EE660 PROJECT

SANIL REGO

USC ID: 9233942261

REGRESSION ANALYSIS ON THE QSAR FISH TOXICITY DATASET

1.ABSTRACT

A thorough exploration of various regression techniques as well as boosting methods on the qsar fish toxicity dataset. Exploratory data analysis and data visualization is used to infer characteristics about the dataset, such as feature distributions, correlation between features and other statistical methods. Regression methods like Linear Regression as well as regularized linear models such as Ridge, Lasso and ElasticNet are used. Regression trees and Boosting methods like Gradient Boosting and AdaBoost are also used for comparison. We also explore our model parameters and hyperparameter tuning using cross validation, model stacking and ensemble learning.

2.INTRODUCTION

2.1 PROBLEM STATEMENT

The qsar dataset represents a classic regression problem. The regression models that I have learnt during this course are used with thorough analysis of each model. The variable we are trying to predict is the quantitative response*, LC50 (-LOG(mol/L)),* which is the concentration of the chemical in water that is lethal for 50% of the exposed population of fish. and we evaluate the predictions using root mean squared error.

2. PRIOR AND RELATED WORK

None

2.4 OVERVIEW AND APPROACH

The dataset is evaluated and the necessary preprocessing is performed to check for missing values, categorical variables, outliers etc. The dataset is then divided into a training set and a test set. The test set is used to evaluate the final out of sample error for the models used.

Linear Regression, Lasso, Ridge and ElasticNet are the linear models that are used and compared. Random Forest Regressor and boosting methods like AdaBoost and Gradient Boosting are also used. For each model we perform hyperparameter optimization using cross validation and compare with baseline models. A stacked approach is then used combining aspects of all these models and then compared.

3.IMPLEMENTATION

3.1 DATASET

The dataset consists of 908 samples, which represent 908 chemicals, LC50 data, which is the concentration that causes death in 50% of test fish over a duration of 96 hours. The first five sets of values are shown in the table below, the columns are the feature names.

	CIC0	SM1_Dz(Z)	GATS1i	NdsCH	NdssC	MLOGP
84	5.926	0.134	1.671	0	0	6.203

	CIC0	SM1_Dz(Z)	GATS1i	NdsCH	NdssC	MLOGP
10	2.405	0.134	0.843	0	0	1.769
617	3.052	0.638	0.903	0	0	3.237
250	3.629	0.331	1.478	0	1	2.821
869	2.824	0.223	1.000	0	1	2.102

The features are the 6 molecular descriptors which are as follows

- 1.MLOGP(molecular properties)
- 2.CIC0 (information indices)
- 3. GATS1i (2D Autocorrelations)
- 4.NdssC (atom-type counts)
- 5.NdsCH (Atom type counts)
- 5.SM1_Dz(Z) (2D matrix-based desctriptors)

The target quantity is the quantitative response, LC50 (-LOG(mol/L))

3.2 PREPROCESSING

There were no categorical features in the dataset. All the feature quantities were numerical and there were no missing values.

3.3 DATASET METHODOLOGY

The entire dataset was used for visualization and exploratory data analysis. I visualized the features against the target variable to observe any linear or nonlinear trends.

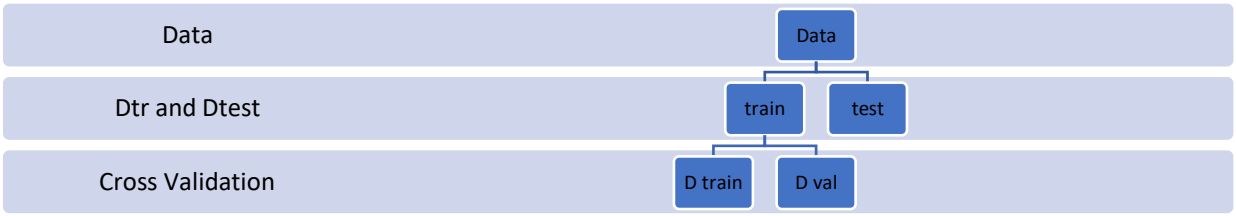
The dataset was split into a training set and a testing set, of which $1/3^{rd}$ of the entire dataset was used as the testing set.

