time which is non-trivial and it was a continuous value rather than a discrete value. The data was nonlinear and there were many attributes out of which, choosing principal components was challenging.

2.2. Algorithm:

It was necessary that the given dataset needed to be preprocessed. We have decided to do preprocessing using Python after a bit of research. The preprocessed data was then trained and tested using neural nets, kNN, deep learning and decision trees.

3. Dataset description

The given dataset consists of test.csv and train.csv.

Train data:

Number of Features - **376**

Number of Instances - **4209**

Among the 376 attributes given, one attribute is ‘y’, which is the predicted value

	ID	y	X10	X11	X12	X13	X14	X15	X16	X17	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	...	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000
mean	4205.960798	100.669318	0.013305	0.0	0.075077	0.057971	0.428130	0.000475	0.002613	0.007603	...	0.318841	0.057258	0.314802	0.020670	0.009503	0.008078	0.007603	0.001663	0.000
std	2437.608688	12.679381	0.114590	0.0	0.263547	0.233716	0.494867	0.021796	0.051061	0.086872	...	0.466082	0.232363	0.464492	0.142294	0.097033	0.089524	0.086872	0.040752	0.021796
min	0.000000	72.110000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2095.000000	90.820000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	4220.000000	99.150000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	6314.000000	109.010000	0.000000	0.0	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	...	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	8417.000000	265.320000	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Fig 1. Train dataset description

Test data:

Number of Features **- 375**

Number of Instances **- 4209**

There was no predicted value attribute in the test data and it needs to found.

	ID	X10	X11	X12	X13	X14	X15	X16	X17	X18	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
count	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	...	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000
mean	4211.039202	0.019007	0.000238	0.074364	0.061060	0.427893	0.000713	0.002613	0.008791	0.010216	...	0.325968	0.049656	0.311951	0.019244	0.011879	0.008078	0.008791	0.000475		
std	2423.078926	0.136565	0.015414	0.262394	0.239468	0.494832	0.026691	0.051061	0.093357	0.100570	...	0.468791	0.217258	0.463345	0.137399	0.108356	0.089524	0.093357	0.021796	0.02	
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	
25%	2115.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	
50%	4202.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	
75%	6310.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	...	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00	
max	8416.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00	

Fig 2. Test dataset description

4. Data Preprocessing:

Languages used - Python

Tools used - Anaconda Navigator

The following graph shows the change in y value with the indices given in the dataset.

