JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Comparative Analysis Of Stacked Models And Other Machine Learning Algorithms In Estimating Claim Cost For Automobile Insurance

[1]Jitesh Rawat, [2]Satya Sai Baba Mudigonda, [3]Rohan Yashraj Gupta, [4]Phani Krishna Kandala, [5]Pallav Kumar Baruah

[1]M.Sc Data Science & Computing, [2] Adjunct Professor, [3]Visiting Faculty
[4]Visiting Faculty, [5]Professor,
Department of Mathematics and Computer Science, Sri Sathya Sai Institute of Higher Learning, Prashanti Nilayam, India

Abstract : Estimation of claim cost for setting the premium is one of key issues in insurance ratemaking. This type of analysis is aided by big data and the use of machine learning models with it improves predictive capability. We created Stacked models to estimate claims cost, and have performed a comparison of its efficacy with various machine learning models. A comparative analysis of performance with Generalized Linear models, Decision tree, Random forest regressor and Gradient Boosting regressor is reported in this paper. We demonstrate that the stacked models indeed perform better than any of the individual models. We also show how our applications scale well when implemented with pyspark on big data.

***IndexTerms* - Big data, Vehicle insurance, Machine learning, Regression, Stacked models, Pyspark, Decision tree, Random forest, Gradient boost, Generalized linear models.**

I. INTRODUCTION

Several academics in the actuarial and insurance literature suggested using generalized linear regression models (GLM) to model claim expenses as a function of risk factors. Not only do risk considerations play a part in insurance ratemaking[1], but so does predicting claims costs. In today's world of big data, the capture, storage, and transmission of enormous data can help with risk assessment, generating subtle forecasts, and making smarter judgments[2].This research is based on general insurance pricing for automobile insurances, where we utilize the custom implementation of stacked models in pyspark as well as several other machine learning methods to estimate claim costs for auto-mobile insurance.As we can see in today's society, the rise of Big Data has made it feasible to address answers to many problems that were previously insurmountable. One of them is estimating claim costs based on historical data. Additionally, the convenience of data with various data gathering methods aids us in formulating the method for the desired result. Because claims cost distributions have traits like positive support and right skewness, implementation of such distributions using Generalized linear models are widely used in insurance literature[3]. Our focus is to compare this distribution model with other regression algorithms which generally perform well in the machine learning area.

The study's dataset is made up of synthesis data from New Zealand's Crash Analysis System. The data includes entries from New Zealand police reports on crashes.From January 2000 through 2021, data is currently available. The crash variables in the dataset are non-personal. Around 7.5 million records with varied characteristics make up the dataset. The goal of this task is to check that the ratemaking factors for our response variable, claim amount, are correct.

This paper has 7 sections including introduction. Section 2 contains the literature study. The figure 1 in section 3, which is methodology, talks about the flow of work starting with data description and regression models. Section 4 is about the performance metrics considered in the work to evaluate the models. Section 5 proposes the results and analysis of the work. Section 6 contains the implementation and training time analysis of stacked models in python vs pyspark. In the last Section 7 contains the conclusion of the study.

II. LITERATURE REVIEW

Navarun Jain's research article is titled "Towards Machine Learning: Alternative Methods for Insurance Pricing – Poisson-Gamma GLM, Tweedie GLM, and Artificial Neural Networks"[4]. The author discusses three methods for predicting claim severity: generalized linear models, Tweedie compound models, Poisson-Gamma distributions, and artificial neural networks. It also includes a section on estimating the frequency of claims. The paper presents four methods for comparing these models, including MSE mean squared error, AIC, residual plot analysis, and fivefold cross validations. The Poisson-Gamma GLM model scored better in terms of MSE for test data, despite the author's conclusion that Tweedie distribution and neural networks offer significant potential for pricing claims. In our study we would focus on comparing Tweedie GLMs with other machine learning algorithms in a big data platform.

Arthur Charpentier's book "Computational Actuarial Science with R" [5] provides an in-depth look into General Insurance Pricing and the use of Generalized Linear Models with exponential distributions like Gamma, Poisson, Tweedie, and others in estimating and ratemaking for insurance. Various approaches for estimating individual claims, as well as Gamma regression and modeling compound sum with Tweedie regression, are discussed in Chapter 14 of the book.

The paper "Generalized Linear models in vehicle insurance" [6] by Silvie Kafková and Lenka Kivánková estimates annual claim frequency using GLMs and compares predictor variables using analysis of deviance and AIC (Akaike information criteria). The R environment was used for the analysis and prediction.

Mohamed Hanafy and Ruixing Ming published "Machine Learning Approaches for Auto Insurance Big Data" [7]. A claim occurrence prediction analysis. This was conducted using the machine learning techniques Decision trees, XGBoost, Naive bayes, and four additional algorithms on a big data platform. The problem was designed as classification.

