Contents

Description of Regression Models Used

- **Linear Regression** works well for linear models or the models that can convert to linear. It tries to minimize the loss function w.r.t each parameter. While minimizing, it makes some hidden ***assumptions.*** Why's that so ? Lets say the dataset comprises 100 entries and the sum of errors = 500.
  - ➔ Firstly, there can be outliers. These outliers need to be dealt with carefully otherwise the model will end up fitting those along with others. In math words, we can say that in the equation y = mx + c, the slope of the line will face a sudden increase/decrease due to a few ***outliers***.
  - ➔ Secondly, putting the outliers aside, let's say there were 40 entries responsible for 350 of the error and rest 60 for 150. This means that error was taken care of as a whole but not individually. In math words, it assumes that errors are ***normally*** distributed.
  - ➔ Thirdly, when we move forward to the multivariable regression, let's say we've the equation y = ax + by + cz + d. Here, this model will assign (x,y,z) some coefficients. Here, linear regression makes another assumption. It will assign (a,b,c) to (x,y,z) independently. While dealing with "ax", it will minimize the equation w.r.t the variable x only. Maybe, x & z have some relation too in between. Linear Regression puts this aside when dealing the values. In math words, it assumes that (x,y,z) are not ***correlated***.

- **Polynomial Regression** is similar to the linear regression but supports the non-linear models too.
  - ➔ If we are doing polynomial regression with the equation, y = ax + bx^2 + cy + dy^2 + e, it suffers from the same problems as the linear regression.
  - ➔ If we use the equation y = ax + bx^2 + cy + dy^2 + e(xy) + f, we can get rid of the ***correlation*** problem.

- **Decision Tree** Unlike linear & polynomial, a decision tree asks questions to the data provided and builds up a regression model on that. It looks for binary answers in data. Upon every answer, it splits the data into 2 parts. Whether the split is good or not is answered by the sum of errors (residuals). The split with the least error is selected and this process continues till end or till the limit (if specified).

  Decision tree, however, ***overfits*** the data. The more the questions asked the more it fits. To avoid this, trees are pruned.
  In the ***pruning*** process, split is based not only on the sum of residuals. Instead, it also penalizes the depth (or the number of questions asked).

  Aside from overfitting, there's one more very ***serious concern***. Let's say we are testing 5 splits and the minimum sum of residuals is 1200. This might not be a very good split either but we have to go with this anyway. This can be fine with the classification problems but with regression, the predictions suffer very much.

  Another ***problem with regression*** using decision trees is that if data is spreaded, the predictions will be the average of all. These factors result in not so good predictions when it comes to regression with decision trees.

- **Random Forest** is a group of decision trees trying to predict. If data is not in favor of the decision tree, they can end up building an inaccurate model especially for regression. Random forest builds different trees for the same data. Moreover, data can also be bootstrapped (picking out random data for trees). When predicting the values, the random forest asks the trees. The value with which most trees agree on is selected as predicted value.

  Due to this process, random forest do better than a single decision tree. However, sometimes, random forests face the same problems (in regression especially).

- **XGBoost Tree** combines trees, gradient boosting, similarity, regularization, pruning and some more features into one. Trees face problems like overfitting, asking the right question, giving proper weight/importance to variables, averaging the spreaded data. Unlike other models discussed above, it keeps minimizing the residuals instead of target value (y). While doing this, it moves forward with a learning rate hoping it will get better and better (residuals/errors keep decreasing).

Following features of XGBoost Trees compensate for the problems tree-based models usually face.