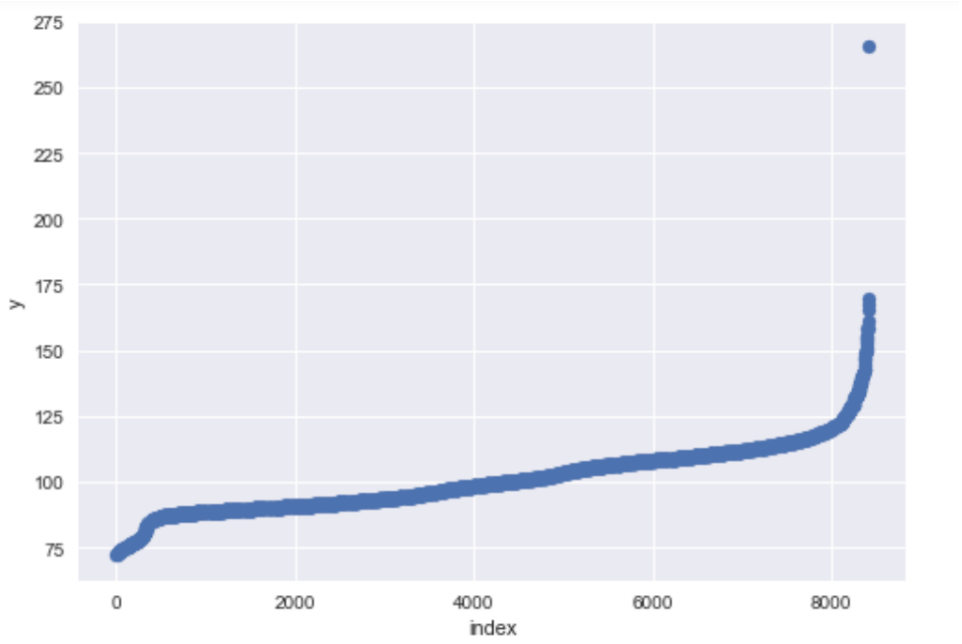

Fig 3. ID vs y value

The following graph gives the distribution of y values

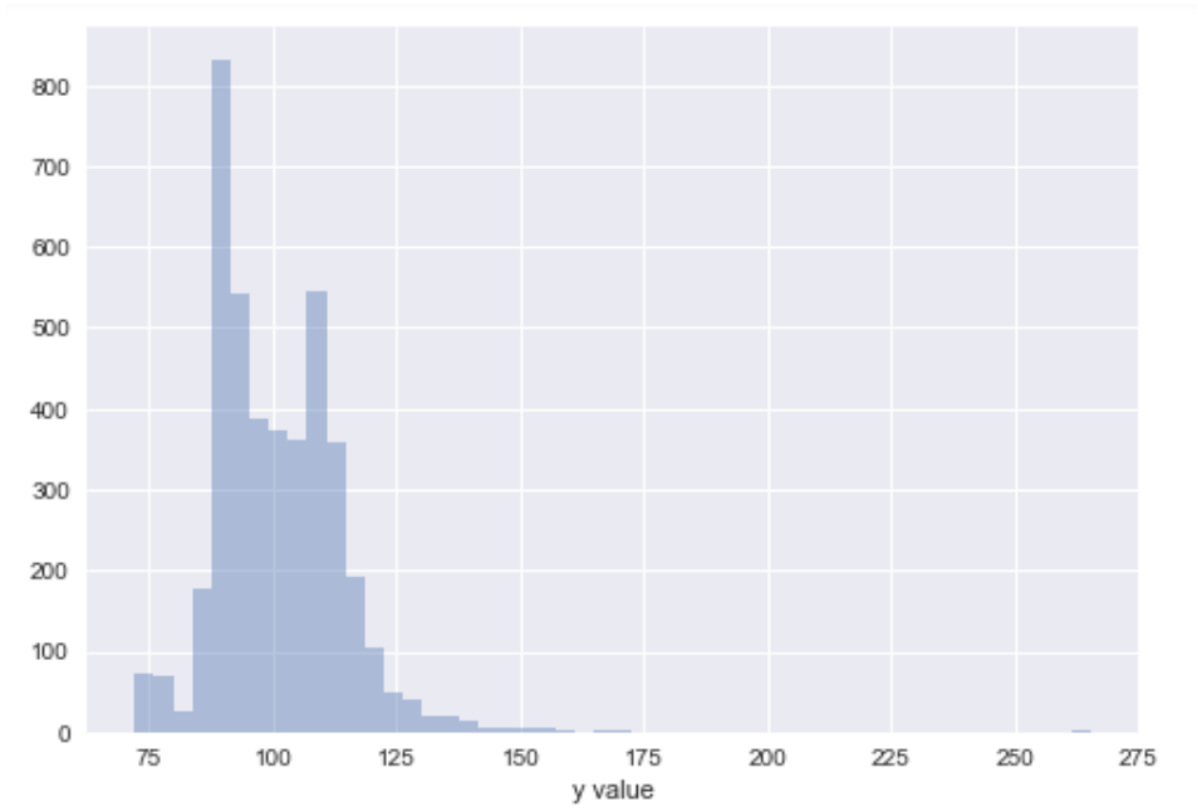

Fig 4. Distribution of y values

Based on this distribution we have decided to do the following tasks:

1. Removing duplicate and null values
2. Giving numerical labels to categorical labels
3. 80/20 split of training data
4. Dimensionality reduction using Principal Component Analysis

1) Removing duplicate and null values

We have found the attributes with the same value for all the instances and dropped those columns from the training data. Instances with null values are deleted.

2) Giving numerical labels to categorical labels

There were 8 attributes with categorical values as below.

array([['k', 'v', 'at', 'a', 'd', 'u', 'j', 'o', 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, ... 0, 0, 0, 0]], dtype=object)

These 8 attributes with categorical values are transformed to numerical values using Label Encoding. These 8 attributes were transformed as below.

array([[32, 23, 17, 0, 3, 24, 9, 14, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,.....0, 0, 0, 0, 0, 0]], dtype=object)

Following is the distribution of the 8 attributes that are transformed from categorical type to numeric type