In this study we address the similar approach to the previous paper. However, we are dealing with a regression problem of estimating claims costs with additional custom implementation of stacked models in pyspark as well as its comparative analysis.

III METHODOLOGY

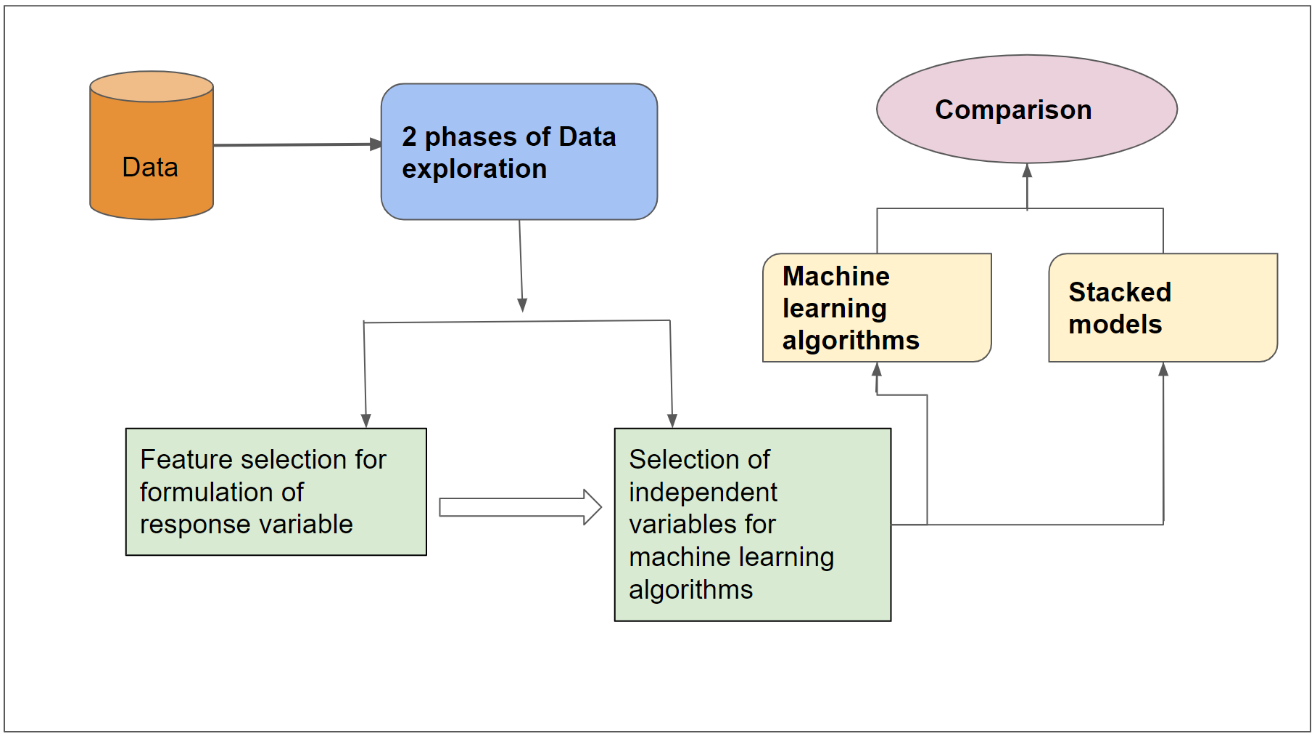

Data
2 phases of Data exploration
Comparison
Machine learning algorithms
Stacked models
Feature selection for formulation of response variable
Selection of independent variables for machine learning algorithms

Figure 1 Workflow

From figure 1 we can see that in the initial stage we prepare the data for the model and in the 2nd phase we perform feature selection for our data. To do so we use the Crash Analysis System's dataset which has information about New Zealand crashes and perform data synthesis to formulate our response variable which is claims cost in this case. To keep this synthesis as close approximate to real world data as possible we make sure that there are large numbers of no claims which is generally the case. The data distribution follows a Tweedie distribution as we can see below.
Tweedie distributions are common in actuarial research as part of GLMs[8], as well as other fields such as healthcare for cancer metastasis and genomic testing. Tweedie distribution can be used wherever there is a mix of zeros and non-negative data points[12].

