Predict Bike Sharing Demand with AutoGluon Template

Project: Predict Bike Sharing Demand with AutoGluon

This notebook is a template with each step that you need to complete for the project.

Please fill in your code where there are explicit `?` markers in the notebook. You are welcome to add more cells and code as you see fit.

Once you have completed all the code implementations, please export your notebook as a HTML file so the reviews can view your code. Make sure you have all outputs correctly outputted.

`File-> Export Notebook As... -> Export Notebook as HTML`

There is a writeup to complete as well after all code implememtation is done. Please answer all questions and attach the necessary tables and charts. You can complete the writeup in either markdown or PDF.

Completing the code template and writeup template will cover all of the rubric points for this project.

The rubric contains "Stand Out Suggestions" for enhancing the project beyond the minimum requirements. The stand out suggestions are optional. If you decide to pursue the "stand out suggestions", you can include the code in this notebook and also discuss the results in the writeup file.

Step 1: Create an account with Kaggle

Create Kaggle Account and download API key

Below is example of steps to get the API username and key. Each student will have their own username and key.

1. Open account settings.
2. Scroll down to API and click Create New API Token.
3. Open up `kaggle.json` and use the username and key.

Step 2: Download the Kaggle dataset using the kaggle python library

Open up Sagemaker Studio and use starter template

1. Notebook should be using a `ml.t3.medium` instance (2 vCPU + 4 GiB)
2. Notebook should be using kernal: `Python 3 (MXNet 1.8 Python 3.7 CPU Optimized)`

Install packages

```
!pip install -U pip
!pip install -U setuptools wheel
!pip install -U "mxnet<2.0.0" bokeh==2.0.1
!pip install autogluon --no-cache-dir
# Without --no-cache-dir, smaller aws instances may have trouble installing
```

```
    Found existing installation: grpcio 1.47.0
    Uninstalling grpcio-1.47.0:
      Successfully uninstalled grpcio-1.47.0
  Attempting uninstall: cloudpickle
    Found existing installation: cloudpickle 1.3.0
    Uninstalling cloudpickle-1.3.0:
      Successfully uninstalled cloudpickle-1.3.0
  Attempting uninstall: xgboost
    Found existing installation: xgboost 0.90
    Uninstalling xgboost-0.90:
      Successfully uninstalled xgboost-0.90
  Attempting uninstall: numba
    Found existing installation: numba 0.51.2
    Uninstalling numba-0.51.2:
      Successfully uninstalled numba-0.51.2
  Attempting uninstall: markdown
    Found existing installation: Markdown 3.3.7
    Uninstalling Markdown-3.3.7:
      Successfully uninstalled Markdown-3.3.7
  Attempting uninstall: hyperopt
    Found existing installation: hyperopt 0.1.2
    Uninstalling hyperopt-0.1.2:
      Successfully uninstalled hyperopt-0.1.2
  Attempting uninstall: torchvision
    Found existing installation: torchvision 0.13.0+cu113
    Uninstalling torchvision-0.13.0+cu113:
      Successfully uninstalled torchvision-0.13.0+cu113
  Attempting uninstall: statsmodels
    Found existing installation: statsmodels 0.10.2
    Uninstalling statsmodels-0.10.2:
      Successfully uninstalled statsmodels-0.10.2
  Attempting uninstall: scikit-image
    Found existing installation: scikit-image 0.18.3
    Uninstalling scikit-image-0.18.3:
      Successfully uninstalled scikit-image-0.18.3
  Attempting uninstall: lightgbm
```

```
    Found existing installation: lightgbm 2.2.3
    Uninstalling lightgbm-2.2.3:
      Successfully uninstalled lightgbm-2.2.3
  Attempting uninstall: dask
    Found existing installation: dask 2.12.0
    Uninstalling dask-2.12.0:
      Successfully uninstalled dask-2.12.0
  Attempting uninstall: distributed
    Found existing installation: distributed 1.25.3
    Uninstalling distributed-1.25.3:
      Successfully uninstalled distributed-1.25.3
  Attempting uninstall: fastai
    Found existing installation: fastai 2.7.6
    Uninstalling fastai-2.7.6:
      Successfully uninstalled fastai-2.7.6
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the follo
torchtext 0.13.0 requires torch==1.12.0, but you have torch 1.11.0 which is incompatible.
torchaudio 0.12.0+cu113 requires torch==1.12.0, but you have torch 1.11.0 which is incompatible.
panel 0.12.1 requires bokeh<2.4.0,>=2.3.0, but you have bokeh 2.0.1 which is incompatible.
gym 0.17.3 requires cloudpickle<1.7.0,>=1.2.0, but you have cloudpickle 2.1.0 which is incompatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is incompatible.
albumentations 0.1.12 requires imgaug<0.2.7,>=0.2.5, but you have imgaug 0.2.9 which is incompatible.
```

Setup Kaggle API Key

```
# create the .kaggle directory and an empty kaggle.json file
!mkdir -p /root/.kaggle
!touch /root/.kaggle/kaggle.json
!chmod 600 /root/.kaggle/kaggle.json
```

```
# Fill in your user name and key from creating the kaggle account and API token file
import json
kaggle_username = "FILL_IN_USERNAME"
kaggle_key = "FILL_IN_KEY"

# Save API token the kaggle.json file
with open("/root/.kaggle/kaggle.json", "w") as f:
    f.write(json.dumps({"username": kaggle_username, "key": kaggle_key}))
```

Download and explore dataset

Go to the bike sharing demand competition and agree to the terms

```
# 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
```

```
401 - Unauthorized
```

```
import pandas as pd
from autogluon.tabular import TabularPredictor
```

```
# 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("/content/train.csv")
train.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

```
# Simple output of the train dataset to view some of the min/max/varition of the dataset features.
train.describe()
```

```
# Create the test pandas dataframe in pandas by reading the csv, remember to parse the datetime!
test = pd.read_csv("/content/test.csv")
test.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014

```
# Same thing as train and test dataset
submission = pd.read_csv("/content/sampleSubmission.csv")
submission.head()
```

	datetime	count
0	2011-01-20 00:00:00	0
1	2011-01-20 01:00:00	0
2	2011-01-20 02:00:00	0
3	2011-01-20 03:00:00	0
4	2011-01-20 04:00:00	0

Step 3: Train a model using AutoGluon's Tabular Prediction

Requirements:

- We are prediting `count`, so it is the label we are setting.
- Ignore `casual` and `registered` columns as they are also not present in the test dataset.
- Use the `root_mean_squared_error` as the metric to use for evaluation.
- Set a time limit of 10 minutes (600 seconds).
- Use the preset `best_quality` to focus on creating the best model.

```
train.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

