Report: Predict Bike Sharing Demand with AutoGluon Solution

NAME HERE

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?

The first issue I faced was with the date column—it wasn't useful because the model couldn't understand it in its original format. At first, I created extra features from it, but later I realized that the next question asked me to make the same features. So, I had to go back, delete them, and retrain the model. It was a bit frustrating, but I learned a lot from doing it again.

Another thing I realized, thanks to your feedback, was that two columns were in the training set but missing in the test set. Now I won't forget that both sets should have the same columns and structure.

Lastly, I had to set any negative values to 0 because negative counts don't make sense in this case.

What was the top ranked model that performed?

WeightedEnsemble_L3 -53.171528 root_mean_squared_error

Exploratory data analysis and feature creation

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

From the exploratory analysis, I found that most values in the holiday column were 0, meaning most bike-sharing activity happened on non-holiday (working) days. The distribution of season, temperature (temp, atemp), and humidity was fairly normal. However, weather conditions were mostly skewed to the left, indicating that most rides occurred under clear sky conditions.

For the second question, I added additional features by first converting the date column to datetime format. Then, I extracted useful features from it, such as year, day, hour, and weekday. In the end, I dropped the original date column since it was no longer needed.

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

I noticed a significant improvement in my model's performance after adding additional features. The RMSE (Root Mean Squared Error) decreased from -52 to -32, which shows that the model's predictions became much more accurate.

I believe this improvement happened because the model gained more relevant features that helped it better understand the relationship between time-related variables and other columns. Since time plays a key role in bike-sharing patterns, adding features like year, day, hour, and weekday allowed the model to capture these trends more effectively.

I'm confident that the improvement came from the new features because I didn't change anything else in the training process, like hyperparameters. The only change was adding more meaningful information, which helped the model make better predictions.

Hyper parameter tuning

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

After hyperparameter tuning, the RMSE was -33, whereas before it was -32.

Even though I increased the hyperparameter training time from 10 minutes to 30 minutes, the surprising part was that the Kaggle score improved (from 47 to 44). This suggests that, despite the higher RMSE, the model performed better on unseen test data, indicating that the hyperparameter tuning positively impacted its ability to make better predictions in the competition setting.

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

If I had more time with this dataset, I would focus on adding as many new features as possible. I already observed a significant improvement in the model's performance after adding additional features, which suggests that further feature engineering could lead to even better results. By extracting more meaningful patterns from the data, the model might capture deeper relationships and improve its predictive accuracy.

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

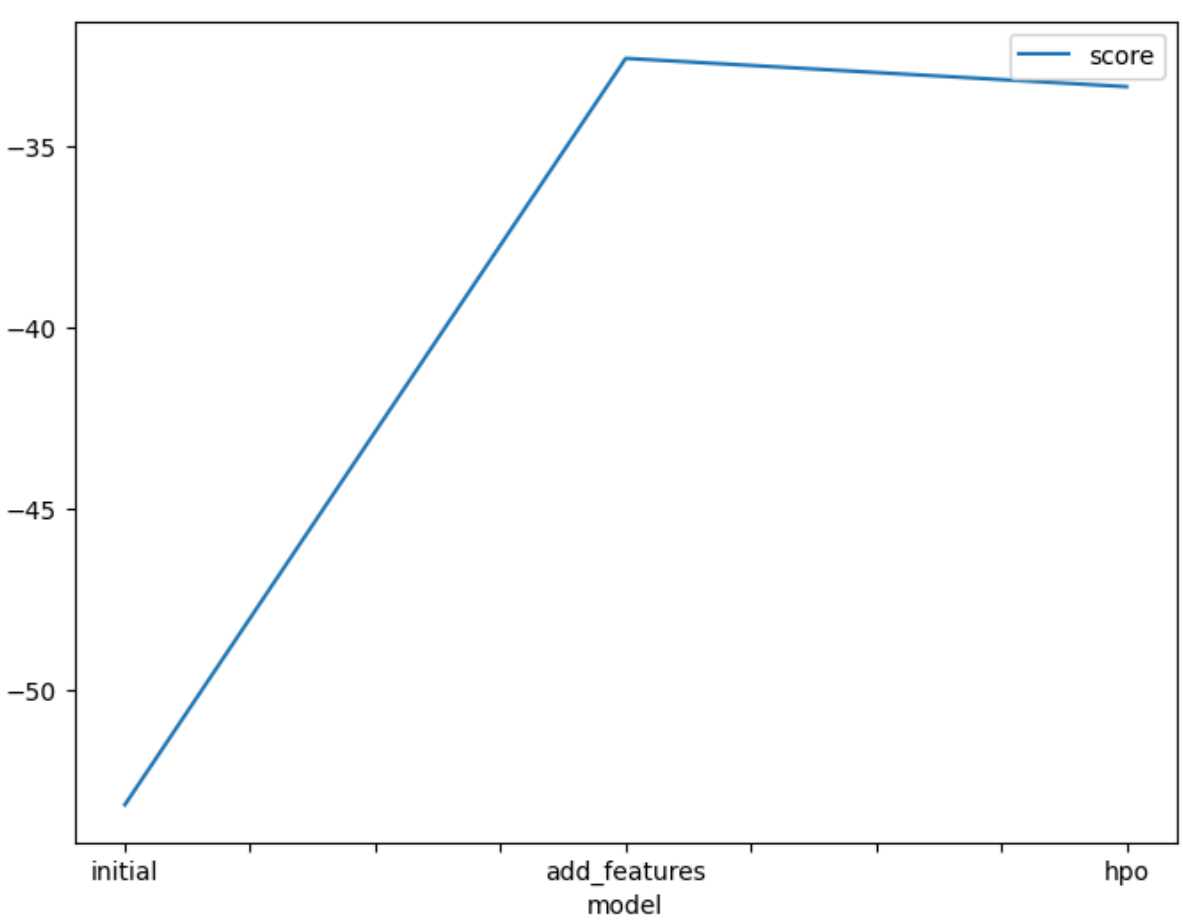
```
import pandas as pd
pd.DataFrame({
    "model": ["initial", "add_features", "hpo"],
```

```
    "hpo1": ["AutoGluon Default", "AutoGluon Default", 1000],
    "hpo2": ["AutoGluon Default", "AutoGluon Default", 1000],
    "hpo3": ["AutoGluon Default", "AutoGluon Default", 50],
    "score": [1.80310, 0.47137, 0.44574]
})
```

	model	hpo1	hpo2	hpo3	score
0	initial	AutoGluon Default	AutoGluon Default	AutoGluon Default	1.80310
1	add_features	AutoGluon Default	AutoGluon Default	AutoGluon Default	0.47137
2	hpo	1000	1000	50	0.44574

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

```
fig = pd.DataFrame(
    {
        "model": ["initial", "add_features", "hpo"],
        "score": [-53.171528, -32.598851, -33.379680]
    }
).plot(x="model", y="score", figsize=(8, 6)).get_figure()
fig.savefig('model_train_score.png')
```



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

```
fig = pd.DataFrame(
    {
        "test_eval": ["initial", "add_features", "hpo"],
        "score": [1.80310, 0.47137, 0.44574]
    }
).plot(x="test_eval", y="score", figsize=(8, 6)).get_figure()
fig.savefig('model_test_score.png')
```