3.1 Data description

Although the data provides 72 collision elements, we are only interested in the variables that influence the ratemaking for third-party responsibility claims for auto insurance.The response variable is the result of data synthesis performed using features of the CAS dataset these features are considered after an Exploratory data analysis which resulted that the following features shown in table 1 can be considered as independent variables :

Feature	Description
carStationWagon	Derived variable to indicate how many cars or station wagons were involved in the crash.
ditch	Derived variable to indicate how many times a 'ditch' or 'waterable drainage channel was struck in a crash.
fatalCount	A count of the number of fatal casualties associated with this crash.
minorInjuryCount	A count of the number of minor injuries (inj) associated with this crash.
seriousInjuryCount	A count of the number of serious injuries (inj) associated with this crash.

speedLimit	The speed (spd) limit (lim) in force at the crash site at the time of the crash. May be a number, or 'LSZ' for a limited speed zone.
vanOrUtility	Derived variable to indicate how many vans or utes were involved in the crash.
urban_Open	To indicate whether the road was open road or not
urban_Urban	To indicate whether the road was urban or not
postOrPole	Derived variable to indicate how many times a post or pole was struck in the crash. This includes light, power, phone, utility poles and objects practically forming part of a pole (i.e. 'Transformer Guy' wires)
NumberOfLanes	The number(num) of lanes on the crash road.
weather	Indicates weather at the crash time/place. See wthr_b. Values that are possible are 'Fine', 'Mist', 'Light Rain', 'Heavy Rain', 'Snow', 'Unknown'.

Table 1 : Data description

Because the variables used to simulate our response variable have a direct association with it, they are removed, and the rest of the data is designated as our predictor features. During additional processing, more than 80% of independent variable features with missing values are removed, and categorical variables are converted to numerical using one hot encoding. The dataset was then split into two sets, one for training and one for testing, with an 80:20 split ratio.

3.2 Regression models

The regression models are trained using the train set and validated using the test set. There are 5 regression approaches used for estimation of claim cost - GLMs with Tweedie distribution[9], Random forest, Gradient boost, Decision tree and Stacked models approach. We have compared these model performances based on errors mainly mean absolute error, mean square error, root mean square error and r square. In the below section stacked models used in this study are explained.

3.3 Stacked models.

This section contains the explanation about the stacked model implementation that is performed in pyspark. Model stacking is a technique for improving model predictions by mixing the outputs of numerous models and putting them through a meta-learner, which is a machine learning model[10]. The main idea of stacking is using predictions of machine learning models from the previous level as input variables for models on the next level[16]. This is a type of ensemble machine learning model often quite popular at kaggle competitions. In this study, a set of stacked models is devised, and the outcomes and assessment metrics are compared, as well as how these models perform on prediction. In figure 2 we can see a sample stacked model design in our work.

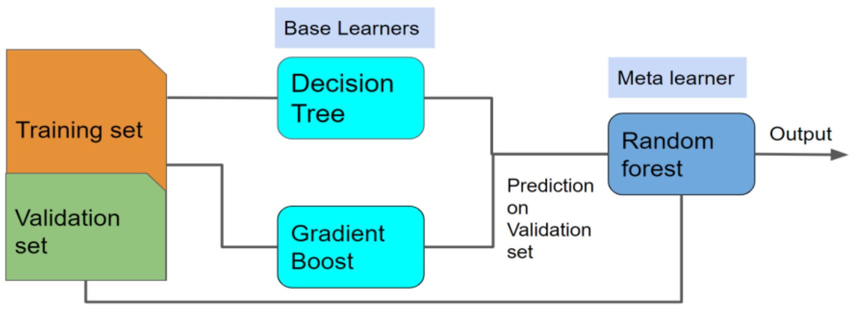


Figure 2 Sample stacked model

Algorithm

1. The training data is divided into k folds and validation data equal parts depending upon the number of models.
2. The base learner algorithms are trained upon the k folds.
3. Then the predictions of these base learners are added to validation data and trained with the final estimator also known as meta learner.

IV. Performance metrics

Performance of a regression model is generally carried out by understanding how well the model is able to explain the dependent variable and by loss functions. In our study we would consider different loss functions as measures of performances by both individual and stacked models.

MSE - A regression line's mean squared error (MSE) indicates how near it is to a set of points. It accomplishes this by squaring the distances between the points and the regression line (these distances are the "errors")[13] which can be observed in below formula :

$$MSE = (1/n) * \Sigma(y - y\hat{})\hat{}2 \qquad (4.1)$$

RMSE - The root mean square error (RMSE) is the residuals' standard deviation (prediction errors). The RMSE is a measurement about how spaced out the residuals are, and the residuals are a metric of how much the data points stray from the regression line. The formula can be shown as :

$$RMSE = [\Sigma(x - x)2/N]^{\frac{1}{2}} \qquad (4.2)$$

MAE - The level of mistake in your measurements is known as absolute error. It's the distinction between what's measured and what's true value[14]. The average of these absolute errors is mean absolute error. We can observe the formula for mean absolute error as :

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} | x_i - x | \qquad (4.3)$$

R squared - In a regression issue, the R squared value is a statistical measure that indicates how much variance is explained by the independent variables[15]. Let SSres be the sum of squared residuals. And SStot to be the total sum of squares. Then the R squared value is observed as :

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \qquad (4.4)$$

V. Results and Analysis

Individual models are developed in pyspark with the spark MLlib module[20]. The MSE, RMSE, MAE and R squared metrics are used to analyze the performance. MAE would be the primary emphasis statistic (Mean Absolute Error)[11].

