Uncertainty Management for Learned Cardinality Estimation, with Conformal Predictions and Bayesian Neural Networks

Nikhil Basavaraj
Data and Knowledge Engineering (FIN)
Otto von Guericke University
Magdeburg, Germany
nikhil.basavaraj@st.ovgu.de

Sourima Dey
Digital Engineering (FIN)
Otto von Guericke University
Magdeburg, Germany
sourima.dey@ovgu.de

Bhavana Malla
Data and Knowledge Engineering (FIN)
Otto von Guericke University
Magdeburg, Germany
bhavana.malla@st.ovgu.de

Brinda Rao
Data and Knowledge Engineering (FIN)
Otto von Guericke University
Magdeburg, Germany
brinda.rao@st.ovgu.de

Adrija Ghosh
Digital Engineering (FIN)
Otto von Guericke University
Magdeburg, Germany
adrija.ghosh@ovgu.de

Sunkra Tushar Rao
Data and Knowledge Engineering (FIN)
Otto von Guericke University
Magdeburg, Germany
sunkra.rao@st.ovgu.de

***Abstract*—By now, the fact that data is the most important asset for a modern business is old news. The importance of data and the ways we can deal with them is well established, and the advantages speak for themselves. However, one aspect that we fail to observe in the discussion about data-driven business is that this information requires to pass through a database to generate values and it's here that many organizations run into trouble. One of the most pervasive problems that affect databases is Cardinality Estimation which plays a significant role in generating high-quality query plans for a query optimizer. According to recent research most systems routinely produce large errors by overestimating the estimates which leads to bad execution plans. In this paper, we propose using learning methods such as Multi-Set Convolutional Network and NARU for cardinality estimation but this gives us a point prediction that may not be able to solve the problem of the system producing large errors by misestimating the cardinality, so we propose incorporating BNN and CP with the learning methods which helps us to solve the problem of cardinality mismatch to some extent and then finally we implement ensembles of models that rely on coverage and tightness of the interval to obtain accurate estimates. Results on a real-world data set show that the ensembles significantly improve estimation as well as overall plan quality. The proposed technique can be easily integrated into most systems.**

I. Introduction

The problem of finding a good execution plan is one of the most studied problems in the database field. Fig.1 illustrates the classical, cost-based approach, which dates back to System R . In Cost-based query optimization, a **plan space enumeration** is an algorithm that finds a plan to execute a query and then uses the **cost model** to find the cost of that plan and finally selects the plan with the lowest cost. In the cost-model, the **cardinality estimate** is the number of rows the optimizer believes will be returned.

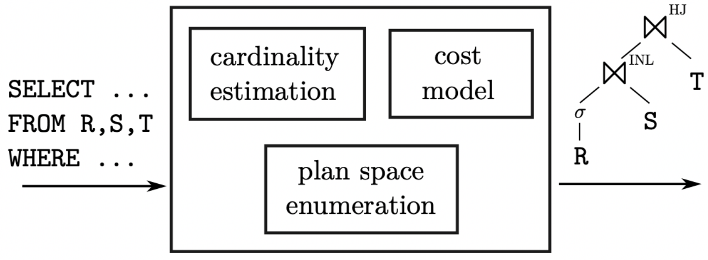


Fig. 1: Query Optimizer Architecture. [12]

According to [7] theoretically, as long as the cardinality estimations and the cost model are accurate, this architecture obtains the optimal query plan. In reality, cardinality estimates are usually computed based on simplifying assumptions like uniformity and independence. In real-world data sets, these assumptions are frequently wrong, which may lead to sub-optimal and sometimes disastrous plans.
Hence we need to deal with these assumptions so that we can get accurate estimate. Cardinality Estimation is the most important aspect of an execution plan because it strongly influences all of the other decisions the optimizer makes.
In this paper, we use two models **Multi-Set Convolutional Network(MSCN)** and **NARU** for cardinality estimation. These models give us the estimated number of rows, the optimizer believes will be returned by a specific operation in the execution plan. But these models gives us the **point prediction** and there is no way of knowing if the model is confident in it's prediction or it is just giving us some value even when the model has no definite answer for the query. In the machine learning world, we call it uncertainty.

In simple words uncertainty means when the model is not confident in it's prediction may be because the model was trained on imperfect or incomplete information. And there are two reasons why modeling uncertainty can be really helpful. The first is **transparency**. Imagine you are building a machine learning model that is applied in medical image analysis. So, the doctors using your tool heavily depend on its capabilities to make a correct diagnosis. If your model now makes a prediction it is highly uncertain about but does communicate this information to the doctor, the consequences for the treatment of the patient can be fatal. Therefore, having an estimation of the model's uncertainty can help the doctor massively when judging the model's prediction. The second is **showing room for improvement**. No machine learning model is perfect. Therefore, knowing the uncertainties and weaknesses of your model can inform you what improvements to make to your model.
When we manage uncertainty in a regression setting, the model instead of giving point prediction gives us an interval with a certain confidence. The confidence of the model is a user-specified parameter called the significance level. The significance level of an event is the probability that the event could have occurred by chance. If the level is quite low, that is, the probability of occurring by chance is quite small, we say the event is significant.
In our paper, we are dealing with uncertainty with the following two methods.

- **Conformal Predictors** which is an External Framework
- **Bayesian Neural Network** which is a Model-Based Framework

The rest of this paper is organized as follows: We first discuss important background in Section II. In Section 3 we briefly discuss some of the research questions that we hope to answer once we are done with our experiments and the framework which we followed for our experiments. In Section 4 we try to show how can we incorporate BNN and CP into learning methods such as MSCN and NARU so that our model instead of giving point predictions gives us an interval with a certain confidence. In Section 5 we extensively evaluate our approach of using learning methods like MSCN and NARU with BNN and CP using a real-world data set and then forming an ensemble of models the results show that the ensemble can outperform all the architectures in terms of coverage and average interval width. Finally, in Section VI we concluded that Naru with a Conformal setting is contributing to the prediction intervals for almost all the test instances when significance levels 0.05 and 0.1 are considered.

II. Background

To provide context to the information discussed throughout the paper, we discuss briefly the following topics.

A. *Uncertainty Estimation*

Uncertainty is one of the biggest sources of difficulty in Machine learning or the Deep learning domain. Noise in data, incomplete coverage of the domain, and imperfect models provide the three main sources of uncertainty in machine learning. There are mainly two different sources of uncertainty-

- **Aleatoric uncertainty**: This refers to the notion of randomness present in the data, that is the variability of an outcome of an experiment due to inherently random effects. This type of uncertainty can not be reduced as it is inherent with the data.
  In Fig.2, we see that the model is giving us a precise knowledge about the optimal hypothesis, when a new query comes in (indicated by a question mark), the model is aleatorically uncertain, because the two classes are overlapping in that region.
- **Epistemic Uncertainty**: Epistemic uncertainty refers to uncertainty caused by a lack of knowledge of the best model. It refers to the ignorance of the decision-maker due to limited data and knowledge. As opposed to Aleatoric uncertainty, epistemic uncertainty is reducible. In Fig.2, we see an example of epistemic uncertainty because here the model is not trained with sufficient amounts of training data. Hence when a new query comes in, the model is not able to produce an optimal hypothesis for it due to poor generalization. So this model is epistemically uncertain.

