model_evaluation

December 17, 2025

[1]:
```
import pandas as pd
import numpy as np
import warnings
from final_project.modeling import load_models, EVENT_WEIGHT
from final_project.data import read_data, split_data
from final_project.preprocessing import NUM_FEATURES, CAT_FEATURES, RESPONDER
from final_project.evaluation import evaluate_predictions, get_pred_summary,␣
 ↪get_models_and_val_data
from final_project.plotting import plot_pred_vs_true, plot_day_predictions,␣
 ↪plot_feature_relevance, plot_pdps

# Silence feature name warnings
warnings.filterwarnings(
    "ignore",
    message="X does not have valid feature names",
)
```

```
/opt/homebrew/Caskroom/miniconda/base/envs/final-project/lib/python3.10/site-
packages/dalex/_global_checks.py:1: UserWarning: pkg_resources is deprecated as
an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html. The
pkg_resources package is slated for removal as early as 2025-11-30. Refrain from
using this package or pin to Setuptools<81.
  import pkg_resources
```

0.1 Load Models

[2]:
```
glm_raw, lgbm_raw, X_val_raw, y_val_raw = get_models_and_val_data("clean_data")
glm_clip, lgbm_clip, X_val_clip, y_val_clip =␣
 ↪get_models_and_val_data("clean_data_clipped")
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000821 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 4702
[LightGBM] [Info] Number of data points in the train set: 21452, number of used
features: 74
[LightGBM] [Info] Start training from score -0.789211
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
```

```
testing was 0.000656 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 4702
[LightGBM] [Info] Number of data points in the train set: 21452, number of used
features: 74
[LightGBM] [Info] Start training from score -0.789211
```

[3]:
```
df_pred_raw = get_pred_summary(glm_raw, lgbm_raw, X_val_raw, y_val_raw)
df_pred_clip = get_pred_summary(glm_clip, lgbm_clip, X_val_clip, y_val_clip)
```

0.2 Evaluate Models

Added baseline (past_50m_span_ewm_vol) to compare to models.

[4]:
```
# Evaluate glm
glm_eval_raw = evaluate_predictions(
    df_pred_raw["y_true"], df_pred_raw["glm_y_pred"],
    df_pred_raw["weight"]
)
print("Unclipped:\n")
print(glm_eval_raw)
glm_eval_clip = evaluate_predictions(
    df_pred_clip["y_true"], df_pred_clip["glm_y_pred"],
    df_pred_clip["weight"]
)
print("\nClipped:\n")
print(glm_eval_clip)
```

```
Unclipped:

bias              0.023247
gamma_deviance    0.124871
mae               0.096982
rmse              0.188458
dtype: float64

Clipped:

bias              0.023018
gamma_deviance    0.124766
mae               0.096265
rmse              0.168332
dtype: float64
```

[5]:
```
# Evaluate lgbm
lgbm_eval_raw = evaluate_predictions(
    df_pred_raw["y_true"], df_pred_raw["lgbm_y_pred"],
```

```
    df_pred_raw["weight"]
)
print("Unclipped:\n")
print(glm_eval_raw)
lgbm_eval_clip = evaluate_predictions(
    df_pred_clip["y_true"], df_pred_clip["lgbm_y_pred"],
    df_pred_clip["weight"]
)
print("\nClipped:\n")
print(lgbm_eval_clip)
```

```
Unclipped:

bias              0.023247
gamma_deviance    0.124871
mae               0.096982
rmse              0.188458
dtype: float64

Clipped:

bias              0.007185
gamma_deviance    0.106868
mae               0.086975
rmse              0.156047
dtype: float64
```

[6]:
```
# Evaluate baseline
base_eval_raw = evaluate_predictions(
    df_pred_raw["y_true"], df_pred_raw["baseline_y_pred"],
    df_pred_raw["weight"]
)
print("Unclipped:\n")
print(base_eval_raw)
base_eval_clip = evaluate_predictions(
    df_pred_clip["y_true"], df_pred_clip["baseline_y_pred"],
    df_pred_clip["weight"]
)
print("\nClipped:\n")
print(base_eval_clip)
```

```
Unclipped:

bias             -0.000721
gamma_deviance    0.138242
mae               0.097388
rmse              0.179108
dtype: float64
```