Training set usage for Kfold cross Validation

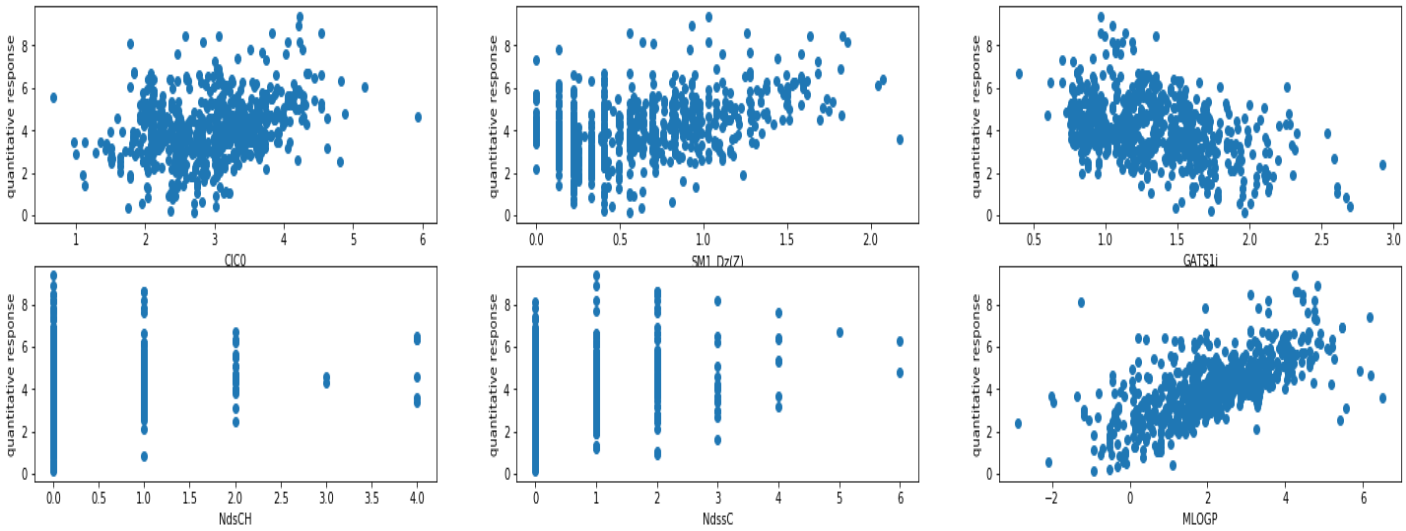
K fold cross validation was used at the training stage when machine learning algorithms were used. For this purpose the training data was split into k folds where k-1 folds were used for training and the Kth fold was used for validation.

For each model chosen, K fold cross validation was used to evaluate the best model parameters on the training set. For model stacking K fold cross validation was used for all the base models on the training set to generate the meta-features. The meta-model was then trained on the meta features and tested on meta features generated by the base models on the test set. The stacking process is described in

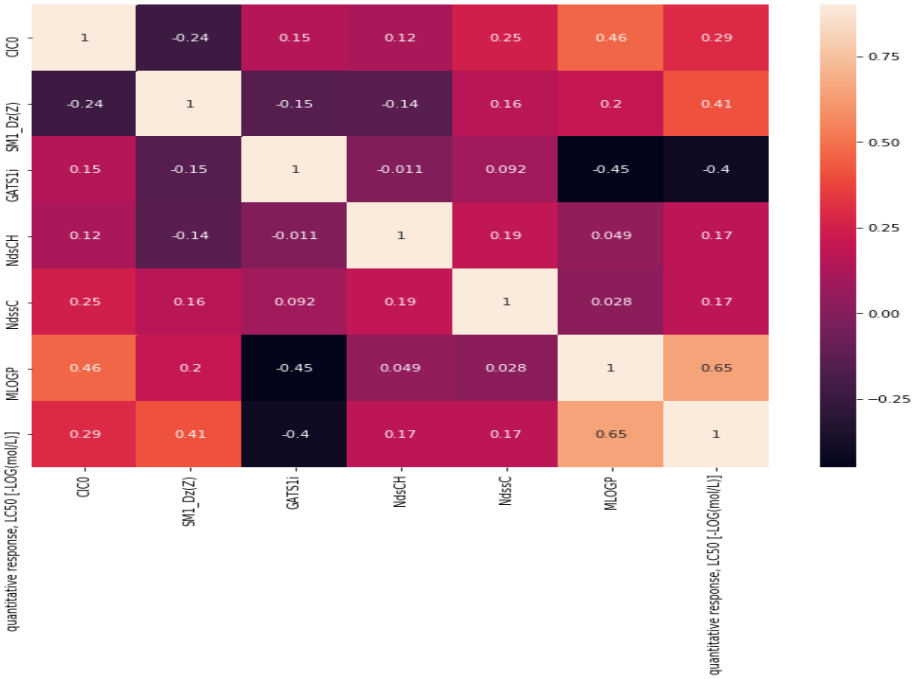
detail in the next section. The test set was also used for testing by the base models so as to have a baseline accuracy.