```
train.drop(columns = ["casual", "registered"],inplace = True)
```

```
train.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	1

```
predictor = TabularPredictor(label = 'count',eval_metric = "root_mean_squared_error").fit(train_data = train ,  time_limit
```

```
        183.45s   = Training   runtime
        0.12s     = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 ... Training model for up to 100.41s of the 300.43s of remaining time.
        -124.5881        = Validation score   (-root_mean_squared_error)
        4.52s     = Training   runtime
        0.5s      = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 ... Training model for up to 94.8s of the 294.82s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -135.9151        = Validation score   (-root_mean_squared_error)
        75.45s    = Training   runtime
        0.32s     = Validation runtime
Fitting model: XGBoost_BAG_L1 ... Training model for up to 15.95s of the 215.97s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -131.6247        = Validation score   (-root_mean_squared_error)
        21.8s     = Training   runtime
        2.27s     = Validation runtime
Completed 1/20 k-fold bagging repeats ...
Fitting model: WeightedEnsemble_L2 ... Training model for up to 360.0s of the 191.13s of remaining time.
        -84.1251         = Validation score   (-root_mean_squared_error)
        0.52s     = Training   runtime
        0.0s      = Validation runtime
Fitting 9 L2 models ...
Fitting model: LightGBMXT_BAG_L2 ... Training model for up to 190.59s of the 190.57s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -60.2698         = Validation score   (-root_mean_squared_error)
        45.76s    = Training   runtime
        2.9s      = Validation runtime
Fitting model: LightGBM_BAG_L2 ... Training model for up to 141.53s of the 141.52s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -55.2437         = Validation score   (-root_mean_squared_error)
        20.25s    = Training   runtime
        0.25s     = Validation runtime
Fitting model: RandomForestMSE_BAG_L2 ... Training model for up to 118.65s of the 118.63s of remaining time.
        -53.4364         = Validation score   (-root_mean_squared_error)
        28.4s     = Training   runtime
        0.6s      = Validation runtime
Fitting model: CatBoost_BAG_L2 ... Training model for up to 88.99s of the 88.98s of remaining time.
```

```
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -55.7381         = Validation score   (-root_mean_squared_error)
        58.36s   = Training   runtime
        0.06s    = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L2 ... Training model for up to 28.28s of the 28.27s of remaining time.
        -53.7567         = Validation score   (-root_mean_squared_error)
        7.84s    = Training   runtime
        0.59s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 ... Training model for up to 19.2s of the 19.19s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -56.7198         = Validation score   (-root_mean_squared_error)
        30.27s   = Training   runtime
        0.39s    = Validation runtime
Completed 1/20 k-fold bagging repeats ...
Fitting model: WeightedEnsemble_L3 ... Training model for up to 360.0s of the -13.54s of remaining time.
        -52.5887         = Validation score   (-root_mean_squared_error)
        0.46s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 614.05s ... Best model: "WeightedEnsemble_L3"
TabularPredictor saved. To load, use: predictor = TabularPredictor.load("AutogluonModels/ag-20220718_183126/")
```

Review AutoGluon's training run with ranking of models that did the best.

```
predictor.fit_summary()
```

```
*** Summary of fit() ***
Estimated performance of each model:
                     model   score_val  pred_time_val    fit_time  pred_time_val_marginal  fit_time_marginal  stack_level  can_infer  fit_order
0      WeightedEnsemble_L3  -52.588686      14.592571  471.715399                0.000719           0.460664            3       True         17
1   RandomForestMSE_BAG_L2  -53.436379      13.368493  412.901057                0.598281          28.396258            2       True         13
2     ExtraTreesMSE_BAG_L2  -53.756692      13.360186  392.340383                0.589974           7.835584            2       True         15
3          LightGBM_BAG_L2  -55.243713      13.017183  404.756034                0.246971          20.251235            2       True         12
4          CatBoost_BAG_L2  -55.738051      12.834581  442.866735                0.064369          58.361936            2       True         14
5   NeuralNetFastAI_BAG_L2  -56.719786      13.156626  414.771658                0.386415          30.266859            2       True         16
6        LightGBMXT_BAG_L2  -60.269798      15.670026  430.269125                2.899814          45.764326            2       True         11
7    KNeighborsDist_BAG_L1  -84.125061       0.103625    0.039399                0.103625           0.039399            1       True          2
8      WeightedEnsemble_L2  -84.125061       0.104821    0.563049                0.001196           0.523650            2       True         10
9    KNeighborsUnif_BAG_L1 -101.546199       0.102778    0.029768                0.102778           0.029768            1       True          1
10  RandomForestMSE_BAG_L1 -116.544294       0.517232   10.038474                0.517232          10.038474            1       True          5
11    ExtraTreesMSE_BAG_L1 -124.588053       0.501942    4.524314                0.501942           4.524314            1       True          7
12         CatBoost_BAG_L1 -130.472284       0.121524  183.454202                0.121524         183.454202            1       True          6
13         LightGBM_BAG_L1 -131.054162       1.135580   24.027062                1.135580          24.027062            1       True          4
14       LightGBMXT_BAG_L1 -131.460909       7.694350   65.132978                7.694350          65.132978            1       True          3
15          XGBoost_BAG_L1 -131.624665       2.271722   21.804194                2.271722          21.804194            1       True          9
16  NeuralNetFastAI_BAG_L1 -135.915121       0.321460   75.454408                0.321460          75.454408            1       True          8
Number of models trained: 17
Types of models trained:
{'StackerEnsembleModel_XGBoost', 'WeightedEnsembleModel', 'StackerEnsembleModel_CatBoost', 'StackerEnsembleModel_LGB', 'StackerEnsembleModel_XT', 'S
Bagging used: True  (with 8 folds)
```

```
Multi-layer stack-ensembling used: True  (with 3 levels)
Feature Metadata (Processed):
(raw dtype, special dtypes):
('float', [])                : 3 | ['temp', 'atemp', 'windspeed']
('int', [])                  : 3 | ['season', 'weather', 'humidity']
('int', ['bool'])            : 2 | ['holiday', 'workingday']
('int', ['datetime_as_int']) : 5 | ['datetime', 'datetime.year', 'datetime.month', 'datetime.day', 'datetime.dayofweek']
Plot summary of models saved to file: AutogluonModels/ag-20220718_183126/SummaryOfModels.html
*** End of fit() summary ***
{'leaderboard':                      model   score_val  pred_time_val    fit_time  \
 0      WeightedEnsemble_L3  -52.588686      14.592571  471.715399
 1   RandomForestMSE_BAG_L2  -53.436379      13.368493  412.901057
 2     ExtraTreesMSE_BAG_L2  -53.756692      13.360186  392.340383
 3          LightGBM_BAG_L2  -55.243713      13.017183  404.756034
 4          CatBoost_BAG_L2  -55.738051      12.834581  442.866735
 5   NeuralNetFastAI_BAG_L2  -56.719786      13.156626  414.771658
 6        LightGBMXT_BAG_L2  -60.269798      15.670026  430.269125
 7    KNeighborsDist_BAG_L1  -84.125061       0.103625    0.039399
 8      WeightedEnsemble_L2  -84.125061       0.104821    0.563049
 9    KNeighborsUnif_BAG_L1 -101.546199       0.102778    0.029768
 10  RandomForestMSE_BAG_L1 -116.544294       0.517232   10.038474
 11    ExtraTreesMSE_BAG_L1 -124.588053       0.501942    4.524314
 12         CatBoost_BAG_L1 -130.472284       0.121524  183.454202
 13         LightGBM_BAG_L1 -131.054162       1.135580   24.027062
 14       LightGBMXT_BAG_L1 -131.460909       7.694350   65.132978
 15          XGBoost_BAG_L1 -131.624665       2.271722   21.804194
 16  NeuralNetFastAI_BAG_L1 -135.915121       0.321460   75.454408

     pred_time_val_marginal  fit_time_marginal  stack_level  can_infer  \
 0                 0.000719           0.460664            3       True
 1                 0.598281          28.396258            2       True
 2                 0.589974           7.835584            2       True
```