5.1.1 Generalized linear model with tweedie distribution.

Implementation tweedie regression on training data and evaluation using a test dataset is carried out. Even though our response variable, the claim amount, is observed to follow the tweedie distribution, there is some form of non-linearity that our model is not able to match with. The outcome and actual vs prediction can be observed as :

MSE	293708981801.53
RMSE	541949.24
MAE	306179.94
R^2	0.7466

Table 2 : GLM's performance metrics

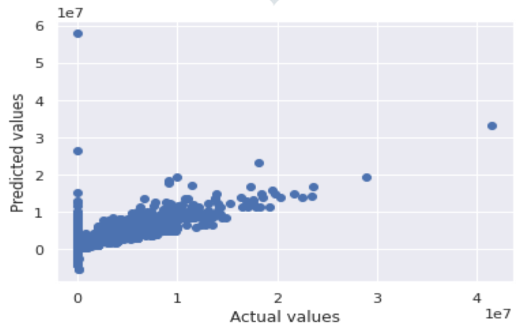


Figure 3 Actual vs Predicted values for Generalized linear models

- There is little collinearity between the real and projected values, and there is a lot of bias.
- From table 2 we see that the R^2 score indicates that the model can explain 74% of the variance around the mean for the dependent variable.

5.1.2 Random forest

Random forest regressor uses ensembles of decision trees on random samples of training set[17]. In our case we consider our number of trees to be 50 and max_depth of tree to be 10 i.e. it can go deep upto 10 nodes. The results can be observed as :

MSE	91498927705.16
RMSE	302487.89
MAE	106959.84
R^2	0.9327

Table 3 : Random forest's performance metrics

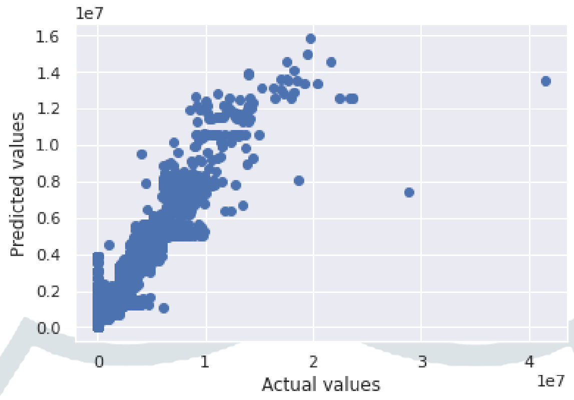
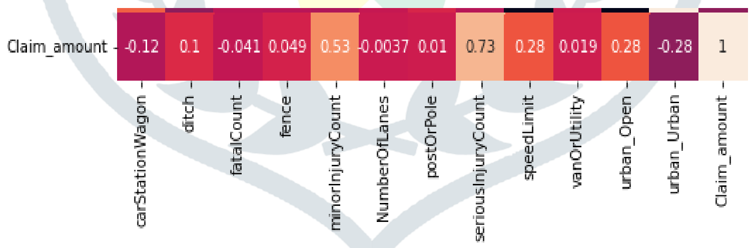


Figure 4 Actual vs Predicted values for Random forest regressor

- It is evident from the real versus forecasted graphic in figure 4 that random forest can diminish bias and increase collinearity with the regression trees.
- From the Rsquared value in table 3, it's apparent that random forest outperforms GLM, as it accounts for 93 percent of variance.
- Another benefit of the random forest algorithm is it enables us to get feature importances through gini gain calculations[19]. This is helpful in terms of dimension reduction if required.
- We will not undertake dimension reduction in our research, instead focusing on the importance of attributes and the relationship between these variables and the claim amount.
- The features that were ranked high in terms of importance are :
  bus , carStationWagon , fatalCount, fence, guardRail, minorInjuryCount, NumberOfLanes, overBank, vanOrUtility, weatherA, weatherB, urban_Open, urban_Urban.