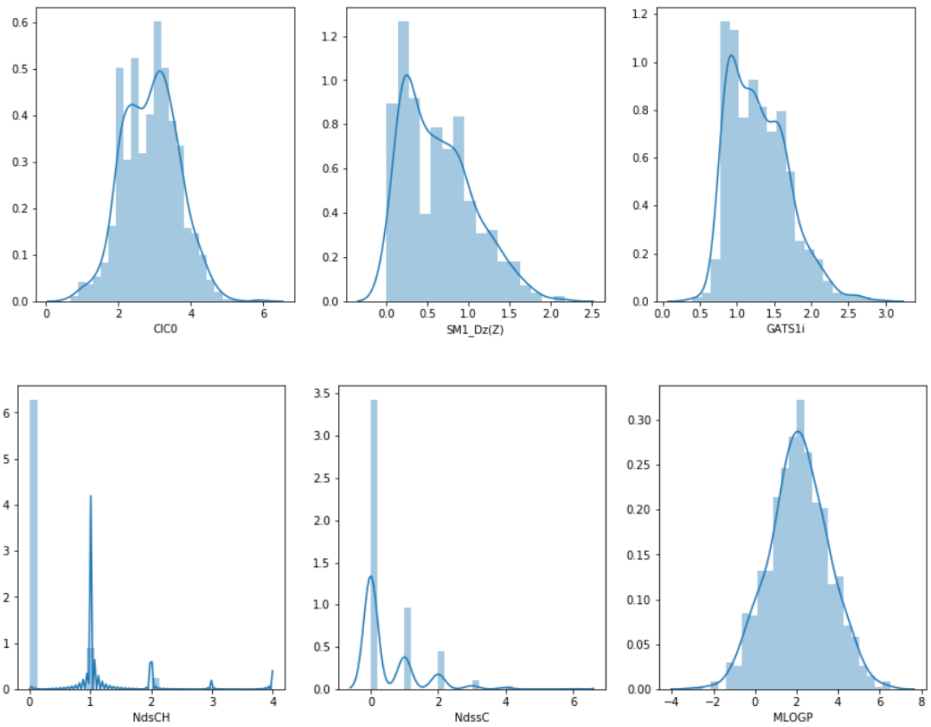
3.4 EXPLORATORY DATA ANALYSIS



The plots above are scatter plots between each feature and the target variable. Visually we can see that there is a linear relationship between the molecular descriptors CIC0 ,SM1_Dz(Z), MLOGP and GATS1i . We can further explore correlations between our data using a correlation matrix. MLOGP has a relatively high correlation with the LC50 concentration.



Now to check for linearity of the features we can plot feature distributions for each of the features. The gaussian kernel density estimate for each variable is plotted. MLOGP is normally distributed while CIC0, SM1_Dz1(Z) and GATS1i are all skewed.



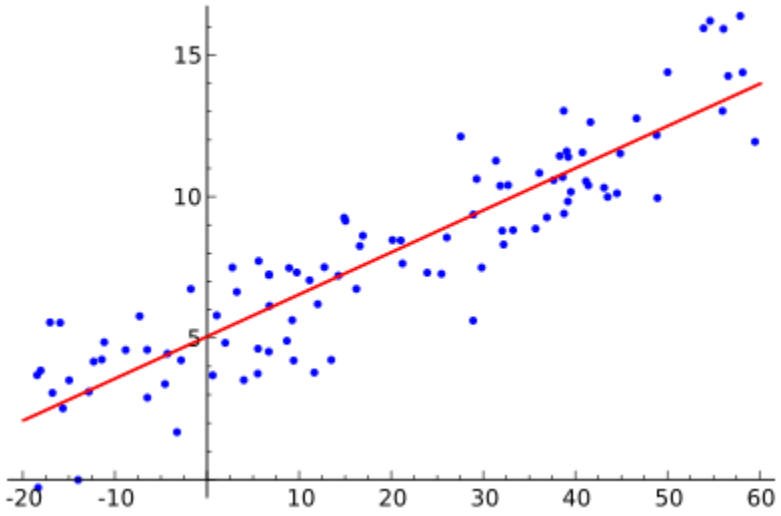
3.5 TRAINING PROCESS

The models used are as follows:

- Linear Regression
- Ridge Regression
- Lasso Regression
- ElasticNet
- Kernel Ridge regression
- Randomforest Regressor
- AdaBoost and
- Gradient Boosting

The model parameters for each model were tuned using K fold cross validation using mean squared error as our error metric.

LINEAR REGRESSION



Linear regression is a supervised learning method used to model a linear relationship between a target variable and its independent features. The equation for linear regression is given below,

$$y = w^t x + w_0$$

We can also obtain a polynomial fit to our data, by using the kernel trick,

$$y = w^t \varphi(x) + w_0$$

Linear regression assumes that there is a linear relationship between the features and the target. It also assumes that there is little or no multicollinearity among the features. This can be determined using the correlation matrix as well as visual scatter plots between features. The error is assumed to follow a normal distribution[1].