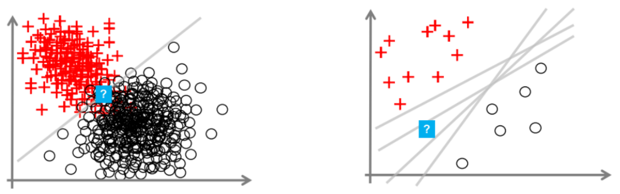


Fig. 2: Aleatoric Uncertainty(left) and Epistemic Uncertainty(right) [4]

Uncertainty modeling techniques: Over the course of time, many techniques have been followed to model uncertainty in machine learning. The diagram below describes a few important techniques of uncertainty modeling. On

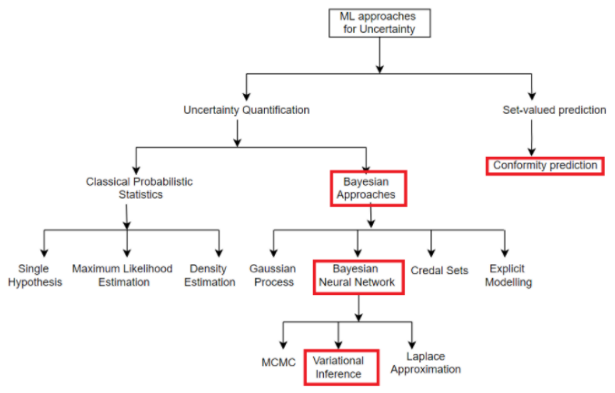


Fig. 3: Uncertainty Modeling Techniques

the left-hand side we have all the techniques which require learning of a model to estimate the uncertainty i.e they are model-dependent. But on the right-hand side we have the approach of conformal prediction which is model-agnostic. In our project, we are following the Bayesian Neural Network approach with variational inference and conformal prediction as highlighted in the fig 3.In the next section, we discuss about these uncertainty modeling techniques.

1.Conformal Predictions:

Today, a wide range of black-box machine learning methods are routinely used in high-risk settings, such as medical diagnostics. They demand uncertainty quantification as these models are more ubiquitous. Distribution-free uncertainty quantification is an easy-to-use paradigm to avoid model failures by providing statistically rigorous confidence intervals for the predictions made by the models. Conformal Prediction is one such distribution-free uncertainty quantification [2] method. The framework can be built on top of any standard machine learning algorithm which makes it model agnostic. Unlike Bayesian Uncertainty Quantification, the method is not dependent on model performance and the distributions. To be precise, it does not require prior probabilities. It can be applied to both classification and regression problems. In the case of regression which is our current task at hand, this framework extends the models to generate outputs as intervals around the predictions with attached significance levels that give valid statistical reliability to the predictions.

Essentially the key motivation behind this framework is it provides guarantees for prediction error. The probability of predicting incorrectly is bound by a user-specified confidence threshold which is also called the significance level. Most of the available alternative uncertainty quantification methods provide a bound on the prediction error for the entire distribution whereas the conformal prediction framework assesses the uncertainty of every single prediction. This is very valuable in many critical and practical applications such as the medical domain where it makes sense to assess the error as well as confidence in the prediction of every single patient rather than groups of patients.

NonConformity Functions: The CP framework derives the estimates of the difference between a new example and a previous set of examples based on real-valued functions called nonconformity functions. Essentially, it is any function that measures the degree of disagreement between the predicted value(ŷ) and the true value(y). The underlying intuition is that inputs less similar to training data should lead to less certain estimates and this is captured by nonconformity scoring functions. [3]

Various nonconformity functions can be designed for a specific prediction model, and each one defines a different conformal predictor. While there are significant differences in terms of efficiency between these various conformal predictors, they should all be valid. In some extreme cases, a function that returns the same nonconformity score for all examples is still valid, although the prediction region will be relatively wide [5].The non-conformity function we used in our experiments is standard regression absolute error. [11] [9]

$$\alpha_i = |y_i - \hat{y}_i| \tag{1}$$

CP frameworks come in two types. The first type is Transductive Conformal Prediction(TCP) [13]and the other is Inductive Conformal Prediction(ICP) [10]. Due to the fact that the whole model must be retrained for every test example, TCP is computationally complex and expensive. On the other hand, ICP was introduced to overcome the complexity issue of the former. In ICP, the training data is used to induce a model, which is then used to predict all test instances. However, ICP requires the calculation of the nonconformity scores to be based on a separate data set (referred to as calibration set C) that was not used during the learning process. To illustrate, let us use $S = \alpha_1, ..., \alpha_{(|C|)}$ as the sorted sequence of non-conformity scores corresponding to the calibration set. When a new object(x_{new}, y_{new}) is presented and its prediction $\hat{y}_{new}$ is estimated using underlying model M, then the prediction region (PR) is then estimated as

$$(\hat{y}_{new} - \alpha_s, \hat{y}_{new} + \alpha_s) \tag{2}$$

Where α_s is the conformity score at index s and $s = \lfloor \varepsilon(|C| + 1) \rfloor$, where $\varepsilon \in (0, 1)$ which is user chosen significance level, and $(1 - \varepsilon)$ is the confidence level.

In the next section, we formalize the inductive conformal prediction approach [11] [3].

ICP Algorithm:
Input: Training dataset:Z_t, object to predict: x_{new}(a novel test instance), regression model M, Calibration Conformity scores
Output: Prediction Region (PR) for the object x_{new}
Step 1: The data set D is divided into two sets: training set $Z(x_1, y_1), ..(x_m, y_m)$ and test set $N(x_n, y_n), ..(x_z, y_z)$. The training set Z is further divided into two disjoint subsets Proper Training set Z_t and the Calibration set Z_c.
Step 2: Train the underlying model M using the training set Z_t in a supervised fashion.
Step 3: Using the induced model M, predict each instance in the calibration data set Z_c.
Step 4: Compute the non-conformity scores of the calibration instances $S = 1, 2, ..C$.
Step 5: Sort the non-conformity scores and identify a prediction probability rule(s) of the calibration non-conformity scores
Step 6: For a novel test instance x_{new}, compute the prediction $\hat{y}_{new}$ using model M
Step 7: Compute the prediction interval from eq-2 at a user-specified significance level ε and confidence level $(1 - \varepsilon)$

Advantages of ICP:

- CP framework is model agnostic and distribution free uncertainty method.
- Unlike BNN, CP doesnt need prior probabilities
- User-friendly paradigm for creating statistically rigorous confidence intervals for predictions.

- Local levels of confidence at the sample level.
- Intervals are valid without distributional/model assumptions.
- Framework is adaptiveness, computationally inexpensive, easily comprehensible.

Disadvantages of ICP:

- There could be the possibility of potential informational inefficiency as some part of training data is divided for calibration phase.

