Report: Predict Bike Sharing Demand with AutoGluon Solution

Haila Bushlaibi

Initial Training

What did you realize when you tried to submit your predictions? What changes were needed to the output of the predictor to submit your results?

At the beginning of the project, I downloaded the necessary packages and set up a Kaggle API key. I then uploaded the dataset and reviewed its properties using the describe function to review key values such as min, max, and mean.

After initial testing, I trained the model using the AutoGluon library, using the Root Mean Squared Error (RMSE) metric to evaluate model performance. Upon completion of training, predictions were generated based on the test data.

While reviewing the predictions, I noticed some negative values, and Kaggle would reject the submission if we didn't make all values greater than zero. Therefore, I converted all negative values to zero. The predictions were then added to a submission dataframe, and the file was then saved and submitted to Kaggle.

What was the top ranked model that performed?

The top ranked model is WeightedEnsemble_L3 model and provided the best performance compared to other models, as it obtained the lowest value for the Root Mean Squared Error measure,and the best submission score for kaggle 0.45804.

Exploratory data analysis and feature creation

What did the exploratory analysis find and how did you add additional features?

A histogram was created for all the features in the dataset to display the distribution of each feature relative to the data.

Next, I added new features based on the datetime column in the data, extracting the year, month, day, and hour. These additions were applied to both the training and test data. I also con the season and weather columns, into categorical data types.

How much better did your model preform after adding additional features and why do you think that is?

The model’s performance improved after adding additional features, as the RMSE score decreased from 1.99863 to 0.61168. I believe these features helped the model better recognize seasonal and temporal patterns in the data, such as the effect of the time of day, month, or weather conditions, which contributed to improving prediction accuracy.

Hyper parameter tuning

How much better did your model preform after trying different hyper parameters?

The model's performance improved significantly after tuning the hyperparameters, as the Root Mean Squared Error (RMSE) decreased from 1.99863 in the baseline model to 0.61168 after adding features, and then to 0.45804 after experimenting with different hyperparameter.

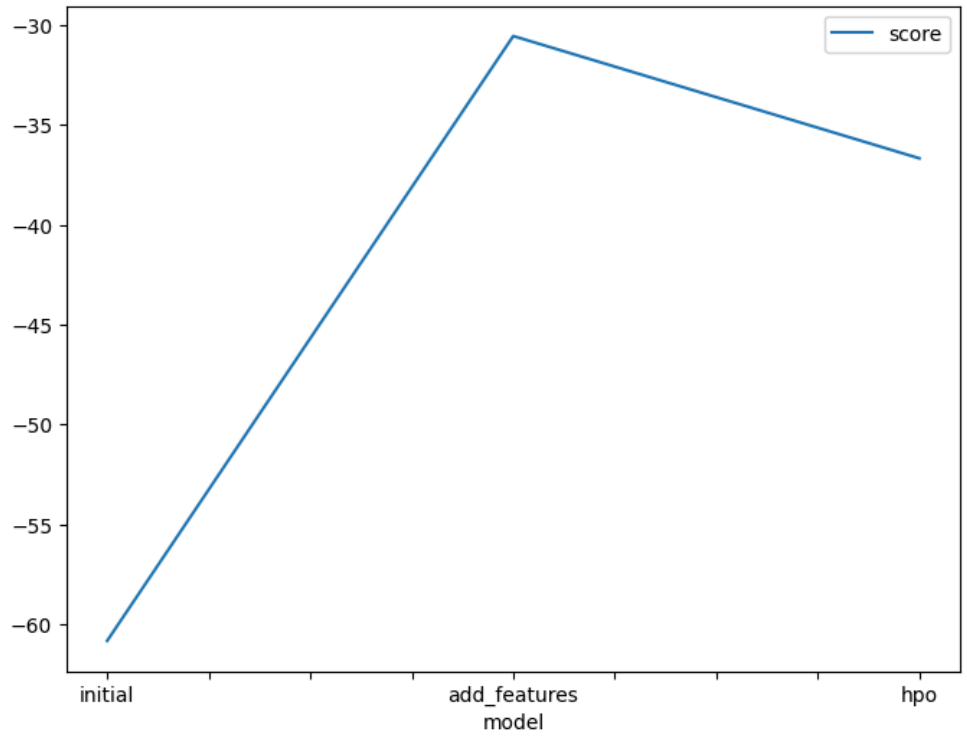
If you were given more time with this dataset, where do you think you would spend more time?