The cost function to be minimized is the residual sum of squares:

$$J(w) = \frac{1}{2} RSS(w) = \frac{1}{2} \sum_{i=1}^{N} (y_i - w^t x_i) \quad \ldots . (1)$$

$$J(w) = \frac{1}{2} (y - \underline{X} w)^T (y - \underline{X} w)$$

We can use the ordinary least squares solution to optimize *w*,

$$w_{OLS} = (\underline{X}^T \underline{X})^{-1} X^T y$$

Training

From the assumptions stated above it was evident that some of the features had a linear relationship with the target variable, so I used Linear Regression on the model. I used a Q-Q plot to assess for normality for the target variable. The results of the Q-Q plot are shown below.

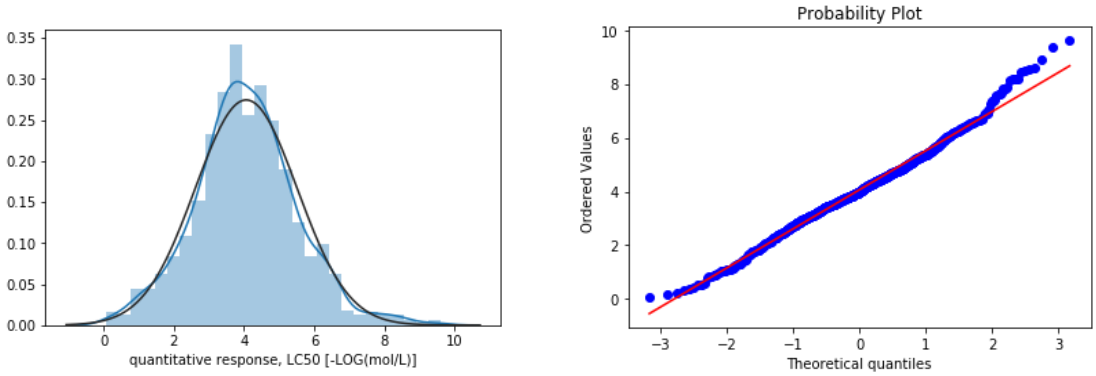
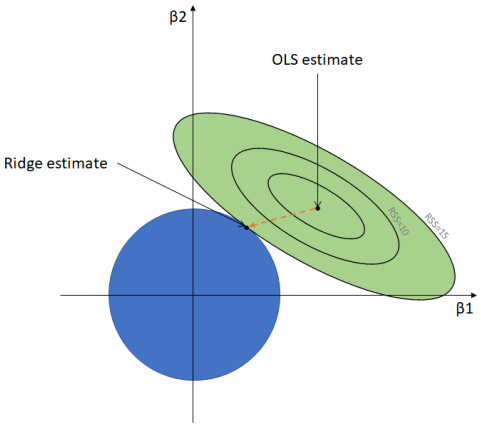


Figure 1 We can see that the target variable is normally distributed, from the distribution plot and the Q-Q plot

A linear Regression model from Python's *sklearn* package was trained on the training data set *X_train,* the results of the model are discussed in the next section.

RIDGE REGRESSION



In Ridge regression we make a change to the cost function given in equation (1) by adding a penalty on the L2 norm of the weights. Because of this penalty the weight vector chosen has a smaller L2 norm, and the coefficients take on smaller values. This can reduce the model complexity and can in turn diminish overfitting. The cost function now becomes,

$$J(w) = \sum_{i=1}^{N} (y_i - w^t x_i) + \boldsymbol{\lambda} ||\boldsymbol{w}||_2^{\,2}$$

This is equivalent to minimizing the cost function of Linear Regression, subject to the constraint

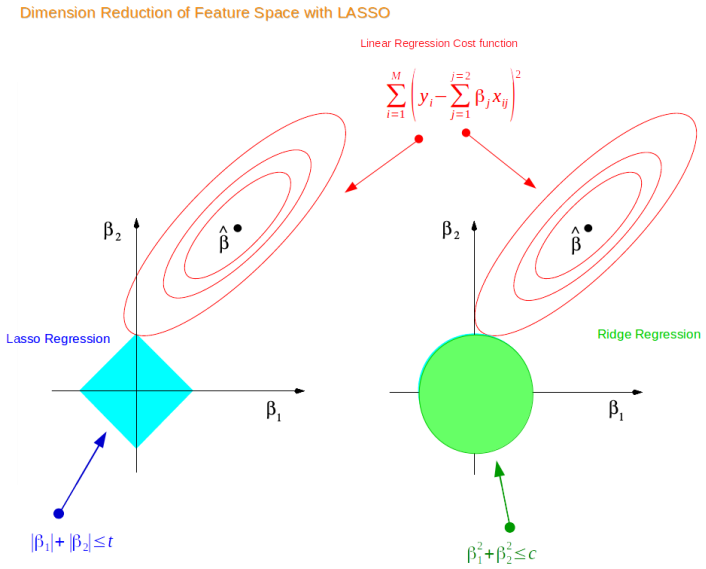
$$For\ c > 0, \quad \sum_{j=0}^{D} w_j^2 < c$$

When the penalty term is increased, if the coefficients take on large values then the Cost function is penalized. As the penalty term $\boldsymbol{\lambda}$ approaches 0, the cost function becomes similar to that of linear Regression. This is how Ridge regression reduces model complexity.