2. Bayesian Neural Network (BNN):

BNN is a stochastic artificial neural network trained using Bayesian inference. Below, we have listed the key points of BNN which are being implemented in our project-

- In a BNN, the weights are assigned with probability distribution as opposed to a single value as in standard neural networks as shown in Fig-4.
- BNN is a probabilistic neural network that gives output as a distribution instead of a point prediction.These probability distributions describe the uncertainty in weights and can be used to estimate uncertainty in predictions. We can take samples from the distribution to train and get predictions.
- The main goal is to obtain a better picture of the uncertainty associated with the underlying processes. This is accomplished by comparing the predictions of multiple sampled model parameterizations. Hence, BNNs can be seen as a special case of ensemble learning. If the different models agree on their predictions, then the uncertainty is low. If the models disagree, then the uncertainty will be high.
- Both types of uncertainties can be captured.

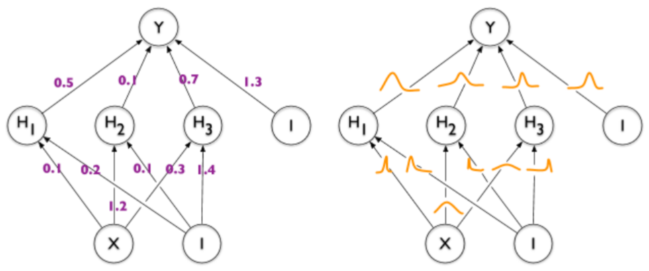


Fig. 4: Standard NN VS BNN. [1]

Bayesian statistics and Variational Inference:

As we know in according to Bayes theorem, after getting the knowledge from the data or observations, the prior belief of a model is updated to its posterior.

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} = \frac{p(y|\theta)p(\theta)}{\int p(y|\theta')p(\theta')d\theta'},$$

Fig. 5: Bayesian equation. [1]

The equation from Fig.5 governs the Bayes theorem where the denominator part is practically often intractable as there's no analytical solution to this integral. So We approximate the posterior with a distribution q (distribution in yellow), called a variational distribution,minimizing the KL divergence between them as shown in Fig.6. We'll find the closest probability distribution to the posterior(distribution in blue) that is represented by a small set of parameters—like means and variances of a Gaussian distribution and we will take samples from it.

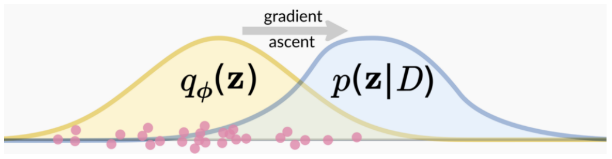


Fig. 6: Variational Bayesian Inference [1]

Re-parameterization trick in Bayesian Backprop: If we assume the parameterized family is Gaussian then the parameters are mean and variance.

$$\phi(\mu, \sigma) \tag{3}$$

A forward pass in training involves a stochastic sampling step where we have to apply the so-called re-parameterization trick for back-propagation to work. The trick is to sample from a parameter-free distribution and then transform the sampled with a deterministic function t(,,) for which a gradient can be defined.

Bringing Uncertainty into context: From the KL divergence between the distributions,

$$DKL(Q(\theta)||P(\theta|D)) \tag{4}$$

, we can get to the Evidence Lower Bound (ELBO).

$$\overbrace{\log P(\mathcal{D})}^{\text{Evidence}} - \overbrace{D_{\mathrm{KL}}[Q(\theta)||P(\theta|\mathcal{D})]}^{\text{Our objective}} = \underbrace{\overbrace{\mathbb{E}_{\theta\sim Q}[\log P(Y|X,\theta)]}^{\text{Classification loss}} - \overbrace{D_{\mathrm{KL}}[Q(\theta)||P(\theta)]}^{\text{Regularisation loss}}}_{\text{Evidence Lower Bound or ELBO}}$$

Fig. 7: Equation for ELBO Loss. [1]

This is called variational inference which converts the inference problem more into an optimization problem. By optimizing the right-hand side we minimize the KL divergence between two distributions i.e the approximated distribution and the true posterior. The first term in right hand side of the above equation in Fig.7 can be termed as likelihood cost which indicates the likelihood of our evidence given the estimated parameter.This contributes to the aleatoric uncertainty. The second term of the last equation can be called as complexity cost which comes from the difference between parameters of variational distribution and the parameters of the true posterior, hence giving rise to the epistemic uncertainty.

Thus a BNN is able to capture both types of uncertainty inherently for which in this paper we have chosen to use BNN as an approach to uncertainty prediction.

Advantages of BNN:

- The model inherently gives us both types of uncertainty estimations.
- We can easily sample from the variational distribution where as methods like MCMC,Laplace Estimation are computationally more expensive.
- BNN is self regularized which reduces the chance of overfitting.

Disadvantages of BNN:

- Unlike Conformal BNN is model-based, hence it forces us to learn a new model for uncertainty estimation.
- Model takes significant amount of time to be trained.

B. Cardinality Estimation

Below we describe two models MSCN and Neurocard which we use for cardinality estimation.

1. MSCN:

A.kipf et al. [6] introduced a Deep Learning Neural Network model termed as multi-set convolutional network(MSCN). MSCN is built on a sampling-based approach that represents relational query plans by capturing query features and true cardinalities by learning to predict the (join-crossing) correlations in the data. MSCN uses bitmaps or cardinalities that are derived from samples into the training signal and addressed the weakness of sampling-based approaches (0-tuple) situation and join correlations.

MSCN consists of three modules (i.e., table, join, and predicate modules). Each of these three modules comprises a one-two layer neural network MLP module with ReLU as the activation function. MSCN consists of multiple sets as its query representation. The model then learns a set-specific, per-element neural network having shared parameters. Then an average pooling is performed on the outputs from these different modules to have an even distribution. These averaged outputs are then concatenated and fed into the final output network which is another MLP, having a sigmoid activation function at the last layer, so it outputs only a scalar value. MSCN enriches the training data with a materialized sample. A predicate will be evaluated on a sample, and a bitmap, where each bit indicates whether a tuple in the sample satisfies the predicate or not, will be added to the feature vector.MSCN model trains to reduce the mean q-error, which is the factor between an estimate and the true cardinality(or vice-versa).

2. Neurocard:

NeuroCard is a learning-based join cardinality estimator. It aims to capture the correlations across multiple joins(all the tables in a database) using the single probabilistic model. A distinctive feature of Naru is that it does not consider any independence assumptions since unlike previous density estimators, Naru and in turn NeuroCard are based on deep autoregressive(AR) models. Neurocard is primarily based on Naru, a cardinality estimator that captures the correlations among all columns of a single table. This essentially acts as a drawback which is leveraged in Neurocard. In Neurocard, the full outer join of all the tables in a schema is computed and the data distribution is learnt on this table. Once the model is trained, it can be used to estimate the cardinalities of any query given to the schema.

