

Assignment 5 Project - Predicting Energy Demand

Karan Kapur (1008384874) and Keagan Rankin (1008578692)

1 Part B: problem statement

Anticipating energy demand is increasingly essential for both private and public entities. For private companies, predicting demand can help mitigate risk and optimize power usage/sell back to the grid for profit. For the government, knowing future demand helps them predict buyer response and the best time to import/export power from neighboring provinces. As renewable energy and battery deployment start to dominate the energy sector, it will also become increasingly important to predict demand in order to facilitate grid support services (e.g. energy buy-back from batteries, two-way power flow from solar PV). In response, we propose two machine learning models for predicting future energy demand using open-source data provided by provincial and federal agencies. We train our models on hourly data from 2018-2023 (~50,000 data points), engineer features using time-series properties and dimensional reduction of climatic data, and deploy the model using Azure's cloud framework and AutoML.

2 Data set and exploratory analysis

2.1 Power data

All power data was taken from Ontario's Independent Electricity System Operator (IESO). They have well-documented, hourly data for energy demand (in MWh), power price (in \$/MW), as well as net imports and exports to the province between 2018-2023 (Figure 1).

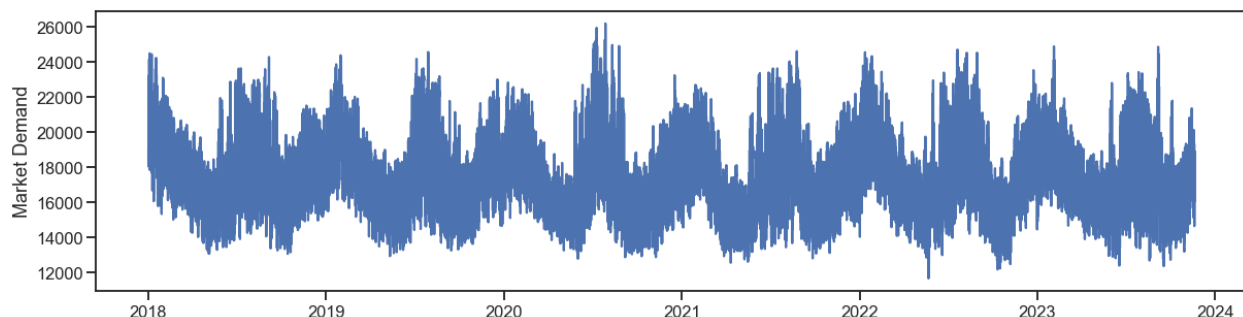


Figure 1 - Ontario hourly demand in Megawatts from 2018-2023

Figure 2 shows that demand for power is relatively stationary across years, with some outlying dates in 2020 (likely due to the pandemic). Demand for power is lowest overnight and peaks around 6-8pm when people arrive home from work. This daily demand trend curve is consistent across many years.

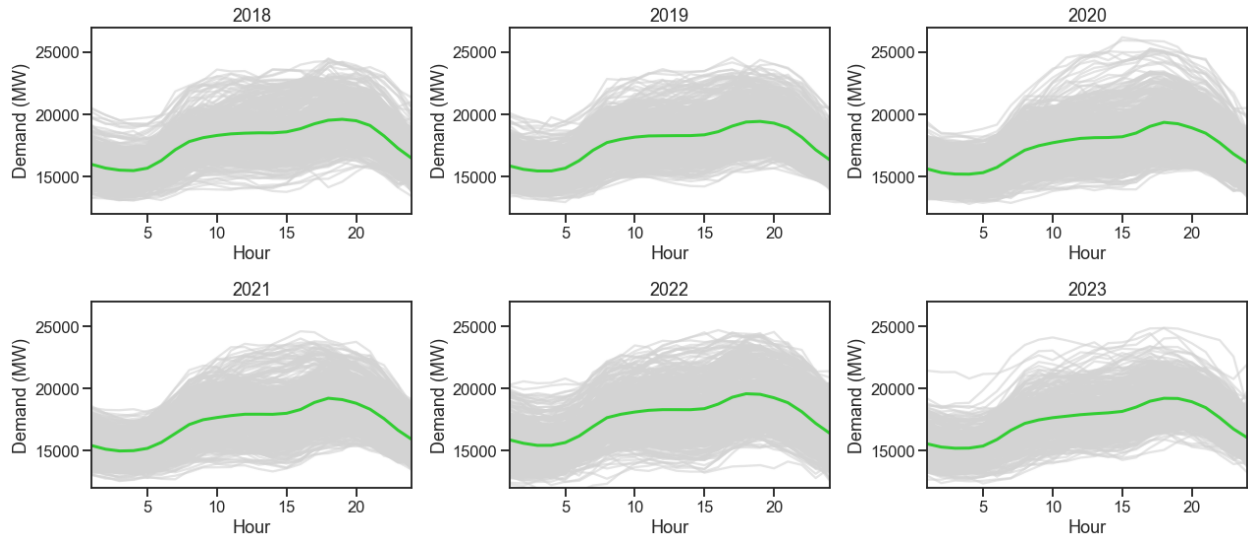


Figure 2 - Daily Ontario power demand. The individual time-series for each day are plotting in grey, and the mean is plotting in green.

By contrast, a variable like energy price is much more inconsistent. Prices see spikes which tend to happen around large changes in demand and energy import/exports to the province (dubbed “ramp” by the IESO and other energy operators) (Figure 3).

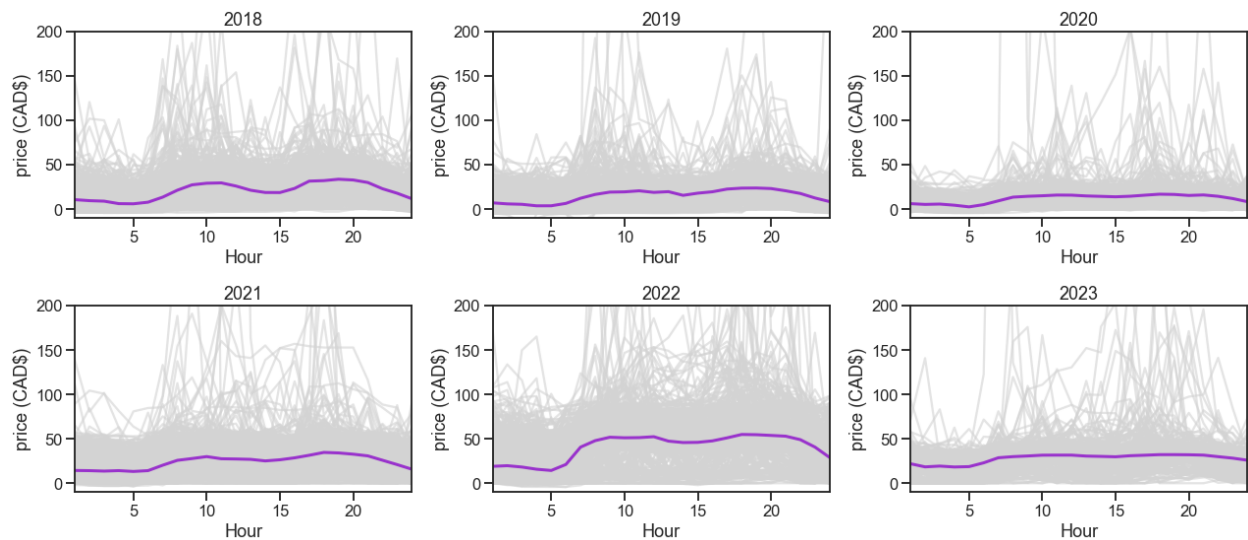


Figure 3 - Daily Ontario energy price. The individual time-series for each day are plotting in grey, and the median is plotting in purple.

