UDACITY

‹ Return to Classroom

Predict Bike Sharing Demand with AutoGluon

REVIEW

CODE REVIEW

HISTORY

Requires Changes

1 specification requires changes

Great job on this submission 👏👏

I want to commend you on submitting this project that showcases your hard work, dedication, and the skills you have developed. It is evident that you have put in a significant amount of effort and time into completing this project, and the results are impressive. Your commitment to learning and growing as a AWS ML Engineer is evident throughout the project, and I appreciate your dedication to delivering a high-quality solution.

However, there were a few areas where some of the required items were not fully met.

I have provided you feedback in greater detail with the attached references so that you can easily move forward. Have a look at each carefully.

Useful References:

- Logging Wisdom: How to Log [Article]
- How to write a good readme for your github project? [Article]
- Pandas get_dummies() [Documentation]
- Sklearn OneHotEncoder [Documentation]

I encourage you to reflect on the areas where improvements could be made and consider them as valuable feedback for your future projects. Use this opportunity to further develop your skills, expand your understanding, and continue to strive for excellence.

Remember that setbacks and challenges are a natural part of the learning process. It's through perseverance and a commitment to improvement that we can overcome obstacles and achieve even greater success in the future. 💪💪

I'm wishing you success. Awaiting your upcoming contribution.
You can also ask questions on the Knowledge Portal to receive instant assistance from the mentors.

Enjoy your day! 🏆🏆

DON'T FORGET TO RATE MY WORK AS PROJECT REVIEWER! YOUR FEEDBACK IS VERY HELPFUL AND APPRECIATED.

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).

❌ You failed using the kaggle cli with the kaggle API token to download

❌ You have correctly unzipped the Bike Sharing Demand dataset in your project notebook

The code cell containing the command to download the dataset via kaggle API is missing.

We need to look at the logs to confirm the successful download of the dataset. It should look something similar to this:

```
bike-sharing-demand.zip: Skipping, found more recently modified local copy (use --force to force download)
Archive:  bike-sharing-demand.zip
  inflating: sampleSubmission.csv
  inflating: test.csv
  inflating: train.csv
```

If you encounter any specific error messages or issues while executing the command, please reach out to the mentors at knowledge portal to get assistance.

References:

Kaggle API Documentation. [Documentation]

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.

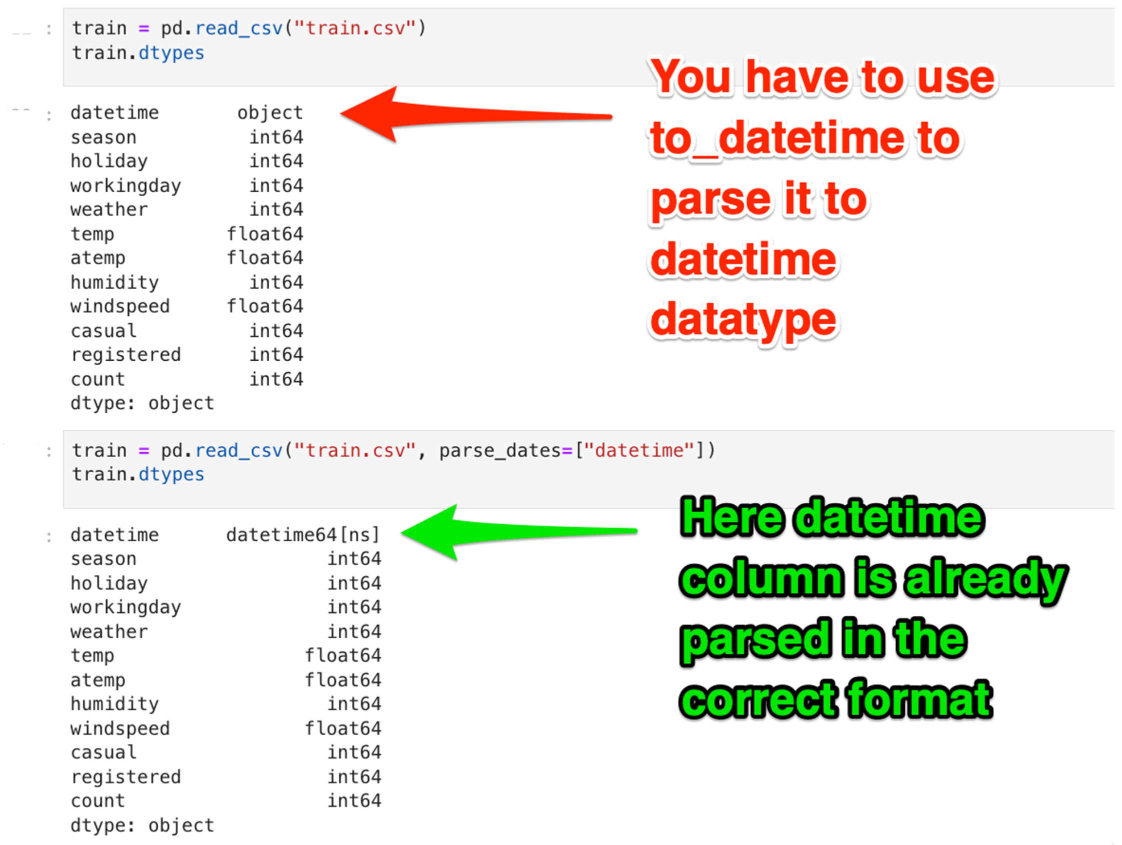
✅ You have successfully performed the read operation to load train, test, and sample submission files to respective dataframes.