Create predictions from test dataset

```
predictions = predictor.predict(test)
predictions.head()
```

```
0    23.550512
1    40.808868
2    44.458626
3    47.286266
4    50.420288
Name: count, dtype: float32
```

NOTE: Kaggle will reject the submission if we don't set everything to be > 0.

```
# Describe the `predictions` series to see if there are any negative values
predictions.describe()
```

```
count    6493.000000
mean      101.748779
std        89.129524
min         2.031049
25%        23.832951
50%        67.130913
75%       170.800186
max       359.129395
Name: count, dtype: float64
```

```
# How many negative values do we have?
print(len(predictions))
(predictions[predictions < 0] )
```

```
6493
Series([], Name: count, dtype: float32)
```

```
# Set them to zero
predictions[predictions < 0 ] = 0
predictions.head()
```

```
0    23.550512
1    40.808868
2    44.458626
3    47.286266
4    50.420288
Name: count, dtype: float32
```

Set predictions to submission dataframe, save, and submit

```
submission["count"] = predictions
submission.to_csv("submission.csv", index=False)
```

```
#!kaggle competitions submit -c bike-sharing-demand -f submission.csv -m "first raw submission"
```

View submission via the command line or in the web browser under the competition's page - `My Submissions`

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

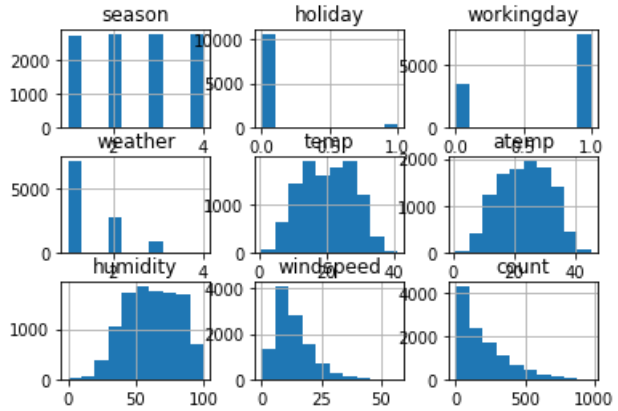
Initial score of `?`

Step 4: Exploratory Data Analysis and Creating an additional feature

- Any additional feature will do, but a great suggestion would be to separate out the datetime into hour, day, or month parts.

```
# Create a histogram of all features to show the distribution of each one relative to the data. This is part of the exploritory data analysis
train.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fe9993f7810>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe9995051d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe999357b10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe997a8cc10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe99955fd90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe997a993d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe997aeca50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe997ae9b10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe9979ae050>]],
      dtype=object)
```




```
train["datetime"] = pd.to_datetime(train["datetime"])
test["datetime"] = pd.to_datetime(test["datetime"])
```

```
# create a new feature
train['second'] =  train.datetime.dt.second
train['minute'] =  train.datetime.dt.minute
train['hour'] =  train.datetime.dt.hour
train['day'] =  train.datetime.dt.day
train['month'] =  train.datetime.dt.month

train['year'] =  train.datetime.dt.year
```

```
#test[?] = ?
train.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	count	second	minute	hour	day	month	year
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	16	0	0	0	1	1	2011
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	40	0	0	1	1	1	2011
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	32	0	0	2	1	1	2011
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	13	0	0	3	1	1	2011
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	1	0	0	4	1	1	2011

```
test['second'] =  test.datetime.dt.second
test['minute'] =  test.datetime.dt.minute
test['hour'] =  test.datetime.dt.hour
test['day'] =  test.datetime.dt.day
test['month'] =  test.datetime.dt.month
test['year'] =  test.datetime.dt.year
```

```
test.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	second	minute	hour	day	month	year
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027	0	0	0	20	1	2011
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000	0	0	1	20	1	2011
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000	0	0	2	20	1	2011
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014	0	0	3	20	1	2011
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014	0	0	4	20	1	2011

Make category types for these so models know they are not just numbers

- AutoGluon originally sees these as ints, but in reality they are int representations of a category.
- Setting the dtype to category will classify these as categories in AutoGluon.

