Eesti Energy

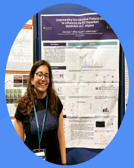
Building a Scalable Data & Modeling Architecture for Prosumer Energy Forecasting

Group 1

Group Leader /
Data Architect

Data Engineer

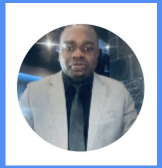
Kenichi
Data Engineer



Selin
Data Scientist

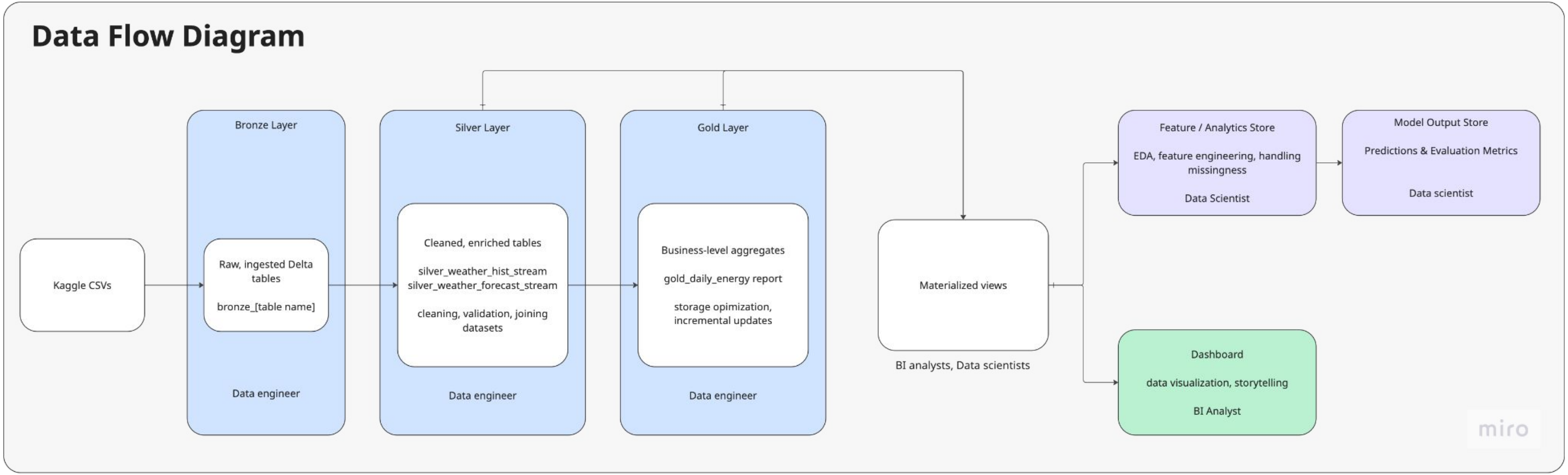
Liwei
Data Scientist /
BI Analyst

Peiran
BI Analyst

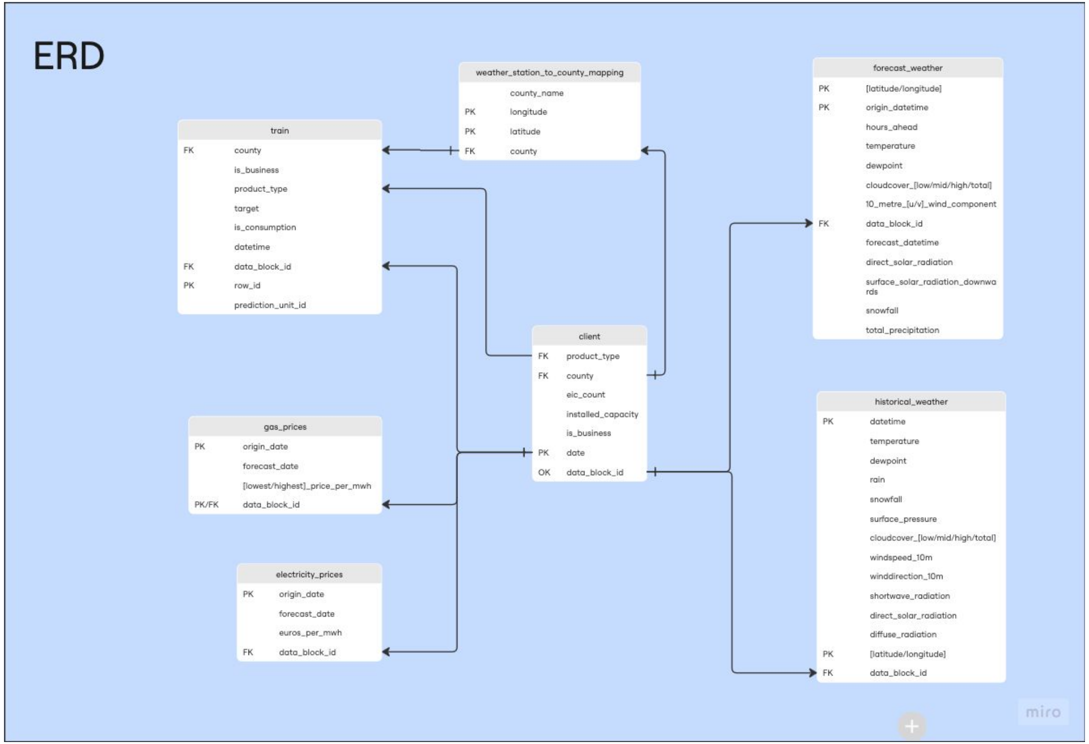


Chijioke
Data Architect

Data Architecture | Process Overview



Data Architecture | Data Overview



Data Engineering | Key Data Challenges

Fragmented Data Sources

Operational data arrived in inconsistent, volatile formats including daily dumps, ad-hoc corrections, and siloed CSVs, complicating the creation of a reliable analytics foundation.

Geospatial Data Incompatibility

Weather data indexed by latitude and longitude conflicted with grid operations organized by county boundaries, preventing seamless integration of datasets.

Diverging Analytical Needs

Machine-learning models demanded high-granularity historical data, while BI teams required stable, daily KPI reporting, creating conflicting data requirements.

Need for Scalable Engineering

These challenges required a scalable engineering approach to support real-time operational reporting alongside advanced forecasting models.

Data Engineering | Solving Data Challenges with Medallion Pipeline

