Prediction Intervals of Machine Learning Models for Taxi Trip Length

Ella Morgan, Ryan Zhou, Wenying Feng*

Abstract Errors are always present in predictions produced by machine learning models. Producing a quantitative estimate of the uncertainty in a model's output is crucial for many fields, especially those where predictive models drive important decisions. In this paper we discuss two methods for producing confidence estimates for neural network, random forest, and gradient boosted tree models. We then evaluate the prediction intervals produced by each algorithm by predicting expected ride length for a NYC taxi trip dataset. We show that inductive conformal prediction produces the most reliable intervals for all machine learning models investigated.

1 Introduction

No model is correct all the time. This is a reality for all models, from simple linear regressions to deep neural networks. Recognizing when a model may produce inaccurate predictions is of increasing importance, especially with the growing usage of predictive models in the real world. As such it is helpful to obtain, along with the point estimate produced by the model, a quantitative measurement of a model's uncertainty in the form of a variance or a prediction interval.

Methods for producing prediction intervals for models such as linear regression are well studied. However, these methods are not easily generalizable to more complex models such as those used in machine learning. We will present two distinct methods - decomposition into model variance and inherent noise, and inductive conformal prediction - for producing a prediction interval for common machine learning techniques. To our knowledge, these methods have not been directly compared in the literature. Our primary contribution with this paper is an empirical comparison

Ella Morgan, Ryan Zhou and Wenying Feng
Trent University, Peterborough, ON Canada. e-mail: {ellamorgan, ryanzhou, wfeng}@trentu.ca

* Corresponding author

of these methods applied to multiple machine learning algorithms. To this end, we develop a method of estimating model uncertainty on tree-based algorithms as well as an improvement on the inherent noise calculation by dividing the calibration set into subsets based on target value.

We evaluate these methods on a dataset consisting of taxi trips in New York City, using machine learning algorithms to predict the duration of a trip. This problem is ideal for showcasing the techniques for a few reasons. Taxi trip duration can be extremely difficult to predict as it relies on unpredictable external factors which are difficult to quantify. Traffic, weather, and road conditions can change rapidly and unpredictably, varying the actual duration of a trip by a large amount and ensuring that errors will be inevitable to some degree. In addition, measurable factors such as the starting and ending location and the time of day do not have simple relationships with the trip duration, making this a complex problem ideal for machine learning algorithms.

The algorithms we use to generate predictions are random forest, gradient boosting, and neural networks. Random forest [4] is an ensemble method that aggregates the predictions of multiple decision trees to produce a single final prediction. Each tree is trained on a subset of the training data produced by drawing with replacement, and a random subset of features is used to build the tree. The prediction is then the average prediction between all trees.

Gradient boosting [6] is a tree-based ensemble method closely related to random forest. However, rather than training all trees independently, it creates the trees iteratively and attempts to improve on the ensemble of existing trees by using gradient descent over the loss function. Each individual tree can be used as a weak estimator as well, a property which we use for uncertainty estimation.

Neural networks consist of many nodes or neurons, each of which produces an output that is a linear combination of its inputs modified by a nonlinear activation function. Neural network models link together many neurons in multiple layers in order to learn complex relationships between input and output. Regularization is often used on neural networks to alleviate overfitting; one such regularization technique is dropout, which randomly mutes the outputs from a certain fraction of neurons. This approximates the training of a large number of similar neural networks with shared weights [12], forcing the network to generalize better by relying less on specific weights. As we describe in the next section, this approximation to ensemble behaviour can also be used for uncertainty estimation.

2 Prediction Interval Algorithms

We provide two techniques for constructing a prediction interval. The first involves decomposing the total uncertainty into two components, one which captures uncertainty within the model and one which describes the uncertainty in the input data.

The second technique, conformal prediction, approaches the problem from a different angle, by attempting to evaluate the difficulty of each input data point by comparing it to other known points.