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

```
# View are new feature
train.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	count	second	minute	hour	day	month	year
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	16	0	0	0	1	1	2011

```
# View histogram of all features again now with the hour feature
train.hist()
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fe999573290>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe9972a3690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe997258ad0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe997284b10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe9971d2510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe997207810>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe9971bfd10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe997182190>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe9971821d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe9971387d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe9970b1110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe997069610>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe99701fb10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe996fd5fd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe996f98550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fe996f4fa50>]],
      dtype=object)
```

Step 5: Rerun the model with the same settings as before, just with more features

```
predictor_new_features = predictor_new_hpo = TabularPredictor(label = "count" ,eval_metric = "root_mean_squared_error" ).fit(time_limit = 600 , presets = "b
```

```
        2.53s     = Validation runtime
Fitting model: RandomForestMSE_BAG_L1 ... Training model for up to 278.34s of the 478.31s of remaining time.
        -38.442   = Validation score   (-root_mean_squared_error)
        13.25s    = Training   runtime
        0.58s     = Validation runtime
```

```
Fitting model: CatBoost_BAG_L1 ... Training model for up to 263.86s of the 463.83s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -33.698  = Validation score   (-root_mean_squared_error)
        219.78s  = Training   runtime
        0.27s    = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 ... Training model for up to 41.61s of the 241.57s of remaining time.
        -38.3437         = Validation score   (-root_mean_squared_error)
        6.03s    = Training   runtime
        0.56s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 ... Training model for up to 34.35s of the 234.32s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -65.0623         = Validation score   (-root_mean_squared_error)
        40.99s   = Training   runtime
        0.37s    = Validation runtime
Completed 1/20 k-fold bagging repeats ...
Fitting model: WeightedEnsemble_L2 ... Training model for up to 360.0s of the 190.81s of remaining time.
        -32.0862         = Validation score   (-root_mean_squared_error)
        0.62s    = Training   runtime
        0.0s     = Validation runtime
Fitting 9 L2 models ...
Fitting model: LightGBMXT_BAG_L2 ... Training model for up to 190.16s of the 190.15s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -31.2247         = Validation score   (-root_mean_squared_error)
        26.19s   = Training   runtime
        1.04s    = Validation runtime
Fitting model: LightGBM_BAG_L2 ... Training model for up to 161.08s of the 161.07s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -30.7226         = Validation score   (-root_mean_squared_error)
        20.66s   = Training   runtime
        0.25s    = Validation runtime
Fitting model: RandomForestMSE_BAG_L2 ... Training model for up to 137.8s of the 137.79s of remaining time.
        -31.6092         = Validation score   (-root_mean_squared_error)
        31.4s    = Training   runtime
        0.62s    = Validation runtime
Fitting model: CatBoost_BAG_L2 ... Training model for up to 105.15s of the 105.13s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -30.4095         = Validation score   (-root_mean_squared_error)
        88.33s   = Training   runtime
        0.14s    = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L2 ... Training model for up to 14.43s of the 14.41s of remaining time.
        -31.4373         = Validation score   (-root_mean_squared_error)
        9.04s    = Training   runtime
        0.63s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L2 ... Training model for up to 4.11s of the 4.09s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        Time limit exceeded... Skipping NeuralNetFastAI_BAG_L2.
Completed 1/20 k-fold bagging repeats ...
Fitting model: WeightedEnsemble_L3 ... Training model for up to 360.0s of the -9.43s of remaining time.
        -30.2148         = Validation score   (-root_mean_squared_error)
        0.31s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 609.77s ... Best model: "WeightedEnsemble_L3"
TabularPredictor saved. To load, use: predictor = TabularPredictor.load("AutogluonModels/ag-20220718_184209/")
```

```
predictor_new_features.fit_summary()
```

```
*** Summary of fit() ***
Estimated performance of each model:
                     model   score_val  pred_time_val    fit_time  pred_time_val_marginal  fit_time_marginal  stack_level  can_infer  fit_order
0      WeightedEnsemble_L3  -30.214835      13.966882  560.471848                0.000734           0.307906            3       True         15
1          CatBoost_BAG_L2  -30.409463      12.059606  481.910322                0.144569          88.333001            2       True         13
2          LightGBM_BAG_L2  -30.722643      12.165332  414.238742                0.250295          20.661421            2       True         11
3        LightGBMXT_BAG_L2  -31.224679      12.952870  419.768789                1.037833          26.191467            2       True         10
4     ExtraTreesMSE_BAG_L2  -31.437290      12.543442  402.613799                0.628405           9.036478            2       True         14
5   RandomForestMSE_BAG_L2  -31.609205      12.533451  424.978053                0.618414          31.400732            2       True         12
6      WeightedEnsemble_L2  -32.086164      10.884509  347.128956                0.000919           0.621614            2       True          9
7          CatBoost_BAG_L1  -33.697982       0.267037  219.782176                0.267037         219.782176            1       True          6
8          LightGBM_BAG_L1  -33.917338       2.526168   38.163175                2.526168          38.163175            1       True          4
9        LightGBMXT_BAG_L1  -34.305322       7.404306   75.284508                7.404306          75.284508            1       True          3
10    ExtraTreesMSE_BAG_L1  -38.343705       0.557975    6.029046                0.557975           6.029046            1       True          7
11  RandomForestMSE_BAG_L1  -38.442020       0.582637   13.246457                0.582637          13.246457            1       True          5
12  NeuralNetFastAI_BAG_L1  -65.062265       0.369486   40.991025                0.369486          40.991025            1       True          8
13   KNeighborsDist_BAG_L1  -84.125061       0.103442    0.031026                0.103442           0.031026            1       True          2
14   KNeighborsUnif_BAG_L1 -101.546199       0.103987    0.049909                0.103987           0.049909            1       True          1
Number of models trained: 15
Types of models trained:
{'WeightedEnsembleModel', 'StackerEnsembleModel_CatBoost', 'StackerEnsembleModel_LGB', 'StackerEnsembleModel_XT', 'StackerEnsembleModel_KNN', 'Stack
Bagging used: True  (with 8 folds)
Multi-layer stack-ensembling used: True  (with 3 levels)
Feature Metadata (Processed):
(raw dtype, special dtypes):
('category', [])             : 2 | ['season', 'weather']
('float', [])                : 3 | ['temp', 'atemp', 'windspeed']
('int', [])                  : 4 | ['humidity', 'hour', 'day', 'month']
('int', ['bool'])            : 3 | ['holiday', 'workingday', 'year']
('int', ['datetime_as_int']) : 5 | ['datetime', 'datetime.year', 'datetime.month', 'datetime.day', 'datetime.dayofweek']
Plot summary of models saved to file: AutogluonModels/ag-20220718_184209/SummaryOfModels.html
*** End of fit() summary ***
{'leaderboard':                      model   score_val  pred_time_val    fit_time  \
 0      WeightedEnsemble_L3  -30.214835      13.966882  560.471848
 1          CatBoost_BAG_L2  -30.409463      12.059606  481.910322
 2          LightGBM_BAG_L2  -30.722643      12.165332  414.238742
 3        LightGBMXT_BAG_L2  -31.224679      12.952870  419.768789
 4     ExtraTreesMSE_BAG_L2  -31.437290      12.543442  402.613799
 5   RandomForestMSE_BAG_L2  -31.609205      12.533451  424.978053
 6      WeightedEnsemble_L2  -32.086164      10.884509  347.128956
 7          CatBoost_BAG_L1  -33.697982       0.267037  219.782176
 8          LightGBM_BAG_L1  -33.917338       2.526168   38.163175
 9        LightGBMXT_BAG_L1  -34.305322       7.404306   75.284508
 10    ExtraTreesMSE_BAG_L1  -38.343705       0.557975    6.029046
 11  RandomForestMSE_BAG_L1  -38.442020       0.582637   13.246457
 12  NeuralNetFastAI_BAG_L1  -65.062265       0.369486   40.991025
 13   KNeighborsDist_BAG_L1  -84.125061       0.103442    0.031026
 14   KNeighborsUnif_BAG_L1 -101.546199       0.103987    0.049909
```