Bronze Layer: Resilient Ingestion

```
    # Write to Delta table
    # model("overwrite"), overwriteSchema - ensures idempotency/robustness
    df_enriched.write.format("delta") \
        .mode("overwrite") \
        .option("overwriteSchema", "true") \
        .saveAsTable(target_table_name)

    print(f"Success: refreshed table: {target_table_name}")

# Ingest all files required for pipeline
ingest_bronze("client", "client.csv")
ingest_bronze("train", "train.csv")
ingest_bronze("gas_prices", "gas_prices.csv")
ingest_bronze("electricity_prices", "electricity_prices.csv")
ingest_bronze("weather_hist", "historical_weather.csv")
ingest_bronze("weather_forecast", "forecast_weather.csv")
ingest_bronze("weather_mapping", "weather_station_to_county_mapping.csv")

print("\nBronze Layer Ingestion Complete")
```

Key Engineering Decisions

Idempotent Ingestion Strategy

- Utilize Spark overwrite mode rather than simple appends
- This treats each daily batch as a complete snapshot, preventing data duplication even if the pipeline needs to be re-run multiple times due to upstream failures.

Automated Schema Evolution

- Enables the overwriteSchema option during write operations
- This future-proofs the system against “Schema Drift.” If the source system adds new columns, the Bronze layer automatically updates the table definition without breaking the pipeline, significantly reducing maintenance overhead.