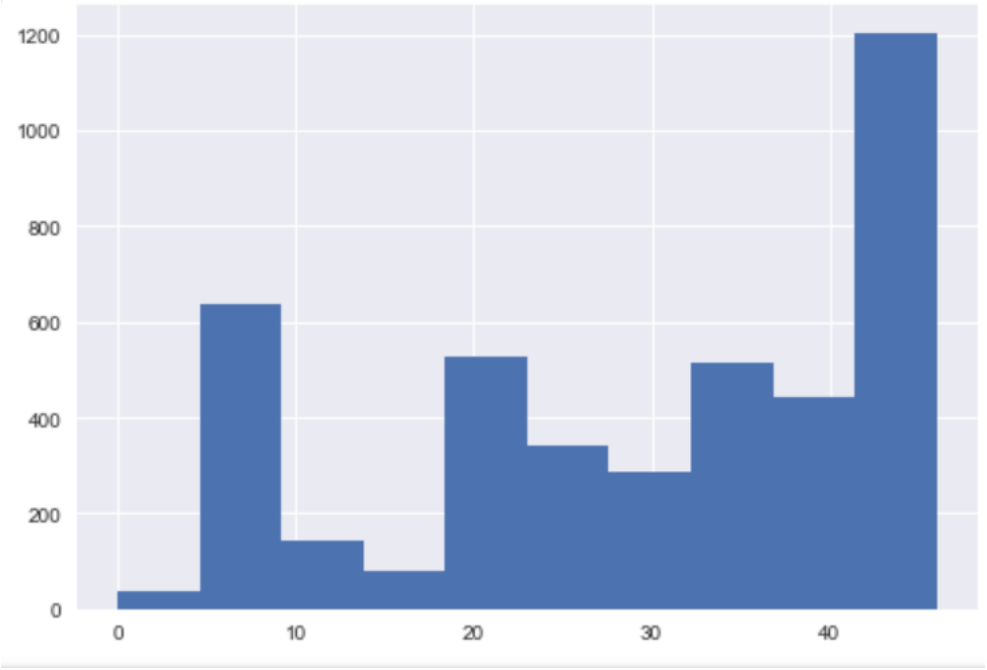

Fig 5. Distribution of the attribute ‘X0’

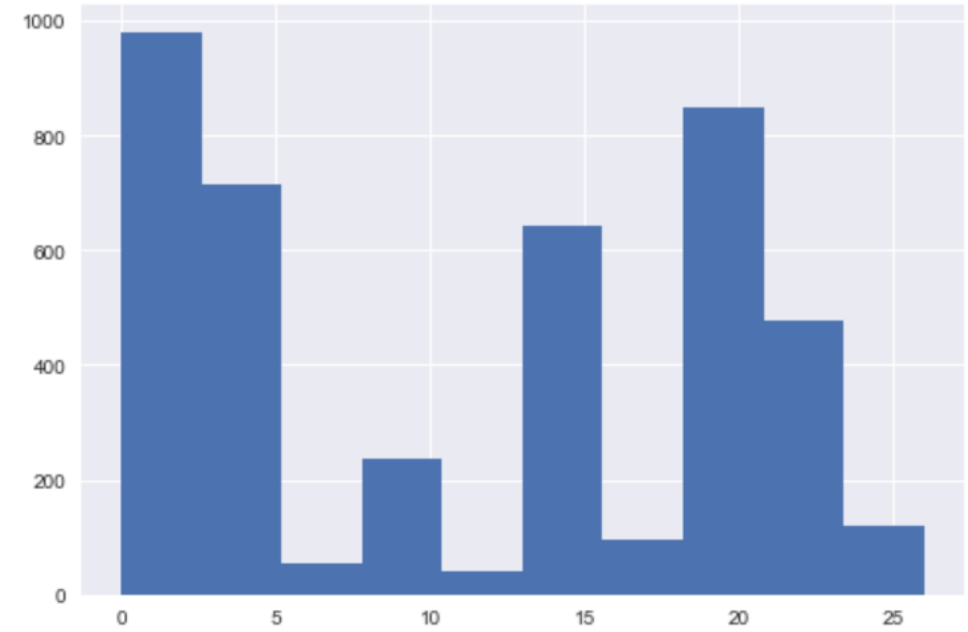

Fig 6. Distribution of the attribute ‘X1’