2.2 Weather data

Climatic data comes from Environment and climate change Canada. We take the daily temperature readings from Pearson International Airport and join them to the

hourly power data using datetime features in Pandas. Figure 4 shows the first two components of a singular value decomposition on climatic data. Seasonal variations are the largest drivers of variance in the data, with temperature and Month (both correlated) changing with component 1.

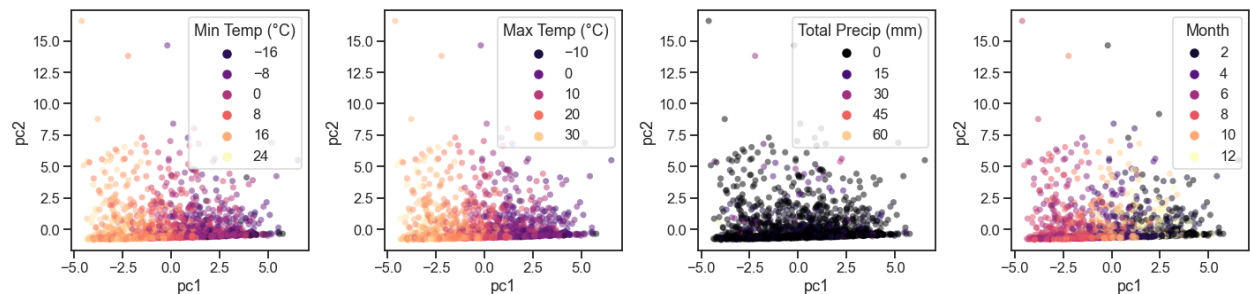


Figure 4 - PCA of Pearson Airport daily climatic data from 2018-2023. Specific variable values are overlaid on each subplot and indicated in the legends.

2.3 Data cleaning and split

After the initial exploratory analysis above, we combined all of the data based on a datetime (as it came from multiple sources). We split the data into training and testing sets according to best practices with time series. The training set contained data from 2018 up to Sept 2022, and the test set contained data from Sept 2022 to November 2023. We then created a function to clean and impute missing data (which was concentrated in the climatic features). Table 1 summarizes our cleaning/imputation method.

Table 1 - Missing values and imputation method

Feature	Imputation	Reasoning
Temperature and heating/cooling degree days	Monthly median	The monthly average temperature is a good guess at the temperature of any given day. We saw no correlation between extreme weather events (e.g. heavy precipitation) and missing data.
Precipitation (rain and snow)	Zero	We saw no correlation between extreme weather and missing precipitation data. We assume that these missing values are days when sensors were taken down for repair, meaning there was probably no precipitation (there were only around 14 missing days)

Maximum daily gust speed and gust direction	K-nearest neighbourhood	There were more missing values for the maximum wind gust and direction. We did not have good priors for this data, so we filled it using K-NN on other weather data (e.g. assuming that variables like min temp may be associated with the wind direction)
---	-------------------------	--

3 Feature engineering and Results

3.1 Choosing features and time-series dependence

We used the training set to explore several features. Most of the time series had some autocorrelation as shown in Figure 5. The periodicity ended up being useful as we used 24-hour lagged values to predict the values for the next day.

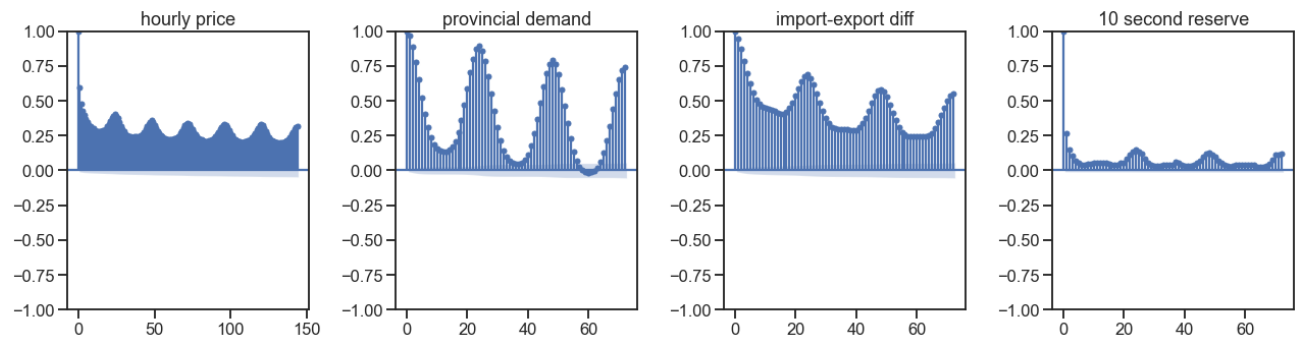


Figure 5 - Autocorrelation of IESO time-series data. Price data demonstrates no more autocorrelation if differenced, while other variables do.

From the IESO time-series data, we chose the following lagged features to predict demand:

- The demand from 24 hours before the prediction.
- The 3-hour moving average from 23, 24, and 25 hours for 2, 3, and 4 days before the prediction.

We chose to not use features from the day we were trying to predict because 1) we wanted to avoid information leakage (specifically from weather data) 2) we wanted our model to forecast an entire day ahead to improve its flexibility for real-world use cases, and 3) lagged features closer to the time of prediction (e.g. using the demand from the previous hour as a variable) actually resulted in a worse model that was overpowered by this predictor and resembled a random-walk prediction. We found that lagged price was not a very significant feature for demand, nor was other data from

IESO (demand could be quite inelastic to price as big consumers like industrial operations need energy to function through the day regardless of price).

We also analyzed the climatic data in relation to demand. We included the previous day's cooling degree days, gust direction and the month of prediction as features in our model as they were strongly correlated with Demand over a year period (Figure 6)