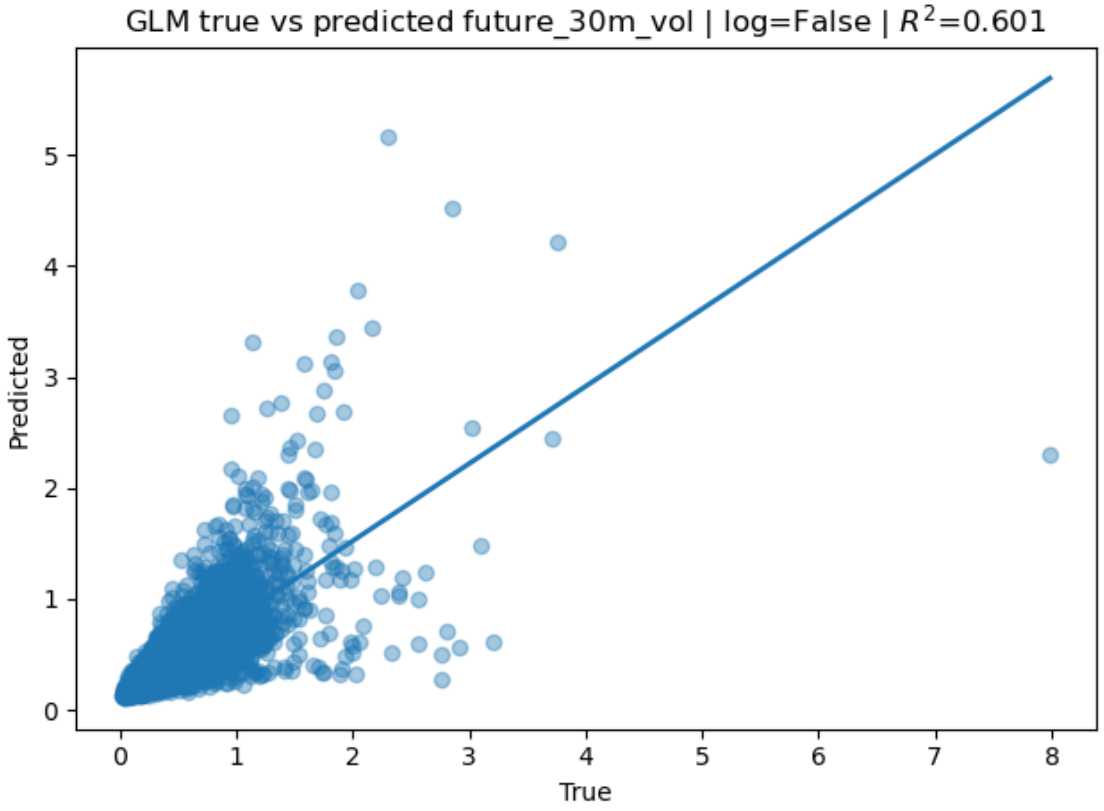
```
Clipped:

bias              -0.001047
gamma_deviance     0.138200
mae                0.097066
rmse               0.175826
dtype: float64
```

