# Predict Bike Sharing Demand with AutoGluon

REVIEW

CODE REVIEW

HISTORY

## Meets Specifications

Congratulations on completing the **Predict Bike Sharing Demand with AutoGluon** project. It was a nice experience reviewing your project. You have demonstrated good understanding in using the tool AutoGluon. Good luck with the projects ahead.

Here are some good articles on AutoGluon:

- Getting Started with AutoML and AWS AutoGluon
  https://www.philschmid.de/getting-started-with-automl-and-aws-autogluon
- Amazon's AutoGluon helps developers deploy deep learning models with just a few lines of code
  https://www.amazon.science/amazons-autogluon-helps-developers-get-up-and-running-with-state-of-the-art-deep-learning-models-with-just-a-few-lines-of-code
- AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data
  https://arxiv.org/abs/2003.06505
- AutoGluon: A revolutionary framework for landslide hazard analysis
  https://www.sciencedirect.com/science/article/pii/S2666592121000305

## Loading the Dataset

✓

**Student uses the kaggle cli with the kaggle API token to download and unzip the Bike Sharing Demand dataset into Sagemaker Studio (or local development).**

### Correct Implementation

- The dataset is successfully downloaded from Kaggle.

### Additional resources

Kaggle is a fantastic platform for beginners to learn and improve their machine learning skills

- Kaggle Competitions: The Complete Guide
  https://www.datacamp.com/blog/kaggle-competitions-the-complete-guide
- Using Kaggle in Machine Learning Projects
  https://machinelearningmastery.com/using-kaggle-in-machine-learning-projects/

✓

**Student uses Panda's `read_csv()` function to load the train/test/and sample submission file into DataFrames. Once loaded, they can view the dataframe in their jupyter notebook.**

### Correct Implementation

- Nice job loading the data using Pandas

### Additional resources

- Here's a cheatsheet for easy reference: https://www.analyticsvidhya.com/blog/2015/07/11-steps-perform-data-analysis-pandas-python/

## Feature Creation and Data Analysis

✓

**Student uses data from one feature column and extract data from it to use in a new feature column.**

### Correct Implementation

- The `hour` and other component of the datetime fields are appropriately extracted.

## Suggestions:

- Extract more features. For example, **day of the week**, Is it peak hour? etc.
- **Time of Day Segmentation**: Divide the day into segments, such as morning, afternoon, evening, and night, to capture different time-of-day patterns.
- **Day/Night Indicator:** Create a binary feature to indicate whether it's daytime or nighttime based on the hour of the day.
- **Cyclic Encodings:** For periodic features like hour of the day, use cyclic encodings (e.g., sine and cosine transformations) to capture the cyclical nature of time features without introducing discontinuities.

## Additional resources

- Three Approaches to Encoding Time Information as Features for ML Models https://developer.nvidia.com/blog/three-approaches-to-encoding-time-information-as-features-for-ml-models/
- Cyclical features encoding, it's about time! https://towardsdatascience.com/cyclical-features-encoding-its-about-time-ce23581845ca

✓

**Student creates a matplotlib image showing histograms of each feature column in the train dataframe.**

## Correct Implementation

- Nice work adding the histograms.

Histograms helps you understand the range, central tendency, and spread of the data.

## Suggestions:

- You can make it better by increasing the size of the figure by passing in an appropriate figsize to the hist() function and increasing the number of bins.
  Example:

```
train.hist(bins=30, figsize=(16, 10))
```

## Additional resources

- Python Histogram Plotting: NumPy, Matplotlib, pandas & Seaborn https://realpython.com/python-histograms/

✓

**Student assigns category data types to feature columns that are typed as numeric values.**

### Correct Implementation

- The categorical columns are selected appropriately.

  It ensures that the 'season' and 'weather' columns are of the 'category' data type, which can be beneficial for memory optimization and consistency between the training and testing datasets.

## Model Training With AutoGluon

✓

**Student uses the TabularPredictor class from AutoGluon to create a predictor by calling .fit().**

- Nice job calling the `.fit()` of the TabularPredictor
- Rather than selecting the columns, please use the `ignored_columns` parameter while fitting the data:

Example:

```
TabularPredictor(
learner_kwargs={"ignored_columns": ["casual", "registered"]}
)
```

Check out this documentation: https://auto.gluon.ai/dev/api/autogluon.tabular.TabularPredictor.html

✓

**Student provides additional arguments in the TabularPredictor .fit() function to change how the model uses hyperparameters for training.**

- Nice job with the use of `hyperparameter` and `hyperparameter_tune_kwarg` arguments.

Here you can refer the documentation for more details:
https://auto.gluon.ai/dev/api/autogluon.tabular.TabularPredictor.fit.html#autogluon.tabular.TabularPredictor.fit

✓

**Student uses the predictor created by fitting a model with TabularPredictor to predict new values from the test dataset.**

- Predictions are successfully obtained for test set from the trained model.

## Compare Model Performance

✓

Student uses the kaggle cli to submit their predictions from the trained AutoGluon Tabular Predictor to Kaggle for a public score submission.

- Kaggle scores are successfully retrieved using the CLI.

✓

Student uses matplotlib or google sheets/excel to chart model performance metrics in a line chart. The appropriate metric will be derived from the either `fit_summary()` or `leaderboard()` of the predictor. Y axis is the metric number and X axis is each model iteration.

- The model performance metrics are correctly plotted.
- Rather than hard-coding scores, please use `fit_summary()` or `leaderboard()` of the predictor for getting the metrics.

Example:

```
"score": [
            predictor.leaderboard(silent=True)['score_val'][0],
            predictor_new_features.leaderboard(silent=True)['score_val'][0],
            predictor_new_hpo.leaderboard(silent=True)['score_val'][0]
        ]
```

✓

Student uses matplotlib or google sheets/excel to chart changes to the competition score. Y axis is the kaggle score and X axis is each model iteration.

- The training and kaggle submission scores are appropriately visualized.

## Competition Report

✓

The submitted report makes use of `fit_summary()` or `leaderboard()` to detail the results of the training run and shows that the first entry will be the “best” model.

- Weighted Ensemble performed better compared to other models. Here's a good article on model averaging.

How to Develop a Weighted Average Ensemble for Deep Learning Neural Networks
https://machinelearningmastery.com/weighted-average-ensemble-for-deep-learning-neural-networks/

The submitted report discusses how adding additional features and changing hyperparameters led to a direct improvement in the kaggle score.

- Clear explanation of how the additional features improved model score. The **hour** of the day has the most predictive power in our dataset.

The submitted report contains a table outlining each hyperparameter uses along with the kaggle score received from each iteration.

The report contains an explanation of why certain changes to a hyperparameter affected the outcome of their score.

- The score from the different submissions are presented in tabular form.
  You can create a table using markdown. Here's a tutorial:
  How to Create Tables in Markdown
  https://itsfoss.com/markdown-table/

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START