Neurocard achieves the following goals:

- It is a single estimator
- Efficient sampling of the full join
- Support any subset of tables
- Accurate density estimation

NeuroCard uses a standard AR architecture, ResMADE, which is also employed by Naru. Input tuples are represented as discrete, dictionary-encoded IDs. The input is given to residual blocks, each consisting of two masked linear layers. The masked layers ensure the autoregressive property. The output layer produces logits. Next, we compute a cross-entropy loss on the logits and perform back-propagation. NeuroCard can use any advanced AR architectures, if desired. We also instantiate NeuroCard with an advanced architecture such as Transformers. [15]

Comparison with MSCN:
Naru uses the joint distribution method, while MSCN is based on a regression method. It prepares the data in such a way that it can be used as the training data for a joint distribution model. Whereas in the case of MSCN, a query pool is constructed and the labels associated to each query is obtained. The MSCN model then trains on this $\langle vector, label \rangle$ pairs. Naru employs the state of the art deep autoregressive models to capture the joint distribution. Considering this architecture, Neurocard achieves a maximum Q-error of 8.5 times on the JOB-Light benchmark. Also, it outperforms MSCN by 15-34x in the JOB-light-ranges benchmark. Although, Naru is a very accurate method, it requires a longer time to update the model. It maybe inaccurate when higher update rates are required. [14] [15]

III. Research Questions Prototype Design

A. Research Questions:

After conducting a thorough qualitative and quantitative study, we try to answer the below mentioned research questions in our paper:

1) Do we have epistemic uncertainty in a model or aleatoric in another and does uncertainty relate to errors better in some models than to others?
2) To what extent can we build an ensemble of agents with their uncertainty values, to reduce the overall q-errors and improve the end estimation performance?

B. Prototype Design:

The framework which we have followed for our experiments can be depicted by the prototype design shown in Fig.8.

- Datasets: In this paper, we have chosen the real-world IMDB (Internet Movie Data Base) dataset. This is one ofthe most common dataset often used in cardinality estimation as it is full of correlations and skewed data which

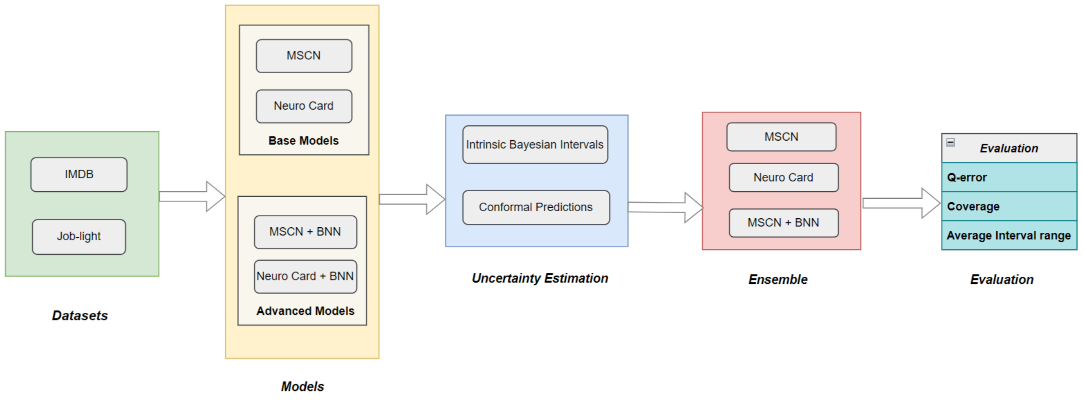


Fig. 8: Prototype architecture

makes cardinality estimation harder and challenging compared to synthetic datasets. The IMDB dataset contains information about movies,actors,directors,producers. Fig represents structure of the queries used in the experiments.We have used synthetic queries as the training data in MSCN, BNN+MSCN and BNN+Naru. JOB-Light benchmark dataset has been used to evaluate the performance of the model.

- Base Models: We train the two base models MSCN and Neurocard especially for cardinality estimation with the above mentioned dataset. Though these models are not able to give us information about model's confidence. That is why we use these models as a pretrained model for training BNN to get uncertainty estimations.We take the hidden layers of MSCN and Neurocard and use them as input embedding for the BNN.
- Advanced Models: BNN with MSCN and Naru-We train MSCN and Naru separately and consider them as base models to obtain the input embeddings. We consider the weights from the last hidden layers of both the models individually, and pass that as input to the BNN.We then train BNN to create a Beta distribution prediction over the cardinality estimates.During training we take samples from the distribution and learn the parameters to minimize the KL divergence.After the learning is done, we use the model for prediction where we take the mean of the model's predictive distribution to get a point prediction per query.The entire flow of training the BNN with embedding from MSCN/Neurocard is described in Fig.9.
- Uncertainty quantification:

1) Intrinsic Bayesian Intervals-: We investigated how well our BNN+MSCN model is capturing uncertainty by examining how accurate its predictive intervals are and also we diagnosed where our model is having problems capturing uncertainty. We have intentionally kept BNN+Neurocard out of the scope of uncertainty quantification due to severe performance issues which we discussed in later sections.

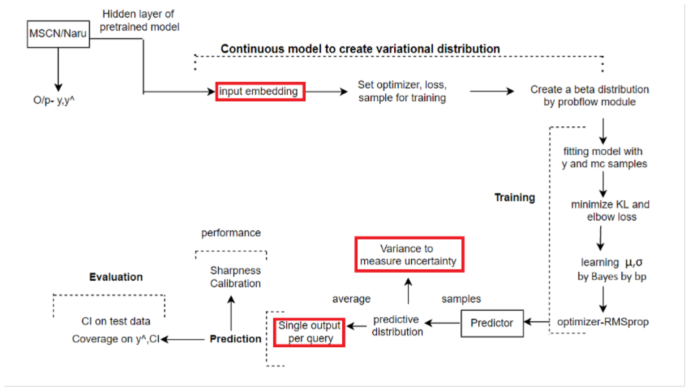
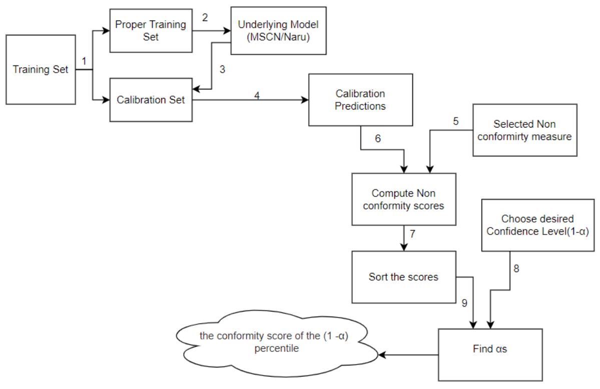

MSCN/Naru
Hidden layer of pretrained model
O/p- y,y^
Continuous model to create variational distribution
input embedding
Set optimizer, loss, sample for training
Create a beta distribution by probflow module
fitting model with y and mc samples
minimize KL and elbow loss
Training
learning μ,σ by Bayes by bp
optimizer-RMSprop
Predictor
samples
predictive distribution
Variance to measure uncertainty
average
Single output per query
Prediction
performance
Sharpness Calibration
Evaluation
CI on test data
Coverage on y^,CI

Fig. 9: Architecture for BNN+MSCN and BNN+Naru

2) Conformal Predictions with MSCN and Naru-: In this paper, we implemented the standard ICP settings. The dataset which we used is the IMDB data set.