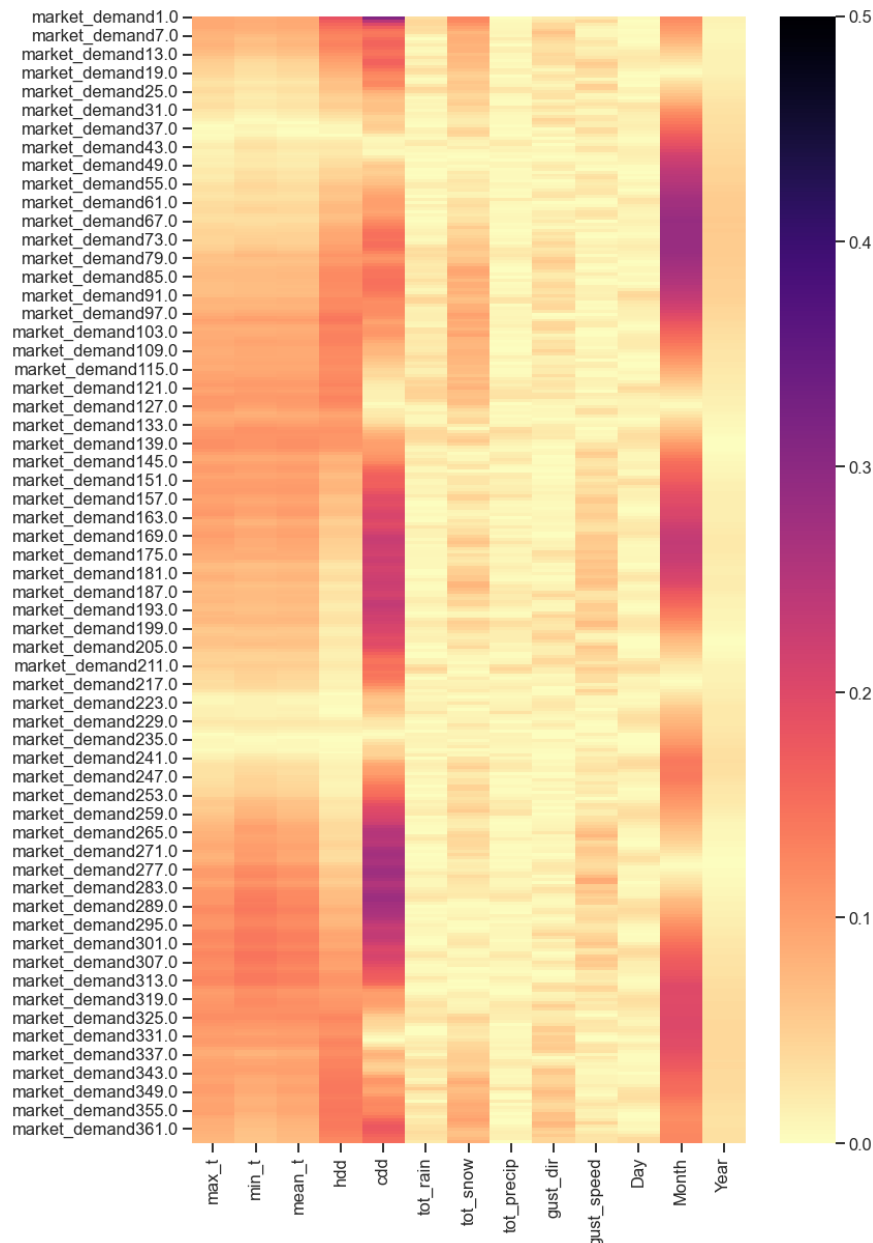


Figure 6 - Correlation of demand N-hours back with different climatic variables. The legend shows the Pearson R coefficient.

3.2 Two models and results

We chose to apply a linear regression and a gradient boosting regression model to our data. These models were easy to implement in scikit-learn on Azure and were of appropriate complexity for our 7 feature model (XGBoost could be overkill). We fit our baseline linear model using L^1 (lasso) regularization with $\alpha = 1.0$. Our model achieved a mean squared error (MSE) of 953086 (≈ 976 MW off on average) and an R^2 of 0.78 on the test set. These errors were a bit lower than how the model performed in validation on just the training set. Figure 7 shows results of the one-day ahead linear model predictions.

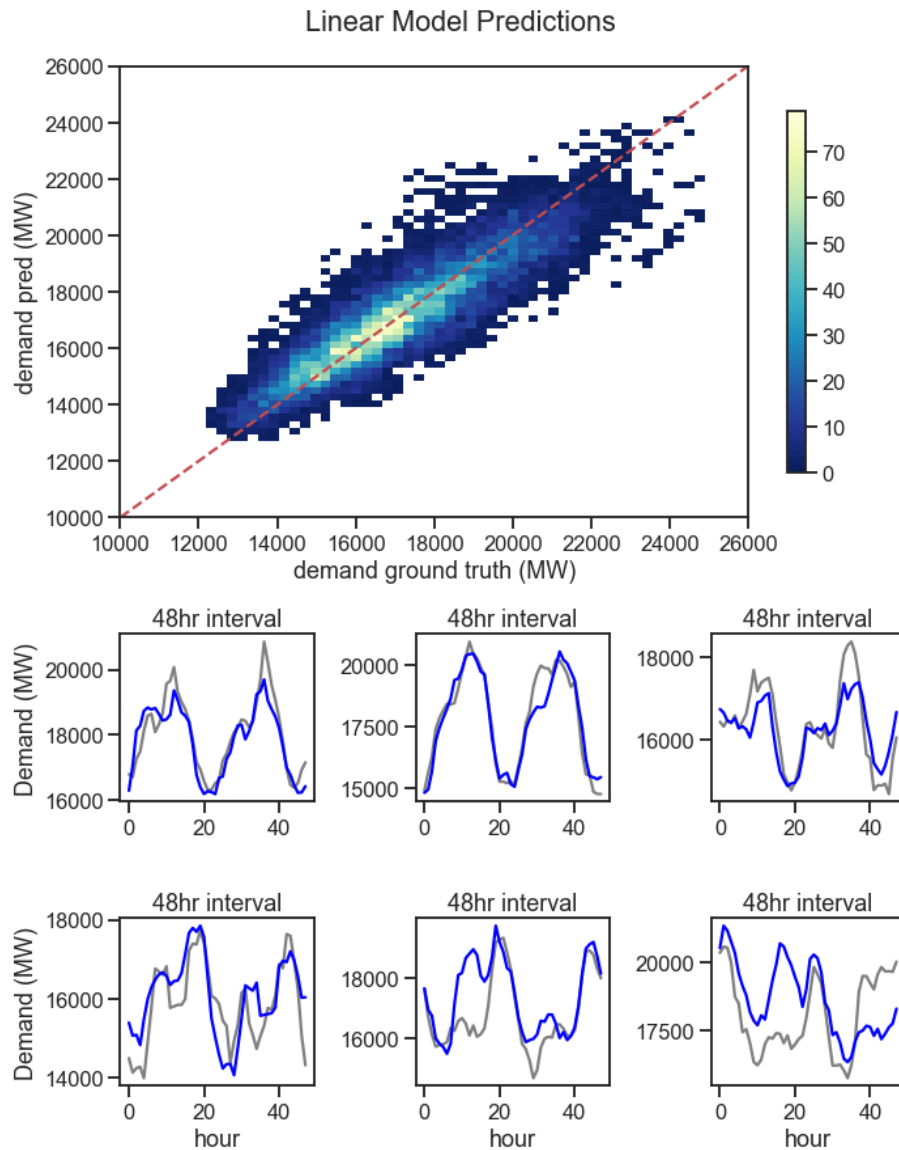


Figure 7 - Results of testing the linear model. The six plots below are random draws of 48-hour intervals from the test set, with predictions in blue and ground-truth in grey.

Our gradient boosting model was initialized with a minimum sample split+leaf of 5 and limited to using 5 features per ensemble unit to avoid overfitting. It did only slightly better than the linear model (without hyperparameter tuning), achieving a MSE of 909269 (average error of 950 MW) and an R^2 of 0.79 on the test set. Figure 8 shows results for the gradient boosting model.