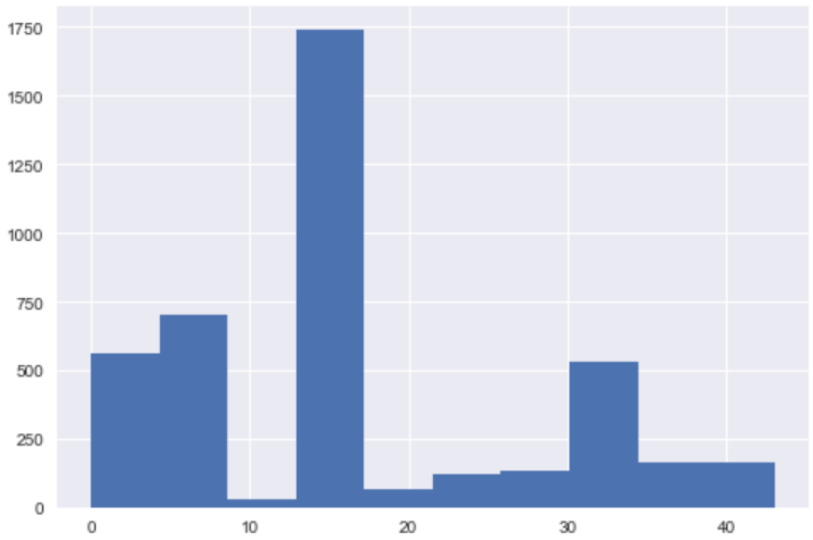

Fig 7. Distribution of the attribute ‘X2’

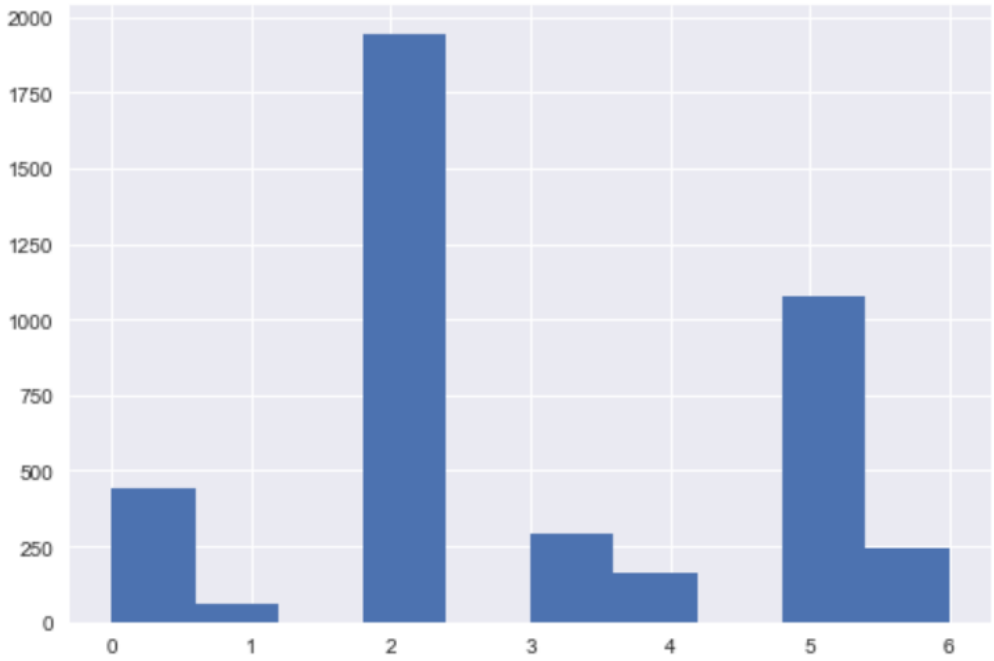

Fig 8. Distribution of the attribute ‘X3’