If I had more time, I would improve hyperparameter tuning by experimenting with different parameters to achieve better model performance. I would also enhance feature engineering by extracting additional features. Additionally, I would explore different methods to handle negative values instead of simply converting them to zero.

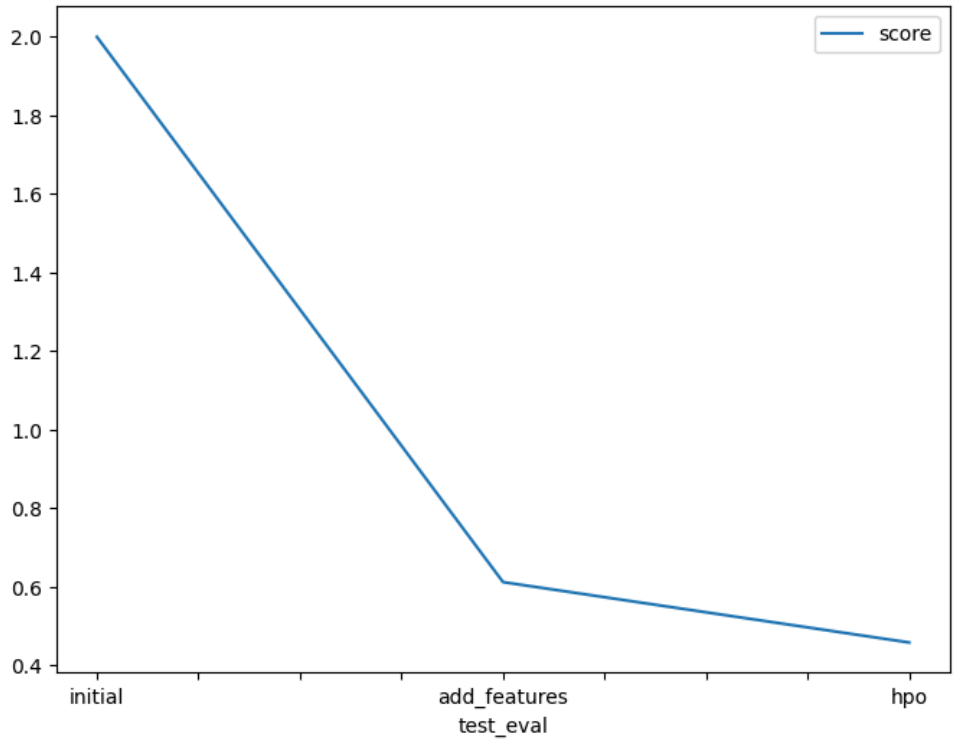
Create a table with the models you ran, the hyperparameters modified, and the kaggle score.

Model	hpo1	hpo2	hpo3	Score
initial	default	default	default	1.99863
add_features	default	default	default	0.61168
hpo	num_layers: space.Int(2,5)	dropout_prob: space.Real(0.1, 0.3)	learning_rate: space.Real(1e-4, 1e-1, log=True)	0.45804

Create a line plot showing the top model score for the three (or more) training runs during the project.



Create a line plot showing the top kaggle score for the three (or more) prediction submissions during the project.



Summary

In this project, I used the AutoGluon library to train a regression model that predicts bike sharing demand. I started with exploratory data analysis to understand the distribution of features and added new features extracted from the date and time column, such as year, month, day, and hour. Additionally, I converted some columns to categorical data types to improve the model's performance.

After adding the new features, the model's performance improved significantly, with the Root Mean Squared Error (RMSE) decreasing from 1.99863 to 0.61168. Then, I tuned the model's hyperparameters using techniques such as adjusting the number of layers, dropout probability, and learning rate, which led to a further improvement in performance, lowering the RMSE to 0.45804.