  Using a correlation heatmap shown in figure 5, the association of these features with Claim amount can be perceived:



	carStationWagon	ditch	fatalCount	fence	minorInjuryCount	NumberOfLanes	postOrPole	seriousInjuryCount	speedLimit	vanOrUtility	urban_Open	urban_Urban	Claim_amount
Claim_amount	-0.12	0.1	-0.041	0.049	0.53	-0.0037	0.01	0.73	0.28	0.019	0.28	-0.28	1

Figure 5 Correlation heatmap of variables with Claim_amount

- The variables showing most correlation through feature importance from random forest can be observed to match with the variables chosen earlier for modeling.
- The correlation of Claim_amount with seriousInjuryCount and minorInjuryCount appears to be more than 0.5.
- Then, with the exception of speedLimit and urban, there doesn't appear to be much of a correlation with our response variable.
- In this way, in terms of dimension reduction modeling, using only the most significant variables can yield results that are roughly equal.

5.1.3 Gradient Boosting

In the same manner how above individual machine learning algorithms were modeled, gradient boosting regression algorithm is fit to our data.

MSE	68555604303.35
RMSE	261831.25
MAE	100401.92
R^2	0.9499749527695116

Table 4 : Gradient boost's performance metrics

- Gradient Boosting, as the best and most efficient assembling strategy, is able to perform well.
- As the name implies, the purpose of this approach is to use regression trees to adjust to non-linearity in the data.
- It is evident that GBTrees are the most effective for our data so far. From table 4, we see that R^2 value seems to be slightly better than random forest and it explains 94% of variance.

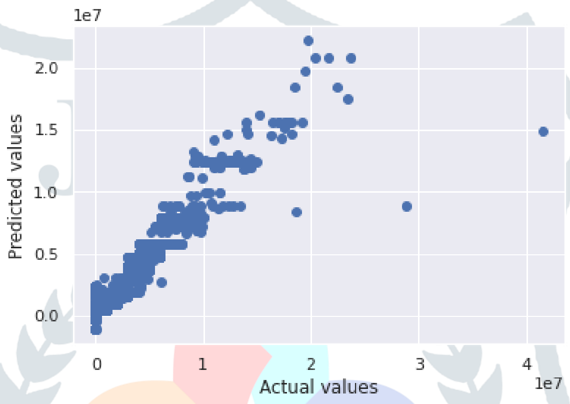


Figure 6 Actual vs Predicted values for Gradient boost regressor

- From the actual vs predicted in figure 6, clearly the dispersion is good enough although gradient boosting trees fail to recognise that it has to predict "0" whenever there are no claims.

5.1.4 Decision Tree

Decision Tree regressor trains well on training data and predicts well but overfits[18]. This is the main reason people use ensemble models so that the overfitting can be dealt with. The metrics can be seen as :

MSE	69552397986.17
RMSE	263727.88
MAE	98809.41
R^2	0.954727587979342

Table 5 : Decision tree's performance metrics

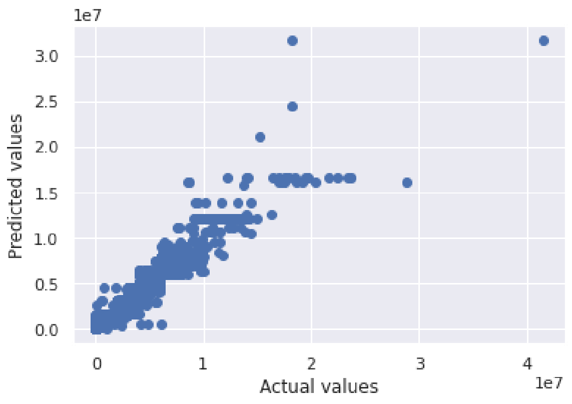


Figure 7 Actual vs Predicted values for Decision tree regressor

- Other algorithms perform better, but decision trees are light and quick to train, which is why they are chosen in some cases with a lot of data and a need to train quickly.
- The decision tree revealed that fatalCount, minorInjuryCount, seriousInjuryCount, speedLimit, and urban Urban were the most important features for prediction.

5.1.5 Evaluation

All in all if the MSE, RMSE and MAE values of the individual models are looked at:

Algorithm	MAE	RMSE	MSE	R^2
GLM with Tweedie	306179.94	541949.24	293708981801.53	0.7466
Decision Tree	98809.41	263727.88	69552397986.17	0.9327
Random forest	106959.84	302487.89	91498927705.16	0.9499
Gradient Boost	100401.92	261831.25	68555604303.35	0.9547

Table 6 : Evaluation of machine learning algorithm's metrics

- In terms of mean absolute error from table 6, the decision tree outperforms all other models considering their loss functions.
- Furthermore, when comparing rmse and mse for gradient increase, rmse appears to be superior.
- Because our data contains many outliers, or extreme values, mean absolute error is our focus metric because it appears to eliminate the effect of outlier predictions.
- Even after performing so well on predictions, the gradient boost overlooks the phase when it should consider no claims and assign a claim cost of 0 to the related claim.
- And on the other hand random forest is able to learn such rules and predicts the corresponding claim cost to be 0 correctly.
- This proves the point that no matter what loss function is considered there is a possibility that individual models fail to generalize on the data.
- And this leads to the introduction of Stacked models which are capable of using the strengths of each individual model and provide proper generalization over the data for unseen data.

