POLITECNICO
MILANO 1863

Financial Engineering

Assignment 3 EPLF, Group 16

2023-2024

Alice Vailati - CP: 10683600 - MAT: 222944
Andrea Tarditi - CP: 10728388 - MAT: 251722
Jacopo Stringara - CP: 10687726 - MAT: 222456
Nicolò Toia - CP: 10628899 - MAT: 247208

Contents

1. Introduction

In this Assignment we implemented a new probabilistic forecasting method (Johnson's SU distributional DNN) and compare the methods already implemented through the pinball function as well as other metrics.
The aim in probabilistic forecasting is to capture boundaries in which a good percentage of the data lies: for example if we set as threshold 80% we should find boundaries where 80% of the data is captured. In order to do so we introduced two methods:

- Quantile regression NN
- Distributional NN

In quantile regression we take as the objective function some function (such as the Pinbal loss) whose value is related to a certain quantile.

2. Quantile regression NN

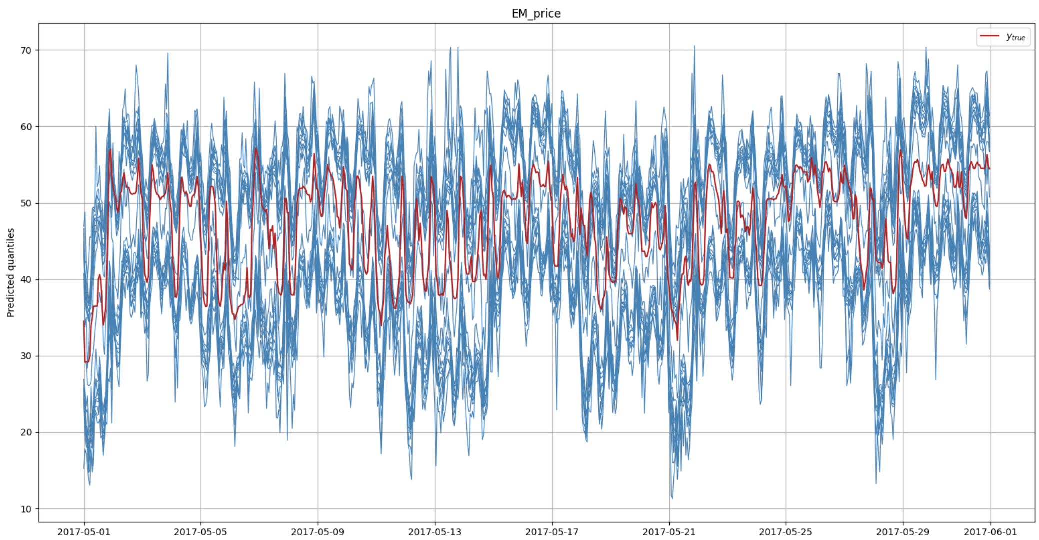
The key point for obtaining a quantile regression NN is to extend the DNN output layer. In order to do so we practice the common DNN regression

$$\begin{cases} l_1 & = g(x_i \cdot W_1 + b_1) \\ l_2 & = g(l_2 \cdot W_2 + b_2) \cdot W_3 + b_3 \\ & \vdots \\ W_1 & \in \mathbb{R}^{n_x \times n_{u_1}},\ W_2 \in \mathbb{R}^{n_{u_1} \times n_{u_2}}, \\ W_3 & \in \mathbb{R}^{n_{u_2} \times H \cdot n_p}, n_{u_1}, n_{u_2} \in \mathbb{Z}^+, \\ b_1 & \in \mathbb{R}^{n_{u_1}},\ b_2 \in \mathbb{R}^{n_{u_2}},\ b_3 \in \mathbb{R}^{H \cdot n_p} \end{cases}$$

Then we try to approximate the deciles of our data's distribution. In order to do that, we need an objective function that captures the quantiles of the data.
We use the pinball loss function:

$$n_p = \#\Gamma \text{ (number of quantiles)}$$
$$\sum_i \sum_h \sum_\gamma (y_i^h - \hat{q}_\gamma^h(x_i))^+ \gamma + (y_i^h - \hat{q}_\gamma^h(x_i))^- (1-\gamma)$$

The result's obtained are the quantiles (in blue) of distribution of the data around the mean (in red).