Data Engineering | Solving Data Challenges with Medallion Pipeline

Silver Layer: Schema Normalization

```
# Join weather with the mapping table to get 'county'
enriched_df = weather_df.join(
    broadcast(mapping_df),
    on=["latitude", "longitude"],
    how="left"
)

# Write to delta
enriched_df.write.format("delta") \
    .mode("overwrite") \
    .option("overwriteSchema", "true") \
    .saveAsTable(target_table)

print(f"Created {target_table}")
```

Key Engineering Decisions

Geospatial Resolution

- Execute a transformation that maps weather data (indexed by Lat/Long) to grid operations (indexed by County).
- This creates a "common currency" (county_id) across all datasets, unlocking the ability to join weather features with operational metrics for downstream ML models.

Compute Optimization (Broadcast Join)

- Utilize a Broadcast Join to send the smaller static mapping table to all worker nodes.
- This eliminates the need to shuffle the massive weather dataset across the network. This structural optimization reduced join time significantly compared to a standard shuffle-sort join.

Data Engineering | Solving Data Challenges with Medallion Pipeline

Gold Layer: Optimized Reporting

```
# Merge with upsert logic
deltaTable = DeltaTable.forName(spark, target_table_name)

deltaTable.alias("target").merge(
    report_df.alias("source"),
    "target.date = source.date AND target.county = source.county"
).whenMatchedUpdateAll(
).whenNotMatchedInsertAll(
).execute()

print(f"Merge/Upsert complete for {target_table_name}")
```

Key Engineering Decisions

Efficient Upsert Logic

- Instead of truncating and reloading the entire reporting table, we use DeltaTable.merge().
- This allows us to modify only the specific records that have changed. This drastically reduces I/O overhead compared to full reloads.

Latency-Aware Data Correction

- The code distinguishes between existing records (Update) and new records (Insert).
- This ensures BI dashboards reflect the "latest and greatest" truth without maintaining multiple versions of the same record, providing the C-suite with a stable, single source of truth.

Data Engineering | Key Contributions

Silver Structured Streaming Pipeline

Implemented a robust Structured Streaming pipeline to incrementally process and join data from Bronze tables, writing the results to Silver streaming tables.

Data Quality/Configuration

Added table existence and schema validation checks, centralized catalog/schema/volume configuration, and improved error surfacing to enhance pipeline robustness and daily execution readiness.

Gold Layer Optimization (ZORDER)

Optimized the Gold layer by applying ZORDER partitioning on the 'gold_daily_energy_report' table, improving BI query performance and data skipping.

End-to-End Data Lineage Diagram

Created a comprehensive data lineage diagram showcasing the end-to-end flow from Bronze to Silver (Batch and Streaming) to Gold, providing visual clarity and an anchor for the final presentation.

Helper Utilities & Documentation

Developed a suite of helper utilities, including table_info(), compare_schemas(), preview(), and validate_columns(), to enhance team productivity and support the data science and BI teams.

Bronze Weather Tables → Streaming Engine

The data flow starts with the Bronze Weather Tables, which are then processed by the Streaming Engine.

NOTE: spark.readStream.table()