5.2 Stacked models

5.2.1 Stacked model 1

The first combination is created with an ensemble of decision trees and gradient boost as base learners and then a Generalized linear model with tweedie regression as our meta learner.

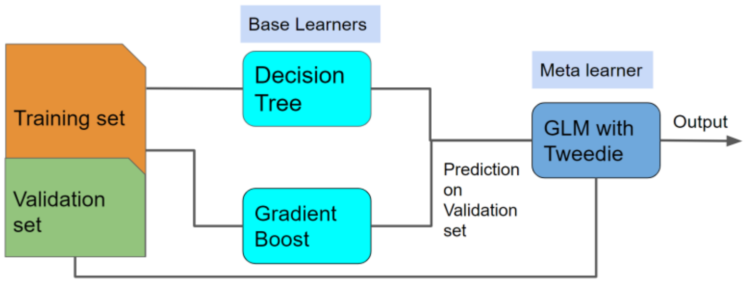


Figure 8 Stacked model 1 design

The results can be observed as :

MSE	62989763194.70
RMSE	250977.61
MAE	104480.47
R^2	0.9574

Table 7 : Stacked model 1 performance metrics

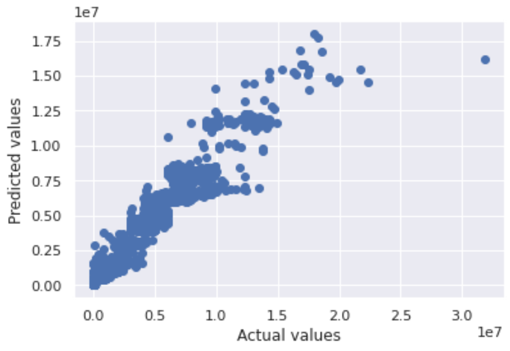


Figure 9 Actual vs Predicted values for Stacked model 1

- We can deduce from the data that glms convert discrete decisions made by tree-based algorithms into continuous results.
- This is why a multi-decision tree model with GLM has a greater MAE value as seen in table 7 than a single decision tree model. In terms of RMSE, however, the stacked model outperforms all other models.

5.2.2 Stack model 2

GLM and Decision Tree are our foundation learners in this combination. They were chosen because GLM with Tweedie ensures consistency in output decision-making, and Decision tree outperformed the other models individually. Then there's Random Forest, which acts as a meta learner.

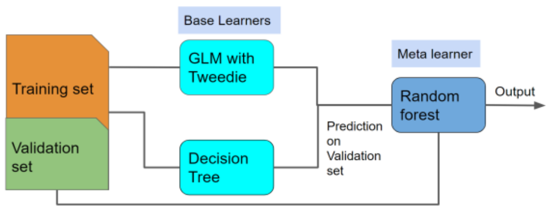


Figure 10 Stacked model 2 design

MSE	65803352320.86
RMSE	256521.64
MAE	99357.12
R^2	0.9555

Table 8 : Stacked model 2 performance metrics

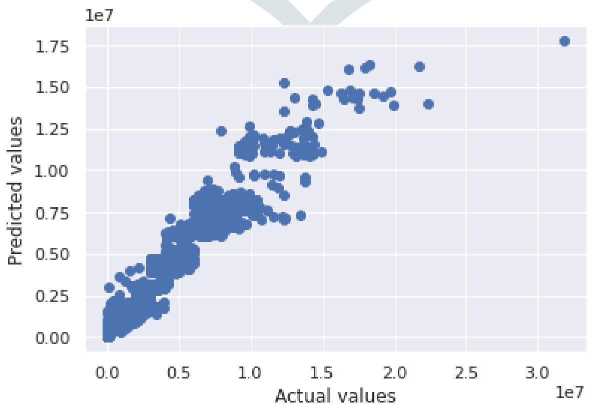


Figure 11 Actual vs Predicted values for Stacked model 2

- All in all it is observed that these models are performing pretty well except the deal with extreme values or outliers as seen in figure 11.

5.2.3 Stack model 3

The final model is an ensemble of decision tree algorithms. So considerations are Decision tree and Gradient boost as the first part of stack and Random forest as final estimator.