```
   pred_time_val_marginal  fit_time_marginal  stack_level  can_infer  \
0                0.000734           0.307906            3       True
1                0.144569          88.333001            2       True
2                0.250295          20.661421            2       True
3                1.037833          26.191467            2       True
4                0.628405           9.036478            2       True
5                0.618414          31.400732            2       True
6                0.000919           0.621614            2       True
```

```
predictor2 =  predictor_new_features.predict(test)
```

```
predictor2.head()
```

```
0    16.686827
1    11.861774
2    11.025127
3     9.817257
4     8.336437
Name: count, dtype: float32
```

```
# Remember to set all negative values to zero
predictor2[predictor2<0]
```

```
Series([], Name: count, dtype: float32)
```

```
predictor2[predictor2<0] = 0
```

```
predictor2.head()
```

```
0    16.686827
1    11.861774
2    11.025127
3     9.817257
4     8.336437
Name: count, dtype: float32
```

```
sumbission_new_features =  pd.read_csv("/content/sampleSubmission.csv")
```

```
sumbission_new_features.head()
```

	datetime	count
0	2011-01-20 00:00:00	0
1	2011-01-20 01:00:00	0
3	2011-01-20 03:00:00	0

```
sumbission_new_features["datetime"] = pd.to_datetime(sumbission_new_features["datetime"])
```

```
# Same submitting predictions
sumbission_new_features["count"] = predictor2
sumbission_new_features.to_csv("submission_new_features.csv", index=False)
```

```
#!kaggle competitions submit -c bike-sharing-demand -f submission_new_features.csv -m "new features"
```

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

New Score of `?`

Step 6: Hyper parameter optimization

- There are many options for hyper parameter optimization.
- Options are to change the AutoGluon higher level parameters or the individual model hyperparameters.
- The hyperparameters of the models themselves that are in AutoGluon. Those need the `hyperparameter` and `hyperparameter_tune_kwargs` arguments.

```
import autogluon.core as ag

num_trials  = 10
search_strategy = 'auto'
nn_options = {
    'num_epochs': 50,
    'learning_rate': ag.space.Real(1e-4, 1e-2, default=5e-4, log=True),
    'activation': ag.space.Categorical('relu', 'softrelu', 'tanh'),
    'dropout_prob': ag.space.Real(0.0, 0.5, default=0.1),
}

gbm_options = {
    'num_boost_round': 100,
    'num_leaves': ag.space.Int(lower=26, upper=66, default=36),
}

hyperparameters = {
```

```
                    'GBM': gbm_options,
                    'NN_TORCH': nn_options,
}
hyperparameter_tune_kwargs = {
    'num_trials': num_trials,
    'scheduler' : 'local',
    'searcher': search_strategy,
}
```

```
predictor_new_hpo = TabularPredictor(label = "count" ,eval_metric = "root_mean_squared_error" ).fit(time_limit = 600 , presets = "best_quality" ,  hyperpara
```

```
No path specified. Models will be saved in: "AutogluonModels/ag-20220718_185251/"
Presets specified: ['best_quality']
Warning: hyperparameter tuning is currently experimental and may cause the process to hang.
Beginning AutoGluon training ... Time limit = 600s
AutoGluon will save models to "AutogluonModels/ag-20220718_185251/"
AutoGluon Version:  0.5.0
Python Version:     3.7.13
Operating System:   Linux
Train Data Rows:    10886
Train Data Columns: 15
Label Column: count
Preprocessing data ...
AutoGluon infers your prediction problem is: 'regression' (because dtype of label-column == int and many unique label-values observed).
        Label info (max, min, mean, stddev): (977, 1, 191.57413, 181.14445)
        If 'regression' is not the correct problem_type, please manually specify the problem_type parameter during predictor init (You may specify pro
Using Feature Generators to preprocess the data ...
Fitting AutoMLPipelineFeatureGenerator...
        Available Memory:                    11929.25 MB
        Train Data (Original)  Memory Usage: 1.15 MB (0.0% of available memory)
        Inferring data type of each feature based on column values. Set feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator...
                        Note: Converting 3 features to boolean dtype as they only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator...
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator...
                Fitting CategoryFeatureGenerator...
                        Fitting CategoryMemoryMinimizeFeatureGenerator...
                Fitting DatetimeFeatureGenerator...
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator...
        Useless Original Features (Count: 2): ['second', 'minute']
                These features carry no predictive signal and should be manually investigated.
                This is typically a feature which has the same value for all rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('category', []) : 2 | ['season', 'weather']
                ('datetime', []) : 1 | ['datetime']
                ('float', [])    : 3 | ['temp', 'atemp', 'windspeed']
                ('int', [])      : 7 | ['holiday', 'workingday', 'humidity', 'hour', 'day', ...]
        Types of features in processed data (raw dtype, special dtypes):
                ('category', [])             : 2 | ['season', 'weather']
                ('float', [])                : 3 | ['temp', 'atemp', 'windspeed']
                ('int', [])                  : 4 | ['humidity', 'hour', 'day', 'month']
                ('int', ['bool'])            : 3 | ['holiday', 'workingday', 'year']
                ('int', ['datetime_as_int']) : 5 | ['datetime', 'datetime.year', 'datetime.month', 'datetime.day', 'datetime.dayofweek']
        0.2s = Fit runtime
        13 features in original data used to generate 17 features in processed data.
        Train Data (Processed) Memory Usage: 1.1 MB (0.0% of available memory)
Data preprocessing and feature engineering runtime = 0.22s ...
AutoGluon will gauge predictive performance using evaluation metric: 'root_mean_squared_error'
        This metric's sign has been flipped to adhere to being higher_is_better. The metric score can be multiplied by -1 to get the metric value.
```

```
        To change this, specify the eval_metric parameter of Predictor()
AutoGluon will fit 2 stack levels (L1 to L2) ...
Fitting 2 L1 models ...
Hyperparameter tuning model: LightGBM_BAG_L1 ... Tuning model for up to 22.49s of the 599.77s of remaining time.
```

100% 10/10 [00:03<00:00, 2.88it/s]