Training:

I used Ridge regression to see if I could reduce the model complexity, so the *Ridge* model from *sklearn* was used. I tuned the penalty term on the L2 norm of the weight vector described as alpha in the Ridge model. I used K fold cross validation, and I evaluated the performance on the validation set at each fold using root mean squared error.

LASSO REGRESSION



Just like Ridge regression which places a penalty on the L2 norm of the coefficients, Lasso Regression places a penalty on the magnitude of the coefficients. The cost function is thus,

$$J(w) = \sum_{i=1}^{N} (y_i - w^t x_i) + \lambda \sum_{j=0}^{D} |w|_j$$

Which is also equivalent to minimizing eq.xx such that,

$$\sum_{j=0}^{D} |w|_j < c, for\ c > 0,$$

Lasso Regression promotes sparsity and may place a hard constraint on the hypothesis, that is, some of the coefficients may be reduced to 0. From the figure above if we try to optimize the cost function *J(w)* subject to the constraints of ridge regression, the coefficients will may not be reduced to 0, but will take on smaller values. However in the case of lasso regression, the optimal *regularized* value of the weights w, will be at the point where the contour of J(w) touches the *peak* of the function of the constraint on the L1 norm[2].

Training

Sklearn's Lasso model was used for lasso regression. K fold cross validation was used to optimize the penalty term on the L1 norm of the coefficients.

ELASTIC NET (L1 AND L2 NORM)

Elastic Net uses the L1 penalty from lasso regression as well as the L2 penalty from ridge regression and also overcomes the shortcomings of both Ridge and Lasso Regression. The objective function to be minimized is,

$$J(w) = \sum_{i=1}^{N} (y_i - w^t x_i) + \lambda(\frac{1-\alpha}{2} \sum_{j=0}^{D} w_j^2 + \alpha \sum_{j=0}^{D} |w_j|)$$

Where α is the mixing parameter between ridge and lasso.

Training:

Elastic Net was trained using sklearns Enet. K fold was used to tune the parameter lambda in the equation above which is given as alpha in the Enet model.

RANDOM FOREST REGRESSOR

Random forest is an ensemble learning technique that uses bagging. Bagging is a process where the multiple trees are trained on subsets of the training set drawn with replacement. This is also known as bootstrap aggregating.

Random forest algorithm:

For each of the trees b=1 to B

- Draw a sample dataset $\acute{D}$ with replacement from D_{tr}.
- Train the tree b on the dataset $\acute{D}$:
    - For each region Rm

1. select a subset of d features at random.

2. Then select the optimal feature to split, the threshold of the feature and the value w.

3. Then split the region into two daughter nodes.

Then repeat until the Classification and regression tree halting conditions.

- Now get the set if trained trees b=1 to B.
- The final regressor is $\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f_b}(x)$

Training

For the random forest regressor, the number of trees, n_estimators and depth of each tree, max_depth were optimized using grid search which is nested cross validation on these two parameters. For each pair of parameters, K fold cross validation was performed.

ADABOOST REGRESSOR

Adaboost also known as adaptive boosting is a forward stagewise additive model which converts a set of weak learners into a strong one.

If we consider a binary classification problem with exponential loss,

At the m_{th} step, we need to minimize,

$$L_M = \sum_{i=1}^{m} \exp\left(\left[-\tilde{y}_i\left(f_{m-1}(x_i) + \beta\varphi(x_i)\right)\right]\right.$$

$$Let\ w_{i,m} = \exp\left(-\tilde{y}_i\acute{\beta}_m\varphi\left(\underline{x}_i, \underline{\gamma}_m\right)\right),\ \acute{\beta}_m > 0$$

$$\text{Then } L_m = \sum_{i=1}^{N} w_{i,m}\exp\left(-\tilde{y}_i\,\acute{\beta}_m\varphi(\underline{x}_i, \acute{\gamma}_m)\right)$$

Consequently the above equation can be rearranged to further provide the equations for the adaboost algorithm.