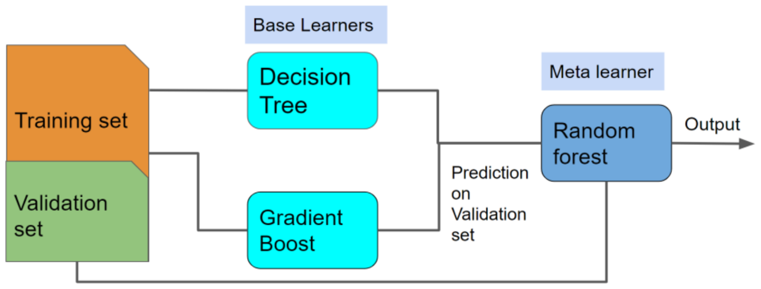


Figure 12 Stacked model 3 design

MSE	64475563937.20
RMSE	253920.38
MAE	98874.49
R^2	0.9552

Table 9 : Stacked model 3 performance metrics

- Finally, the ensemble of decision tree algorithms alone performs slightly better than any individual models in terms of both MAE and R^2 as observed in table 9.

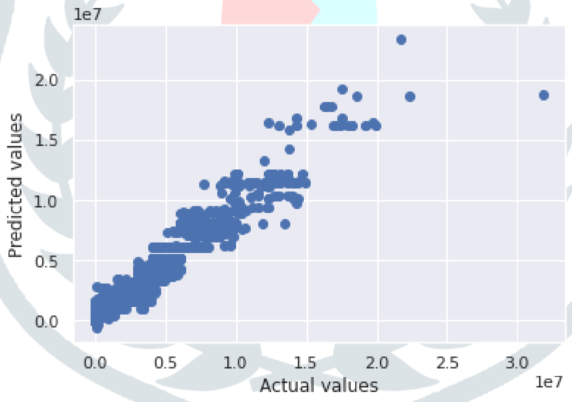


Figure 13 Actual vs Predicted values for Stacked model 3

- The R^2 value from table 9 tells that these models are able to explain 95% of variance from the mean of our response variable. Also we can see that the same is reflected from figure 13.
- Training time for these models for a huge quantity of data takes longer in conventional python, it's presented in the next chapter how pyspark implementation is significantly faster.

VI. Implementation and Performance Analysis.

In this section we are discussing how the implementation of a stacked model for large amounts of data works in python and pyspark. Scikit learn was used to implement it in python. Whereas in pyspark, as we had no readily available library for stacked models we have to go forward with custom implementation of training and transformation pipelines.
The data processing and modeling is performed on google colab notebook. The colab provides 8gb of free memory on a single core processor. In terms of pyspark the available memory would be considered as a single node where all partitions will be processed. The motive of this study was to see how much training time was required for varying numbers of data points, ranging from 1 lakh to 5 million rows. Also, in both the Python and Pyspark frameworks, to observe how long it takes the model to train on different sizes. The below plot shows the training time for both.

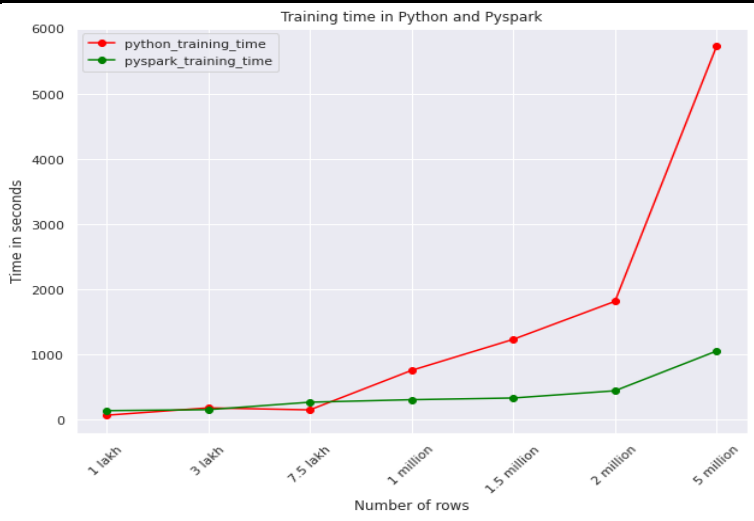


Figure 14 Python vs Pyspark stacked models training time

From figure 14 it is observed that with an increase in data points how the train time increases in normal python whereas in pyspark being a distributed framework its relatively less time. As the quantity of the data grows larger, the training time disparity increases after 7.5 lakh rows. And this will continue as data sizes expand, as training in standard Python will be challenging, necessitating the use of pyspark for large datasets.