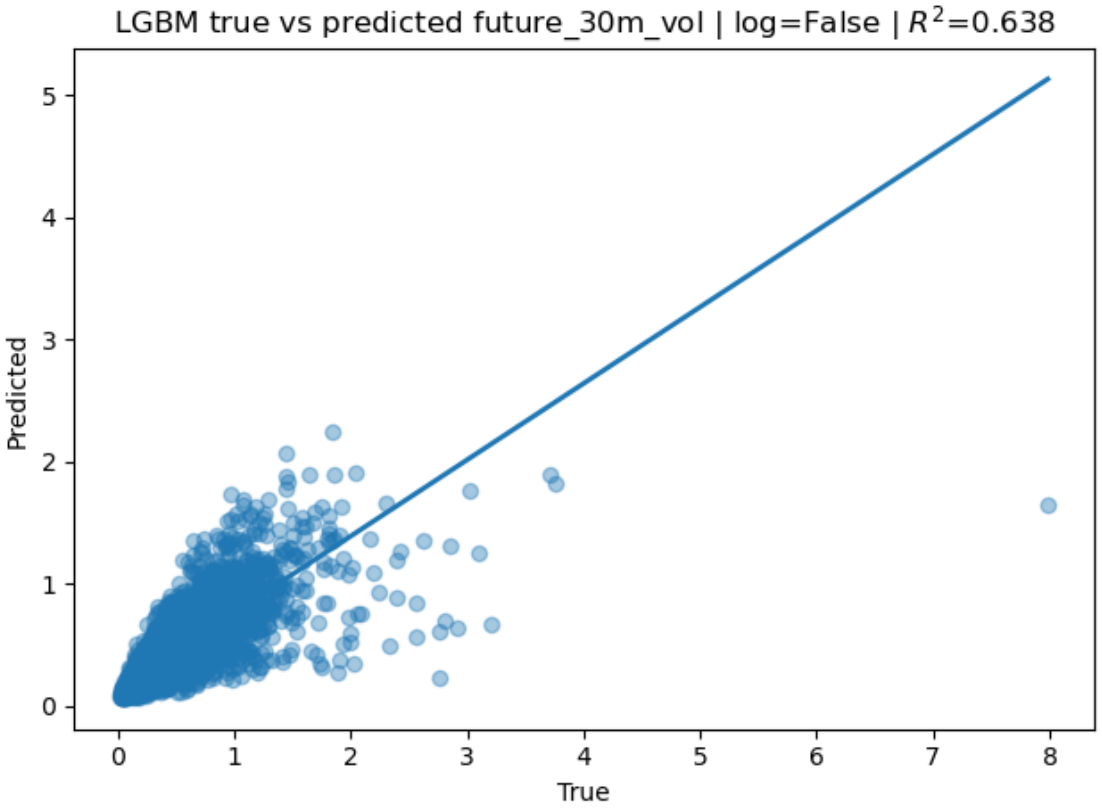
Looks like the GBT outperformed the GLM on all measures! And clipped data is significantly better than unclipped.

Predicted vs. Actual

[7]:
```
# Plot pred vs. true for glm
fig = plot_pred_vs_true(df_pred_clip, "glm")
```