Streaming Engine → Join with station-to-county Mapping

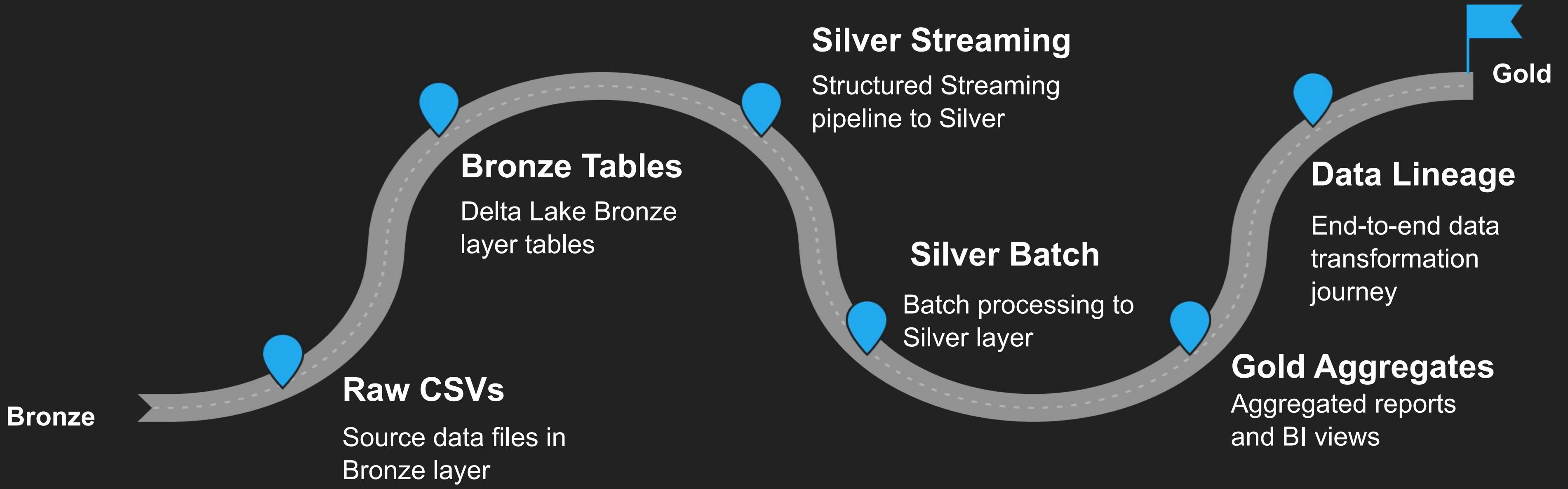
The data from the Streaming Engine is then joined with the Mapping data to enrich the information.

NOTE: trigger(once=True)

Join with Mapping → Silver streaming Tables

The enriched data is then written to the Silver
Streaming Tables, which are the final output of
this pipeline.

NOTE: Checkpointing stored in UC Volume



Data Science | EDA

Questions to ask: What are the key factors that determine energy consumption?

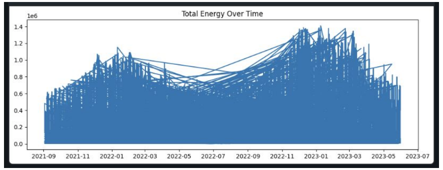
Need to perform exploratory data analysis on the gold layer data first!

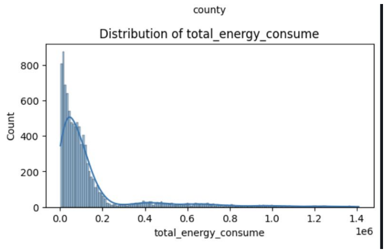
Things to look for

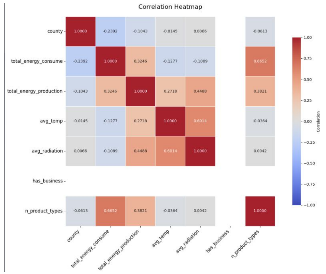
- Skewness/Distribution
- Missing values
- Duplicated rows
- Outliers
- Correlation Matrix (Multicollinearity)
- Visualisations between variables to understand relationships

Actions taken

- Log transformed heavily right skewed variables
- No missing or duplicated values
- There is some minor multicollinearity (ex: temp and radiation) but xgboost can handle it,so chose to leave variables







Data Science | Model Selection & Performance

- AutoML wasn't available in the Databricks free edition environment, used xgboost for both models.
- Split data into train and test
- Used MLFlow for lifecycle management, every run is recorded as an experiment.