Training

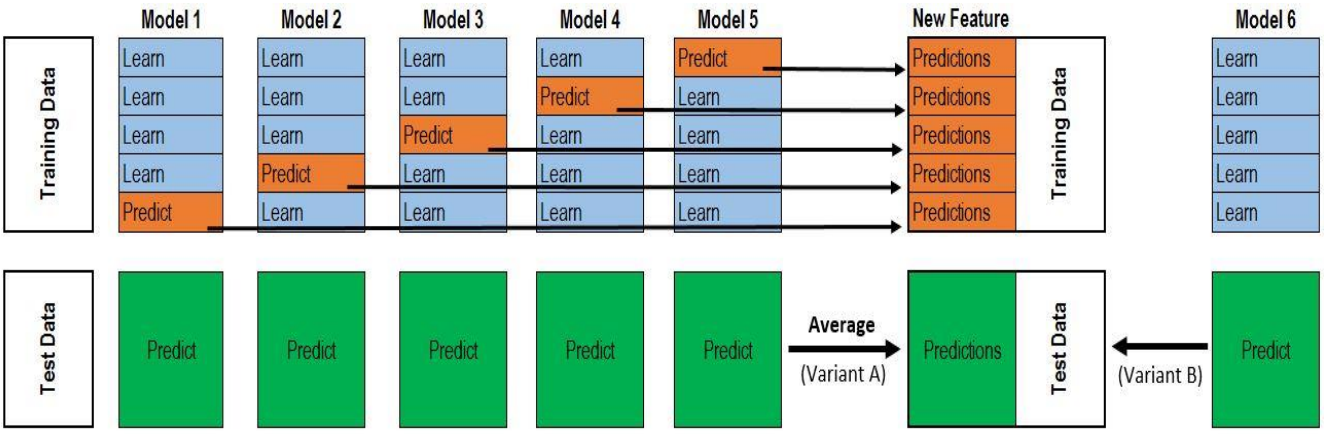
The AdaBoost regressor was trained on the training dataset. The learning rate and the number of estimators were optimized using grid search.

GRADIENT BOOSTING

In similar fashion as adaboost, Gradient boosting use an ensemble of weak learners to give a final prediction model. Gradient Boosting has also been shown as a form of iterative gradient descent in function space.

Training:
Gradient Boosting was also optimized over the number of estimators and the learning rate.

MODEL STACKING



We explore a technique used to improve model accuracy by stacking models and using their predictions as features which are learnt by a meta model. The stacked models make predictions on the test data as well, to generate a test set for the meta model which then uses the test set to make predictions[3].

The procedure for stacking is as follows:

1. For each of the M stacked models, use K fold cross validation, where the model learns from K-1 splits and makes a prediction for the Kth split. This is done over the entire training set N to generate features M_i where $i \in 1…M$. The new generated training data will have the same number of points as the training set N and will have M features corresponding to the number of stacked models.
2. Once we have our meta training data generated by the M stacked models, train the meta model on the meta training data.
3. Now for each of the stacked models M, make predictions for the test set.
4. Now use these predictions as the test data for the meta model and make the final prediction.

4. MODEL SELECTION AND COMPARISON OF RESULTS

Base model Results on training and test set

Base models for each of the regressors were trained on the training set *(Dtr)* and the performance on the training set and test set was recorded.

MODEL	RMSE ON TRAIN SET	RMSE ON TEST SET
Linear Regression	0.92596	1.02662
Ridge Regression	0.92599	1.02637
Lasso Regression	0.92597	1.02660
Elastic Net	0.92597	1.02659

Random forest	0.51028	1.09272
Gradient Boosting	0.31025	1.02919
AdaBoost	0.83408	1.06113

Regularized linear models hyperparameter tuning

The alpha parameter of Ridge, Lasso, Elastic Net and Kernel Ridge regression were tuned with a range of parameters of alpha using five fold cross validation, where root mean squared error was used as the error metric.

Alphas: 0.001, 0.005, 0.01, 0.05, 0.1, 0.3, 1, 3, 5, 10, 15, 30, 50, 70

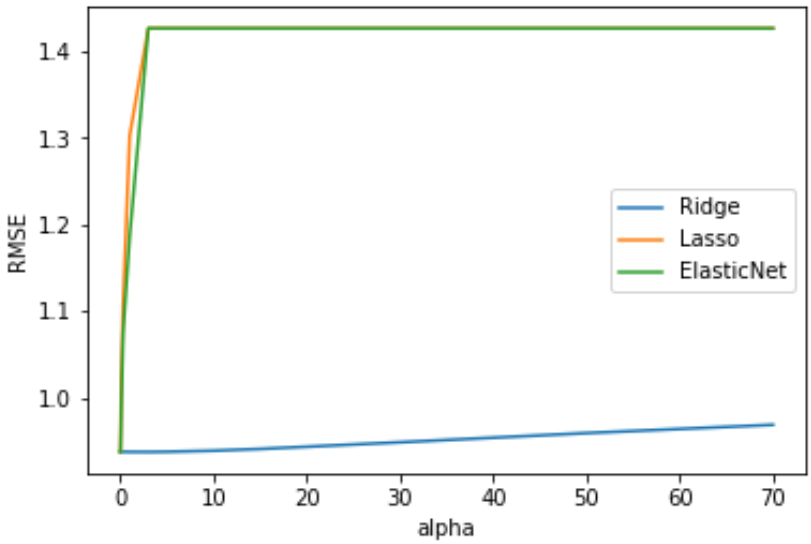


Figure 2 alpha vs mean Value of RMSE on 5 fold cross Validation on the training set