3. Distributional NN

The set of equations that parameterize the DNN are very similar to the previous case:

$$\begin{cases} l_1 & = g(x_i \cdot W_1 + b_1) \\ l_2 & = g(l_2 \cdot W_2 + b_2) \cdot W_3 + b_3 \\ & \vdots \\ W_1 & \in \mathbb{R}^{n_x \times n_{u_1}},\ W_2 \in \mathbb{R}^{n_{u_1} \times n_{u_2}}, \\ W_3 & \in \mathbb{R}^{n_{u_2} \times H \cdot n_p},\ n_{u_1},\ n_{u_2} \in \mathbb{Z}^+, \\ b_1 & \in \mathbb{R}^{n_{u_1}},\ b_2 \in \mathbb{R}^{n_{u_2}},\ b_3 \in \mathbb{R}^{H \cdot n_p} \end{cases}$$

What distinguishes these types of DNN is the parameters we receive as output. Rather than computing the quantiles directly Distributional NN give as output a set of parameter to identify a given distribution function. As an objective function for such models one can either use the Pinball score (or any other quantile-based loss function) or a loss function based directly on the distribution function given the output values (e.g. the log-likelihood ratio function). All other techniques (pruning, tuning, etc. etc.) we have seen so far apply to this kind of models too.

3.1. Normal DNN

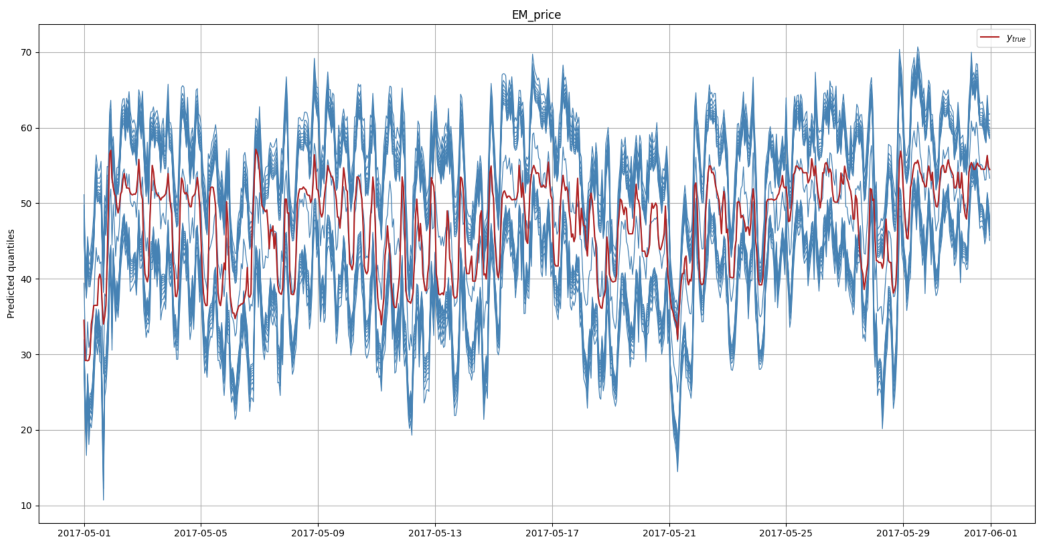
In the normal DNN a normal distribution for the output values is assumed. Now, since a Normal distribution is fully identified by computing its mean and variance, these will be our output parameters.

$$\begin{cases} \mu_i^h & = \ell_2^h \\ \sigma_i^h & = \epsilon + 3 \cdot \text{Softplus}(\ell_2^{[H+h]}) \end{cases}$$

Where the function $\text{Softplus}(x) = \log(1 + \exp(x))$. For such a model we chose to use a log-likelihood objective function. Let us recall that a Normal distribution has the following density function:

$$f(\chi) = \frac{1}{\sigma_i^h \sqrt{2\pi}} \exp\left\{ -\frac{1}{2} \left(\frac{\chi - \mu_i^h}{\sigma_i^h} \right)^2 \right\}$$

Utilizing this method with our data has resulted in the following graph:



3.2. Johnson's SU distribution

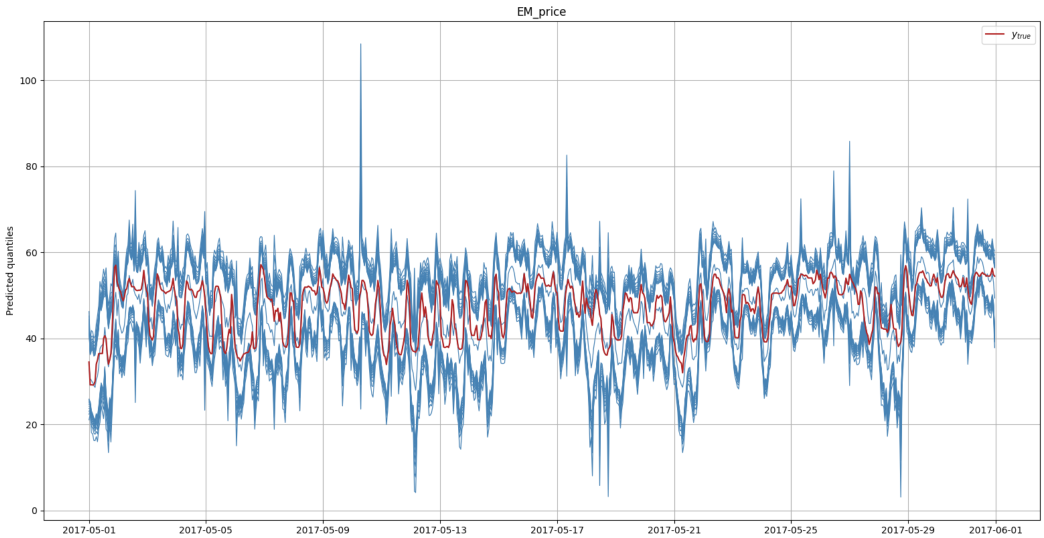
On the other hand, for Johnson's SU distribution, we have the following parameters:

$$\begin{cases} \lambda_i^h &= \ell_2^{[h]} \\ \sigma_i^h &= \epsilon + \gamma \cdot \text{Softplus}(\ell_2^{[H+h]}) \\ \tau_i^h &= 1 + \gamma \cdot \text{Softplus}(\ell_2^{[2H+h]}) \\ \zeta_i^h &= \ell_2^{[3H+h]} \end{cases}$$

And the same objective function as above on the following distribution function:

$$f^h(\chi) = \frac{\tau_i^h}{\sigma_i^h \sqrt{2\pi}} \frac{1}{\sqrt{1 + \left(\frac{\chi - \lambda_i^h}{\sigma_i^h}\right)^2}} \exp\left\{-\frac{1}{2}\left[\zeta_i^h + \tau_i^h sinh^{-1}\left(\frac{\chi - \lambda_i^h}{\sigma_i^h}\right)\right]^2\right\}$$

Applying this technique to the full month of May with hyper-parameters tuned with a random search yields the following graph:



4. Model comparison

4.1. Pinball loss

In order to evaluate the performance of the models introduced above, we proceed to compare them using the Pinball loss, which is a measure of fit for quantiles. For a given calibrated quantile we can compute the following Pinball loss:

$$\mathcal{P}(\alpha, \hat{q}_{\alpha,n}, y_n) = \alpha \cdot [y_n - \hat{q}_{\alpha,n}] \cdot \mathbb{I}\{y_n \geq \hat{q}_{\alpha,n}\} + (1-\alpha) \cdot [\hat{q}_{\alpha,n} - y_n] \cdot \mathbb{I}\{y_n < \hat{q}_{\alpha,n}\}$$

$$\text{PinBall}(\alpha) = \frac{1}{N} \cdot \sum_{n=1}^{N} \mathcal{P}(\alpha, \hat{q}_{\alpha,n}, y_n)$$

where N is the number of samples.

Furthermore, we can aggregate the scores obtained for each of the deciles we have calibrated as follows to obtain the Average Pinball Loss:

$$\text{APL} = \frac{1}{20} \cdot \left(\sum_{\alpha=1\%}^{10\%} \text{PinBall}(\alpha) + \sum_{\alpha=90\%}^{99\%} \text{PinBall}(\alpha) \right)$$

Initially, understand the models' performances, we chose to compute and plot the APL by grouping by hour of the day, which yielded the following graph:

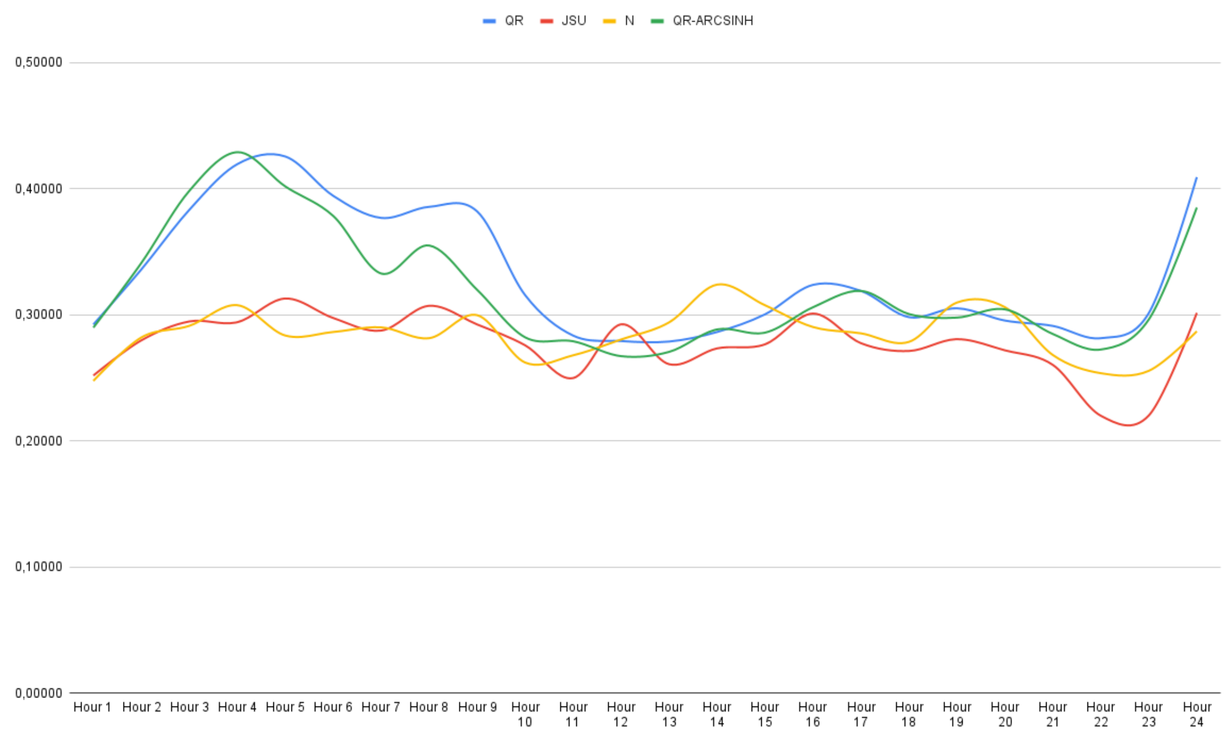


Figure 1: APL by hour

As we can see from the graph, the JSU and Normal models out perform the two others. Between the two the JSU has a slightly better performance overall, but this was to be expected since it is an extension of the Normal distribution. Indeed we have that the Johnson SU distribution variable χ can be written as the following transformation of a standard normal z:

$$\chi = \sigma \sinh\left(\frac{z-\zeta}{\tau}\right) + \lambda$$

Similarly employing an hyperbolic function such as arcsinh for the Quantile Regression leads to a lower APL on almost all hours of day.
Finally, since the data show some consistent patterns without noticeable peaks of error, we chose to aggregate the hours with a simple mean. Such an aggregated metric is particularly useful to easily compare the performance of different models at a glance and summarily. The scores of each of the models we used is reported in the following table:

QR-DNN	N-DNN	JSU-DNN	QR-DNN-ARCSINH
0,33181	0,28497	0,27710	0,32020

Table 1: APL for different models

We can see at a glance that the two distributional models have much lower Pinball scores and overall the JSU is the best performing model. Let us also remark that applying the arcsinh transformation helps the Quantile regression improve slightly.
This was to be expected, in general applying the arcsinh^{-1} transformation, as we have explained for the JSU distribution, is a relatively simple way of making our model for the prices more flexible without over-complicating it since it still possesses some of the properties of the original model.
In particular the arcsinh^{-1} behaves very simlarly to a log transformation for large values of the mean and variance (e.g. in the peak hours of the day) while for lower values of the mean and variance (such as during holidays or night-time) the JSU (and the arcsinh^{-1} in general) tend to better reproduce real observed data.

4.2. Winkler Score

In addition to the above mentioned Pinball score, we also employed a scoring mechanism for intervals, in order to understand their goodness of fit. In particular we chose to use the Winkler score, which is defined as follows:

$$\mathrm{W}_n = \begin{cases} \delta_n & \text{if: } y_n \in [\hat{L}_n, \hat{U}_n] \\ \delta_n + \frac{2}{1-\alpha} \cdot (\hat{L}_n - y_n) & \text{if: } y_n < \hat{L}_n \\ \delta_n + \frac{2}{1-\alpha} \cdot (y_n - \hat{U}_n) & \text{if: } y_n > \hat{U}_n \end{cases}$$

where $\delta_n = \hat{U}_n - \hat{L}_n$ is the width of Interval and $\hat{U}_n$ and $\hat{L}_n$ are the estimated upper and lower bound respectively for said interval.
Computing the Winkler scores for the 4 different models we decided employ yields the following graphs:

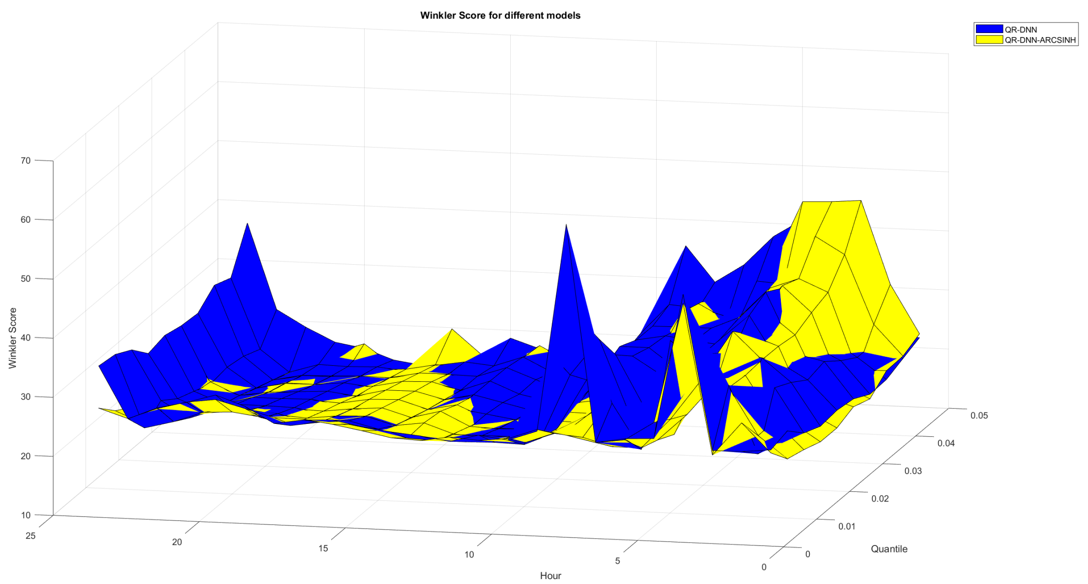


Figure 2: QR with and without Arcsinh corrections

Overall we can see that the two distributional models perform much better than the Quantile Regression, just like we saw for the Pinball Score.
This means that these models, not only perform better on average on the quantile but also possess rather narrow and precise probabilistic intervals.
Again applying the arcsinh transformation improves the QR's score. Indeed, even though it cannot be clearly grasped from the first graph, wherever the transformed QR has an higher score, the vanilla one likewise surges, but the reverse is almost never true.

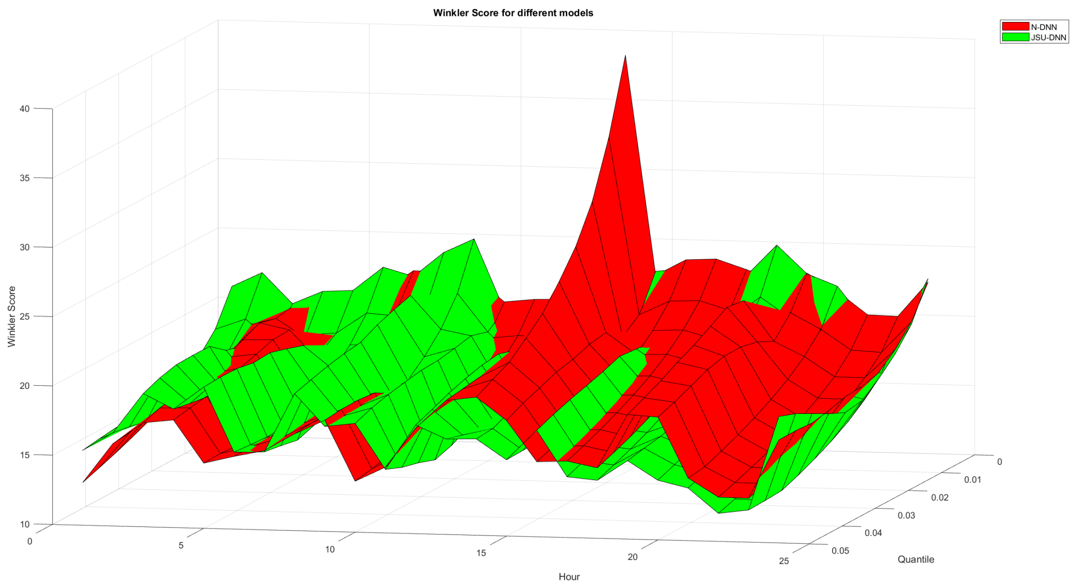


Figure 3: JSU vs Normal approach

5. Appendix

5.1. Code

5.1.1 Winkler Score

```
def compute_winkler_scores(y_true, pred_quantiles, quantiles_levels):
"""
Utility function to compute the winkler score on the test results
return: winkler scores computed for each quantile level and each step in the pred
                                    horizon
"""
score = []
# loop over only half the quantiles
# the other half is symmetric, winkler is only applied to quantiles above (or below)
                                    the median
for i, tau in enumerate(quantiles_levels[:len(quantiles_levels)//2]):
    # get the upper and lower quantiles
    L_tau = pred_quantiles[:, :, i]
    U_tau = pred_quantiles[:, :, -i-1]
    # compute the quantile width
    delta_tau = np.subtract(U_tau, L_tau)
    # compute the errors
    error_L = np.subtract(L_tau, y_true)
    error_U = np.subtract(y_true, U_tau)
    # compute the winkler score
    # use tau, not 1-tau, as the quantiles are symmetric
    loss_q = delta_tau + 2 / tau * (
        np.maximum(error_L, np.zeros(error_L.shape))
        + np.maximum(error_U, np.zeros(error_U.shape))
    )
    score.append(np.expand_dims(loss_q,-1))
score = np.mean(np.concatenate(score, axis=-1), axis=0)

return score
```