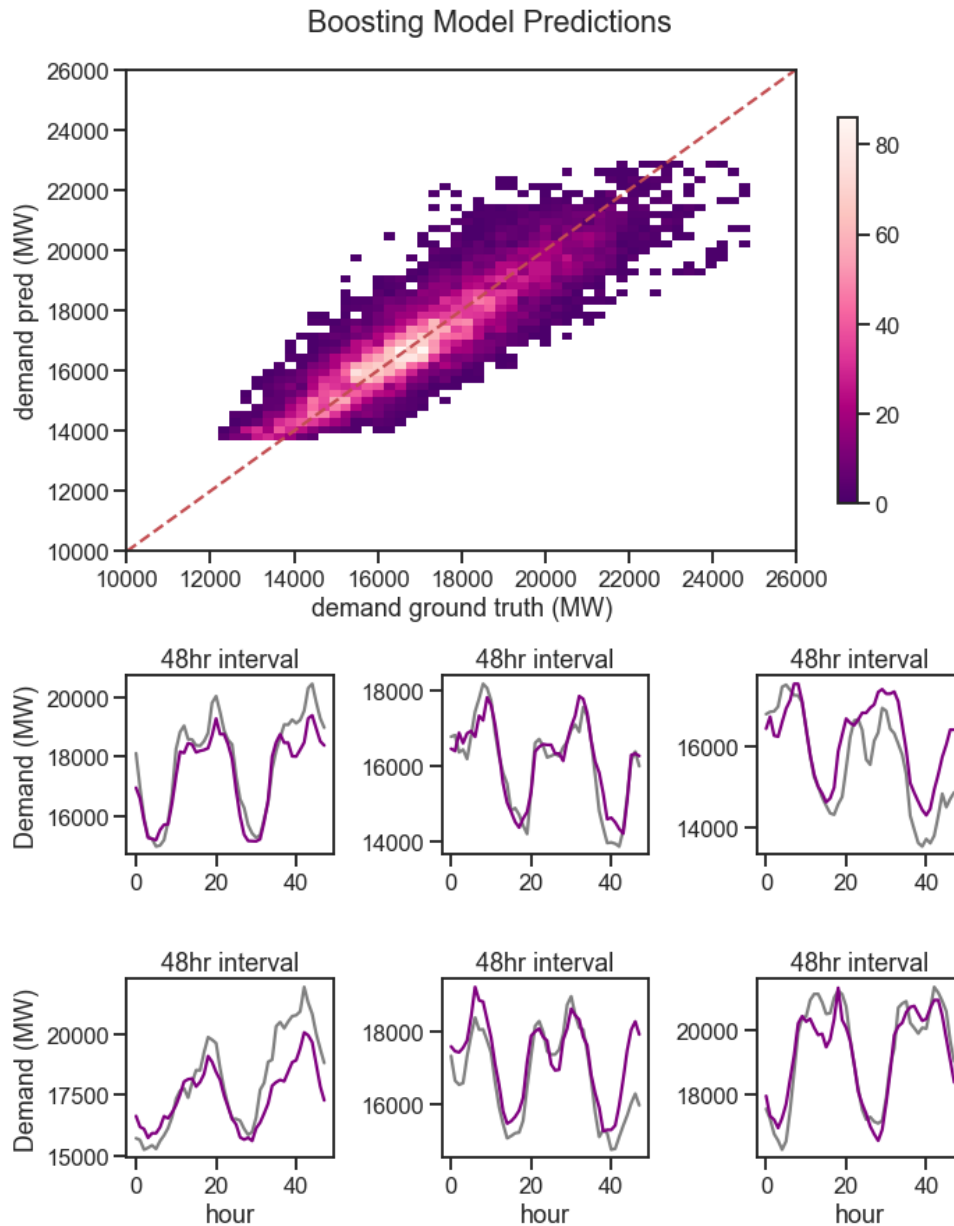


Figure 8 - Results of testing the gradient boosting model. The six plots below are random draws of 48-hour intervals from the test set, with predictions in purple and ground-truth in grey.

3.3 Improving predictions

We theorized methods of improving our demand prediction. Improved features could include one-hot encoded dates like holidays and weekends as well as more diverse climatic data from around Ontario. More lagged features farther back in time could also help improve predictions (though would cut off more training data at the beginning of the feature set). Supply data was poorly-organized on the IESO website but could be cleaned and used to improve demand estimates.

Different models could probably also improve predictions. XGBoost could improve over our current gradient boosting model (see AutoML implementation below). A linear regression with ARIMA errors (ARIMAX) might capture the autocorrelation of the time series better. The time series could also be fed into a deep-learning architecture (e.g. LSTM) which might achieve marginal gains.

4 AutoML implementation

Create data asset:

The screenshot shows the 'Create data asset' dialog in the Azure AI Machine Learning Studio. The 'Data type' tab is selected. The 'Name' field contains 'train_dataset'. The 'Description' field is empty. The 'Type' dropdown is set to 'Tabular'. The 'Use cases for data types' section on the right provides guidance on when to use 'File type', 'Folder type', or 'Table type'.

Set the name and type for your data asset

Customers should not include personal data or other sensitive information in fields marked with the ⓘ because the content in these fields may be logged and shared across Microsoft systems to facilitate operations and troubleshooting. [Learn more](#)

Name ⓘ

train_dataset

Description

Data asset description

Type ⓘ

Tabular

Use cases for data types

When should I use File type?

The File type is recommended in most scenarios when you are working with a single data file of any type (including tabular data). This type allows you to specify a file location by URI in a storage location on your local computer, an attached Datastore, blob/ADLS storage, or a publicly available (https) location. There are many types of supported URIs. In the Azure Machine Learning CLI v2 or Python SDK v2, this data type is called `uri_file`. [Learn more about the uri_file type](#) ⓘ

When should I use Folder type?

The Folder type has all the same capabilities and use cases as the File type, but is used when specifying a folder location. In the Azure Machine Learning CLI v2 or Python SDK v2, this data type is called `uri_folder`. [Learn more about the uri_folder type](#) ⓘ

When should I use Table type?

The Table type is most useful for advanced scenarios where you might need to abstract the schema definition for easier sharing. You should use it when you have complex transformations and schema you want to capture in a reusable asset. For simpler tabular data, the File and Folder types are recommended. If you choose the Table type in the Azure Machine Learning CLI v2 or Python SDK v2, this data type is called `mltable`. [Learn more about the mltable type](#) ⓘ

Table type requires an MTable file (DMML-based file (.tbl)), which can be created using the python sdk.

Back **Next** **Cancel**

Reviewing dataset:

The screenshot shows the 'Create data asset' dialog in the Azure AI Machine Learning Studio, 'Settings' tab. The 'File format' is 'Delimited', 'Delimiter' is 'Comma', and 'Encoding' is 'UTF-8'. The 'Column headers' are 'All files have same headers'. The 'Skip rows' are 'None'. The 'Dataset contains multi-line data' checkbox is unchecked. The 'Data preview' section shows a table of data.

Settings

These settings determine how the data is parsed. The initial settings are automatically detected; you can change them as needed to reparse the data.

File format ⓘ

Delimited

Delimiter ⓘ

Comma

Example

field1,field2,field3

Encoding ⓘ

UTF-8

Column headers ⓘ

All files have same headers

Skip rows ⓘ

None

☐ Dataset contains multi-line data ⓘ

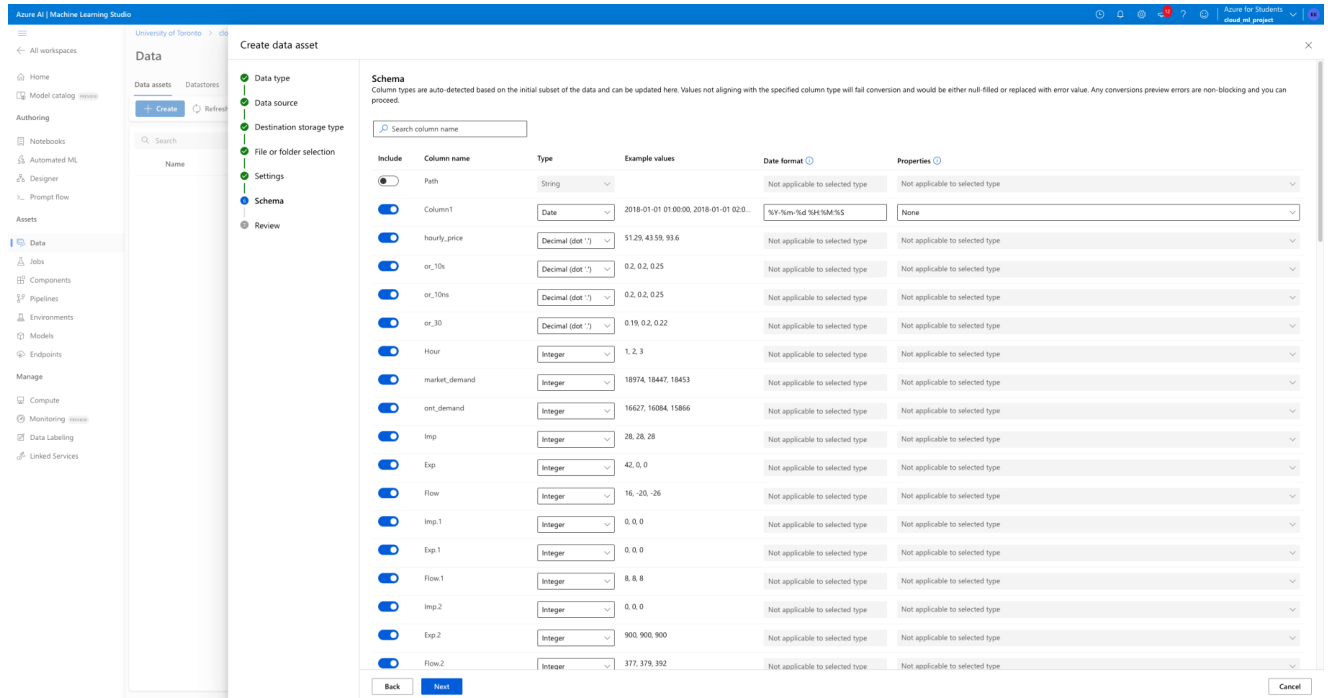
Note: Processing tabular files with multi-line data is slower because multiple CPU cores cannot be used to ingest the data in parallel. Checking this option may result in slower processing times.