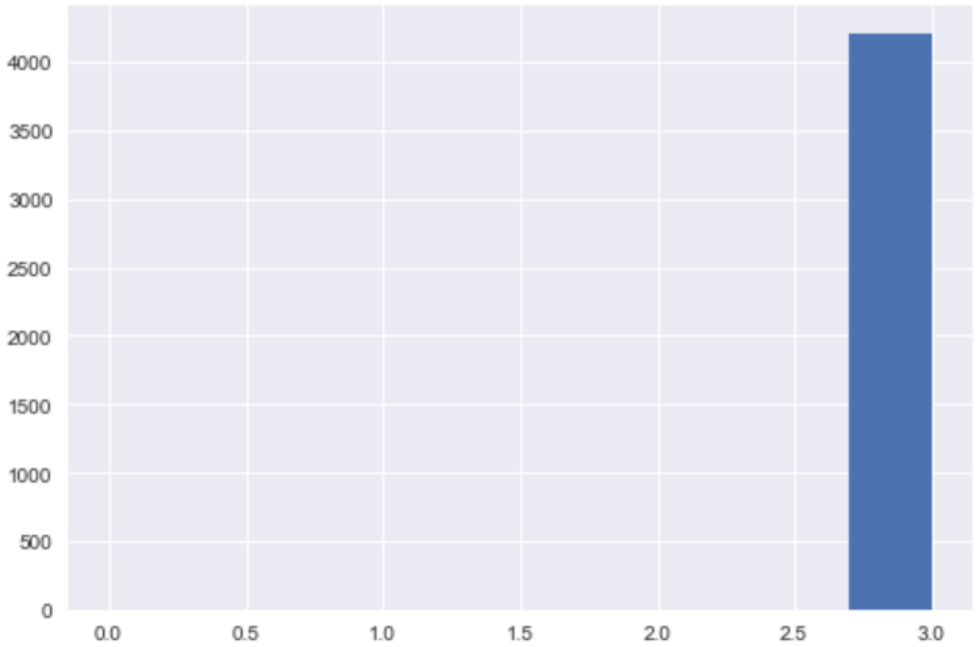

Fig 9. Distribution of the attribute ‘X4’

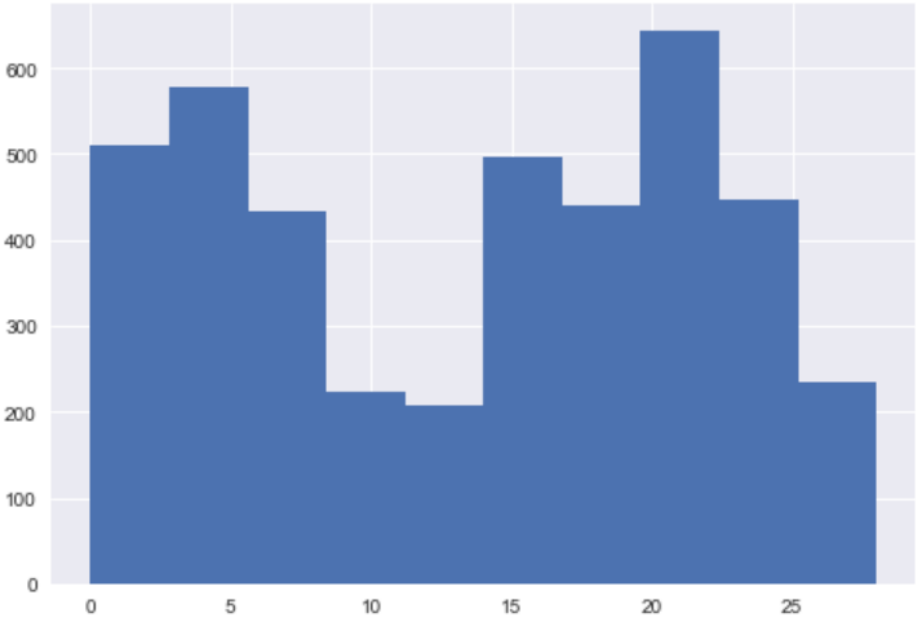

Fig 10. Distribution of the attribute ‘X5’

