Surya Pugazhenthi

Prof. Crawford

Math-160-9652

27 November 2023

AI Modeling with Linear Regression vs. Random Forest Algorithms

In the modern landscape of technology endeavors, in which discoveries in artificial intelligence (AI) and machine learning (ML) are coming so quickly, researchers and businesses are all working to improve existing AI models to handle large-scale operations with greater efficiency. Research and development on AI models existed as early as the 1950s. However, data scientists and engineers have widely used ML algorithms such as “Linear Regression” and “Random Forest” to model datasets. Both algorithms are used similarly to classify tasks but differ in efficiency, analytical functionality/capability, and complexity of interpretation. A comparative analysis between both models in a specific use case will help more closely examine their similarities and differences. In the case of this analysis, a gigantic dataset of Amazon product reviews (from 2018), specifically in the “Office Products” category, is used for the algorithms to parse through and give a predicted measure or label of the review’s sentimental feedback. Jianmo Ni, a Ph.D. Computer Scientist provides this portion of the dataset as free for class project-related work since it is smaller than the complete dataset and is easier to work with despite still being gigantic (over eight hundred thousand reviews!). Both models use a random sample of ten thousand reviews to analyze and make their respective sentiment predictions for one random product review. Unfortunately, the smaller version of the dataset does not include the product's name, but sometimes context clues are left within each review to guess what the product could be about. Knowing the name of the products is not the purpose of this comparative

analysis, but understanding how both ML algorithms lead to their predicted sentiment conclusions is what truly matters.

The beginning of advancements in ML algorithms took place with linear regression between 1950 and 1980 when interest in learning models picked up. The algorithm is beginner-friendly enough to provide entry-level experience for data science and ML researchers. The fundamentals of the linear regression model can help pave the way to understanding more complex learning models that require greater emphasis on data manipulation. Regression is a statistical method used in modeling the relationship between a dependent and an independent variable. Regression-based models are primarily used for predictions and identifying cause-and-effect relationships between variables. A simple linear regression includes only one independent variable, and a linear relationship exists between the independent, typically 'x' variable and the dependent, typically 'y' variable. The relationship in the data points can be modeled with a "line of best fit" using a linear equation: "y = mx+b." Linear regression aims to find the best coefficient values (b_0, b_1,..., b_n) for minimizing the difference between the experimental (true) values and the predicted values. The function is also referred to as the "cost function." Optimizing the model for accuracy is done using the sum of the squared differences between the two values, as mentioned earlier, and then divided by the total number of data points. The coefficients are acquired during the training stage of the model through iteration to minimize the error difference between the predicted and actual results. Once the model is trained, it can make predictions on newer data by adding the values of the independent variables.

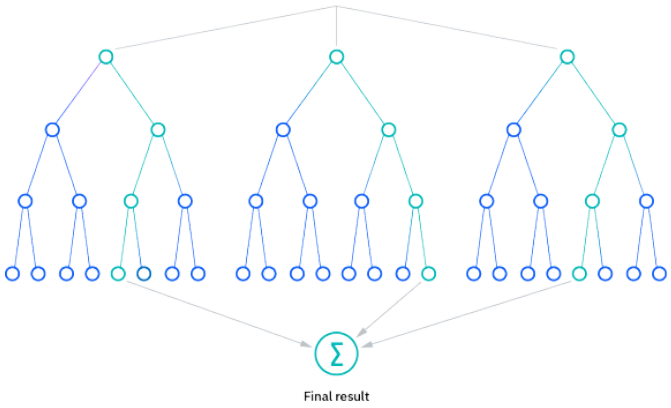
$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

In the linear regression sentiment analysis program, the linear regression ML model comes from a Python library called Scikit-learn. This library offers numerous ML algorithms used for classification, including Random Forest. The library Pandas is used for data handling, in which "TfidfVectorizer" converts the text data to numerical attributes. Splitting the dataset is done by "train_test_split." Out of the eight hundred thousand reviews, only a subset sample of ten thousand reviews is taken randomly from the JSON file (where all of the features for the reviews belong), and a "DataFrame" is made. TF-IDF vectorization transforms the text data properties of the reviews into numerical features, converting textual data into a format explicitly used by ML models for data processing. The linear regression model is created by "model = LinearRegression()" and is trained using the given data "model.fit(X_train, y_train)". "X_train" includes the TF_IDF data properties of the reviews, and "y_train" consists of the sentiment scores. A classification report is given in the output to show details of the accuracy of the model. Based on running this version of the program, accuracy in its predictions is very low as the "accuracy" result is typically on the lower end, ranging below 0.5 (an accuracy score of "1" means that the model predicts with perfect accuracy). If I were to continue working on this model, I would likely use multiple regression models to process the data more because a singular linear regression model does not give accurate enough results. However, the implementation was not too difficult as I have experience working with linear regression ML models from a project activity for a computer science-related club last Spring. Linear regression models are perfect to work with at a beginner level, where technical understanding of the algorithm is not as difficult to interpret in pseudocode and then translate as Python code.