```
Fitted model: LightGBM_BAG_L1/T1 ...
        -39.5016         = Validation score   (-root_mean_squared_error)
        0.34s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L1/T2 ...
        -37.3144         = Validation score   (-root_mean_squared_error)
        0.3s     = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L1/T3 ...
        -36.7779         = Validation score   (-root_mean_squared_error)
        0.37s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L1/T4 ...
        -120.7115        = Validation score   (-root_mean_squared_error)
        0.31s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L1/T5 ...
        -41.8284         = Validation score   (-root_mean_squared_error)
        0.34s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L1/T6 ...
        -109.1997        = Validation score   (-root_mean_squared_error)
        0.36s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L1/T7 ...
        -36.744  = Validation score   (-root_mean_squared_error)
        0.29s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L1/T8 ...
        -35.2159         = Validation score   (-root_mean_squared_error)
        0.35s    = Training   runtime
        0.02s    = Validation runtime
Fitted model: LightGBM_BAG_L1/T9 ...
        -107.6886        = Validation score   (-root_mean_squared_error)
        0.29s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L1/T10 ...
        -35.2249         = Validation score   (-root_mean_squared_error)
        0.3s     = Training   runtime
        0.01s    = Validation runtime
Hyperparameter tuning model: NeuralNetTorch_BAG_L1 ... Tuning model for up to 22.49s of the 595.72s of remaining time.
```

0% 0/10 [00:15<?, ?it/s]

```
        Stopping HPO to satisfy time limit...
Fitted model: NeuralNetTorch_BAG_L1/T1 ...
        -43.7766         = Validation score   (-root_mean_squared_error)
        15.72s   = Training   runtime
        0.02s    = Validation runtime
```

```
        0.02s    = Validation runtime
Fitting model: LightGBM_BAG_L1/T1 ... Training model for up to 379.79s of the 579.81s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -40.2554         = Validation score   (-root_mean_squared_error)
        13.59s   = Training   runtime
        0.13s    = Validation runtime
Fitting model: LightGBM_BAG_L1/T2 ... Training model for up to 360.92s of the 560.94s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -39.0553         = Validation score   (-root_mean_squared_error)
        13.36s   = Training   runtime
        0.13s    = Validation runtime
Fitting model: LightGBM_BAG_L1/T3 ... Training model for up to 345.41s of the 545.43s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -38.5238         = Validation score   (-root_mean_squared_error)
        13.98s   = Training   runtime
        0.16s    = Validation runtime
Fitting model: LightGBM_BAG_L1/T4 ... Training model for up to 329.39s of the 529.41s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -121.6873        = Validation score   (-root_mean_squared_error)
        13.5s    = Training   runtime
        0.1s     = Validation runtime
Fitting model: LightGBM_BAG_L1/T5 ... Training model for up to 313.65s of the 513.68s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -43.3082         = Validation score   (-root_mean_squared_error)
        14.52s   = Training   runtime
        0.13s    = Validation runtime
Fitting model: LightGBM_BAG_L1/T6 ... Training model for up to 296.18s of the 496.2s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -109.5652        = Validation score   (-root_mean_squared_error)
        13.89s   = Training   runtime
        0.11s    = Validation runtime
Fitting model: LightGBM_BAG_L1/T7 ... Training model for up to 280.13s of the 480.15s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -38.3912         = Validation score   (-root_mean_squared_error)
        13.31s   = Training   runtime
        0.12s    = Validation runtime
Fitting model: LightGBM_BAG_L1/T8 ... Training model for up to 264.56s of the 464.59s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -36.344  = Validation score   (-root_mean_squared_error)
        13.85s   = Training   runtime
        0.19s    = Validation runtime
Fitting model: LightGBM_BAG_L1/T9 ... Training model for up to 248.47s of the 448.49s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -108.68  = Validation score   (-root_mean_squared_error)
        13.58s   = Training   runtime
        0.1s     = Validation runtime
Fitting model: LightGBM_BAG_L1/T10 ... Training model for up to 232.69s of the 432.72s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -36.2414         = Validation score   (-root_mean_squared_error)
        13.82s   = Training   runtime
        0.17s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1/T1 ... Training model for up to 216.72s of the 416.74s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -43.9628         = Validation score   (-root_mean_squared_error)
```

```
        123.56s  = Training   runtime
        0.33s    = Validation runtime
Completed 1/20 k-fold bagging repeats ...
Fitting model: WeightedEnsemble_L2 ... Training model for up to 360.0s of the 306.36s of remaining time.
        -35.1035         = Validation score   (-root_mean_squared_error)
        0.38s    = Training   runtime
        0.0s     = Validation runtime
Fitting 2 L2 models ...
Hyperparameter tuning model: LightGBM_BAG_L2 ... Tuning model for up to 17.21s of the 305.94s of remaining time.
```

100% 10/10 [00:05<00:00, 1.96it/s]

```
Fitted model: LightGBM_BAG_L2/T1 ...
        -38.1252         = Validation score   (-root_mean_squared_error)
        0.64s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L2/T2 ...
        -38.0837         = Validation score   (-root_mean_squared_error)
        0.43s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L2/T3 ...
        -38.0777         = Validation score   (-root_mean_squared_error)
        0.58s    = Training   runtime
        0.03s    = Validation runtime
Fitted model: LightGBM_BAG_L2/T4 ...
        -101.9393        = Validation score   (-root_mean_squared_error)
        0.45s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L2/T5 ...
        -38.368  = Validation score   (-root_mean_squared_error)
        0.52s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L2/T6 ...
        -98.9992         = Validation score   (-root_mean_squared_error)
        0.56s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L2/T7 ...
        -38.0485         = Validation score   (-root_mean_squared_error)
        0.41s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L2/T8 ...
        -38.1322         = Validation score   (-root_mean_squared_error)
        0.56s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L2/T9 ...
        -89.2872         = Validation score   (-root_mean_squared_error)
        0.46s    = Training   runtime
        0.01s    = Validation runtime
Fitted model: LightGBM_BAG_L2/T10 ...
        -37.6909         = Validation score   (-root_mean_squared_error)
        0.44s    = Training   runtime
        0.01s    = Validation runtime
Hyperparameter tuning model: NeuralNetTorch_BAG_L2 ... Tuning model for up to 17.21s of the 300.16s of remaining time.
```

0% 0/10 [00:13<?, ?it/s]