Data preview

Column1	hourly...	or_10s	or_10ms	or_30	Hour	market...	ont.de...	Imp	Exp	Flow	Imp.1	Exp.1	Flow.1	Imp.2	Exp.2	Flow.2	Imp.3	Exp.3	Flow.3	Imp.4	Exp.4	Flow
2018-01...	51.29	0.2	0.2	0.19	1	18974	16627	28	42	16	0	0	8	0	900	377	0	94	90	0	1150	1688
2018-01...	43.59	0.2	0.2	0.2	2	18447	16084	28	0	-20	0	0	8	0	900	379	0	15	15	0	1150	1688
2018-01...	93.6	0.25	0.25	0.22	3	18453	15866	28	0	-26	0	0	8	0	900	392	0	39	36	0	1098	1690
2018-01...	54.78	0.2	0.2	0.2	4	18662	15725	28	0	-36	0	0	8	0	900	405	0	39	36	0	1109	1715
2018-01...	14.35	0.2	0.2	0.2	5	18060	15470	28	42	6	0	0	8	0	426	352	0	40	36	0	1150	1311
2018-01...	18.6	0.2	0.2	0.2	6	18429	15502	28	0	-6	0	0	7	0	504	598	0	5	0	0	1150	1073
2018-01...	21.72	0.2	0.2	0.2	7	19337	15750	0	76	66	0	0	7	50	950	941	0	0	-1	0	1150	1170
2018-01...	40.89	0.2	0.2	0.2	8	19470	15887	0	93	78	0	0	7	50	950	1033	0	79	75	0	1370	1166
2018-01...	20.75	0.2	0.2	0.2	9	19333	15923	0	42	46	0	0	7	50	950	793	0	39	44	0	1079	1090
2018-01...	74.57	6.74	6.74	2.11	10	19816	16381	0	54	47	0	0	7	0	900	768	0	39	40	0	1249	1310
2018-01...	4.65	0.21	0.21	0.21	11	19667	16672	0	96	83	0	0	7	130	930	839	0	79	79	0	1450	1435
2018-01...	9.45	0.2	0.2	0.2	12	19682	16720	169	234	77	0	0	7	80	980	720	0	39	41	0	1415	1494
2018-01...	13.65	0.21	0.21	0.21	13	19621	16839	0	85	72	0	0	7	0	900	792	0	79	71	0	1415	1535
2018-01...	14.37	0.22	0.22	0.22	14	19600	16772	62	191	116	0	0	7	0	900	808	55	39	-13	0	1415	1500
2018-01...	26.73	0.21	0.21	0.2	15	19605	16868	100	191	80	0	0	7	0	840	688	32	39	8	0	1400	1550
2018-01...	42.12	0.25	0.25	0.16	16	20561	17314	100	207	110	0	0	7	0	900	835	41	59	21	0	1450	1515
2018-01...	42.28	2.99	2.99	0.16	17	21717	18081	0	0	11	0	0	7	125	1025	865	0	40	39	0	1425	1440
2018-01...	44.84	20.92	20.92	0.16	18	23149	19339	6	54	39	0	0	6	205	1105	682	55	39	-12	0	1450	1665
2018-01...	42.22	16.78	16.78	0.15	19	22944	19249	154	108	-30	0	0	6	174	1074	914	0	39	40	0	1450	1525
2018-01...	41.06	9.69	9.69	0.16	20	22851	18990	95	162	46	0	0	6	205	1105	1019	0	38	34	0	1450	1323
2018-01...	43.76	1.52	1.52	0.16	21	22468	18663	252	204	-43	0	0	6	80	980	982	0	38	36	0	1425	1323
2018-01...	48.38	3.45	3.45	0.21	22	21678	18074	275	272	-14	0	0	6	125	971	893	55	39	-18	0	1425	1370

Back **Next** **Review** **Cancel**

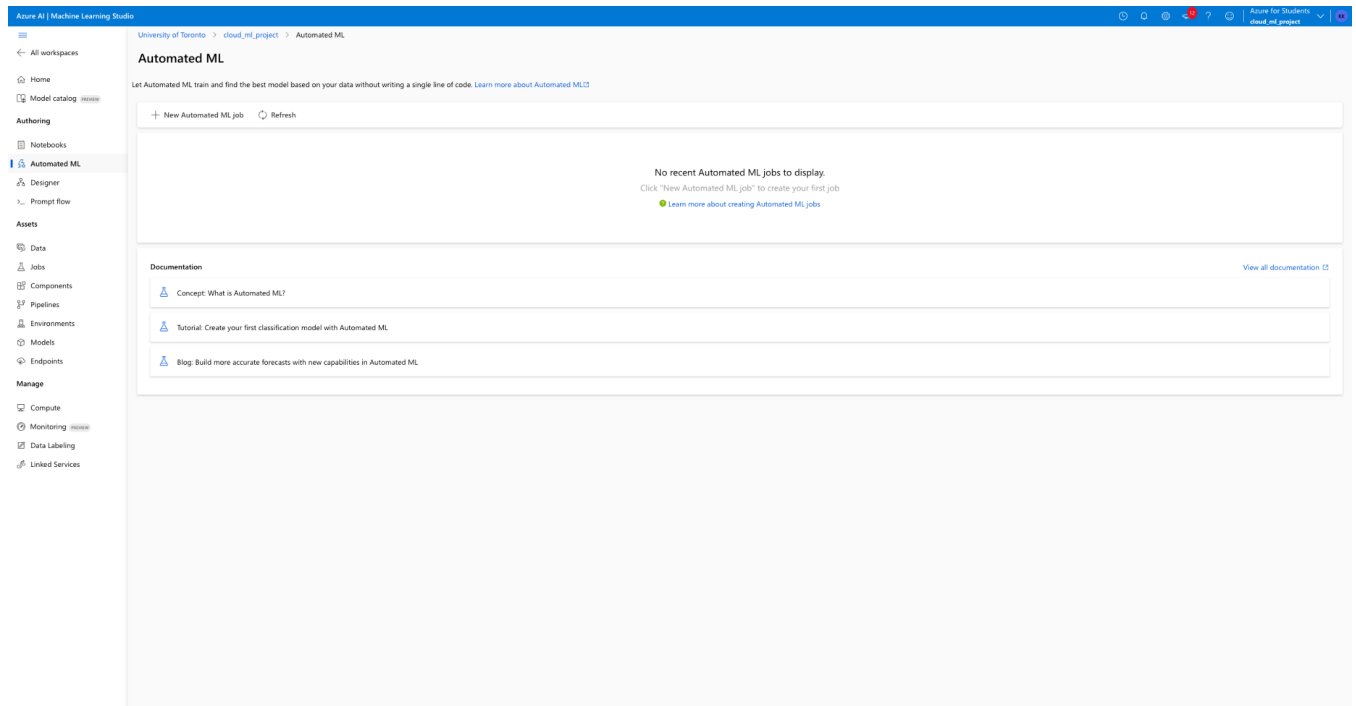
Reviewing schema:



The screenshot shows the 'Create data asset' dialog in Azure AI Machine Learning Studio. The 'Schema' tab is selected, displaying a table of auto-detected columns. The columns include 'Path', 'Column1', 'hourly_price', 'or_10s', 'or_10ns', 'or_30', 'market_demand', 'ont_demand', 'Imp', 'Exp', 'Flow', 'Imp.1', 'Exp.1', 'Flow.1', 'Imp.2', 'Exp.2', and 'Flow.2'. Each column has a type (String, Date, Decimal, Integer) and example values. The 'Date format' is set to '%Y-%m-%d %H:%M:%S'. The 'Properties' column is set to 'None' for all columns.