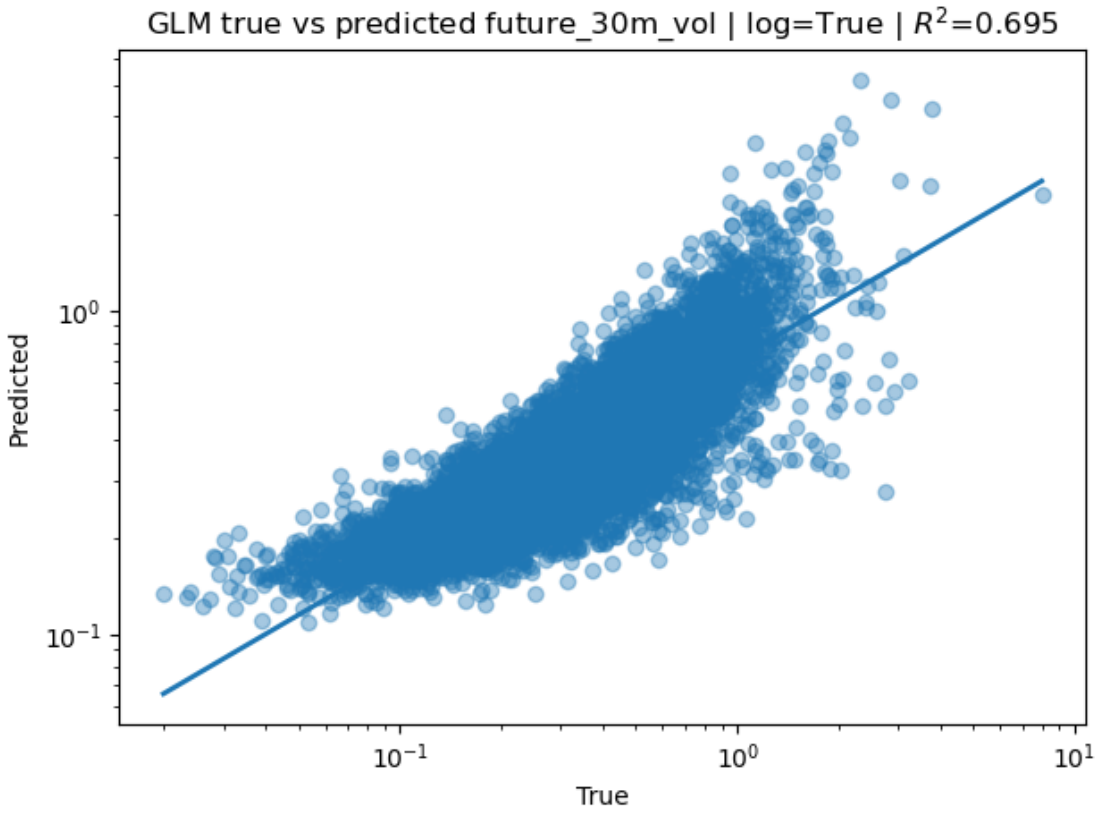


[8]:
```
# Plot pred vs. true for lgbm
fig = plot_pred_vs_true(df_pred_clip, "lgbm")
```