```
        Ran out of time, stopping training early. (Stopping on epoch 41)
        Stopping HPO to satisfy time limit...
Fitted model: NeuralNetTorch_BAG_L2/T1 ...
        -38.1669         = Validation score   (-root_mean_squared_error)
        13.58s   = Training   runtime
        0.04s    = Validation runtime
Fitting model: LightGBM_BAG_L2/T1 ... Training model for up to 286.32s of the 286.3s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -34.4221         = Validation score   (-root_mean_squared_error)
        14.79s   = Training   runtime
        0.13s    = Validation runtime
Fitting model: LightGBM_BAG_L2/T2 ... Training model for up to 269.79s of the 269.77s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -34.2775         = Validation score   (-root_mean_squared_error)
        15.72s   = Training   runtime
        0.1s     = Validation runtime
Fitting model: LightGBM_BAG_L2/T3 ... Training model for up to 251.93s of the 251.91s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -34.4842         = Validation score   (-root_mean_squared_error)
        15.73s   = Training   runtime
        0.16s    = Validation runtime
Fitting model: LightGBM_BAG_L2/T4 ... Training model for up to 234.26s of the 234.24s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -102.2607        = Validation score   (-root_mean_squared_error)
        14.2s    = Training   runtime
        0.1s     = Validation runtime
Fitting model: LightGBM_BAG_L2/T5 ... Training model for up to 218.1s of the 218.08s of remaining time.
        Fitting 7 child models (S1F2 - S1F8) | Fitting with ParallelLocalFoldFittingStrategy
        -34.8599         = Validation score   (-root_mean_squared_error)
```

```
predictor_new_hpo.fit_summary()
```

```
*** Summary of fit() ***
Estimated performance of each model:
                      model   score_val  pred_time_val    fit_time  pred_time_val_marginal  fit_time_marginal  stack_level  can_infer  fit_order
0        WeightedEnsemble_L3  -33.873361       2.642558  426.503501                0.000809           0.365879            3       True         24
1         LightGBM_BAG_L2/T2  -34.277470       1.765036  276.691106                0.095971          15.718854            2       True         14
2         LightGBM_BAG_L2/T7  -34.280354       1.765253  274.778807                0.096188          13.806556            2       True         19
3         LightGBM_BAG_L2/T1  -34.422115       1.794327  275.764373                0.125263          14.792121            2       True         13
4        LightGBM_BAG_L2/T10  -34.458408       1.760924  275.387263                0.091860          14.415011            2       True         22
5         LightGBM_BAG_L2/T3  -34.484224       1.833919  276.698411                0.164854          15.726159            2       True         15
6         LightGBM_BAG_L2/T8  -34.485643       1.777448  276.355587                0.108383          15.383335            2       True         20
7         LightGBM_BAG_L2/T5  -34.859941       1.785930  276.029047                0.116866          15.056795            2       True         17
8   NeuralNetTorch_BAG_L2/T1  -34.946875       2.192876  366.471043                0.523812         105.498791            2       True         23
9        WeightedEnsemble_L2  -35.103487       0.691589  151.607491                0.000755           0.381549            2       True         12
10       LightGBM_BAG_L1/T10  -36.241351       0.169744   13.818541                0.169744          13.818541            1       True         10
11        LightGBM_BAG_L1/T8  -36.343987       0.187953   13.849602                0.187953          13.849602            1       True          8
12        LightGBM_BAG_L1/T7  -38.391192       0.120165   13.313435                0.120165          13.313435            1       True          7
13        LightGBM_BAG_L1/T3  -38.523833       0.164400   13.983031                0.164400          13.983031            1       True          3
14        LightGBM_BAG_L1/T2  -39.055347       0.126059   13.364925                0.126059          13.364925            1       True          2
15        LightGBM_BAG_L1/T1  -40.255449       0.129999   13.586885                0.129999          13.586885            1       True          1
16        LightGBM_BAG_L1/T5  -43.308206       0.129294   14.521622                0.129294          14.521622            1       True          5
```

```
17  NeuralNetTorch_BAG_L1/T1  -43.962834       0.333137  123.557800                0.333137         123.557800            1       True         11
18        LightGBM_BAG_L2/T9  -89.242439       1.780491  277.104753                0.111427          16.132501            2       True         21
19        LightGBM_BAG_L2/T6  -99.161061       1.780571  276.264057                0.111507          15.291805            2       True         18
20        LightGBM_BAG_L2/T4 -102.260746       1.764274  275.169422                0.095210          14.197170            2       True         16
21        LightGBM_BAG_L1/T9 -108.680001       0.100933   13.578286                0.100933          13.578286            1       True          9
22        LightGBM_BAG_L1/T6 -109.565161       0.107608   13.893130                0.107608          13.893130            1       True          6
23        LightGBM_BAG_L1/T4 -121.687263       0.099772   13.504996                0.099772          13.504996            1       True          4
Number of models trained: 24
Types of models trained:
{'WeightedEnsembleModel', 'StackerEnsembleModel_LGB', 'StackerEnsembleModel_TabularNeuralNetTorch'}
Bagging used: True  (with 8 folds)
Multi-layer stack-ensembling used: True  (with 3 levels)
Feature Metadata (Processed):
(raw dtype, special dtypes):
('category', [])             : 2 | ['season', 'weather']
('float', [])                : 3 | ['temp', 'atemp', 'windspeed']
('int', [])                  : 4 | ['humidity', 'hour', 'day', 'month']
('int', ['bool'])            : 3 | ['holiday', 'workingday', 'year']
('int', ['datetime_as_int']) : 5 | ['datetime', 'datetime.year', 'datetime.month', 'datetime.day', 'datetime.dayofweek']
Plot summary of models saved to file: AutogluonModels/ag-20220718_185251/SummaryOfModels.html
*** End of fit() summary ***
{'leaderboard':                        model   score_val  pred_time_val    fit_time  \
 0        WeightedEnsemble_L3  -33.873361       2.642558  426.503501
 1         LightGBM_BAG_L2/T2  -34.277470       1.765036  276.691106
 2         LightGBM_BAG_L2/T7  -34.280354       1.765253  274.778807
 3         LightGBM_BAG_L2/T1  -34.422115       1.794327  275.764373
 4        LightGBM_BAG_L2/T10  -34.458408       1.760924  275.387263
 5         LightGBM_BAG_L2/T3  -34.484224       1.833919  276.698411
 6         LightGBM_BAG_L2/T8  -34.485643       1.777448  276.355587
 7         LightGBM_BAG_L2/T5  -34.859941       1.785930  276.029047
 8   NeuralNetTorch_BAG_L2/T1  -34.946875       2.192876  366.471043
 9        WeightedEnsemble_L2  -35.103487       0.691589  151.607491
 10       LightGBM_BAG_L1/T10  -36.241351       0.169744   13.818541
 11        LightGBM_BAG_L1/T8  -36.343987       0.187953   13.849602
 12        LightGBM_BAG_L1/T7  -38.391192       0.120165   13.313435
 13        LightGBM_BAG_L1/T3  -38.523833       0.164400   13.983031
 14        LightGBM_BAG_L1/T2  -39.055347       0.126059   13.364925
 15        LightGBM_BAG_L1/T1  -40.255449       0.129999   13.586885
```