2.1 Decomposition into Model Variance and Inherent Noise

Total prediction uncertainty is a combination of model uncertainty, denoted by η_1^2, and inherent noise in the data, denoted by η_2^2 [13]. Model uncertainty is specific to the underlying trained model, and results from uncertainty in the model parameters. In the following, we will discuss different methods for evaluating this type of uncertainty

The inherent noise η_2^2 is calculated by finding the mean squared error on a separate validation set. Typically, one value for η_2^2 is found for the full validation set and this single value is used for all prediction intervals. We propose a modification to this method, where instead the validation set is split into n separate groups and the inherent noise is evaluated separately for each group. We sort the elements of the validation set based on their predictions and divide the sorted set into n groupings. η_2^2 is then calculated separately for each grouping. At inference time, we use η_2^2 of the grouping whose range contains the test prediction. The two variances, η_1^2 abd η_2^2, are then combined into a total uncertainty measure. The full algorithm is described in Algorithm 1 below.

Algorithm 1 Variance and Inherent Noise

```
Input: Data $x^*$, validation set $x'$ prediction network $h(\cdot)$, model uncertainty $\eta_1^2$
Output: Prediction $\hat{y}_{mc}$, uncertainty $\eta$
  // prediction
  $\hat{y} \leftarrow h(x^*,\ p)$
  // inherent noise
  for $x'_v$ in validation set $x'$ do
      $\hat{y}'_v \leftarrow h(x'_v)$
  end for
  $\eta_2^2 = \frac{1}{V}\sum_{v=1}^{V}(y_v - \hat{y}_v)^2$
  // total prediction uncertainty
  $\eta = \sqrt{\eta_1^2 + \eta_2^2}$
  return $\hat{y}_{mc}$, $\eta$
```

We now describe some methods for calculating η_1^2, the model uncertainty. These methods are specific to the model used, whereas inherent noise is independent of the model and depends only on the data.

Variance Estimation Through Monte Carlo Dropout

As outlined earlier, dropout is a regularization technique used on neural networks to prevent overfitting. This is done by muting a certain fraction of neurons at random each training epoch, effectively changing the structure of the neural network slightly each time [12]. While this prevents the network from being overly reliant on specific nodes by forcing it to make predictions when the nodes in question are dropped out, it can also be seen as performing an averaging of the weights over the ensemble of possible subnetworks. As such, the final dropout-trained network can be viewed as approximating an ensemble prediction.

Following the method described by Gal in [7], dropout can also be used during inference time to effectively reconstruct the ensemble and gather predictions from random subnetworks, a technique known as Monte Carlo dropout. By making a large number of predictions with dropout, we are able to sample from the distribution of ensemble predictions and estimate their variance as follows:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b, \tag{1}$$

$$\eta_1^2 = \frac{1}{B} \sum_{b=1}^{B} (\hat{y}_b - \hat{y})^2. \tag{2}$$

In the equations above, $\hat{y}_b$ represents a single prediction made with dropout and B the number of times the predictions were repeated. The average of all predictions is returned as the point prediction $\hat{y}$, and the variance is found using equation (2). This variance is the model uncertainty, representing the disagreement in the predictions of each subnetwork produced by dropout, despite having all been trained on the same data.

Variance Estimation for Tree Based Estimators

Although dropout regularization is a neural network technique and not used with tree based models, random forest and gradient boosted trees are innately ensemble methods. Therefore, we propose to calculate the model uncertainty using the same method as the one described above, by finding the variance of the predictions produced by each submodel in the ensemble. With $\hat{y}_b$ in equation 1 now being the predictions from individual trees, we similarly calculate the model uncertainty for the tree based model with equation 2. This method can also be used with other ensemble algorithms beyond those described in this paper.

Variance Estimation Through Proper Scoring

It is possible to take a different approach, by training a neural network to predict its own variance. We modify the neural network to output a probability distribution instead of a single value; as described in [9], this amounts to outputting the mean and variance for a Gaussian distribution. The loss function to be minimized then is the negative log-likelihood criterion:

$$-\log p_\theta(y_n|x_n) = \frac{\log \sigma_\theta^2(x)}{2} + \frac{(y - \mu_\theta(x))^2}{2\sigma_\theta^2(x)} + \beta, \tag{3}$$

where μ_θ and σ_θ^2 are the outputted mean and variance respectively. By modifying the neural network in this manner, the mean and variance can be estimated without producing multiple predictions as with Monte Carlo dropout, at the cost of increasing the number of parameters in the neural network. The model uncertainty produced by this method η_1^2 is then the value σ_θ^2 predicted by the network: $\eta_1^2 = \sigma_\theta^2(x)$.