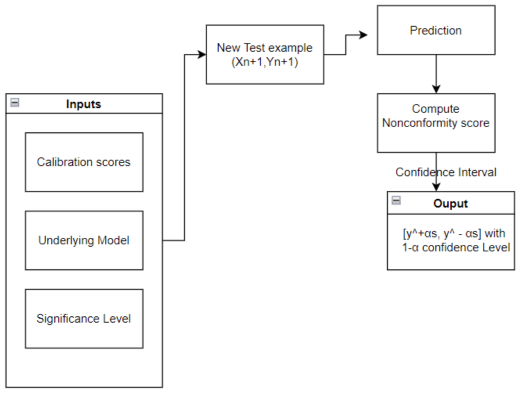
- We divided the dataset into training and test set and the training set was further partitioned randomly into a proper training set and calibration set using 90: 10 ratio.Hence the observed data shapes for training set , calibration sets and the test sets are 9000, 1000 and 70 respectively.
- We used 2 underlying machine learning models MSCN and Naru described in previous sections for conformal setting as the base models separately. These models were tuned to get the best hyper-parameters.
- Each induced model are used separately for estimating the true values of the calibration instances. Then the non-conformity scores of the calibration set were calculated using the induced models separately and the calibration score(α_s) was identified.Fig.10 highlights the detailed CP workflow.
- Now to calculate the prediction intervals of the test set, we are using the non conformity scores of the calibration set, and the user chose significance level and the induced model as input.



Training Set
Proper Training Set
Underlying Model (MSCN/Naru)
Calibration Set
Calibration Predictions
Selected Non conformirty measure
Compute Non conformity scores
Choose desired Confidence Level(1-α)
Sort the scores
Find αs
the conformity score of the (1 -α) percentile
1
2
3
4
5
6
7
8
9

Fig. 10: Detailed CP Architecture

– The induced model was first used to predict the test instance and with (α_s) at chosen significance level, that intervals are predicted for test instance which is highlighted in the Fig.11.



New Test example (Xn+1,Yn+1)
Prediction
Inputs
Calibration scores
Underlying Model
Significance Level
Compute Nonconformity score
Confidence Interval
Ouput
[y^+αs, y^ - αs] with 1-α confidence Level

Fig. 11: CP Test set Interval Prediction Workflow

– We used 100 significance levels between 0.1 to 1 and in this paper we reported the CP's performance at 0.1, 0.01, 0.05 significance levels and the corresponding confidence levels are 0.90, 0.99, 0.95 respectively.
– The efficiency was reported in terms of coverage and mean interval width at specified significance and confidence levels for the conformal set ups (MSCN+CP, MSCN+Naru).
– We also presented the Calibration plots for the above mentioned architectures which reports how the error changes for different significance levels
– Finally, we also investigate an ensemble approach, where we considered the intervals reported by every setting (MSCN+CP, Naru+CP, MSCN+BNN+CP , MSCN+BNN) and we reported the average interval width, coverage and q error distributions for the ensemble setting.

- Ensemble: As a scope of this project we also have created ensembles on the architectures we have discussed so far that identifies the best interval for each test instance based on the coverage and the tightness of the interval.Our expectation is that with ensembles we get better Q errors than the individual models and produce the best intervals around the predictions.The outcome of of the ensemble has been described in section V.
- Evaluation Metrics:We have used three evaluation metrics as described below-
1) Q-Error: Moerkotte et al. present in their paper [8] the error function Q-Error. They show, that this Q-Error is suitable for the cardinality estimation task, by deducing it from the Join Order Optimization (JOO) task and by showing, that it can guarantee some cost boundaries. The authors define the Q-Error as:

$$Q-Error(\hat{y},y) = \begin{cases} max(\frac{y}{\hat{y}},\frac{\hat{y}}{y}) & \frac{y}{\hat{y}} >= 0 \\ \infty & \text{else,} \end{cases} \quad (5)$$

where ŷ are the estimated cardinalities and y are the true cardinalities. What we can observe from the above equation is that if the estimation is perfect the resulting q-error will be equal to 1 and it can not go below this value. However, if the estimated value is worse, then q$\rightarrow\infty$. The q-error is a good choice when it comes to cardinality estimates since one obviously wants to get as close as possible to the true cardinality, but also the estimation needs to be "punished" when it's off by a large percent from the true cardinality.
2) Coverage: Coverage is used to assess the quality of the constructed prediction intervals around the true output values(targets).In this paper, we are reporting coverage which indicates how many test instances have its true value within the two bounds of the prediction interval.
3) Average Interval Width: We consider the validity and efficiency of a Conformal predictor when assessing its performance. As a part of the validity assessment, the calibration plots are presented, which plot the error rate against significance levels (0,1).As part of efficiency measurement, the width of the prediction region is reported. A narrower prediction region indicates better informational efficiency, hence a more effective model.

IV. Implementation Details

- For implementing the base models: - We have used the original repository from Github for MSCN[1] and for Neurocard [2].
- We are running our models on Pytorch framework. The version we have used is 1.10.0.
- We have used the ProbFlow package for building the probabilistic Bayesian model and to perform stochastic variational inference as described in section II. The reason for choosing probFlow module is that it is easier to develop, build, fit, and evaluate custom or already exiting

[1]https://github.com/andreaskipf/learnedcardinalities
[2]https://github.com/neurocard/neurocard.git

Bayesian models which run on top of either TensorFlow or Pytorch. The problow version we are using is 2.4.1 in our project.

- The observed training data shape for MSCN+BNN is (9000,256) and validation data shape is (1000,256).The dimension of hidden layer is 256. For Neurocard+BNN the shape of training data is (9000,672) where 672 is the hidden layer dimension. The validation data remains the same as in the case of MSCN.
- The way we have implemented Conformal Prediction in this project is motivated from the github repository of 'Synergy Conformal Prediction for Regression'[3]

V. Experiments-Results and Evaluation

A. MSCN with BNN

1) Hypothesis: The MSCN with BNN is expected to outperform the standalone MSCN model.We expect that MSCN with BNN will significantly perform better in terms of Q-error compared to the traditional MSCN and also gives us an interval prediction by which we will be able to capture both the types of uncertainties in the model. We also expect it to produce a good coverage value for the JOB-Light dataset.

2) Experiments: We have conducted various experiments on the MSCN with Bayesian model. The idea is to pre-train the MSCN model with the IMDB data set and use the weight embeddings from the MSCN model as input to the BNN. We evaluate the final performance of the models on the JOB-Light dataset. We are considering 10,000 queries during the training phase, and 70 queries during the testing phase. We use the validation set (10% of training data) to fine tune the parameters.

Learning Rate	1e-3
Epochs	100
Hidden Layers	256

TABLE I: HYPER PARAMETERS

We have experimented with multiple sampling rate (1,25,50,100) and two different optimizers (Adam, RMSProp). We have used one of the most widely used evaluation metric in cardinality estimation which is the Q-error. Finally, the following hyper parameters mentioned in table were selected after conducting several experiments.