5.1.2 Archsinh transformation

```
dir_path = os.getcwd()
ds = pd.read_csv(os.path.join(dir_path, 'data', 'datasets', configs['data_config'].
                                        dataset_name))
ds.set_index(ds.columns[0], inplace=True)

if apply_arcsinh_transf:
    ds['TARG__'+PF_task_name] = np.arcsinh(ds['TARG__'+PF_task_name])

...

# apply inverse sinh transformation to all the predictions
if apply_arcsinh_transf:
    test_predictions = np.sinh(test_predictions)
```

5.1.3 JSU implementation

```
class TensorflowRegressor():
    ...
    def __pred_JSU_params__(self, pred_dists: tfp.distributions):
        loc = tf.expand_dims(pred_dists.loc, axis=-1)
        scale = tf.expand_dims(pred_dists.scale, axis=-1)
        tailweight = tf.expand_dims(pred_dists.tailweight, axis=-1)
        skewness = tf.expand_dims(pred_dists.skewness, axis=-1)
        # Expand dimension to enable concat in ensemble
        return tf.expand_dims(tf.concat([loc, scale, tailweight, skewness], axis=-1),
                                                    axis=2)

...

class Ensemble():
    ...
    @staticmethod
    def __build_JSU_PIs__(preds_test, settings):
        # for each de component, sample, aggregate samples and compute quantiles
        pred_samples = []
        for k in range(preds_test.shape[2]):
            pred_samples.append(tfd.JohnsonSU(
                loc=preds_test[:,:,k,0],
                scale=preds_test[:,:,k,1],
                tailweight=preds_test[:,:,k,2],
                skewness=preds_test[:,:,k,3]).sample(10000).numpy())
        return np.transpose(np.quantile(np.concatenate(pred_samples, axis=0),
        q=settings['target_quantiles'], axis=0),
            axes=(1, 2, 0)).reshape(-1, len(settings['target_quantiles']))
```

5.2. Numerical Results

5.2.1 Hyperparameters

Model	hidden size	learning rate
QR-DNN	64	0.0005255314302093036
QR-DNN-ARCSINH	192	0.0006856111192110566
N-DNN	896	0.0007802798828823418
JSU-DNN	448	0.0006305512659056026

Table 2: Tuned Hyperparameters

5.2.2 APL by hour

Hour	QR-DNN	QR-DNN-ARCSINH	N-DNN	JSU-DNN
1	0,29231	0,28984	0,247568	0,25183
2	0,33598	0,34112	0,282023	0,27975
3	0,38324	0,39829	0,291244	0,29471
4	0,41945	0,42894	0,307608	0,29411
5	0,42554	0,40178	0,283546	0,31290
6	0,39430	0,37834	0,286357	0,29744
7	0,37681	0,33269	0,290020	0,28748
8	0,38561	0,35493	0,281448	0,30712
9	0,38194	0,31975	0,299677	0,29231
10	0,3155	0,28210	0,262115	0,27578
11	0,2835	0,27901	0,267859	0,24982
12	0,2791.	0,26718	0,280416	0,29243
13	0,2787	0,27057	0,293974	0,26081
14	0,2863	0,28837	0,323927	0,27334
15	0,3003	0,28583	0,307560	0,27657
16	0,3237	0,30596	0,290298	0,30098
17	0,3189	0,31890	0,285273	0,27746
18	0,2981	0,30058	0,278519	0,27128
19	0,3051	0,29773	0,309691	0,28066
20	0,2954	0,30432	0,306046	0,27181
21	0,2911	0,28475	0,268080	0,26023
22	0,2814	0,27244	0,253664	0,21992
23	0,3013	0,29630	0,255552	0,22016
24	0,4091	0,38504	0,286920	0,30148