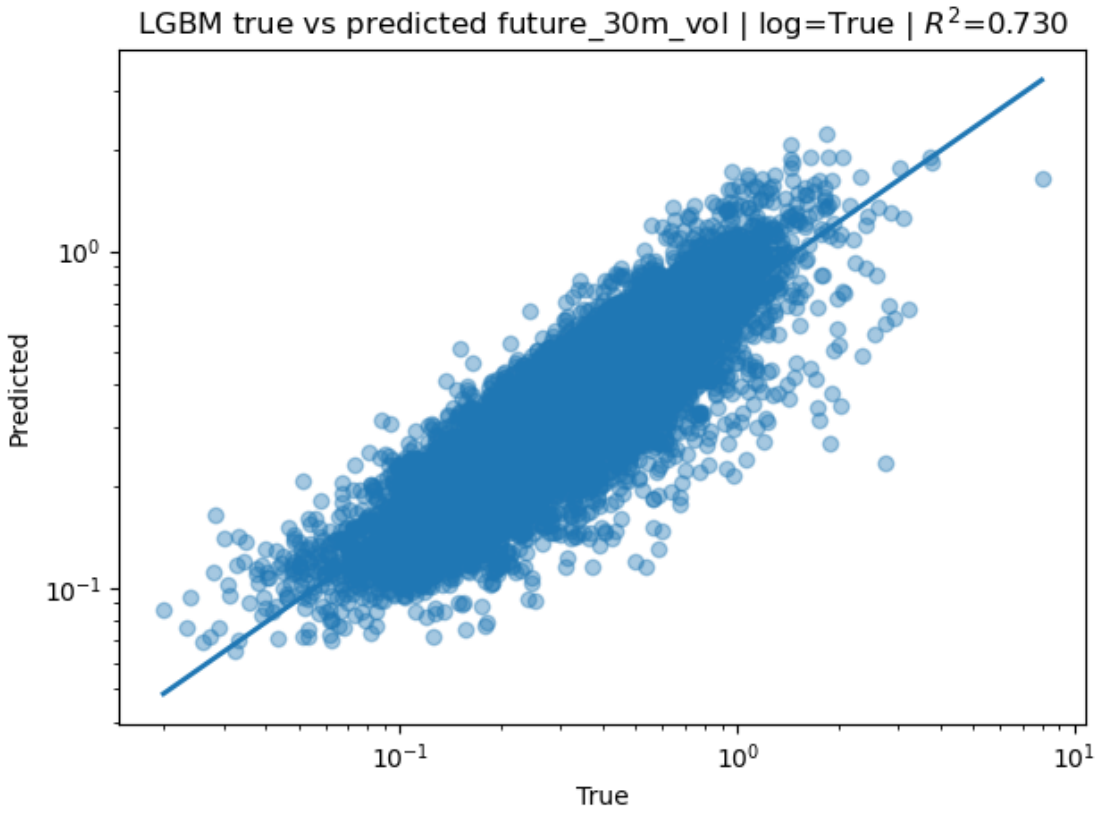


This should look better with logs!

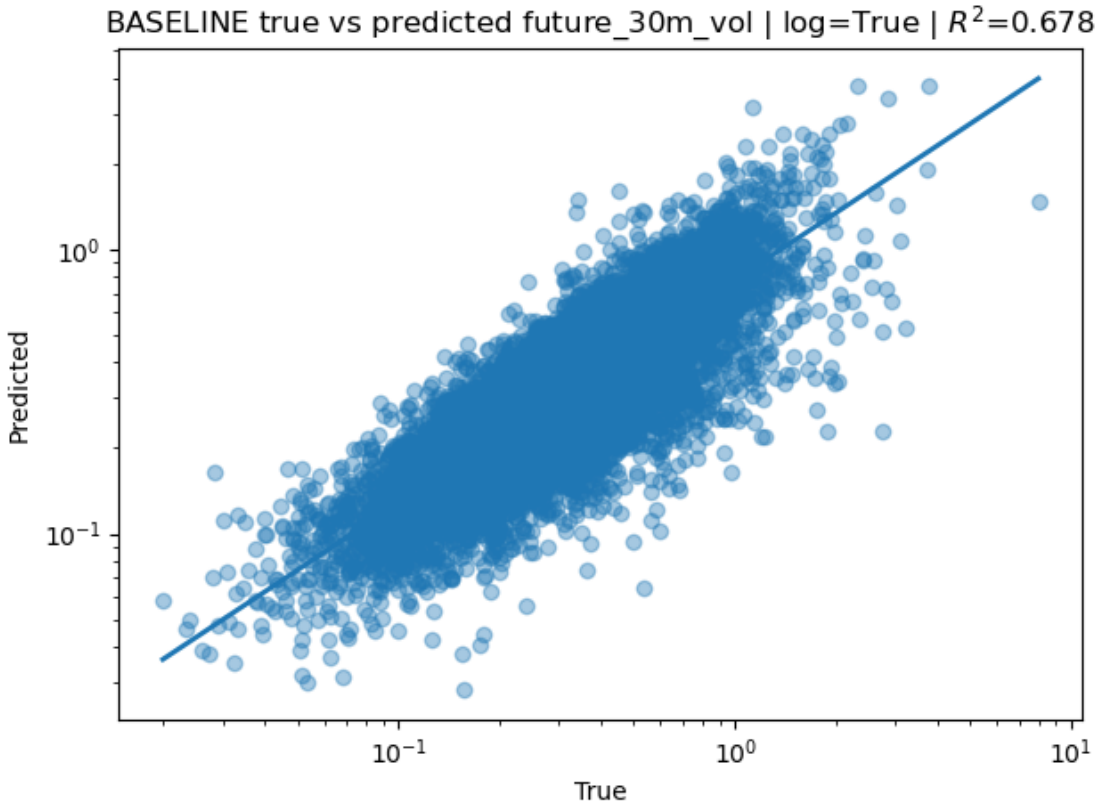
[9]:
```
# Plot pred vs. true for glm, log axes
fig = plot_pred_vs_true(df_pred_clip, "glm", log=True)
```