Before the creation of random forest, similar ML models belonged to a category of learning methods known as ensemble learning methods. These were made up of decision trees in

which their predictions are clustered by popular vote to identify the most popular outcome. One of the most well-known early examples of the ensemble learning method was developed by Leo Breiman in 1996, who introduced the bagging method. This method involves a random sample of training set data having individual data points chosen multiple times through replacement. The models are trained independently when enough data samples are produced. The average/majority of the predictions reveal a more accurate final prediction. Likewise, the random forest ML algorithm uses the ensemble learning method of bagging and incorporates feature randomness to generate an uncorrelated forest of decision trees. Feature randomness creates a random subset of features to reduce the correlation between decision trees.



The difference between random forest and decision trees is how decision trees look for all possible feature splits, whereas random forest chooses only a subset of the features. All random forest algorithms have three primary “hyperparameters”: node size, the number of trees, and the number of features sampled. The random forest ML algorithm itself comprises multiple decision trees. Each tree in the ensemble consists of a data sample taken from the training dataset known as the bootstrap sample. One-third of this training sample is set aside to be used as “testing data” referred to as the out-of-bag sample. Diversifying the dataset further, randomness through feature bagging occurs again to reduce the correlation between decision trees. In a regression task, the average of the individual decision trees is used in determining the prediction. In a classification task in which the random forest version of the sentiment analysis program performs, the most

frequent outcome for a category is the outcome for the predicted class done by a majority vote. The base prediction is returned from the out-of-bag sample for comparison to make a final prediction. The benefit of having a robust number of decision trees in a random forest algorithm reduces the overall risk of overfitting samples from the training data due to the averaging of uncorrelated trees and decreases prediction error. However, random forest ML algorithms can have longer processing times as data is being computed for every decision tree, requiring more resources to store all of the data. Unlike linear regression, the predicted outcome of a forest of individual decision trees becomes more complicated to interpret and understand how it leads to a conclusion.

The random forest sentiment analysis program serves the same purpose as the linear regression version. However, critical differences exist in its functionality to perform with greater accuracy in the model's prediction output. Instead of giving sentiment scores from a scale of "1 to 10" for product reviews, the program provides a sentiment label that generalizes the emotional context of the review as either "positive," "negative," or "neutral." Since random forest ML algorithms primarily serve classification tasks, they are designed to predict a categorical outcome rather than a numerical value. The benefit of using labels instead of scores is that it allows for a more straightforward interpretation of the output because the categories are pre-defined.
In contrast, the user must best judge the output without any predefined measures for the numerical value sentiment scores in the linear regression program. This version of the sentiment analysis program also takes a subset sample of ten thousand random reviews from the JSON file. Loading the data and processing also works similarly to the other program to make a "DataFrame." TF-IDF vectorization is also involved in converting text data properties into numerical features, and the data is split into training and testing sets using "train_test_split." The

random forest classification model is created using the three primary hyperparameters mentioned earlier and a few additional parameters for scalability. The “n_estimators=300” parameter sets the number of decision trees in the random forest to be three hundred. The “max_depth=20” parameter controls the depth of each decision tree to capture enough patterns within the data without the model overfitting. The “min_samples_split=2” parameter sets the number of samples required to split a single node. A node can split as long as it has more than one sample in this scenario. The “class_weight=’balanced’” parameter balances the “classes” (categories) to prevent favoring bias towards the majority. Finally, the reproducibility of outcomes uses the “random_state” parameter to ensure that results are replicable. A classification report is also given in the output to show info about the model’s accuracy. Based on running this version of the program, the random forest model is more likely to make accurate predictions as the “accuracy” result ranges on the higher end above 0.5 (remember that a score of “1” means perfect accuracy). As demonstrated by the two models’ different outputs, this is a much better result for the model’s overall accuracy than the linear regression model. When it comes to predicting the emotional context of the product review and interpreting its sentiment analysis, the random forest program serves a greater use for this classification task because we know the predefined contexts behind the meaning of “positive,” “negative,” and “neutral” when reading product reviews than a number scale that may be left open to interpretation. Although this model's range of accuracy results may seem adequate, it can be improved by experimenting with different parameter values, such as changing the number of decision trees in the random forest or the number of samples required for each node to split. Adding multiple random forest models would also improve the range of accuracy results. However, it would quickly add a much higher difficulty level in code implementation than just a random forest. Random forest models are