3) Results: As expected MSCN+BNN model outperforms the individual MSCN model in terms of Q-error as we can see in Table II . We got the best Q-error and calibration curve for sampling rate as 25 and optimizer as RMSProp after experimenting with the parameters of the BNN model as depicted in Table-II and Fig.14 respectively.We also can interpret the model's uncertainty from the predictive distributions i.e. Wider the distribution more is the model's uncertainty as represented in Fig.12. We are also able to give a different representation for the predictive interval in Fig.13, where we can see towards the start and the end of the curve the interval is low indicating low variance meaning low uncertainty and variance in the middle.

[3]https://github.com/pharmbio/SCPRegression

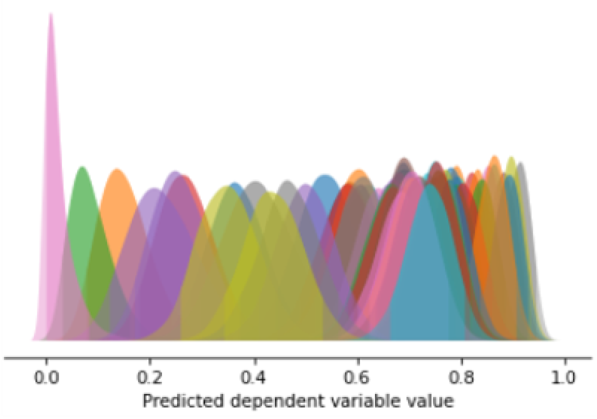


Fig. 12: Predictive distributions for MSCN+BNN model

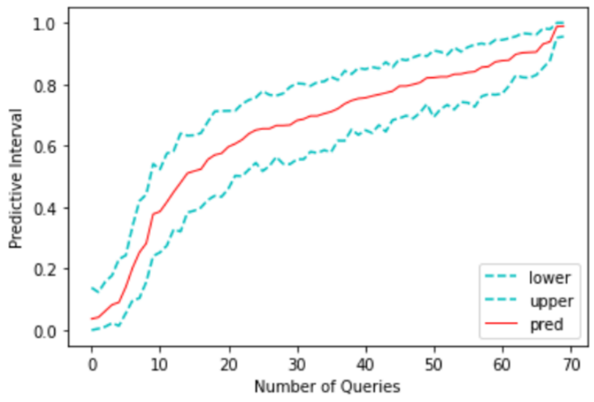


Fig. 13: Predictive Intervals for MSCN+BNN model

B. Neurocard with BNN

1) Hypothesis: We expect the Neurocard model to outperform all other models in terms of Q-error as Neurocard is the current state of the art cardinality estimator. It has a better performance compared to other cardinality estimators of the same type i.e., DeepDB which is data driven estimators and different types i.e., MSCN is a query driven estimator. We also expect to get good predictive interval and coverage from Neurocard+BNN.

2) Experiments: In the case of Neurocard, we aim to obtain the weights produced by the residual blocks which contains two masked linear layers. We extract the weights from the last layer of the hidden layer. The shape of the weights we tried to obtain is of the format [number of queries x number of hidden layers]. Since the model does not learn from representative queries in the training phase, we pass the representative queries in the testing phase of the model to extract the values for the predicted and actual estimations. The obtained resultant weight embeddings are passed on to the BNN model.

3) Results: The combination of Neurocard with BNN did not yield us with good results as compared to MSCN with

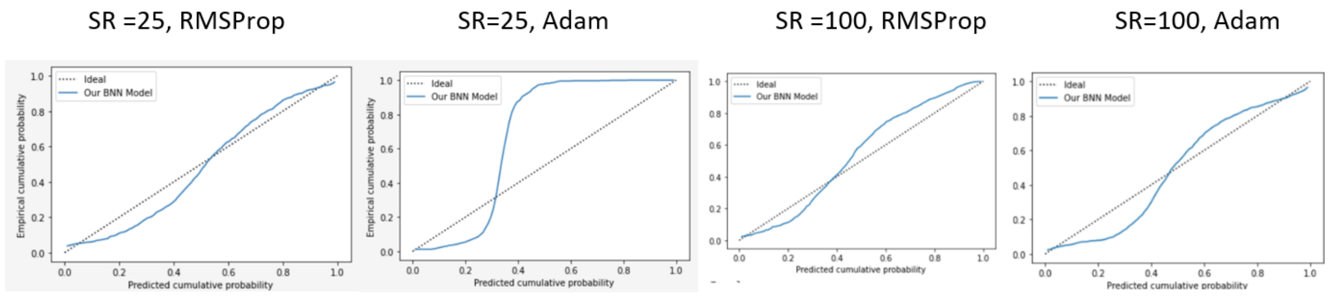


Fig. 14: Calibration curves for MSCN+BNN model

Estimator	Sampling Rate/Optimizer	median	95th	99th	max	mean
BNN +MSCN	SR=25, RMSProp	2.2	29.7	226.6	**493.2**	**14.4**
	SR=25,Adam	3.1	410.6	1744.6	2227	83.7
	SR=100,RMSProp	3.0	449.2	2021.8	3137.6	103.6
	SR=100,Adam	3.3	370.3	2003.3	2714.6	97.6
BNN+Naru	SR=25,RMSProp	271.09	23413.89	186226	489092	10048
Naru	Naru	1.36	4.48	8.89	11.0	48.58
MSCN	Adam	2.37	93.89	382.50	569.49	22.23

TABLE II: Estimation errors on the JOB-light workload

BNN. We suspect this behaviour to be because of the way the training weights were fed into the BNN model. With the case of MSCN with BNN, each weight provided by MSCN was in tandem with the representative queries the model was trained on. Whereas in the case of Neurocard, the weights were learnt on the data distribution over the entire schema and did not represent any training queries. Future enhancement can include choosing a different BNN design which handles the weights provided by a data driven estimator as well.

C. MSCN and Naru with Conformal Predictions

1) Hypothesis: We expect to compare the uncertainties captured by the standalone alone models i.e., MSCN and Naru when applying a conformal predictions framework to it. Along with experimenting on the two settings MSCN with CP, Naru with CP, we also experimented with conformal predictions on MSCN with BNN. We expect to capture uncertainty by reporting statistically valid rigorous confidence intervals around predictions with known error bounds of significance levels. Besides reporting prediction intervals, we expect to report efficiency in terms of coverage and average interval width, of the JOB-Light data set. We also expect that Naru with Conformal outperforms the other two settings in terms of coverage and average and interval width.

2) Experiments: We conducted experiments with the induced standalone models MSCN and Naru at 100 significance levels ranging from 0.1 to 0.99 on the test set. Additionally, we also considered MSCN with BNN in the conformal setting as another standalone model in our experiments.We are considering 9000 queries during the training phase and 1000 queries for calibration phase and 70 queries during the testing phase.The key idea is to use the induced models separately for the predictions.We use the predictions of test set and the calibration score α_s to report prediction intervals of the test.