Performance metrics to look for:

- Plot Actual vs Prediction values
- RMSE, R2, MAE, MAPE
- Residual plot
- Error distribution
- Feature importance values from the model

Important Considerations

- The Date column had to be split into day, month and year to make data more meaningful/
- The County column (ID) has to be categorical instead of an integer, otherwise the model would interpret the ID numbers as numbers instead of categories.
- Need to convert metrics after running the model, since target variable was initially log transformed

Data Science | Key takeaways and improvement

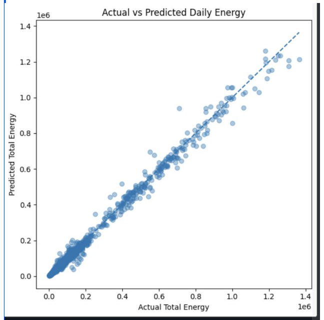
Findings:

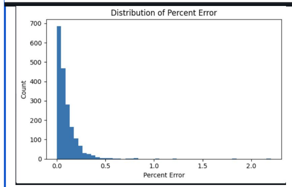
- Simple vs Complex model aren't that different in prediction power (R2: 0.9831 vs 0.9863). Models explain at least 98% of variability in both cases.
- County is the most important variable, followed by n_product_types and year.
  - This makes sense since geographical location and how industrial or rural the area is deterministic in energy consumption
  - Product type reflects economic activity
  - Year tells us there are also trends over time

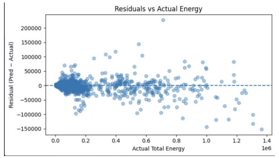
How to improve?

- Didn't optimize for the best model possible. Could use autoML and try many more
- Geographical location has a distinct effect, could remove that and focus on other variables to see what else impacts energy consumption (eg: more focus on weather radiation etc)

Performance Metrics:







metric	value
MAE_log	0.09723301550345761
RMSE_log	0.14555206859747313
R2_log	0.9863574496094758
MAE	9638.462477599674
RMSE	19043.365698977428
MAPE_%	9.849041804500796

	feature	gain
7	county_enc	16.483137
2	n_product_types	9.594130
4	year	1.032624
6	dayofweek	0.682672
5	month	0.634705
0	avg_temp	0.525710
3	total_energy_production_log	0.511178
1	avg_radiation_log	0.348388

Data Science | MLOps Feature Pipelines

Key Improvements:

1. Create Streaming at the Bronze layer for incremental ingestions
2. Added Materialized View at Silver layer for cleansed data
3. Created County level Weather and Energy aggregated view for data analysis and feature engineering

Data Science | MLOps Model Pipeline

Registered Models

Share and serve machine learning models. Learn more

fp | Owned by me | Owner

Name	Catalog	Schema
fp-model-5	cscie103_catalog_final	gold
fp-model-10	cscie103_catalog_final	gold
fp-model-15	cscie103_catalog_final	gold
fp-model-14	cscie103_catalog_final	gold
fp-model-8	cscie103_catalog_final	gold
fp-model-7	cscie103_catalog_final	gold
fp-model-4	cscie103_catalog_final	gold
fp-model-13	cscie103_catalog_final	gold
fp-model-2	cscie103_catalog_final	gold
fp-model-3	cscie103_catalog_final	gold
fp-model-11	cscie103_catalog_final	gold
fp-model-6	cscie103_catalog_final	gold
fp-model-0	cscie103_catalog_final	gold