2.2 *K Nearest Neighbors Inductive Conformal Prediction*

Conformal prediction classifies data points based on a nonconformity measure - or intuitively, how "strange" each data point is for the underlying model - based on the error the model produces when labeling that point. This is measured using a reserved validation set, as shown in Algorithm 2. A calibration score is chosen by ordering the nonconformity measures and finding the value which captures all scores up to a given significance level. As each score is an explicit function of error, this can be directly translated into a prediction interval.

Newer formulations [10] of conformal prediction also employ normalization using a difficulty measure, which is used to adjust the sizes of the uncertainty intervals based on the estimated difficulty of the data point. For this study we use the average error, weighted by distance, of the K nearest neighbours as the difficulty estimate. This version of the metric is described in further detail in [3].

3 Empirical Evaluation

In this section we describe the setup for an empirical evaluation of the different algorithms. We report the results from the tests in the following section.

3.1 *Dataset*

The dataset used for the experiments was obtained from the New York City (NYC) Taxi and Limousine Commission website
https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, for the month of August, 2016. In order to prevent data leakage, we used only features which would be known before a taxi trip, including pickup latitude and longitude, drop off latitude and longitude, distance travelled (which could be estimated with the routing software used by taxi drivers), and the day and hour of pickup.

The dataset was augmented with features such as average taxi speed that hour, pickup and drop off neighborhood (by grouping locations based on density), number of taxi rides that hour, direction of travel, and whether the taxi ride started or ended

Algorithm 2 kNN ICP

```
Input: Data $x^*$, validation set $x'$, prediction algorithm $h(\cdot)$, kNN algorithm $g(\cdot)$, number of nearest
  neighbors $k$, significance level $\delta$
Output: Prediction interval $\hat{Y}_i^\delta$
  // prediction
  $\hat{y} \leftarrow h(x^*)$
  $\hat{y}' \leftarrow h(x')$
  for $x_i'$ in validation set $x'$ do
    // find K nearest neighbors
    $\{x_1', ..., x_k'\} = g(x_i', x', k)$
    for $x_j'$ in $\{x_1', ..., x_k'\}$ do
      // euclidean distance
      $d_j = d(x_i', x_j')$
      $o_j = |\hat{y}_j' - y_j'|$
    end for
    // difficulty measurement
    $\mu_i = \frac{\Sigma_{j=1}^k o_j/d_j}{\Sigma_{j=1}^k 1/d_j}$
    // nonconformity measure
    $\alpha_i = \frac{o_i}{\mu_i + \beta}$
  end for
  sort $\{\alpha_1, ..., \alpha_m\}$ incrementally
  $\alpha_\delta = \alpha_{\lfloor \delta(m+1) \rfloor}$
  $\hat{Y}_i^\delta = \hat{y}_i \pm \alpha_\delta(\mu_i + \beta)$
  return $\hat{Y}_i^\delta$
```

at an airport. Records were removed if they failed to meet any of the following criteria: a trip time between 10 seconds and 20 hours, a trip distance greater than 0, an origin and destination within NYC and with valid coordinates, and an average trip speed (time/distance) under 100km/h.

3.2 Models

The models used for making predictions were as follows: random forest, gradient boosting, a standard neural network, a neural network with dropout layer, and a neural network using a proper scoring method.

The variance and inherent noise methods were used to estimate confidence intervals for predictions made by random forest, gradient boosting, the neural network with a dropout layer, and the neural network using a proper scoring method.

The *K* nearest neighbours inductive conformal prediction method was used to estimate confidence intervals for predictions made by random forest, gradient boosting, and the standard neural network.