VII. Conclusion

In conclusion, the individual models tend to perform well but either there is a loss of information and or issues related to overfitting. It was observed how decision trees individually could give a good result but lacked explanation of variance whereas in stacked models we could provide almost the same mean absolute error with better R squared value. Therefore the stacked models provided a better approach in generalization while learning. The advantage of stacked models is that they capture the data's generality, allowing it to avoid overfitting and learn more than individual models can. Therefore the combined strength of availability of big data and the stacked machine learning models provided the opportunity to combine strength of both to get better results. Besides, we saw the effect of implementation of this approach in a big data framework like pyspark reduces training time approximately 10 times relative to python.

Acknowledgement

We dedicate this work to the Revered Founder Chancellor of Sri Sathya Sai Institute of Higher Learning, Bhagawan Sri Sathya Sai Baba. We acknowledge expert advice received from Mr. T.V.S Aditya, Actuarial Data Scientist.

References

[1] Sukono, Riaman, E. Lesmana, R. Wulandari, H. Napitupulu, and S. Supian, "Model estimation of claim risk and premium for motor vehicle insurance by using Bayesian method," in IOP Conference Series: Materials Science and Engineering, Feb. 2018, vol. 300, no. 1. doi: 10.1088/1757-899X/300/1/012027.
[2] H. Hassani, S. Unger, and C. Beneki, "Big data and actuarial science," Big Data and Cognitive Computing, vol. 4, no. 4, pp. 1–29, Dec. 2020, doi: 10.3390/bdcc4040040.
[3] Resti, "ESTIMATION OF CLAIM COST DATA USING ZERO ADJUSTED GAMMA AND INVERSE GAUSSIAN REGRESSION MODELS," Journal of Mathematics and Statistics, vol. 9, no. 3, pp. 186–192, Mar. 2013, doi: 10.3844/jmssp.2013.186.192.
[4] N. Jain and D. Brillinger, "Towards Machine Learning: Alternative Methods for Insurance Pricing-Poisson-Gamma GLM, Tweedie GLM and Artificial Neural Networks."
[5] Charpentier and Arthur, "Computational Actuarial Science with R."
[6] S. Kafková and L. Krivánková, "Generalized linear models in vehicle insurance," Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis, vol. 62, no. 2, pp. 383–388, 2014, doi: 10.11118/actaun201462020383.
[7] M. Hanafy and R. Ming, "Machine learning approaches for auto insurance big data," Risks, vol. 9, no. 2, pp. 1–23, 2021, doi: 10.3390/risks9020042.
[8] "Predictive Modeling Applications in Actuarial Science, Volume II: Case Studies in Insurance."
[9] M. Koissi, A. Professor, and V. Whitledge, "Emerging Data Analytics Techniques with Actuarial Applications Innovation and Technology 2 Emerging Data Analytics Techniques with Actuarial Applications HERSCHEL DAY," 2019.
[10] F. Güneş, R. Wolfinger, and P.-Y. Tan, "Stacked Ensemble Models for Improved Prediction Accuracy," 2017.
[11] Chai, Tianfeng & Draxler, R.R.. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?– Arguments against avoiding RMSE in the literature. Geoscientific Model Development. 7. 1247-1250. 10.5194/gmd-7-1247-2014.
[12] "Tweedie Distribution: Definition and Examples." 2016. Statistics How To. https://www.statisticshowto.com/tweedie-distribution/.
[13] Botchkarev, Alexei. 2019. "Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology." Arxiv 1 (1): 37. https://arxiv.org/abs/1809.03006.
[14] "Absolute Error & Mean Absolute Error (MAE)." 2016. Statistics How To. https://www.statisticshowto.com/absolute-error/.

[15] “RMSE: Root Mean Square Error.” n.d. Statistics How To. Accessed June 12, 2022. https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/.
[16] B. Pavlyshenko, "Using Stacking Approaches for Machine Learning Models," 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), 2018, pp. 255-258, doi: 10.1109/DSMP.2018.8478522.
[17] Liaw, Andy & Wiener, Matthew. (2001). Classification and Regression by RandomForest. Forest. 23.
[18] Kumar, Satyam. 2021. “3 Techniques to Avoid Overfitting of Decision Trees | by Satyam Kumar.” Towards Data Science. https://towardsdatascience.com/3-techniques-to-avoid-overfitting-of-decision-trees-1e7d3d985a09.
[19] Lewinson, Eryk. 2019. “Explaining Feature Importance by example of a Random Forest.” Towards Data Science. https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e.
[20] Bandi, Raswitha & Jayavel, Amudhavel & Karthik, R.. (2018). Machine Learning with PySpark - Review. Indonesian Journal of Electrical Engineering and Computer Science. 12. 102-106. 10.11591/ijeecs.v12.i1.pp102-106.