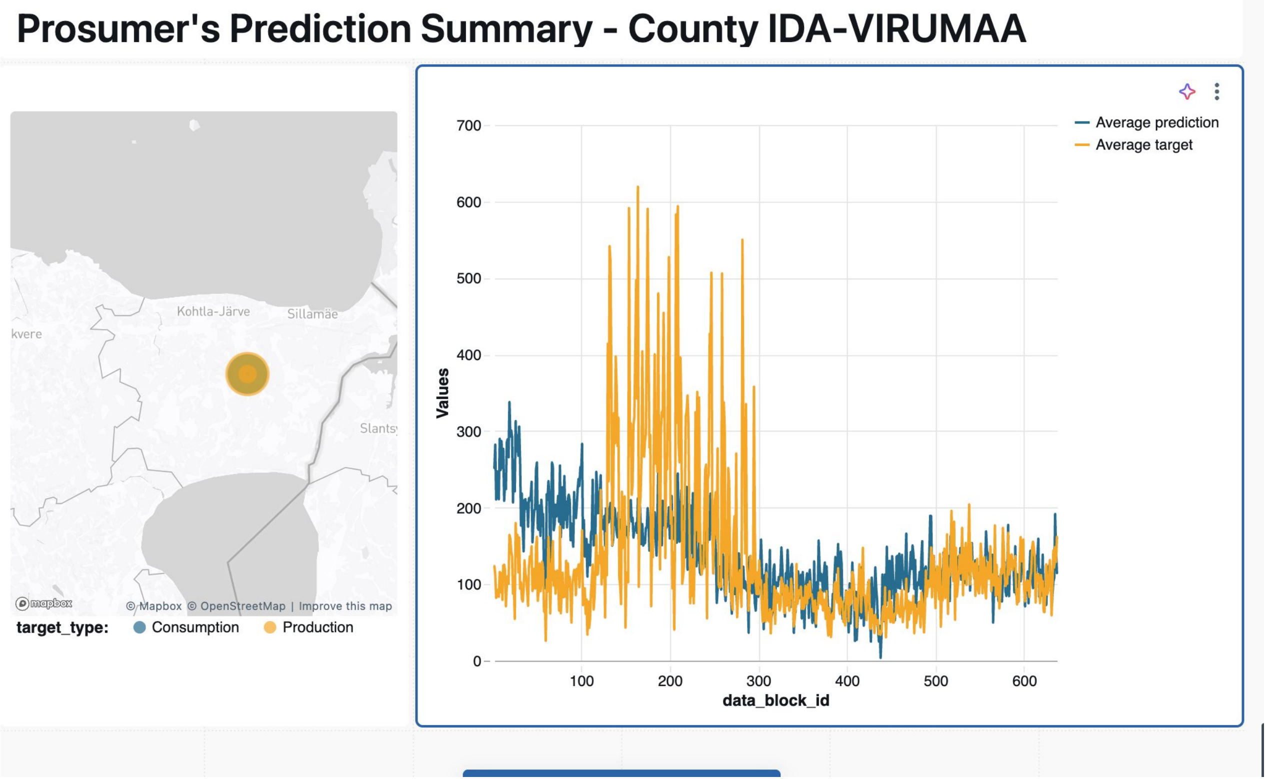
Key Streamlines:

1. **Model training by County based on findings**
2. **Feature Reduction and Selection**
3. **Use mlflow log model training input and output and save model in UC model registry**
4. **Retrieve model from registry for prosumer energy prediction**

Data Science | Model Evaluations

Model	County	Train	Test	RMSE	Rank
fp-model-1	HIIUMAA	85264	21548	50.792517	1
fp-model-6	LÄÄNEMAA	28392	7212	82.672502	2
fp-model-8	PÕLVAMAA	85264	21548	136.156688	3
fp-model-4	JÕGEVAMAA	136879	34311	198.750356	4
fp-model-7	LÄÄNEMAA	160629	40007	262.997524	5
fp-model-2	IDA-VIRUMAA	106998	26908	299.840072	6
fp-model-11	TARTUMAA	183526	45772	1042.838145	7
fp-model-0	HARJUMAA	198229	49319	1733.095094	8