Table 3: APL by hour

5.2.3 Winkler Scores

Hour	0,005	0,01	0,015	0,02	0,025	0,03	0,03	0,04	0,045	0,05
1	23,849	24,709	22,016	20,811	21,163	20,746	19,197	19,342	20,548	21,727
2	23,349	20,598	29,476	28,021	26,102	26,859	24,663	22,880	22,904	22,044
3	23,327	32,003	26,632	31,973	28,919	28,009	27,084	25,998	29,377	36,506
4	49,557	34,125	30,600	32,572	36,591	38,195	36,501	33,802	32,405	38,927
5	24,914	39,179	34,513	32,259	32,339	33,155	32,930	32,734	32,234	40,846
6	24,678	20,508	24,552	24,685	25,781	24,969	26,412	28,867	32,599	37,784
7	23,827	28,119	24,353	22,518	25,840	24,370	26,932	26,751	26,342	32,808
8	60,556	41,690	36,380	31,405	28,859	27,941	27,597	26,766	27,575	29,638
9	25,313	35,589	30,157	26,964	26,066	27,928	26,678	26,662	27,989	35,593
10	23,587	20,501	19,229	18,561	20,020	20,595	21,871	21,730	22,773	22,580
11	24,217	20,645	19,290	18,188	17,263	16,146	15,252	14,512	13,334	16,660
12	22,956	20,527	19,269	18,136	17,232	16,451	15,686	14,827	13,659	13,024
13	23,672	20,547	19,223	17,981	17,979	17,580	16,675	16,770	15,643	16,162
14	24,140	20,709	19,112	18,853	18,728	18,699	18,674	18,597	18,050	17,092
15	24,790	22,061	20,335	19,237	18,353	17,596	16,917	16,915	16,326	18,687
16	25,739	23,236	21,712	20,313	19,309	18,565	17,890	17,013	16,794	15,115
17	25,517	22,789	21,515	20,355	19,385	18,500	17,615	16,797	15,990	15,711
18	24,711	22,012	20,293	19,251	18,430	17,591	16,734	16,017	14,793	13,439
19	27,406	23,133	21,472	20,147	19,232	18,229	17,309	16,106	15,316	14,142
20	29,105	24,471	21,770	20,257	19,121	17,945	16,841	15,896	15,154	15,888
21	27,643	23,600	21,860	20,283	19,117	17,643	16,759	15,815	15,295	16,725
22	26,520	21,825	20,119	18,971	17,785	16,902	15,813	14,810	13,778	19,048
23	24,430	20,550	22,841	20,819	19,724	18,601	19,096	18,611	17,453	21,834
24	33,176	32,778	31,207	28,244	28,965	28,257	28,008	30,413	29,304	36,273

Table 4: QR-DNN

Hour	0,005	0,01	0,015	0,02	0,025	0,03	0,03	0,04	0,045	0,05
1	20,464	18,509	17,277	16,337	15,592	14,956	14,422	13,936	13,495	13,074
2	21,381	19,318	18,014	17,255	17,291	17,124	16,786	16,430	16,088	15,948
3	20,127	19,894	20,210	19,640	19,017	18,487	18,302	18,184	17,931	17,506
4	20,466	18,518	19,761	19,707	19,305	18,929	18,334	18,203	17,993	17,762
5	20,753	18,750	17,531	16,580	15,847	15,208	14,656	14,653	14,761	14,704
6	21,410	19,344	18,046	17,097	16,342	15,683	15,106	15,336	15,316	15,248
7	22,928	20,743	19,344	18,326	17,487	16,789	16,423	16,280	15,994	15,720
8	22,848	20,618	19,249	18,194	17,516	17,580	17,363	17,120	17,120	16,987
9	22,607	20,417	19,038	18,014	17,275	17,388	18,075	18,191	18,173	18,030
10	20,844	18,860	17,576	16,644	15,891	15,258	14,698	14,218	13,934	13,682
11	21,081	19,110	17,822	16,850	16,482	16,177	15,840	15,498	15,124	14,750
12	21,145	19,082	17,875	16,923	16,493	17,483	17,822	17,988	17,916	17,631
13	21,335	19,280	18,739	19,080	19,304	19,564	19,949	20,058	19,865	19,676
14	38,864	33,579	29,590	26,980	25,103	23,431	22,297	21,225	20,614	19,974
15	23,045	20,838	19,893	20,842	20,625	20,181	19,905	19,410	18,950	18,348
16	24,219	21,877	20,393	19,305	18,404	17,655	17,009	16,424	15,894	15,429
17	24,331	21,964	20,497	19,374	18,458	17,728	17,090	16,511	15,989	15,528
18	23,626	21,318	19,903	18,853	18,017	17,291	16,650	16,061	15,562	15,097
19	22,826	21,671	22,722	22,895	22,156	21,433	20,837	20,146	19,562	19,004
20	23,476	21,259	19,837	20,444	20,717	20,699	20,378	19,832	19,306	18,871
21	22,218	20,125	18,801	17,792	16,994	16,306	15,721	15,181	14,701	14,532
22	20,590	18,680	17,403	16,493	15,745	15,124	14,566	14,082	13,622	13,208
23	20,532	18,571	17,383	16,470	15,710	15,048	14,510	14,021	13,590	13,184
24	23,062	20,853	19,470	18,417	18,095	18,000	17,633	17,394	17,469	17,454