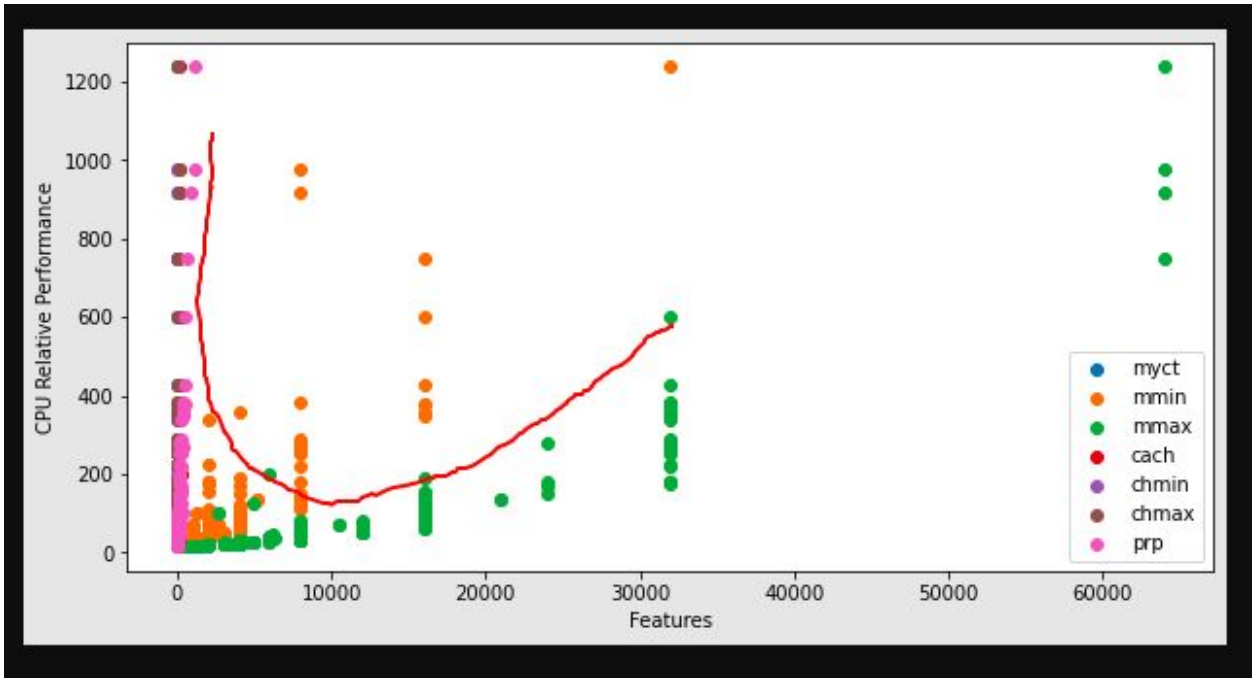
- ➔ ***Gradient Boosting*** : Builds the next tree based on the previous one's performance. It does so by assigning weights to individual entries. This way the values which are predicted wrong get a higher weight. Now, in order to get selected, the next tree has to do better on the mistakes done by the previous trees.
- ➔ ***Similarity*** : While splitting the data, it ensures that the leaves get data that are close/alike. It does so by giving each leave a similarity score. Lower similarity score tells that values in the leaf are not suitable for being together. This way, XGBoost does better splitting.
- ➔ ***Regularization & Pruning*** : It has regularization & pruning parameters that lower the similarity score and ensure the tree depth to avoid overfitting.

Overview of Data

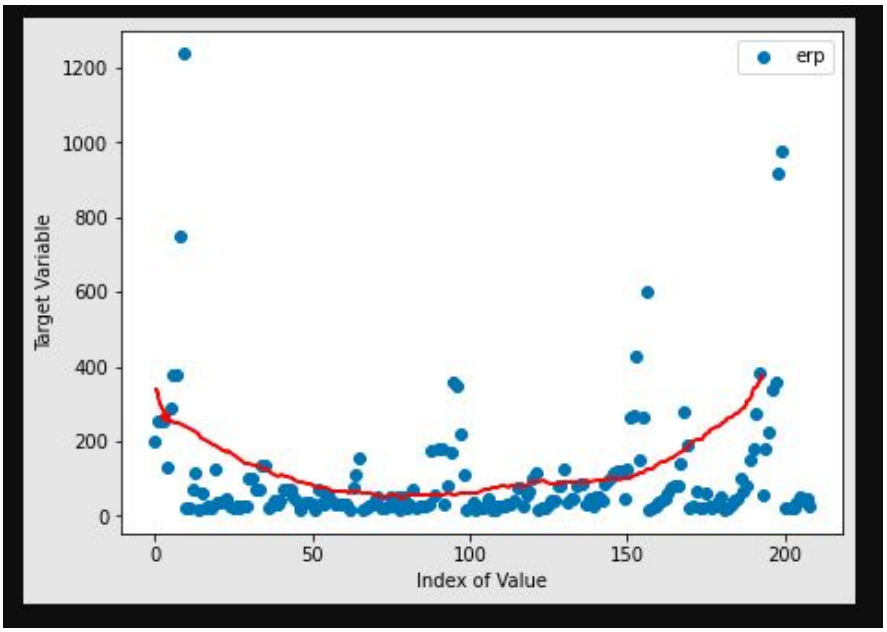
For the following section, please refer here for details.

The dataset had 7 features. Some of them had a linear relation with the targeted variable, some had (1/x) relation and some had no visible pattern. However, the two patterns were very clear as a whole.

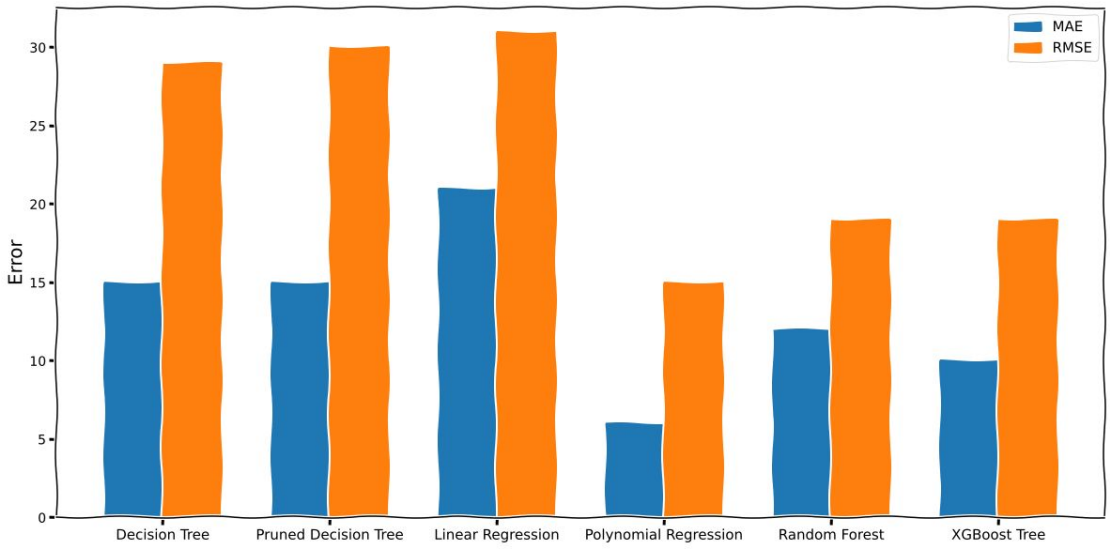
This shows the scatter plot of all features against the target variable