[10]:
```
# Plot pred vs. true for lgbm, log axes
fig = plot_pred_vs_true(df_pred_clip, "lgbm", log=True)
```

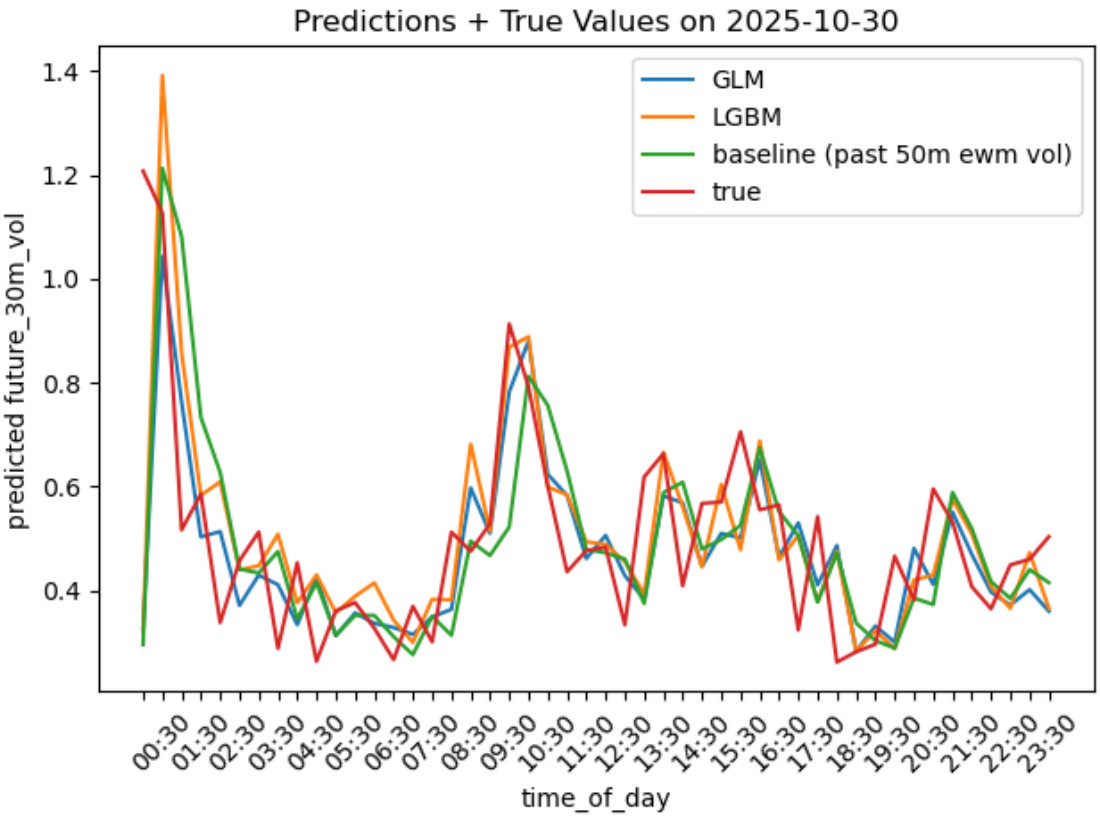


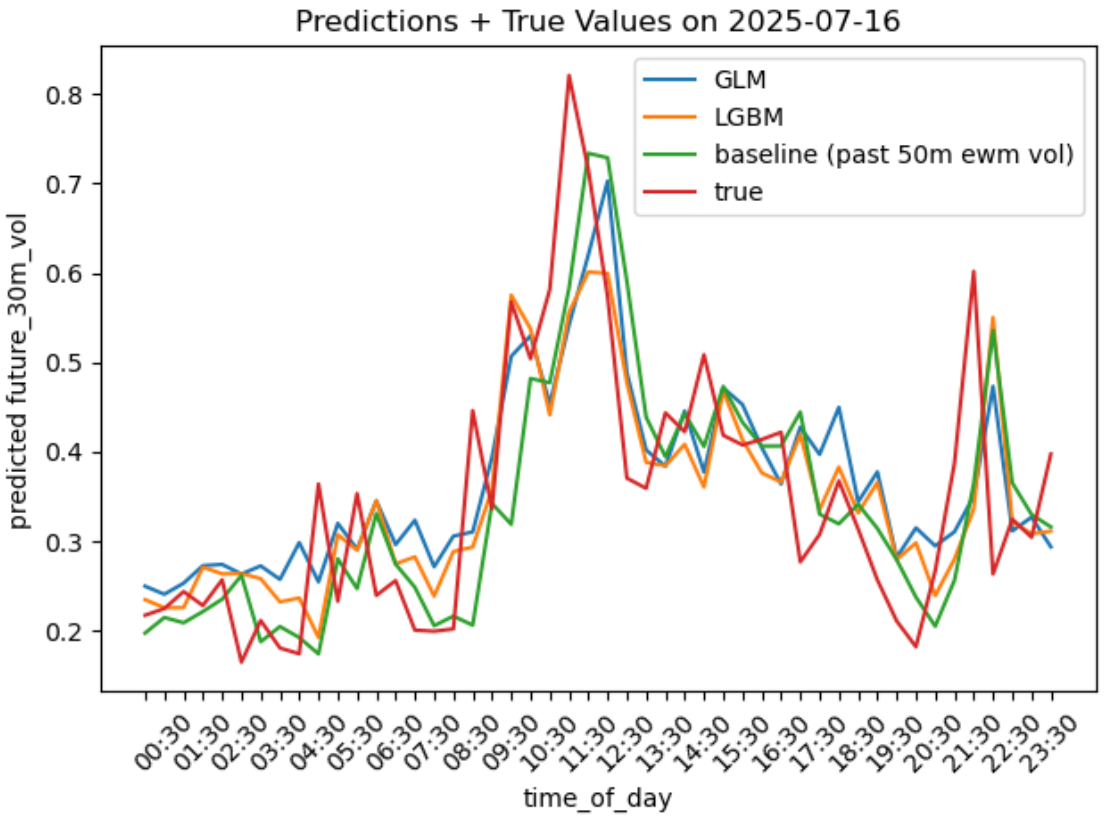
[11]:
```
# Plot pred vs. true for baseline, log axes
fig = plot_pred_vs_true(df_pred_clip, "baseline", log=True)
```



Look at predicted value throughout the day.

[12]:
```
fig = plot_day_predictions(df_pred_clip, "2025-10-30")
fig = plot_day_predictions(df_pred_clip, "2025-07-16")
```


Predictions + True Values on 2025-10-30
GLM
LGBM
baseline (past 50m ewm vol)
true
predicted future_30m_vol
1.4
1.2
1.0
0.8
0.6
0.4
00:30
01:30
02:30
03:30
04:30
05:30
06:30
07:30
08:30
09:30
10:30
11:30
12:30
13:30
14:30
15:30
16:30
17:30
18:30
19:30
20:30
21:30
22:30
23:30
time_of_day



0.3 Feature Relevance and PDPs

[13]:
```
# Plot glm features
glm_top_5 = plot_feature_relevance(glm_clip, X_val_clip, y_val_clip)
```

```
Preparation of a new explainer is initiated

  -> data              : 14301 rows 29 cols
  -> target variable   : Parameter 'y' was a pandas.Series. Converted to a
numpy.ndarray.
  -> target variable   : 14301 values
  -> model_class       : glum._glm.GeneralizedLinearRegressor (default)
  -> label             : Not specified, model's class short name will be used.
(default)
  -> predict function  : <function yhat_default at 0x3021b2e60> will be used
(default)
  -> predict function  : Accepts only pandas.DataFrame, numpy.ndarray causes
problems.
  -> predicted values  : min = 0.109, mean = 0.38, max = 5.16
  -> model type        : regression will be used
  -> residual function : difference between y and yhat (default)
```

```
  -> residuals         : min = -2.86, mean = -0.023, max = 5.69
  -> model_info        : package sklearn

A new explainer has been created!
```