Include	Column name	Type	Example values	Date format	Properties
<input type="checkbox"/>	Path	String		Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Column1	Date	2018-01-01 01:00:00, 2018-01-01 02:00:00	%Y-%m-%d %H:%M:%S	None
<input checked="" type="checkbox"/>	hourly_price	Decimal (dot '.')	5129.4359, 93.6	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	or_10s	Decimal (dot '.')	0.2, 0.2, 0.25	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	or_10ns	Decimal (dot '.')	0.2, 0.2, 0.25	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	or_30	Decimal (dot '.')	0.19, 0.2, 0.22	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Hour	Integer	1, 2, 3	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	market_demand	Integer	18974, 18447, 18453	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	ont_demand	Integer	16627, 16084, 15866	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Imp	Integer	28, 28, 28	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Exp	Integer	42, 0, 0	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Flow	Integer	16, -20, -26	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Imp.1	Integer	0, 0, 0	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Exp.1	Integer	0, 0, 0	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Flow.1	Integer	8, 8, 8	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Imp.2	Integer	0, 0, 0	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Exp.2	Integer	900, 900, 900	Not applicable to selected type	Not applicable to selected type
<input checked="" type="checkbox"/>	Flow.2	Integer	377, 379, 392	Not applicable to selected type	Not applicable to selected type

Created workspace and entered the AutomatedML job centre:



The screenshot shows the 'Automated ML' job center in Azure AI Machine Learning Studio. The page has a header with the title 'Automated ML' and a sub-header 'Let Automated ML train and find the best model based on your data without writing a single line of code. [Learn more about Automated ML](#)'. Below the header, there is a 'New Automated ML job' button and a 'Refresh' button. A message states 'No recent Automated ML jobs to display.' and provides a link to 'Click "New Automated ML job" to create your first job' and a link to 'Learn more about creating Automated ML jobs'. Below this, there is a 'Documentation' section with three links: 'Concept: What is Automated ML?', 'Tutorial: Create your first classification model with Automated ML', and 'Blog: Build more accurate forecasts with new capabilities in Automated ML'. A 'View all documentation' link is also present.

AutomatedML started:

The screenshot shows the Azure AI Machine Learning Studio interface. The breadcrumb navigation at the top indicates the path: University of Toronto > cloud_ml_project > Jobs > my-1st-automl-experiment > energy_demand_pred. The job 'energy_demand_pred' is currently in a 'Running' state, as indicated by the status icon and the 'Running' label. The left sidebar contains navigation options: All workspaces, Home, Model catalog, Authoring (Notebooks, Automated ML, Designer, Prompt flow), Assets (Data, Jobs, Components, Pipelines, Environments, Models, Endpoints), and Manage (Compute, Monitoring, Data Labeling, Linked Services). The main panel displays the 'Overview' tab for the job. It includes a 'Properties' section with details like Status (Running), Job type (Automated ML), Experiment (my-1st-automl-experiment), Arguments (None), and creation/start times. There are also links to 'Raw JSON', 'See all properties', 'See YAML job definition', and 'Job YAML'. The 'Inputs' section shows 'Input name: training_data' and 'Data asset: train_dataset1'. The 'Best model summary' and 'Run summary' sections are currently empty, showing 'No data'.

Model training completed:

The screenshot shows the Azure AI Machine Learning Studio interface for a completed job. The breadcrumb navigation is: University of Toronto > cloud_ml_project > Jobs > 2nd_exp_ml > energy_demand_impute_feat. The job 'energy_demand_impute_feat' is in a 'Completed' state, indicated by a green checkmark. The left sidebar is identical to the previous screenshot. The main panel shows the 'Overview' tab. The 'Properties' section includes a warning: 'Warning: No scores improved over last 20 iterations, so experiment stopped early. This early stopping behavior can be disabled by setting enable_early_stopping = False'. It also lists creation and start times, duration (36m 45.56s), and compute duration (36m 45.56s). The 'Inputs' section shows 'Input name: training_data' (Data asset: train_dataset2) and 'Input name: test_data' (Data asset: test_data2). The 'Outputs' section lists 'Output name: best_model' (Model: azureml_energy_demand_impute_feat_1_output_mflow_log_model_1576007697.1) and 'Output name: full_validation_dataset' (Dataset: c7d12a20-23c6-40f0-b7a4-e514c47a6adc). The 'Best model summary' section shows the 'Algorithm name' as 'MaxAbsScaler, XGBoostRegressor' and the 'Hyperparameters' as 'View hyperparameters'. The 'Normalized root mean squared error' is 0.03780, and the 'Sampling' is 100.00 %.

Models page:

Azure AI | Machine Learning Studio

University of Toronto > cloud_ml_project > Jobs > 2nd_exp_ml > energy_demand_impute_feat

energy_demand_impute_feat Completed

Overview Data guardrails Models Outputs + logs Child jobs

Refresh Deploy Download Explain model View generated code View options

Model catalog

Authoring

Assets

Manage

Jobs

Components

Pipelines

Environments

Models

Endpoints

Compute

Monitoring

Data Labeling

Linked Services

Search

Filter Columns

Algorithm name	Explained	Responsible AI	Normalized ro... ↑	Sampling	Created on	Duration	Hyperpar
VotingEnsemble			0.03780	100.00 %	Nov 29, 2023 12:10 AM	52s	algorithm
MaxAbsScaler, XGBoostRegressor	View explanation	View responsible AI das	0.03780	100.00 %	Nov 28, 2023 11:41 PM	22s	tree_meth
MaxAbsScaler, LightGBM			0.04584	100.00 %	Nov 28, 2023 11:41 PM	22s	min_data,
MinMaxScaler, RandomForest			0.06101	100.00 %	Nov 28, 2023 11:41 PM	21s	bootstrap
MaxAbsScaler, ExtremeRandomTrees			0.06113	100.00 %	Nov 28, 2023 11:41 PM	20s	bootstrap
MinMaxScaler, RandomForest			0.06264	100.00 %	Nov 28, 2023 11:41 PM	19s	bootstrap
StandardScalerWrapper, RandomForest			0.06268	100.00 %	Nov 28, 2023 11:53 PM	39s	bootstrap
StandardScalerWrapper, ExtremeRandomTrees			0.06309	100.00 %	Nov 28, 2023 11:56 PM	40s	bootstrap
MinMaxScaler, ExtremeRandomTrees			0.06393	100.00 %	Nov 28, 2023 11:41 PM	19s	bootstrap
MinMaxScaler, ExtremeRandomTrees			0.06457	100.00 %	Nov 28, 2023 11:41 PM	19s	bootstrap
MinMaxScaler, LightGBM			0.06480	100.00 %	Nov 28, 2023 11:58 PM	38s	boosting,
StandardScalerWrapper, ElasticNet			0.06501	100.00 %	Nov 28, 2023 11:41 PM	19s	alpha : 0.0
StandardScalerWrapper, ElasticNet			0.06501	100.00 %	Nov 28, 2023 11:41 PM	20s	alpha : 0.0