Random forest and gradient boosting were implemented using scikit-learn [11] in Python. For both models 1000 trees were built, and all other parameters were the defaults for scikit-learn. For both random forest and gradient boosting a single model was trained and there were one set of predictions for each, and then confidence intervals were found using two different methods: variance and inherent noise with variance estimation for tree based estimators, and K nearest neighbours inductive conformal prediction.

All neural networks were implemented in Python using Keras [5] and Tensorflow [1]. Three different models were built, all consisting of a dense layer of 1000 neurons using a tanh activation. The first neural network implemented variance estimation through Monte Carlo dropout and incorporated a dropout layer active during both training and testing time. The second network used the proper scoring method to output the mean and variance of a Gaussian distribution. This network was trained using log-likelihood loss implemented as a custom loss function in Keras. The mean of the distribution was taken to be the model's prediction, and the variance was used as model uncertainty. The final neural network was a standard neural network used as the underlying model for KNN inductive conformal prediction.

3.3 Metrics

To compare the accuracy of the point predictions from each machine learning model we use the mean absolute error (MAE), root mean square error (RMSE), and symmetric mean absolute percentage error (SMAPE) scoring metrics. Mean absolute error is calculated as the average difference between the actual and predicted value, and is a commonly used metric which is easy to interpret.

Root mean square error squares the difference between the actual and predicted value, then takes the square root of the sum. The benefit of this is that larger errors get penalized more and have a larger impact on the result, allowing for the ability to evaluate which methods have a larger quantity of outliers in the predictions.

While the previous metrics are useful for comparisons within the same dataset, we also include a percentage error metric to allow comparison between datasets. The symmetric mean absolute percentage error was used due to its popularity in the literature and ability to handle low values [2, 8].

$$\mathrm{SMAPE} = \frac{100}{N} \sum_{n=1}^{N} \frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}. \tag{4}$$

We compare the prediction intervals using two main criteria. First, we investigate whether the intervals capture the desired confidence level by finding the fraction of target values y in the testing set which fall inside the calculated prediction interval $[\hat{y}_{lower}, \hat{y}_{upper}]$, where $\hat{y}_{upper}$ and $\hat{y}_{lower}$ are the upper and lower limits of the interval, respectively. The percentage of records found within the interval, referred to as capture percentage (CP), is calculated as follows and ideally is close to the target confidence level:

$$\mathrm{CP} = 100 \times \frac{1}{N} \sum_{n=1}^{N} x_n, \tag{5}$$

$$x_n = \begin{cases} 1 & \hat{y}_{upper} \geq y \text{ and } y \geq \hat{y}_{lower}, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Next the relative sizes of the prediction intervals are compared. Ideally we prefer methods which produce smaller intervals on average, given that the accuracy of the intervals remains around the intended percentage. To evaluate this we find the mean and median interval size. Using both the mean and median allows us an overview of the comparative sizes of the intervals while remaining robust to outliers.

4 Experimental Results

Table 1 compares the performance of the models based on their point predictions. The random forest model achieved the highest accuracy by all listed metrics while the gradient boosting model proved the least accurate. Of the neural networks, the standard neural network performed slightly better than other neural network models, indicating that dropout and the proper scoring method affect accuracy by a small degree. However, the overall difference in performance between models was slight and all models obtained comparable accuracy on this dataset.

Table 1 Point Prediction Results

Model	**MAE**	**RMSE**	**SMAPE**
Random Forest	143.14	218.24	18.89
Gradient Boosting	153.24	230.55	20.39
Standard Neural Network	146.12	226.29	19.58
Neural Network with Dropout	148.47	232.53	19.40
Neural Network with Proper Scoring	150.60	231.10	19.82

Table 2 evaluates the performance for different methods of computing predictive regions. The results show that K nearest neighbours inductive conformal prediction (models 5-7) consistently outperformed variance and inherent noise methods (models 1-4) in both accuracy of coverage (equation 5) and interval size. The intervals obtained from conformal prediction better captured the target confidence level and as a result, prediction intervals produced by conformal prediction were also smaller due to the variance and inherent noise method consistently overestimating the size of interval required. The variance and inherent noise method produced more conservative estimates which may be desirable if avoiding underestimates is critical, but the significantly larger intervals do not as accurately convey the model's confidence in its prediction.