Comments:

- Perfect! you have loaded all the dataframes correctly without any issues. Your ability to efficiently read and load data is a crucial skill in data analysis and machine learning.

Suggestions:

- You can use the read_csv 🔗parse_dates parameter to parse a datetime column while loading the data. Using the `parse_dates` parameter in pandas simplifies the process of working with time-related data.
- It saves you from manual conversion tasks and unlocks the full range of time-series analysis capabilities available in pandas and other compatible libraries.



```
train = pd.read_csv("train.csv")
train.dtypes
```

```
datetime       object
season          int64
holiday         int64
workingday      int64
weather         int64
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

You have to use to_datetime to parse it to datetime datatype

```
train = pd.read_csv("train.csv", parse_dates=["datetime"])
train.dtypes
```

```
datetime      datetime64[ns]
season                 int64
holiday                int64
workingday             int64
weather                int64
temp                 float64
atemp                float64
humidity               int64
windspeed            float64
casual                 int64
registered             int64
count                  int64
dtype: object
```

Here datetime column is already parsed in the correct format

References:

Pandas Documentation [Documentation]

Feature Creation and Data Analysis

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

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

Awesome! You have effectively used `datetime` feature to derive features like:

- hour

Deriving features from datetime variables provides several benefits in data analysis and machine learning tasks:

- **Seasonality and trends**: Extracting features like month, day, or hour allows you to capture the seasonal patterns or daily/hourly trends in your data. This can help identify recurring patterns, understand cyclic behavior, and detect any time-related variations that might impact your analysis or modeling.
- **Temporal aggregations**: By deriving features at different time granularities (e.g., day, month, year),

you can perform temporal aggregations and summarize the data at different levels. This enables you to analyze trends over time, identify long-term patterns, or compare data across different time periods.
- **Feature engineering**: Derived datetime features can serve as valuable inputs for machine learning models. They provide additional information and context that might be relevant for predicting the target variable.

References:

pandas Documentation - DatetimeProperties **[Documentation]**

Suggestion:

- You can also check 🔗dt.dayofweek to derive one more timestamp-based feature.
- You can also check out the 🔗tsfresh package. It automatically calculates several time series characteristics, the so-called features. Further, the package contains methods to evaluate the power and importance of such attributes for regression or classification tasks.

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

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

Awesome 👍

You have showcased the necessary distributions of the numerical features present is the data. This provides an overview of the data's distribution, allowing us to understand its central tendency, spread, and shape. They give us a sense of the data's characteristics and help identify any patterns, trends, or anomalies present.

Suggestions:

- To increase the figure size when plotting a histogram using `df.hist()`, you can incorporate the figsize parameter to specify the desired width and height of the figure. Here's an example of how to do it:

```
import pandas as pd
import matplotlib.pyplot as plt

# Increase the figure size
plt.figure(figsize=(10, 6))

# Plot the histogram
```

```
df.hist(column='column_name')