Regularized linear model	Best value of alpha after 5 fold Cross Validation	RMSE on test set
Ridge Regression	3.000	1.02601
Lasso Regression	0.001	1.02658
Elastic Net	0.001	1.02657

From the results above we can see that the value of alpha for Lasso and Elastic net are very low, which is also evident from the graph of alpha vs mean value RMSE for five fold Cross Validation at each value of alpha. We can see from the graph that the mean squared error values drastically increase as the penalty term alpha on the L1 norm is increased. This tells us that the constraint on the L1 norm does not have a good effect on the generalization error. Conversely the Ridge regression and Kernel ridge regression have a very slow linear increase in RMSE as alpha increases and optimal values for Ridge regression and

Kernel Ridge regression were found at alpha =3 and alpha=1 respectively, which indicates that some constraint on the L2 norm is actually preferred.

Random forest and boosting methods parameter tuning

The number of trees (n_estimators) and max depth of each tree were tuned for the random forest regressor.

N_estimators: 10, 50, 100, 1000, 3000

Max depth: 3, 4 ,5 ,6

Similarly for Adaboost and Gradient Boosting I optimized over n_estimators and the learning rate

n_estimators: 10, 50, 100, 1000

Learning rate: 0.01, 0.05, 0.3, 0.1, 1

Model	n_estimators	Max_depth	Learning rate	RMSE on test set
Random Forest	50	6	N/A	1.00249
AdaBoost	3000	N/A	1	1.13350
Gradient Boosting	1000	N/A	0.01	1.00248

The boosting methods perform better when the number of weak learners (n_estimators) is relatively high.

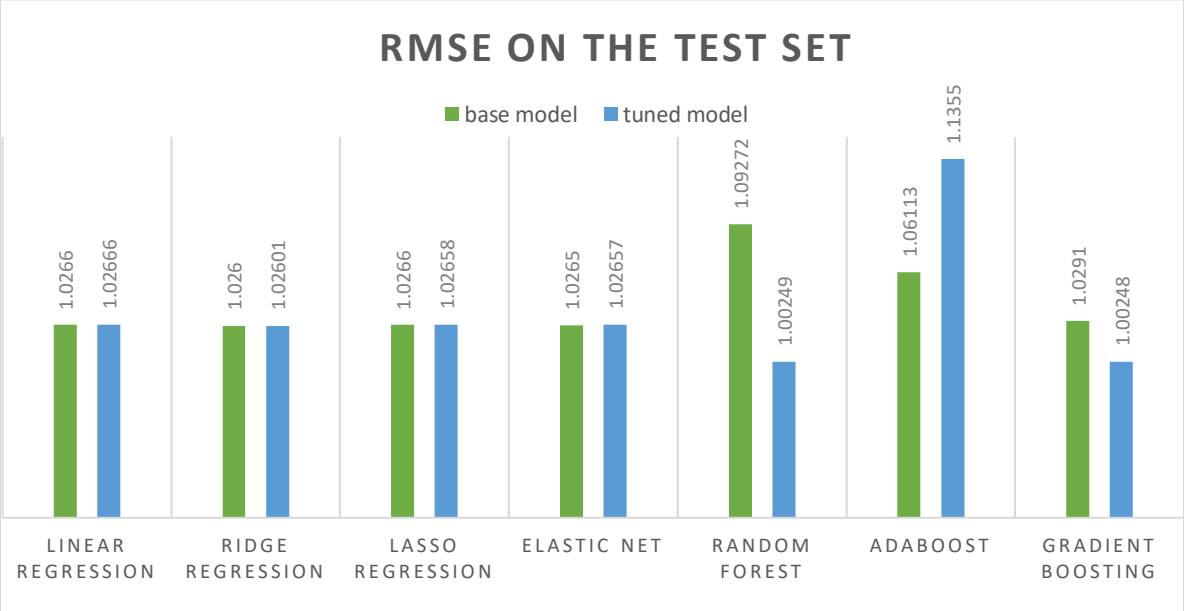


Chart 1: Baseline model and Tuned model Generalization error

Stacking Models

I used model stacking to see if I could obtain a better out of sample error on the Test data set. For this purpose, model stacked is a very effective approach. I used Elastic Net which has L1 and L2 regularization, the boosting methods Adaboost and Gradient Boosting and the random forest regressor, thus incorporating a multitude of regressors to generate our meta features. For our meta model, I used lasso Regression which imposes a penalty on the L1 norm. All the stacked models and the meta model were trained using the previously determined optimal hyperparameters found during cross validation and grid search.

