UDACITY

Return to Classroom

Predict Bike Sharing Demand with AutoGluon

검토

코드 검토

HISTORY

사양 충족

Dear Student,

Congratulations on passing the project in one shot!

We look forward to seeing more of your achievement in the next project.

Regards,
Your Reviewer

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).

```
# Download the dataset, it will be in a .zip file so you'll need to unzip it as well.
!kaggle competitions download -c bike-sharing-demand
# If you already downloaded it you can use the -o command to overwrite the file
!unzip -o bike-sharing-demand.zip
```

👍Great job! You successfully applied kaggle API to download the data.

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.

```python
# Create the train dataset in pandas by reading the csv
# Set the parsing of the datetime column so you can use some of the `dt` features in pandas later
train = pd.read_csv("./bike-sharing-demand/train.csv")
train.head()
```

Python

👍 Well done! Using `head()` to preview the data is very professional!

Pro Tips

You can also use pandas.DataFrame.sample to take a more random peek at the data. For example, instead of viewing the first 5 rows of the data, `train.sample(5)` gives you random 5 rows of the data. This is specifically useful when your data is not shuffled or cannot be shuffled.

Feature Creation and Data Analysis

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

```python
train["hour"] = train_datetime.dt.hour
test["hour"] = test_datetime.dt.hour
train["day"] = train_datetime.dt.day
test["day"] = test_datetime.dt.day
train["month"] = train_datetime.dt.month
test["month"] = test_datetime.dt.month
```

👍 You successfully split the DateTime information. Great job mirroring the operation to the testing data!

👂*Pro Tips:*

Feature engineering is a big topic. Of course, there are tons of blogs that discuss FE, and google can give you some good search results about it, while I tend to recommend exploring Kaggle for competitors' experience in FE here. It is just a simple search, but powerful!

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

👍 Your histogram code is nice and the output is nice.

👂 *Pro Tips*

You can make the visualization even better by changing it to : `train.hist(figsize=(20,20));`

- add `;` to the end of the visualization to get rid of the messy output text:

```
array([[<AxesSubplot:title={'center':'datetime'}>,
        <AxesSubplot:title={'center':'season'}>,
        <AxesSubplot:title={'center':'holiday'}>],
       [<AxesSubplot:title={'center':'workingday'}>,
        <AxesSubplot:title={'center':'weather'}>,
        <AxesSubplot:title={'center':'temp'}>],
       [<AxesSubplot:title={'center':'atemp'}>,
        <AxesSubplot:title={'center':'humidity'}>,
        <AxesSubplot:title={'center':'windspeed'}>],
       [<AxesSubplot:title={'center':'casual'}>,
        <AxesSubplot:title={'center':'registered'}>,
        <AxesSubplot:title={'center':'count'}>]], dtype=object)
```

- set the figure size by adding the `figsize` argument.

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

```
train["season"] = train.season.astype('category')
train["weather"] = train.weather.astype('category')
test["season"] = test.season.astype('category')
test["weather"] = test.weather.astype('category')
```

👍 Great job assigning the correct type to season and weather.

Model Training With AutoGluon

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

```
predictor = TabularPredictor(label="count", problem_type="regression", eval_metric="root_mean_squared_error").fit(
    train_data=train[["datetime","season","holiday","workingday","weather","temp","atemp","humidity","windspeed","count"]], time_limit=600,
)
```

👍 Very Nice! You use the correct evaluation metric for this project.

💡 *Pro Tips*

This article is a nice collection of popular metrics. You can use it as a reference when choosing metrics for your model.

Also, keep in mind the choice of metrics highly depend on the application of your model.

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

```
hyperparameters = {'GBM': {'num_boost_round': 15}, 'XGB': {'eta': 0.2} }

# HPO is not performed unless hyperparameter_tune_kwargs is specified
hyperparameter_tune_kwargs = {
    'num_trials': 3,
    'scheduler' : 'local',
    'searcher': 'auto',
}

predictor_new_hpo = TabularPredictor(label="count", problem_type="regression", eval_metric="root_mean_squared_error").fit(
    train_data=train[["season","holiday","workingday","weather","temp","atemp","humidity","windspeed","hour","day","month","count"]], time_
    hyperparameters=hyperparameters, hyperparameter_tune_kwargs=hyperparameter_tune_kwargs
)
```

👍 Amazing job carefully assigning hyperparameters to your candidate models.

👂 *Pro Tips*

Here we can see your effort on the `hyperparameters` argument. You can also try to play with some parameters under `**kwargs` in the documentation. The reason is that it is more efficient to try them compared to tuning the hyperparameters of each model types.

In the **Maximizing predictive performance** of this quick start guide for auto gluon. The author mentioned:

> counterintuitively, hyperparameter tuning is not the best way to spend a limited training time budgets, as model ensembling is often superior.

So Autogluon use ensembling to build powerful models. I recommend you read this blog to get more insight into it.

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

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 competitions submissions -c bike-sharing-demand | tail -n +1 | head -n 6
```

👍 Perfect! You can consider this as a great milestone to publish your result to kaggle.

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.

```
# Taking the top model score from each training run and creating a line plot to show improvement
# You can create these in the notebook and save them to PNG or use some other tool (e.g. google sheets, excel)
fig = pd.DataFrame(
    {
        "model": ["initial", "add_features", "hpo"],
        "score": [-50.591929, -49.820223, -52.089706]
    }
).plot(x="model", y="score", figsize=(8, 6)).get_figure()
fig.savefig('model_train_score.png')
```

🤔 *TODO (Optional)*

Amazing job using leaderboard information to record the result, while here you should avoid hard-coding the values. Recall that data in `leaderboard()` has a `pandas.DataFrame` structure.

HINTS

Here's an example of how you can fetch the score values of the top models (modify the names of the variables if needed):

```
leader_df = predictor.leaderboard()

# Get the top 3 rows of the DataFrame
top3 =  leader_df.head(3)

# Get the model names and score values for the top 3 rows
top3_models = top3['model']
top3_scores = top3['score_val']
```

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.

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.

👍 Correctly record the best model in the report. Well done!

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

프로젝트 다운로드

경로로 돌아가기

이 리뷰를 평가하세요

시작