The following shows the target variable scatter plot.



Performance of Different Models



- **Linear Regression** couldn't come up with a good model. The model had a high Mean Absolute Error of 21 and Root Mean Squared Error of 31. That was obvious, too, given the non-linear graph. Only 2-3 features were

somewhat linear. However, it still manages to give a 64% accuracy. The reason behind this is the "prp" feature which had a great say in predicting. This feature had a 0.97 collinearity with the target variable and that's why linear regression managed to get this accuracy despite the obvious non-linearity.

- **Polynomial Regression** showed great results with a 2nd degree polynomial. The data also had shown a square-fit. The polynomial regression version used in here took care of correlations between other variables too. Due to this, it got a 93% accuracy. It had the lowest MAE of 6 which shows at average the deviation of prediction from actual value is 6. The RMSE of 15 (more than twice of MAE) tells that there were some values for which the deviation was larger and were penalized more by MSE. So, the polynomial regression was weaker in fitting those values. That, too, is obvious because linear & polynomial regression fit better with more features/degrees (especially polynomial). If we add more degrees to polynomials, it will try to fit those large deviations but will overfit and consequently will not perform well on the testing data.
- **Decision Tree** gave satisfactory results with an accuracy of 85%. The model had RMSE of 29 and MAE of 15. These errors show that there were either large deviations or outliers that the model were penalized for. Though, 85% accuracy looks good but there's one more concern to it. The simple tree had a depth of 13 and the pruned tree (also nearly the same results) had a depth of 8. For the training dataset of around 200 entries, a depth of 8 had around 100 leaves (was not a complete tree) doesn't look that satisfactory. It seems it is because of the tree structure which suits classification but not regression especially with spreaded data (because in these cases it is forced to go deep and if we prune, its accuracy decreases).
- **Random Forest** performed better than the decision tree model as it was expected to. It gave an accuracy of 89%. The model had RMSE of 19 and MAE of 12. This shows that random forest took care of large deviations well. It seems that the selection of different variables with different trees helped dealing the large deviations. Unlike a decision tree, random forest doesn't have to force itself to fit to every data and every feature. This helps in avoiding overfitting and dealing spreaded out values better. Though this gives a better model and a good accuracy along with a good RMSE, the model had to go through 100 decision trees with depths between 10-14. This shows how costly it can be on large datasets.

- **XGBoost Tree** also performed well with an accuracy of 90%. The model had RMSE of 19 and MAE of 10. This model too had a little problem when it comes to large deviations or outliers (as RMSE is large) **. Aside from that, with the tree nature, this model gave very good results. This model is an extension to the random forest as it weighs the trees and ensures similarity. If we compare random forest and XGboost, there's not much difference when it comes to errors and accuracy. However, the XGBoost Model achieved this accuracy with 50 trees with depths between 1-5. Moreover, this model was not properly regularized/pruned and like the tree nature would've overfitted the data but still manages to give these results.

** *The errors for XGBoost can be reduced. This model has tuning parameters that need to be dealt with carefully. So, the overall performance can be improved by pruning and regularizing properly. I couldn't set parameters properly and I think because of that it might be overfitting.*

Choice of Error Metric

The Mean Absolute Average (M.A.E.) was used to calculate accuracy of models. The Mean Squared Error (M.S.E.) is like variance but w.r.t the predicted values (*instead of mean as is the case in variance*). The MAE tells us an average deviation of values and their predictions. MSE, on the other hand, tells about the squares. I chose to plot RMSE on graph (*which is also like standard deviation but w.r.t the predicted values*) against MAE because that was comparable. MSE are huge and RMSE can also give the information MSE is trying to give.

The data was not properly filtered and preprocessed for trying regression models. RMSE penalizes the deviations more than the MAE and gives a better picture of the model. However, in the data there were some values with large deviations (*as visible both from graphs and the errors of models*). So, RMSE penalized those values more than MAE. Though RMSE looks like a better option but using that the models with more sensitivity to outliers or deviations would've suffered more. This would have been misleading because data should be arranged that way before trying the model.

It doesn't mean that the MAE will take care of the unprocessed data, however, MAE will not penalize as much as RMSE.