# Display the plot
plt.show()
```

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

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

Good work!

You have nicely dealt with category-based features. Using the categorical data type can provide several benefits, including:

- **Memory Efficiency**: Categorical data types can significantly reduce memory usage compared to regular object or string data types. Categorical data is internally stored as integers, where each unique category is assigned a unique integer code. This integer representation consumes less memory than storing the actual string values. If you have a column with a limited number of unique values or a column that represents categories, converting it to the categorical data type can lead to substantial memory savings, especially for large datasets.
- **Improved Performance**: Working with categorical data can improve the performance of certain operations. Categorical data allows for efficient operations like grouping, aggregating, and sorting. These operations can be faster when working with categorical data compared to object or string data types.
- **Ordered Categoricals**: Categorical data types can have an inherent order defined among the categories. This ordered nature can be useful for variables with a natural ranking or hierarchy. For example, a categorical variable representing education levels (e.g., "High School," "Bachelor's Degree," "Master's Degree") can be ordered in a meaningful way. You can specify the order of categories using the `ordered=True` parameter when creating the categorical data type.

References:

🔗****Using pandas categories properly is tricky, here’s why...** [Article]**

Model Training With AutoGluon

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

✅ You have instantiated the TabularPredictor class from AutoGluon to create a predictor.

Fantastic 💯

You have correctly instantiated the TabularPredictor object with all the necessary parameters.

Here are certain things you should keep in mind while fitting a TabularPredictor:

- **Consider Time and Resource Constraints**: When instantiating **`TabularPredictor`**, consider specifying time and resource constraints based on your available computing resources and time limitations. This allows you to manage the model training process efficiently and prevents excessive resource usage or lengthy training times.
- **Experiment with Different Presets**: AutoGluon provides various presets that encapsulate different configurations and settings. Experiment with different presets to explore their impact on model performance. Some popular presets include "medium_quality_faster_inference", "high_quality", or "best_quality". Test multiple presets to find the right balance between model performance and computational cost for your specific use case.

✅ Evaluation metric used is `root_mean_squared_error`

✅ Time limit set to “600” seconds.

✅ Presets set to “best_quality”

✅ `casual` and `registered` columns are ignored while fitting the model

Good catch! You have ignored the “ `["casual", "registered"]` ”.

- **Casual Users**: Casual users refer to individuals who rent bikes on a short-term or temporary basis, typically for recreational or leisure purposes. These users may include tourists, occasional riders, or individuals who do not have a long-term subscription or membership with the bike sharing service.
- **Registered Users**: Registered users, on the other hand, are individuals who have subscribed or registered with the bike sharing service. These users often have long-term memberships or subscriptions and use the bike sharing service regularly for commuting or other purposes.

We are dropping these columns so that the model does not discriminate between `casual` and `registered` users since both users are important for the bike-sharing application. They both bring significant traffic.

Suggestions:

- Try to use the `learner_kwargs` parameter to ignore the columns:

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

- You can also set the `problem_type` attribute to `regression` since the target variable is numeric variable.

Reference:

🔗TabularPredictor Documentation

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

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

Outstanding 🙌

You have incorporated a lot of parameters in your hyperparameter tuning.

AutoGluon's hyperparameter tuning functionality helps in automating the process of finding optimal hyperparameters, saving our time and effort compared to manual tuning. It explores the hyperparameter space efficiently and provides us with the best hyperparameters for our model, allowing us to achieve better performance and generalization on our dataset.

Note: It's worth noting that hyperparameter tuning can be computationally expensive, especially when searching a large space or using complex models. Therefore, it's important to allocate sufficient computational resources and set appropriate time limits or budget constraints to ensure efficient tuning without excessive resource consumption.

Suggestions:

- There are many hyperparameter options. You can always reference them in this Autogluon 🔗documentation
- Here is one detailed tutorial on Autogluon Hyperarameter tuning: **Hyperparameter Tuning is Overrated Apply AutoGluon-Tabular, an AutoML Algorithm [Video Lecture]**

Some additional tips (Only for future reference):

- Do not specify the hyperparameter_tune_kwargs argument (counterintuitively, hyperparameter tuning is not the best way to spend a limited training time budgets, as model ensembling is often superior)
- We recommend you only use hyperparameter_tune_kwargs if your goal is to deploy a single model rather than an ensemble.

- Do not specify hyperparameters argument (allow AutoGluon to adaptively select which models/hyperparameters to use).

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

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

Spot on! 🙌

- You have handled all the negative values which might occur during the prediction of the values on the test set. We are doing so to ensure the realistic constraints are intact.
- All the feature modification you did on the training dataset is also applied to the test dataset. Failure to do so can lead to mismatched feature columns and prevent us from making predictions on the test data.

Additional tips:

- When we call predict(), AutoGluon automatically predicts with the model that displayed the best performance on validation data
- We can instead specify which model to use for predictions like this:

```
predictor.predict(test_data, model='LightGBM')
```

- By setting `as_pandas=True`, the predicted output will be returned as a Pandas DataFrame. This can be useful if you want to easily manipulate, analyze, or further process the predictions using Pandas' functionality.

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.

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

Excellent 💯

I didn't find any issues in the logs returned by the kaggle APIs. The submission format was as per the template provided to you.

Additional Tips:

- **Verify File Format**: Ensure that you are submitting the correct file format specified by the competition. Double-check if the competition requires a specific file extension, such as CSV, ZIP, or JSON.
- **Submission File Content**: Confirm that your submission file contains the appropriate data in the correct structure and format. It should align with the submission requirements specified by the competition, such as the number of rows, columns, and the expected data format.
- **Submission Message**: Include a meaningful and descriptive submission message to provide additional information about your submission. This message can help you track and understand the purpose of each submission you make. For example, you can mention the approach or techniques used, or any specific details you want to highlight.

References:

Kaggle API Documentation. [Documentation]

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.

✅ You have successfully used the Matplotlib to chart model performance metrics in a line chart.

✅ Appropriate metric derived from the `**leaderboard**()` method.

✅ Y axis is the metric number and X axis is each model iteration.

Fantastic job on utilizing the `leaderboard()` method in AutoGluon to analyze and present the performance of your models!

Your use of this method demonstrates your ability to assess and compare the performance of different models in a clear and organized manner.

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.

✅ You have successfully used Matplotlib to chart changes to the competition score

✅ Y axis is the kaggle score and X axis is each model iteration

Congratulations on successfully plotting the Kaggle score vs. model chart!

Your efforts in retrieving the scores from the Kaggle competition submission API and visualizing them in a chart format demonstrate your excellent skills in data analysis and visualization.

For the Bike Sharing Demand competition, the evaluation metric used to calculate the Kaggle score is the `Root Mean Squared Logarithmic Error` (RMSLE). The RMSLE metric measures the difference between the logarithm of the predicted count and the logarithm of the actual count. A lower RMSLE score indicates better performance, with a perfect score of 0 representing an exact match between the predicted and actual counts.

$$\text{RMSLE} = \sqrt{(log(y_i + 1) - log(\hat{y}_i + 1))^2}$$

References:

- Hyperparameter tuning a custom model with TabularPredictor [Documentation]
- Hyperparameter Tuning is Overrated Apply AutoGluon-Tabular, an AutoML Algorithm [Video Lecture]

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.

✅ 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.

Additional Tips:

You can also use `[leaderboard](https://auto.gluon.ai/stable/api/autogluon.tabular.TabularPredictor.leaderboard.html)` method to pull the best model name. Here's an example code snippet that demonstrates how to pull the best model using the leaderboard in AutoGluon:

```
from autogluon.tabular import TabularPredictor

# Train multiple models using AutoGluon
predictor = TabularPredictor(...)
```

```
predictor.fit(...)

# Generate the leaderboard
leaderboard = predictor.leaderboard()

# Retrieve the best model
best_model_name = leaderboard.iloc[0]['model']
best_model = predictor.get_model(best_model_name)
```

In this example, `TabularPredictor` is initialized and trained using AutoGluon. The `leaderboard()` method is then called to generate the leaderboard. The first entry in the leaderboard represents the best model. You can retrieve the name of the best model using `leaderboard.iloc[0]['model']` and obtain the corresponding model object using `predictor.get_model(best_model_name)`.

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

✅ Report discusses how adding additional features led to a direct improvement in the kaggle score

Great job on your project report!

I was impressed to see how you discussed the impact of adding additional features on the Kaggle score and how it directly led to an improvement in performance. This demonstrates your strong analytical skills and understanding of feature engineering.

I encourage you to continue exploring different feature engineering techniques and experimenting with additional variables to further enhance your models' performance.

✅ Report how changing hyperparameters led to a direct improvement in the kaggle score

Awesome!

Your report effectively explains the rationale behind modifying the hyperparameters and how each change contributed to the improvement in the Kaggle score. This level of detail helps readers understand the impact of hyperparameter choices on model behavior and performance.

I encourage you to continue exploring different hyperparameter optimization techniques and consider more advanced methods like Bayesian optimization or genetic algorithms. By further refining your hyperparameter tuning process, you can potentially achieve even better results.

References:

- Hyperparameter tuning a custom model with TabularPredictor [Documentation]
- Hyperparameter Tuning is Overrated Apply AutoGluon-Tabular, an AutoML Algorithm [Video Lecture]

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 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.

Great Work 🙌

The inclusion of such a table is vital as it allows readers to easily visualize and analyze the relationship between hyperparameters and the corresponding Kaggle scores. It demonstrates your thoroughness in documenting and presenting the results of your hyperparameter tuning experiments.

References:

- Use this website to get the basic outline for the markdown table. [Website]
- Markdown Cheatsheet [Article]

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review

START