[14]:
```
# Plot lgbm features
lgbm_top_5 = plot_feature_relevance(lgbm_clip, X_val_clip, y_val_clip)
```

```
Preparation of a new explainer is initiated

  -> data              : 14301 rows 29 cols
  -> target variable   : Parameter 'y' was a pandas.Series. Converted to a
numpy.ndarray.
  -> target variable   : 14301 values
  -> model_class       : lightgbm.sklearn.LGBMRegressor (default)
  -> label             : Not specified, model's class short name will be used.
(default)
  -> predict function  : <function yhat_default at 0x3021b2e60> will be used
(default)
  -> predict function  : Accepts only pandas.DataFrame, numpy.ndarray causes
problems.
  -> predicted values  : min = 0.0648, mean = 0.364, max = 2.24
  -> model type        : regression will be used
  -> residual function : difference between y and yhat (default)
  -> residuals         : min = -0.774, mean = -0.00719, max = 6.34
  -> model_info        : package sklearn

A new explainer has been created!
```

[15]:
```
plot_pdps(glm_clip, X_val_clip, y_val_clip, n_top=5)
```

```
Preparation of a new explainer is initiated

  -> data              : 14301 rows 29 cols
  -> target variable   : Parameter 'y' was a pandas.Series. Converted to a
numpy.ndarray.
  -> target variable   : 14301 values
  -> model_class       : glum._glm.GeneralizedLinearRegressor (default)
  -> label             : Not specified, model's class short name will be used.
(default)
  -> predict function  : <function yhat_default at 0x3021b2e60> will be used
(default)
  -> predict function  : Accepts only pandas.DataFrame, numpy.ndarray causes
problems.
  -> predicted values  : min = 0.109, mean = 0.38, max = 5.16
  -> model type        : regression will be used
  -> residual function : difference between y and yhat (default)
  -> residuals         : min = -2.86, mean = -0.023, max = 5.69
  -> model_info        : package sklearn
```

```
A new explainer has been created!

Calculating ceteris paribus: 100%|     | 4/4 [00:00<00:00, 37.37it/s]

Calculating ceteris paribus: 100%|     | 1/1 [00:00<00:00, 58.76it/s]
```

[16]:
```
plot_pdps(lgbm_clip, X_val_clip, y_val_clip, n_top=5)
```

```
Preparation of a new explainer is initiated

  -> data              : 14301 rows 29 cols
  -> target variable   : Parameter 'y' was a pandas.Series. Converted to a
numpy.ndarray.
  -> target variable   : 14301 values
  -> model_class       : lightgbm.sklearn.LGBMRegressor (default)
  -> label             : Not specified, model's class short name will be used.
(default)
  -> predict function  : <function yhat_default at 0x3021b2e60> will be used
(default)
  -> predict function  : Accepts only pandas.DataFrame, numpy.ndarray causes
problems.
  -> predicted values  : min = 0.0648, mean = 0.364, max = 2.24
  -> model type        : regression will be used
  -> residual function : difference between y and yhat (default)
  -> residuals         : min = -0.774, mean = -0.00719, max = 6.34
  -> model_info        : package sklearn

A new explainer has been created!

Calculating ceteris paribus:  75%|      | 3/4 [00:00<00:00,
9.01it/s]/opt/homebrew/Caskroom/miniconda/base/envs/final-
project/lib/python3.10/site-
packages/dalex/predict_explanations/_ceteris_paribus/utils.py:100:
FutureWarning:

Setting an item of incompatible dtype is deprecated and will raise in a future
error of pandas. Value '[0.   0.01 0.02 … 0.98 0.99 1.  ]' has dtype
incompatible with int64, please explicitly cast to a compatible dtype first.

Calculating ceteris paribus: 100%|     | 4/4 [00:00<00:00,  8.98it/s]

Calculating ceteris paribus: 100%|     | 1/1 [00:00<00:00, 16.24it/s]
```