Table 5: N-DNN

Hour	0,005	0,01	0,015	0,02	0,025	0,03	0,03	0,04	0,045	0,05
1	21,454	19,004	17,612	16,578	16,415	16,071	15,760	15,744	15,594	15,402
2	22,506	19,800	18,184	17,003	16,109	15,341	15,962	16,240	16,525	16,731
3	20,319	17,931	18,062	19,191	19,518	20,445	20,452	20,412	20,100	19,611
4	21,263	18,767	17,246	16,794	16,987	16,980	18,020	18,143	18,314	18,550
5	21,387	18,951	18,127	19,342	19,744	20,110	20,019	20,242	20,269	20,194
6	23,123	20,452	18,806	17,632	16,731	15,969	15,462	15,373	15,444	15,555
7	22,511	20,299	18,851	17,790	16,958	16,252	15,610	15,192	15,431	15,625
8	24,297	21,284	22,481	22,095	21,656	21,031	20,858	20,593	20,254	19,888
9	25,332	22,042	20,095	18,752	17,693	17,364	17,521	17,725	17,670	17,574
10	20,242	19,317	19,685	19,416	19,190	19,124	18,896	18,653	18,328	17,853
11	19,274	17,239	16,030	15,120	14,396	13,758	13,357	13,780	14,094	14,593
12	22,806	20,046	19,018	19,379	19,767	19,601	19,267	18,858	18,341	17,850
13	19,970	17,805	16,509	16,057	16,439	16,349	16,013	16,392	16,539	16,900
14	20,561	18,315	19,336	19,633	19,192	18,887	18,270	17,615	17,064	16,960
15	23,334	20,735	19,115	17,989	17,057	16,295	15,631	15,598	15,547	15,507
16	23,553	21,104	19,585	19,703	19,818	19,365	18,978	18,615	18,186	17,693
17	23,757	21,086	19,502	18,358	17,430	16,650	16,001	15,403	14,884	14,394
18	23,207	20,708	19,150	18,009	17,118	16,379	15,723	15,166	14,665	14,199
19	25,451	22,342	20,443	19,132	18,087	17,259	16,527	16,148	15,929	15,670
20	23,423	20,871	19,319	18,199	17,289	16,558	15,893	15,310	14,795	14,315
21	22,595	20,085	18,635	17,532	16,660	15,936	15,304	14,744	14,264	13,808
22	19,885	17,633	16,306	15,307	14,551	13,917	13,362	12,863	12,435	12,036
23	19,177	17,085	15,815	14,861	14,113	13,483	12,932	12,461	12,378	12,339
24	23,323	20,677	19,134	17,974	17,063	16,856	17,407	18,066	18,697	19,144

Table 6: JSU-DNN

Hour	0,005	0,01	0,015	0,02	0,025	0,03	0,03	0,04	0,045	0,05
1	21,454	19,004	17,612	16,578	16,415	16,071	15,760	15,744	15,594	15,402
2	22,506	19,800	18,184	17,003	16,109	15,341	15,962	16,240	16,525	16,731
3	20,319	17,931	18,062	19,191	19,518	20,445	20,452	20,412	20,100	19,611
4	21,263	18,767	17,246	16,794	16,987	16,980	18,020	18,143	18,314	18,550
5	21,387	18,951	18,127	19,342	19,744	20,110	20,019	20,242	20,269	20,194
6	23,123	20,452	18,806	17,632	16,731	15,969	15,462	15,373	15,444	15,555
7	22,511	20,299	18,851	17,790	16,958	16,252	15,610	15,192	15,431	15,625
8	24,297	21,284	22,481	22,095	21,656	21,031	20,858	20,593	20,254	19,888
9	25,332	22,042	20,095	18,752	17,693	17,364	17,521	17,725	17,670	17,574
10	20,242	19,317	19,685	19,416	19,190	19,124	18,896	18,653	18,328	17,853
11	19,274	17,239	16,030	15,120	14,396	13,758	13,357	13,780	14,094	14,593
12	22,806	20,046	19,018	19,379	19,767	19,601	19,267	18,858	18,341	17,850
13	19,970	17,805	16,509	16,057	16,439	16,349	16,013	16,392	16,539	16,900
14	20,561	18,315	19,336	19,633	19,192	18,887	18,270	17,615	17,064	16,960
15	23,334	20,735	19,115	17,989	17,057	16,295	15,631	15,598	15,547	15,507
16	23,553	21,104	19,585	19,703	19,818	19,365	18,978	18,615	18,186	17,693
17	23,757	21,086	19,502	18,358	17,430	16,650	16,001	15,403	14,884	14,394
18	23,207	20,708	19,150	18,009	17,118	16,379	15,723	15,166	14,665	14,199
19	25,451	22,342	20,443	19,132	18,087	17,259	16,527	16,148	15,929	15,670
20	23,423	20,871	19,319	18,199	17,289	16,558	15,893	15,310	14,795	14,315
21	22,595	20,085	18,635	17,532	16,660	15,936	15,304	14,744	14,264	13,808
22	19,885	17,633	16,306	15,307	14,551	13,917	13,362	12,863	12,435	12,036
23	19,177	17,085	15,815	14,861	14,113	13,483	12,932	12,461	12,378	12,339
24	23,323	20,677	19,134	17,974	17,063	16,856	17,407	18,066	18,697	19,144

Table 7: QR-DNN-ARCSINH