```
predictor3 = predictor_new_hpo.predict(test)
```

```
predictor3.head()
```

```
0    11.809039
1     5.627992
2     5.170829
3     5.247394
4     5.147136
Name: count, dtype: float32
```

```
# Remember to set all negative values to zero
predictor3[predictor3 < 0 ]
```

```
Series([], Name: count, dtype: float32)
```

```
predictor3[predictor3<0]  = 0
```

```
submission_new_hpo =  pd.read_csv("/content/sampleSubmission.csv")
```

```
submission_new_hpo.head()
```

	datetime	count
0	2011-01-20 00:00:00	0
1	2011-01-20 01:00:00	0
2	2011-01-20 02:00:00	0
3	2011-01-20 03:00:00	0
4	2011-01-20 04:00:00	0

```
submission_new_hpo["datetime"] =  pd.to_datetime(submission_new_hpo["datetime"])
```

```
# Same submitting predictions
submission_new_hpo["count"] = predictor3
submission_new_hpo.to_csv("submission_new_hpo.csv", index=False)
```

```
submission_new_hpo.tail()
```

	datetime	count
6488	2012-12-31 19:00:00	279.128235
6489	2012-12-31 20:00:00	206.825699
6490	2012-12-31 21:00:00	148.648682
6491	2012-12-31 22:00:00	98.907944
6492	2012-12-31 23:00:00	63.056171

```
#!kaggle competitions submit -c bike-sharing-demand -f submission_new_hpo.csv -m "new features with hyperparameters"
```

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

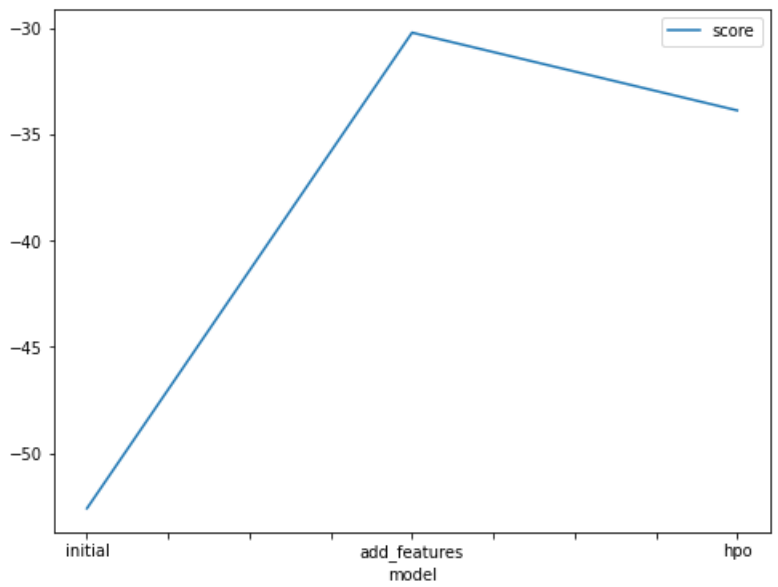
New Score of ``

Step 7: Write a Report

Refer to the markdown file for the full report

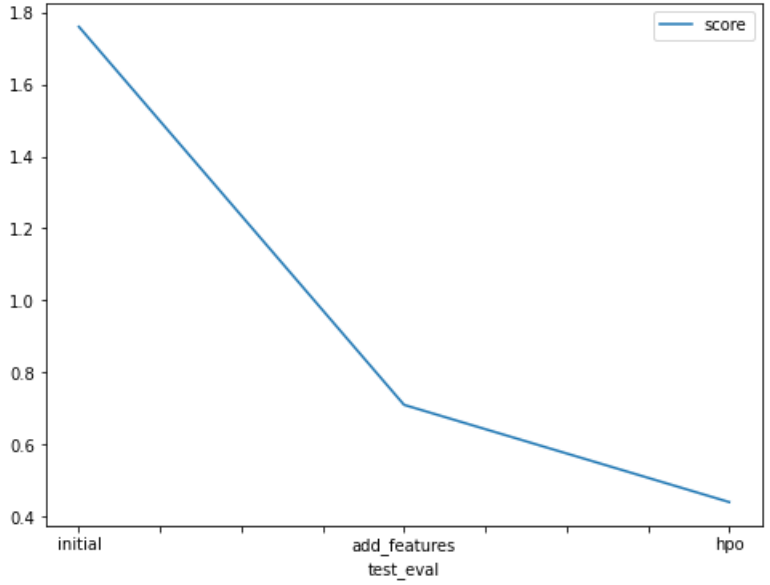
Creating plots and table for report

```
# 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": [-52.588686, -30.214835, -33.873361]
    }
).plot(x="model", y="score", figsize=(8, 6)).get_figure()
fig.savefig('model_train_score.png')
```




```
# Take the 3 kaggle scores and creating a line plot to show improvement
```

```
fig = pd.DataFrame(
    {
        "test_eval": ["initial", "add_features", "hpo"],
        "score": [1.76, 0.71, 0.44]
    }
).plot(x="test_eval", y="score", figsize=(8, 6)).get_figure()
fig.savefig('model_test_score.png')
```



Hyperparameter table

```
# The 3 hyperparameters we tuned with the kaggle score as the result
pd.DataFrame({
    "model": ["initial", "add_features", "hpo"],
    "names": ["first", "second", "third"],
    "time": ["time = 600", "time = 600", "time = 600"],
    "presets": ["best_quality", "best_quality", "best_quality"],
    "score": [1.76, 0.71, 0.44]
})
```

	model	names	time	presets	score
0	initial	first	time = 600	best_quality	1.76
1	add_features	second	time = 600	best_quality	0.71
2	hpo	third	time = 600	best_quality	0.44

```
!jupyter nbconvert --to html /content/bike_sharing.ipynb
```

```
[NbConvertApp] Converting notebook /content/bike_sharing.ipynb to html
[NbConvertApp] Writing 659335 bytes to /content/bike_sharing.html
```

✓ 2s completed at 9:14 PM