Significance = 0.01, Confidence = 0.99			
Model	**Avg Interval Width**	**Coverage**	**#Instances Contributing to Coverage**
MSCN-CP	**0.45**	**1**	**70**
NARU-CP	0.56	1	70
MSCN-BNN-CP	0.57	1	70

TABLE III: Average interval width of prediction region and coverage for **MSCN-CP, NARU-CP, MSCN-BNN-CP** at confidence level 99% and significance level 0.01%

Significance = 0.05, Confidence = 0.95			
Model	**Avg Interval Width**	**Coverage**	**#Instances Contributing to Coverage**
MSCN-CP	0.28	0.84	59
NARU-CP	**0.27**	**1**	**70**
MSCN-BNN-CP	0.35	0.93	65

TABLE IV: Average interval width of prediction region and coverage for **MSCN-CP, NARU-CP, MSCN-BNN-CP** at confidence level 95% and significance level 0.05%

3) Results: In the results,we reported the confidence predictive intervals around the true values for the test instances at significance levels(0.01, 0.05, 0.1) for the three settings MSCN with CP, Naru with CP and MSCN-BNN with CP in Fig.[15, 16 , 17] . It is evident that the intervals of Naru with CP are tighter around its true output values for the three significance levels compared to other two settings.We also reported the average interval width, coverage for all the test instances at these significance levels. As we expected, the Naru+CP model

Significance = 0.1, Confidence = 0.90			
Model	**Avg Interval Width**	**Coverage**	**#Instances Contributing to Coverage**
MSCN-CP	0.18	0.47	33
NARU-CP	**0.19**	**0.94**	**66**
MSCN-BNN-CP	0.22	0.74	52

TABLE V: Average interval width of prediction region and coverage for **MSCN-CP, NARU-CP, MSCN-BNN-CP** at confidence level 90% and significance level 0.1%

Significance = 0.05, Confidence = 0.95			
Model	**Avg Interval Width**	**Coverage**	**#Instances Contributing to Coverage**
MSCN-BNN-CP	0.35	0.95	65
MSCN-CP	0.28	0.84	59
MSCN-BNN	**0.25**	**0.94**	**66**

TABLE VI: Average interval width of prediction region and coverage for **MSCN-BNN-CP, MSCN-CP, MSCN-BNN** at confidence level 95% and significance level 0.05%

is able to capture the uncertainties better than the other two settings in almost all the significance levels. We could see that the average interval width is narrower for Naru+CP model which significantly indicates better informational efficiency and hence a more effective and confident model. The average interval width and coverage of all three settings at the significance levels(0.01, 0.05, 0.1) are illustrated in the tables [III, IV, V].It is observed that with the decrease in significance level, the average interval is getting wider which means that all of the true values falls with in the predicted intervals and hence the coverage and with the increased significance level, there is a reduction in average interval width. We are able to present the conformal calibration curves of the error rates at the mentioned significance levels for all the three settings and it is evident that the error rate was reduced with reducing in significance level and increased confidence level as can be seen in Fig.18. We are also able to report the average interval width curve at all significance levels as depicted in Fig.19 and it indicates that the average interval was in a reducing trend.

D. MSCN with Conformal and BNN

1) Hypothesis: We expect that MSCN with BNN setting will outperforms MSCN with CP and MSCN and BNN with CP in terms of the capturing the uncertainty. We expect that the average interval width of the MSCN with BNN setting outperform the other two settings.We also expect that MSCN with BNN will report high coverage compared to other two settings for the JOB-Light dataset.

2) Experiments: The aim of this experiment is to compare the intervals reported by MSCN with BNN, MSCN with CP and, MSCN and BNN with CP.We experimented at different significance levels ranging from 0.1 to 0.99 for all the three settings.In this paper, we reported the experimental results at significance level 0.05 and confidence level 0.95.

3) Results: The combination of standalone MSCN with CP and MSCN-BNN with CP did not yield us best results as when compared to the intervals reported by standalone MSCN with BNN setting .We reported the results in the Table VI which highlighted the average interval width of each model setting along with coverage and the total no of instances contribute to the coverage.We identified that MSCN in combination with BNN outperforms MSCN and MSCN-BNN in combination to CP framework in terms of both the average interval width and the coverage.

E. Ensembles

1) Hypothesis: We expect that ensembles on all the architectures we have discussed so far identifies the best interval for each test instance based on the coverage and the tightness of the interval.We expect that with ensembles we get better Q errors than the models on their own.We expect that the ensemble is able to produce best and tighter intervals around the predictions.Finally,we also expect that ensemble is able to produce best average interval width and coverage for JOB-Light test data set.

2) Experiments: The aim of this experiment is to perform an ensemble on the models we have experimented with so far. The models that we considered for this experiment are MSCN with BNN, MSCN with CP, MSCN with BNN, MSCN and BNN with CP, MSCN with Naru. Our current experiment is based on the same 70 queries of the JOB-Light test set that we used in our previous experiment. In order to implement the ensemble, we consider the interval width and the coverage of all the models for each test instance. To perform our experiment, we first checked the width of the prediction interval reported by each of the models and chose the model whose interval width is smaller compared to every other model. In addition, we compared the coverage of the model of the chosen interval. Based on these two measures, we are able to get the prediction intervals of the ensemble setting. We are also able to report which model has more contribution to the ensemble prediction intervals.

3) Results: As we expected, ensemble is able to outperforms all the architectures in terms of coverage and average interval width as we can see in Table VII. In this paper, we reported the ensemble at 0.01,0.05 and 0.1 significance levels with 99%, 95%, 90% confidence respectively .We identified that Naru with Conformal setting is contributing the prediction intervals for almost all the test instances when significance levels 0.05 and 0.1 is considered and MSCN with conformal is outperforming the former when significance level is 0.01.Overall we found out that Naru with conformal setting is outperforming all the models in the ensemble setting for most of the significance levels.

VI. Conclusion & Future Work

We have conducted a number of experiments in the scope of this project, and Neurocard has given better results in terms

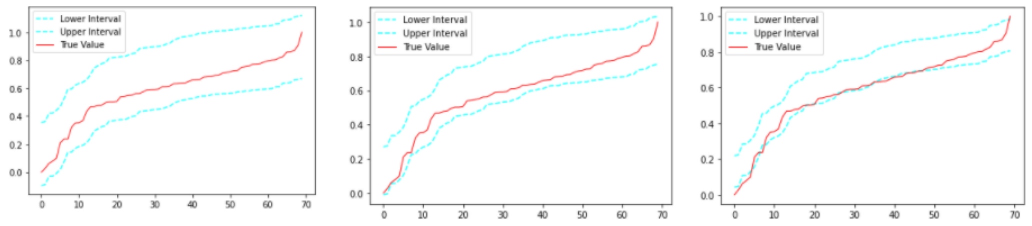


Fig. 15: Predictive Intervals of **MSCN with CP** model at 0.01 ,0.05 and 0.1 significance levels respectively