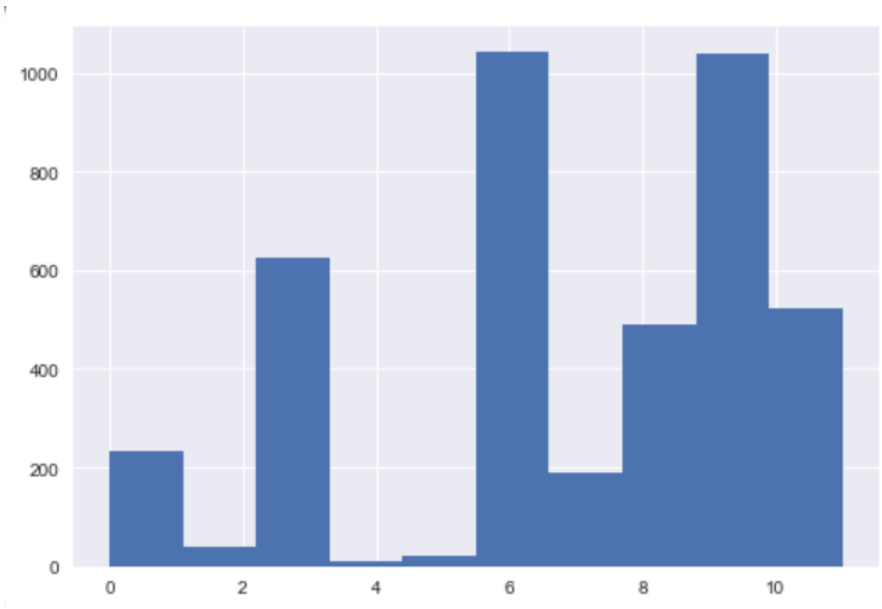

Fig 11. Distribution of the attribute ‘X6’

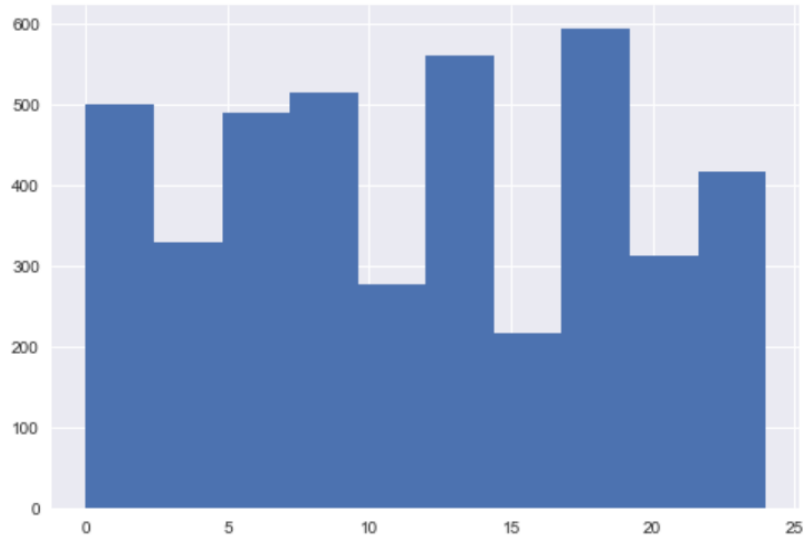

Fig 12. Distribution of the attribute ‘X8’

3) The training data was split into training set and test set using 80/20 split.

4) The training set and test set dimensionality was reduced using Principal Component Analysis from 375 attributes to 25 attributes.

5. Proposed Techniques and Solutions:

1. **Principal Component Analysis :**

PCA is a Feature Extraction technique. The main idea behind using PCA is to reduce the dimensionality of the dataset consisting of many variables correlated with

each other, while retaining the variation present in the dataset, up to the maximum extent. From the m independent variables of a dataset, PCA extracts p<=m new independent variables that explain the most of the variance of the dataset, regardless of the dependent variable. The fact that the Dependent Variable is not considered makes PCA an unsupervised model.
In order to use PCA one has to import the following library in scikit learn.
from sklearn.decomposition import PCA

2. K- Fold Cross Validation Implementation:

To evaluate a model's performance we perform K-Fold Cross Validation. When we get the accuracy on the test set and if we run the model again and test again its performance on the other test set, we can get a very different accuracy. So judging our model performance only on one accuracy, on one test set is not super relevant. So there is this technique called K-Fold Cross Validation that improves this a lot. It will fix this problem. How ?
The answer is it will fix it by splitting the training set into k folds and train the model on k-1 folds and test it on the last remaining fold. Consider 10 folds, we split the training set into 10 folds and train the model on 9 folds and test it on the last remaining fold. Since with 10 folds we can make 10 different combinations of 9 folds to train the model and one fold to test it. That means we can train the model and test the model on 10 combinations of train and test sets. That we give us much better idea of the model performance. Analysis will be much more relevant with k-fold cross validation.