<< < Page 1 of 2 > >>

25/Page

Best model is XGBoostRegressor and following are the model details:

Hyperparameters:

Data Transformation:

Data transformation:

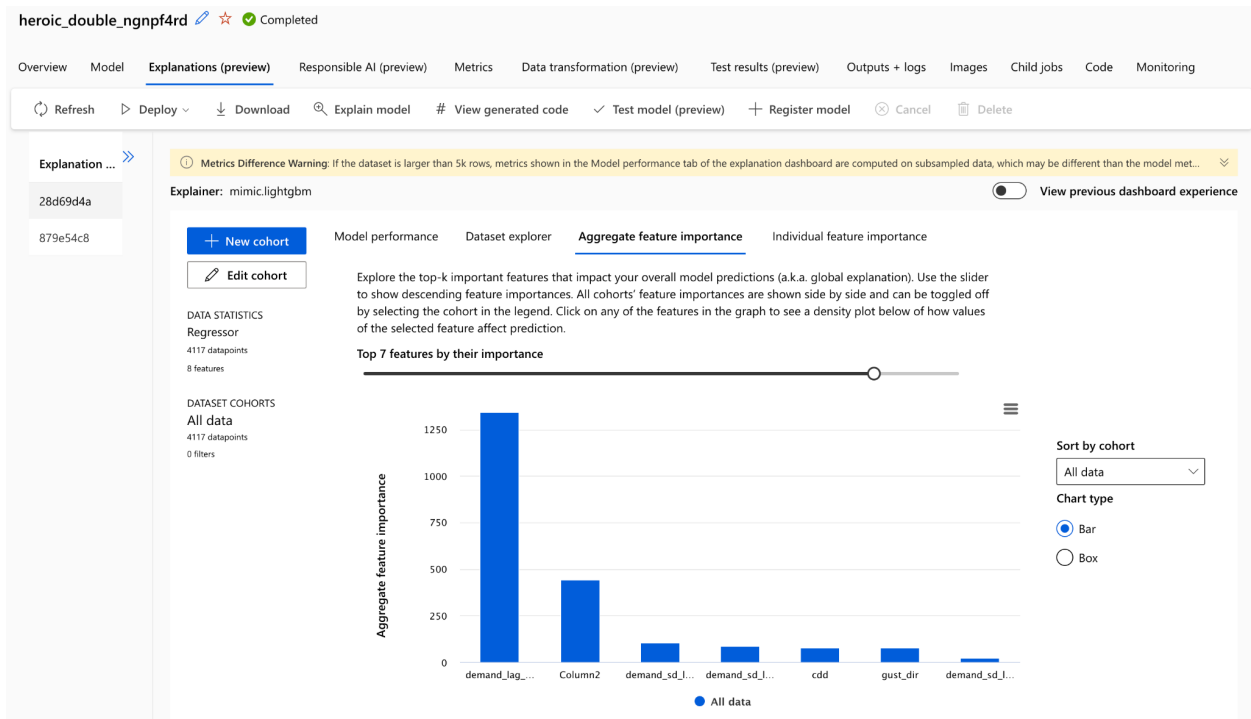
```
1  {
2    "spec_class": "preproc",
3    "class_name": "MaxAbsScaler",
4    "module": "sklearn.preprocessing",
5    "param_args": [],
6    "param_kwargs": {},
7    "prepared_kwargs": {}
8  }
```

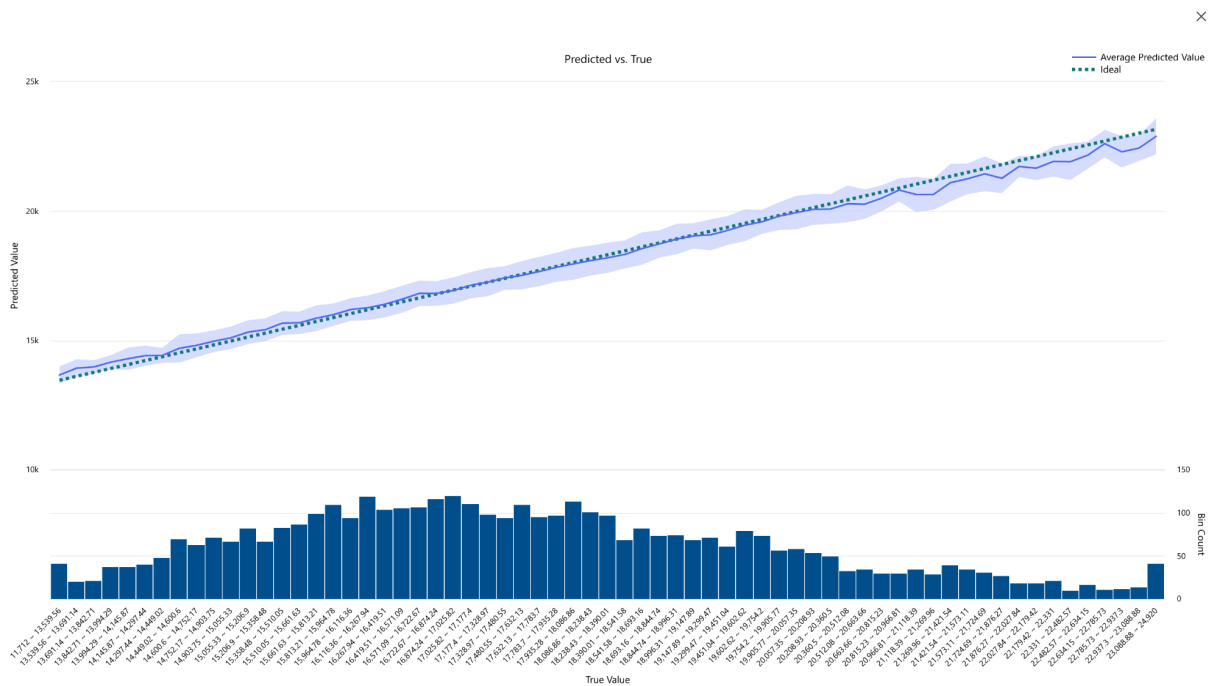
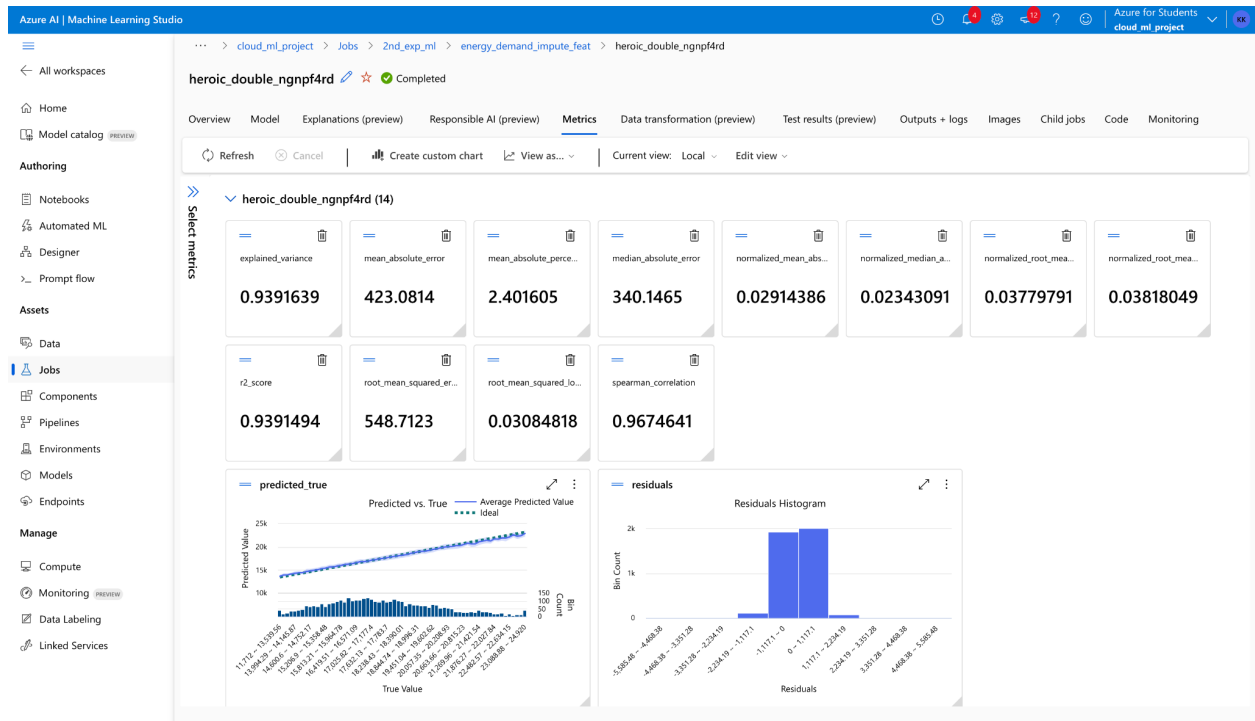
Training algorithm:

Training algorithm:

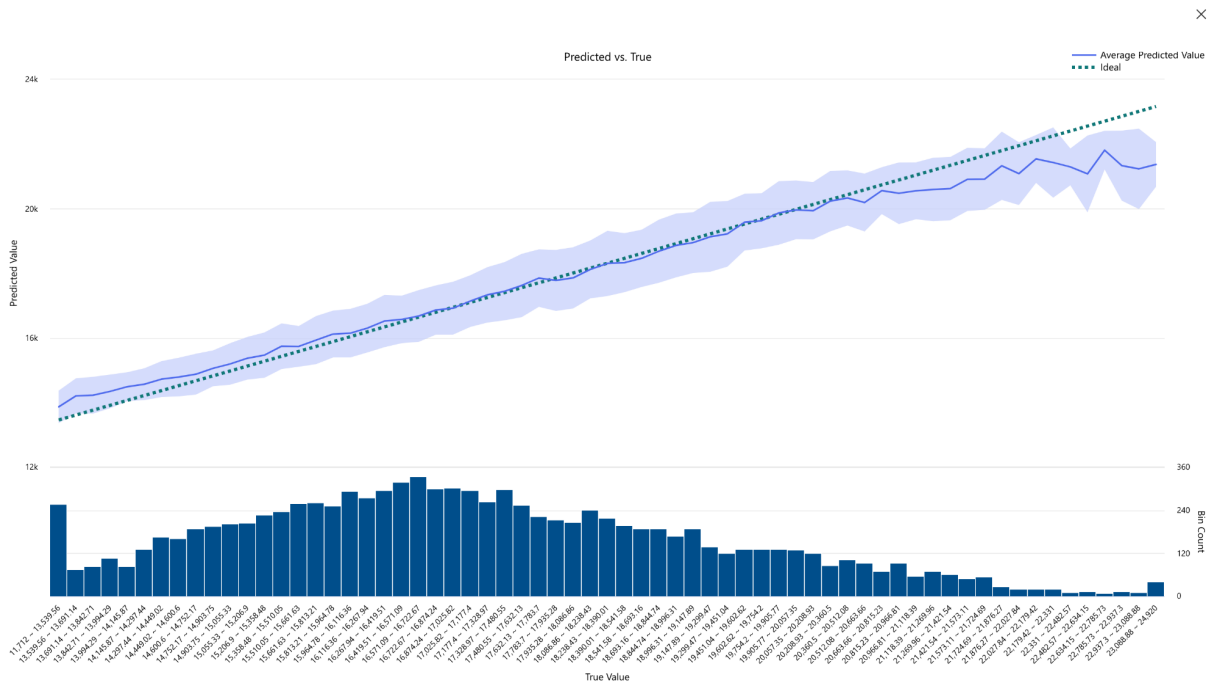
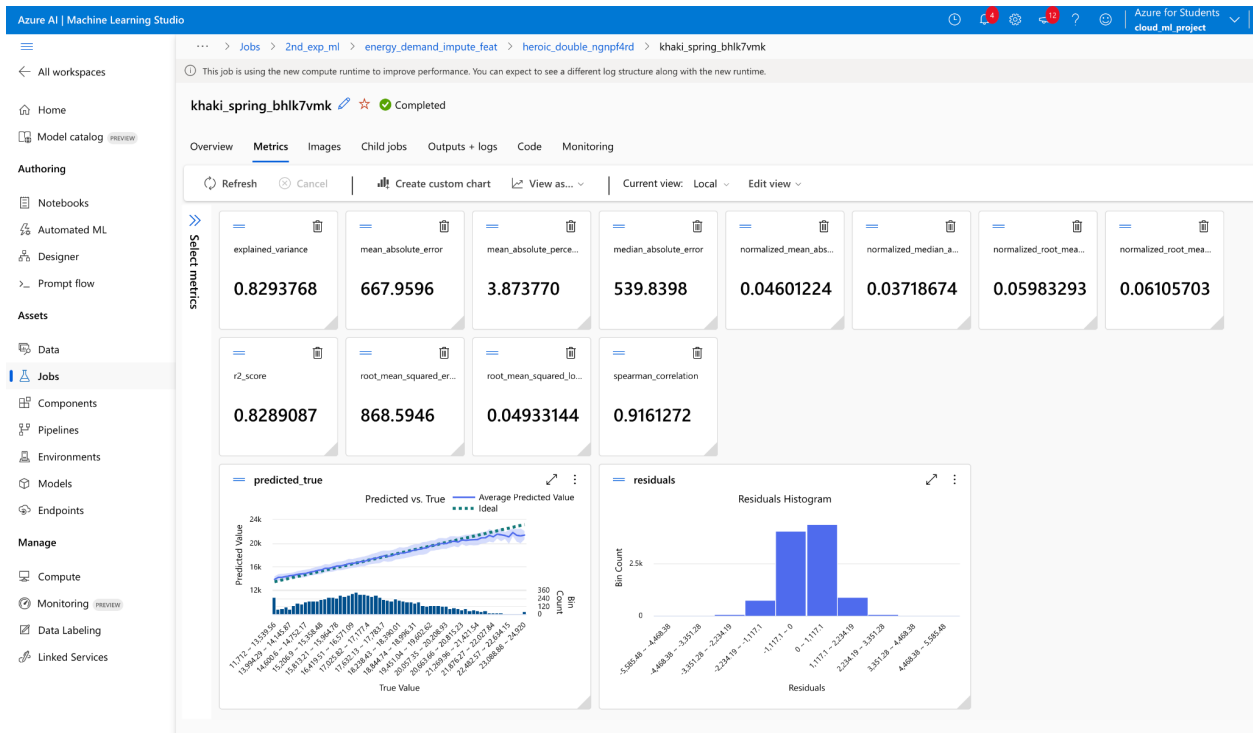
```
1  {
2    "spec_class": "sklearn",
3    "class_name": "XGBoostRegressor",
4    "module": "automl.client.core.common.model_wrappers",
5    "param_args": [],
6    "param_kwargs": {
7      "tree_method": "auto"
8    },
9    "prepared_kwargs": {}
10 }
```

Model explanations:





Results on test set:



5 Concluding Remarks

Energy price forecasting is a difficult but important task in the energy sector, and will become increasingly important with the sustainable energy transition. Models like the one presented here provide an opportunity to both support the grid and gain a competitive advantage by predicting future market changes.

The code and data for our project is stored at this [Github link](#). We have attached screenshots of our model running in Azure and AutoML.