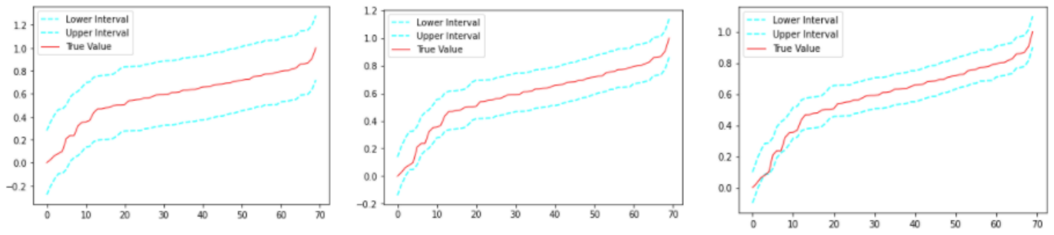


Fig. 16: Predictive Intervals of **Naru with CP** model at 0.01 ,0.05 and 0.1 significance levels respectively

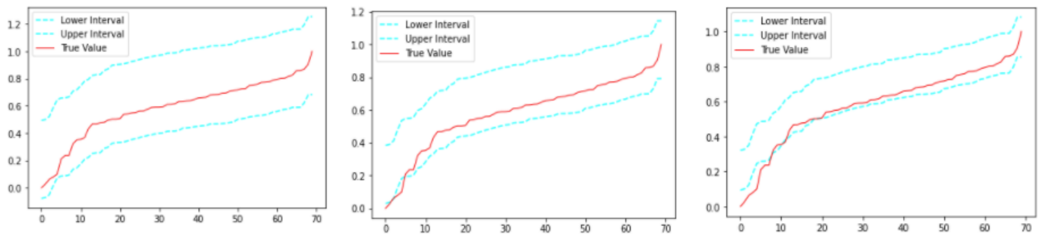


Fig. 17: Predictive Intervals of **MSCN-BNN with CP** model at 0.01 ,0.05 and 0.1 significance levels respectively

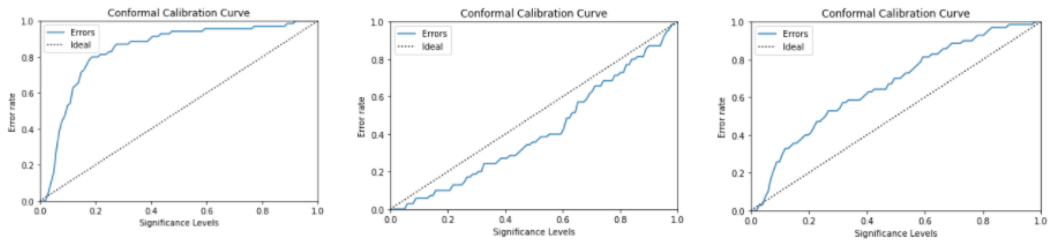


Fig. 18: Calibration plots of **MSCN , Naru and MSCN-BNN with CP** framework at all significance levels

ENSEMBLE				
Significance	**Avg Interval Width**	**Coverage**	**#Instances Contributing to Coverage**	**Q - Error**
0.01	0.45	1	70	382.5
0.05	0.27	1	70	4.48
0.1	0.19	0.94	66	2.28

TABLE VII: Ensemble Average interval width of prediction region and coverage at significance level(0.01, 0.05, 0.1)

of Q-error. In addition to improving MSCN's performance, BNN provided us with an understanding of the uncertainty associated with the model's predictions. Despite this, BNN did not perform well with Neurocard due to the (as we assume) way in which the weights of Neurocard are learned and then used as embeddings in BNN. On the other hand, along with BNN, we are using the model-agnostic distribution free Conformal Prediction to capture the uncertainty of the models and evaluated their validity and efficiency. The key outcome is that CP, regardless of the model performance is able to report prediction intervals with a controllable amount of significance levels and these intervals are statistically valid.In order to get a better Q-error estimation, we are taking the ensemble of the models' intervals, where we have CP with Neurocard as winner.

Future Work:-

1) To improve our work further,the BNN architecture can be modified so that it becomes compatible with Neurocard.
2) We can also include DeepDB to form the ensemble and compare it with other models in order to have a good Q-error estimation.

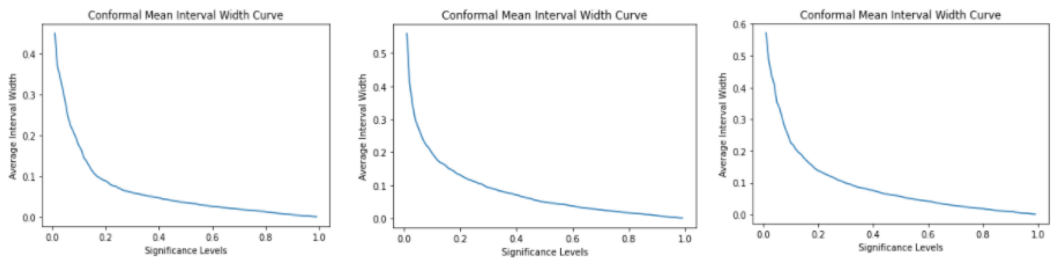


Fig. 19: Average Interval widths of **MSCN , Naru and MSCN-BNN with CP** framweowrk at all significance levels

References

[1] MS Windows NT var.inf. shorturl.at/cCGY3. Accessed: 2010-09-30.

[2] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2021.

[3] Niharika Gauraha and Ola Spjuth. Synergy conformal prediction for regression. pages 212–221, 01 2021.

[4] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110, 03 2021.

[5] Ulf Johansson, Henrik Boström, Tuve Löfström, and Henrik Linusson. Regression conformal prediction with random forests. *Machine Learning*, 97(1):155–176, Oct 2014.

[6] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. Learned cardinalities: Estimating correlated joins with deep learning, 09 2018.

[7] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. How good are query optimizers, really? *Proceedings of the VLDB Endowment*, 9(3):204–215, 2015.

[8] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proc. VLDB Endow.*, 2(1):982–993, aug 2009.

[9] H. Papadopoulos, V. Vovk, and A. Gammerman. Regression conformal prediction with nearest neighbours. *Journal of Artificial Intelligence Research*, 40:815–840, apr 2011.

[10] Harris Papadopoulos. Inductive conformal prediction: Theory and application to neural networks. In Paula Fritzsche, editor, *Tools in Artificial Intelligence*, chapter 18. IntechOpen, Rijeka, 2008.

[11] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. pages 345–356. Springer, 2002.

[12] P Griffiths Selinger, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie, and Thomas G Price. Access path selection in a relational database management system. In *Readings in Artificial Intelligence and Databases*, pages 511–522. Elsevier, 1989.

[13] Vladimir Vovk. Transductive conformal predictors. In Harris Papadopoulos, Andreas S. Andreou, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Intelligence Applications and Innovations*, pages 348–360, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[14] Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. Are we ready for learned cardinality estimation? *Proceedings of the VLDB Endowment*, 14(9):1640–1654, may 2021.

[15] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. Neurocard: One cardinality estimator for all tables. *Proc. VLDB Endow.*, 14(1):61–73, sep 2020.