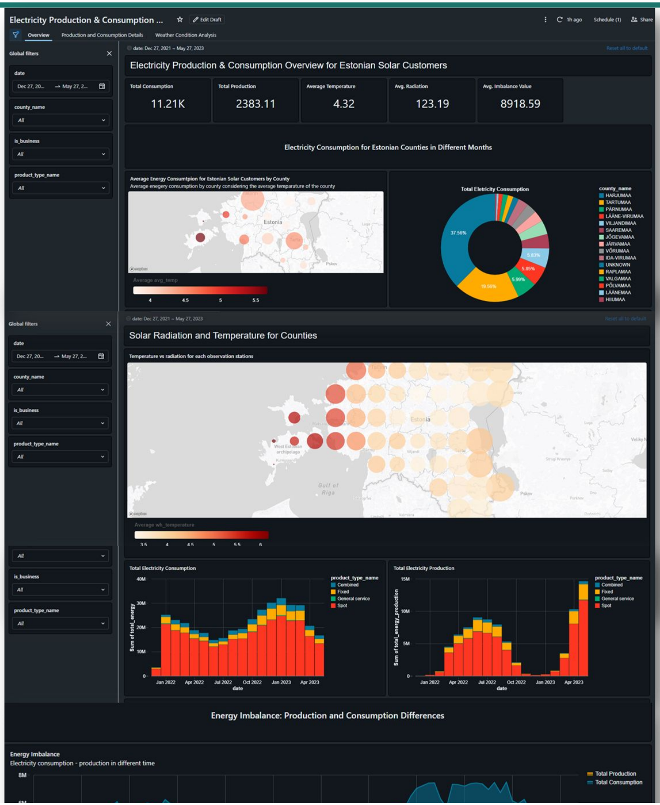
Data Science | Model Prediction Application



Key Observations:

1. Data block id is equivalent to datetime
2. The linear regression model is missing seasonality
3. Need to enhance leverage time series models
4. Setup Prediction pipeline. With mlflow we can load model and get predictions

BI Dashboard | Objectives



01. Identify key drivers of energy demand & supply.

Highlight the main factors influencing energy production and consumption, such as time (month/season), geographic conditions, county-level characteristics, product type, and residential vs. business usage.

02. Detect energy imbalances & actionable root causes.

Expose where and when gaps between production and consumption occur, analyze likely root causes, and indicate the specific counties and time periods where energy providers should prioritize corrective actions.

03. Geographic focus for targeted optimization

Visualize county-level conditions and performance to pinpoint high-risk or high-impact areas, helping stakeholders decide where to focus resources to reduce energy imbalance most effectively.

BI Dashboard | Key Functions

Data Ingestion and Integration

Queries curated data from the Gold layer of the data model.

Enriches core metrics with supporting datasets (e.g., customer attributes, environmental factors)

Why it matters

- Ensures analytics are built on clean, consistent, and business-ready data

Automated Data Refresh

Daily automatic refresh at 0:30

Consistent updates for all user groups

Why it matters

- User always access latest data
- Eliminate manual refresh and reduce operational risk

Data Governance & Security

Role Level Access Control
Off_shore county users: access data for county 1 and 10

Mainland_conty users: Access to all others

Why it matters

- Enforce data privacy and compliance
- Delivers relevant views

Business Insights & Decision Support

Analyze the electricity production, consumption and imbalance
Correlate trends

Why it matters

- Identify drivers of imbalance
- Support data-driven actions

Data Architecture | Governance, Security, Disaster Recovery

- Governance & Security
  - GDPR-compliance
  - Data validation and quality checks embedded throughout
  - Implement role-based access and least-privilege controls across layers
- Disaster recovery
  - Store all code in version control environment
  - For data
    - Durable & replayable storage
    - Cross-region replication for critical datasets
    - Checkpointing for streaming jobs

Data Architecture | Future Considerations

- If business comes back with streaming requirement
  - Kafka streams continuous meter, weather, and market price events
  - Spark Structured Streaming with Databricks/MLflow models for near–real-time predictions
  - Forecasts feed grid operations and alerts
    - all data persisted in Delta Lake for retraining and trends
- Future automation
  - Deploy infrastructure with Terraform
  - Later CI/CD set up via DABs with github actions