Example :
```
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(estimator, X, y, cv)
```

3. Extreme Gradient Boosting (XGBoost) Implementation:

XGBoost is a usage of gradient boosted decision trees intended for speed and execution that is dominative focused machine learning. XGBoost is a powerful machine learning model that prevents from overfitting.
The prime features of the xgboost algorithm are sparse aware, block structure, continued training. we used xgboost because it optimizes the speed and memory used.
The parameters that we tuned are: max_depth=5, n_estimators=100

4. Neural Network (Multilayer Perceptron) Implementation:

Neural Network is a learning algorithm that is based on the structure of biological neural networks(neurons). We implement this by using multiple perceptrons together. there are mainly 3 layers. Input layer, it gets the input and passes it to the hidden layer. Hidden layer, which forms most of the artificial brain, it learns a COMPACT representation of the previous layer and passes it down to the next layer. Output layer, it gets the input from the hidden layer and gives the output. Most neural networks are fully connected, which means each hidden unit and each output unit is connected to every unit in the layers either side. The Neural Network divides the data points on a plane with multiple straight lines.

The parameters that we tuned are:

hidden_layer_sizes=(10,10), learning_rate_init=0.01, early_stopping=False

5. Decision Tree Implementation:

A decision tree is a tree algorithm which the internal node represents the "test case" on the attribute (example: GPA greater than 3.5), the branches represent the decision from the internal node(if true then one branch else the other), the leaf nodes represent the classification value (Example: positive class or negative class).

The Decision tree divides the feature space into rectangles which are parallel to the x axis and y axis. The efficiency can be improved by decreasing the height of the tree. It can be done by selecting appropriate attributes as top as possible (with Information Gain) or by pruning.

The parameters that we tuned are:

max_depth= 6

6. K- Nearest Neighbours Implementation:

In K-nearest neighbours the learning phase consists of simply storing all the training examples which are referenced as instances it also frees us the burden from model creation when we encounter a test query a set of similar instances are retrieved from and used to classify the data In fact it is the most popular IBL algorithm.

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. The curse of dimensionality in the k-NN setting essentially implies that Euclidean separation is unhelpful in high measurements since all vectors are practically equidistant to the search question vector (imagine multiple points lying more or less on a circle with the query point at the center; the distance from the query to all data points in the search space is almost the same).

We have considered the number of neighbours to be 10 (k=10) for our implementation

7. Bagging :

Bagging also called as Bootstrap aggregating is an ensemble technique, that combines the predictions of multiple machine algorithms together. Bootstrap aggregation is a process to reduce the variance for the algorithms which have high variance. We have used bagging to reduce the variance from our model. In our bagging algorithm we have created many sub samples from our dataset. we have used the Bagging regressor from the sklearn.ensemble package
The parameters that we tuned are: n_estimators = 10

8. Deep Learning :

Deep learning is a class of machine learning algorithms that uses a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. It uses some form of gradient descent for training via backpropagation. Deep Learning is used for many highly computing intensive tasks. In fact deep learning can be used for variety of purposes from making predictions for business problems or computer vision like recognizing pictures and videos and also for recommended systems.

We have installed a special library called **keras** library. Keras is a library based on Theano and Tensor flow libraries i.e., it runs on theano and Tensor flow libraries. Keras is an amazing library which helps to build a deep neural network with only few lines of code.We have used keras to build deep learning model very efficiently. We have trained the ANN with Stochastic Gradient Descent. We used two modules, Sequential module required to initialize the neural network and Dense module required to build the layers of ANN.

6. Experimental results and Analysis:

We created models using the following techniques and calculated the Mean Squared Error, Root Mean Squared Error, R Squared value, Mean Absolute Error and Cross validation scores for each model.

The following table gives a comparison of the results from each model.

Techniques	MSE	RMSE	R Squared	Mean Absolute error	Cross validation score
XGBoost	*64.660*	*8.041*	*0.569*	*5.693*	*0.702*
Neural Net	*75.611*	*8.695*	*0.482*	*6.327*	*0.642*
Decision Tree	*73.542*	*8.575*	*0.496*	*6.016*	*0.677*
kNN	*67.737*	*8.230*	*0.536*	*5.765*	*0.578*

Bagging	***77.917***	***8.827***	***0.466***	***6.253***	***0.604***

Fig 13. Results

7. Conclusion :

The techniques such as neural network, deep learning, ensemble, decision trees , k nearest neighbors have been implemented on the data set after doing preprocessing. We have used the performance metrics such as Mean Squared error, Root Mean Squared error, Mean absolute error , R-square and cross-val-score by doing 10 fold cross validation and these results have been summarized in the table above. From the experimental analysis, we have concluded that XGBoost model has given us the best results.

8. References:

[1] https://www.kaggle.com/c/mercedes-benz-greener-manufacturing
[2] http://scikit-learn.org/stable/supervised_learning.html#supervised-learning