With the variance and inherent noise method, the gradient boosting and neural network models produced comparable intervals. Between the proper scoring and dropout neural networks, proper scoring was consistently slightly better and produced smaller overestimates of interval size. The neural network with proper scoring was also significantly faster at generating intervals and single predictions, as sampling a large number of predictions for the dropout network added considerable overhead.

Table 2 Prediction Interval Results

	90% Interval			**95% Interval**			**99% Interval**		
Model	CP	Mean	Median	CP	Mean	Median	CP	Mean	Median
1	98.0%	459.5	414.4	98.9%	547.5	493.8	99.6%	719.6	649.0
2	94.7%	390.1	315.4	97.1%	464.9	375.8	98.9%	611.0	494.0
3	95.5%	388.0	366.9	97.6%	462.3	437.1	99.0%	607.6	574.5
4	94.3%	371.3	338.5	96.8%	442.4	403.4	98.7%	581.4	530.1
5	89.8%	327.2	313.9	94.8%	443.1	425.0	98.9%	766.2	735.1
6	89.6%	346.8	333.7	94.8%	466.6	449.0	98.8%	785.1	755.4
7	89.5%	301.1	264.8	94.8%	385.0	338.6	98.9%	616.2	541.8

The capture percentage (CP) column shows the percentage of intervals that contain the true value (equation 5). The mean and median columns show the average sizes of the prediction intervals.

Model 1: Random forest with variance and inherent noise method for tree based estimators.
Model 2: Gradient boosting with variance and inherent noise method for tree based estimators.
Model 3: Neural network with variance and inherent noise method with Monte Carlo dropout variance.
Model 4: Neural network with variance and inherent noise method using a proper scoring method.
Model 5: Random forest with K nearest neighbours inductive conformal prediction.
Model 6: Gradient boosting with K nearest neighbours inductive conformal prediction.
Model 7: Neural network with K nearest neighbours inductive conformal prediction.

It was observed that inductive conformal prediction was consistently able to achieve the target confidence level, independent of the underlying algorithm. This illustrates a key advantage of conformal prediction in that the algorithm can be treated as a black box, unlike variance decomposition which requires model-specific methods of estimating the model uncertainty. In addition, the inductive version of conformal prediction used in this paper only requires a relatively straightforward calculation of difficulty at inference time, unlike some implementations of variance estimation which may require a computationally expensive ensemble prediction.

5 Conclusion

In this paper we explored different methods for finding prediction intervals for machine learning algorithms. We compared two methods for finding such intervals:

a decomposition into model variance and inherent noise method, and an inductive conformal prediction method utilizing K nearest neighbours. These methods were evaluated on three base models trained to predict taxi trip length. In terms of prediction accuracy we achieved comparable results on all models, with random forest having marginally better results. For prediction intervals we found that K nearest neighbors inductive conformal prediction outperforms the variance with inherent noise methods tested in all cases. It also has the significant advantage of being independent from the underlying algorithm, so it is compatible and can be implemented alongside any prediction model.

Acknowledgements We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
2. J Scott Armstrong and Long-Range Forecasting. From crystal ball to computer. *New York ua*, 1985.
3. Henrik Boström, Henrik Linusson, Tuve Löfström, and Ulf Johansson. Accelerating difficulty estimation for conformal regression forests. *Annals of Mathematics and Artificial Intelligence*, 81(1-2):125–144, 2017.
4. Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
5. François Chollet et al. Keras. https://keras.io, 2015.
6. Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
7. Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.
8. Spyros Makridakis. Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, 9(4):527–529, 1993.
9. David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.
10. Harris Papadopoulos, Vladimir Vovk, and Alexander Gammerman. Regression conformal prediction with nearest neighbours. *Journal of Artificial Intelligence Research*, 40:815–840, 2011.
11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
12. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
13. Lingxue Zhu and Nikolay Laptev. Deep and confident prediction for time series at uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110. IEEE, 2017.