intriguing but require a deeper understanding of language model processing and Python coding skills. It's no wonder ML researchers and data scientists consider random forest algorithms instrumental in classification tasks.

ML algorithms have become a powerful tool for extensive data analysis and decision-making. Linear regression and random forest are highly applicable ML algorithms used in various fields to perform regression or classification tasks. Linear regression is best suited for predicting numerical values (regression tasks) by identifying the relationship between two variables to minimize the difference between experimental and predicted values. Random forest is best suited for classification tasks where the prediction is a class label based on categorizing nonlinear relationships with multiple variables and voting for the most frequent outcome. In the case of analyzing sentiment from product reviews, the random forest model has a better application for performing the task than linear regression because the emotional context of a review is more straightforward to interpret through categorized words such as "positive," "negative," and "neutral" rather than a numerical value (score) that leaves the user to define their interpretation of the range of values given as prediction values (the scale being from 1 to 10). Additionally, a numerical scale can be too broad when product reviews only have a few words, making it more difficult to give an accurate score. Both algorithms have unique strengths and limitations, but the choice of the algorithm used for a particular task depends on whether the task's problem suits regression or classification better. Improvements in ML algorithms and the availability of large-scale data collection are coming quickly in the current race to develop some of the most powerful AI models. Hence, applying these algorithms plays a crucial role in these pursuits.

Appendices

Gandhi, Rohith. *Minimization and Cost Function Image*. 27 May 2018. *Introduction to Machine Learning Algorithms: Linear Regression*, Medium, https://miro.medium.com/v2/resize:fit:640/format:webp/1*wQCSNJ486WxL4mZ3FOYtgw.png.

Ni, Jianmo, et al. "Office_Products." University of California San Diego, 2019.

Pugazhenthi, Surya. "Linear Regression Code and Console Output." *Google Docs*, Google, docs.google.com/document/d/1-JiDLGkrlzooFqwCGQ28aVbZo_DeDDF4ECChl9T8HW8/edit. Accessed 2 Dec. 2023.

Pugazhenthi, Surya. "Random Forest Code and Console Output." *Google Docs*, Google, docs.google.com/document/d/1YKjn8uAO4w3HYKHSzmvuzNaCejKzU6jHvVCYrIpVy-4/edit. Accessed 2 Dec. 2023.

Random Forest Algorithm Decision Making. *What Is Random Forest?*, IBM, https://www.ibm.com/content/dam/connectedassets-adobe-cms/worldwide-content/cdp/cf/ul/g/50/f9/ICLH_Diagram_Batch_03_27-RandomForest.component.simple-narrative-xl.ts=1698245878055.png/content/adobe-cms/us/en/topics/random-forest/jcr:content/root/table_of_contents/body/content_section_styled/content-section-body/simple_narrative0/image. Accessed 1 Dec. 2023.

Works Cited

Breiman, Leo. “Random Forests | U.C. Berkeley.” *Machine Learning*, vol. 45, no. 1, 2001, pp. 5–32, https://doi.org/10.1023/a:1010933404324.

Döring, Matthias. “Supervised Learning: Model Popularity from Past to Present.” *KDnuggets*, Guiding Tech Media, 2018, www.kdnuggets.com/2018/12/supervised-learning-model-popularity-from-past-present.html.

Gandhi, Rohith. “Introduction to Machine Learning Algorithms: Linear Regression.” *Medium*, Towards Data Science, 27 May 2018, towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a.

Ni, Jianmo, et al. “Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects.” *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, https://doi.org/10.18653/v1/d19-1018.

“What Is Random Forest?” *What Is Random Forest? | IBM*, IBM, www.ibm.com/topics/random-forest. Accessed 1 Dec. 2023.