	ElasticNet	Gradient Boosting	AdaBoost	random forest regressor
0	5.996729	4.506217	3.996909	4.167667
1	3.344604	3.365895	3.958183	3.322000
2	4.781019	4.587726	4.665686	5.300333
3	4.090506	4.269137	4.138807	4.166000
4	3.667174	3.404507	4.243959	3.353333

First five samples of the meta features. The predictions of the stacked models are the columns

5.FINAL RESULTS AND INTERPRETATION

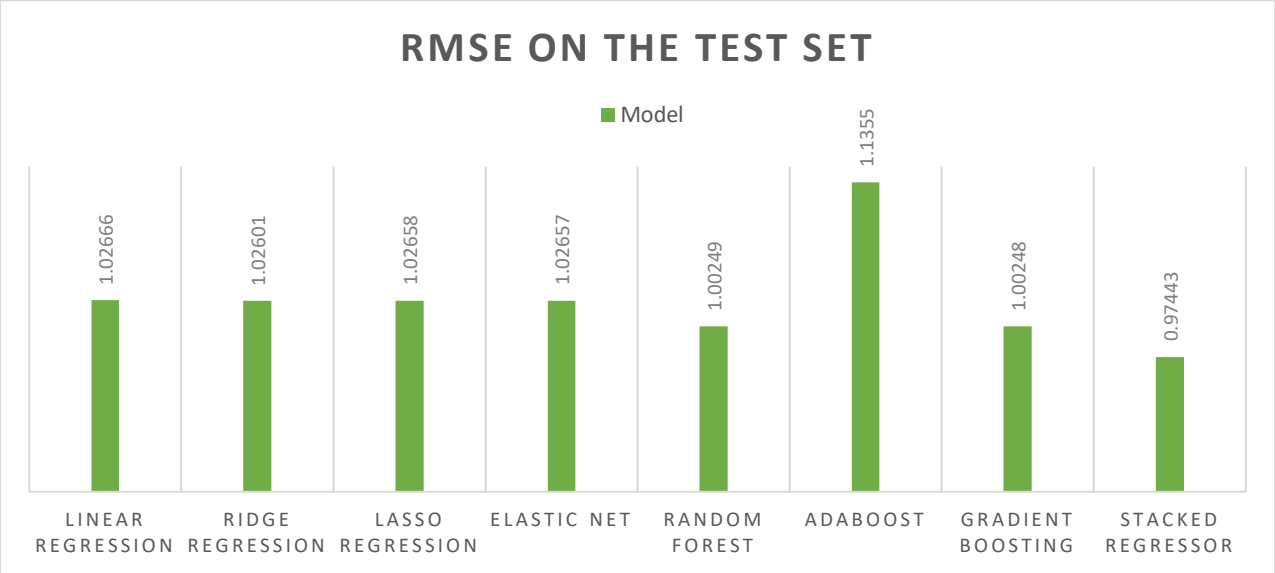


Chart 2: Generalization error (RMSE) on the Test set

The stacked Regressor which combines all of the other regressors has the best performance on the Test set so far.

It was found that higher penalties on L1 regularization was causing the model to severely underfit the data, whilst L2 regularization allowed for some flexibility. Elastic Net which uses both L1 and L2 regularization, had the lowest generalization error among the linear models.

The random forest regressor tends to over fit the data performing poorly on the test set when the depth of each tree is increased, but the generalization error does not increase when the number of estimators is increased, in fact it reduces the variance in the generalization error.

All of the models used performed better than the baseline models when their parameters were tuned. Many of these models were *strong* learners, such as Gradient Boost. But in using the predictions of these models I was able to create an even better model that reduced the generalization error. The stacked regressor outperforms all the other models and has the lowest generalization error as seen from chart 2.

6.SUMMARY AND CONCLUSIONS

A lot of useful insights were made known from the dataset by conducting regression analysis. From linear models to adaptive basis function models such as Random Forest and Boosting methods, the performance and Generalization error for each was determined. Model Stacking was a very effective ensemble learning technique that I learnt and implemented, which further improved the generalization error.

7. REFERENCES

[1] Machine Learning: *A probabilistic Perspective*, Kevin P. Murphy

[2] https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b

[3] Stacked Regressions : Top 4% on LeaderBoard | Kaggle

[4] https://medium.com/@pruchka/intro-to-model-stacking-in-python-70814e95d7a1