BLOOD GLUCOSE PREDICTION

Opencampus - Advanced Time Series

WiSe 24/25

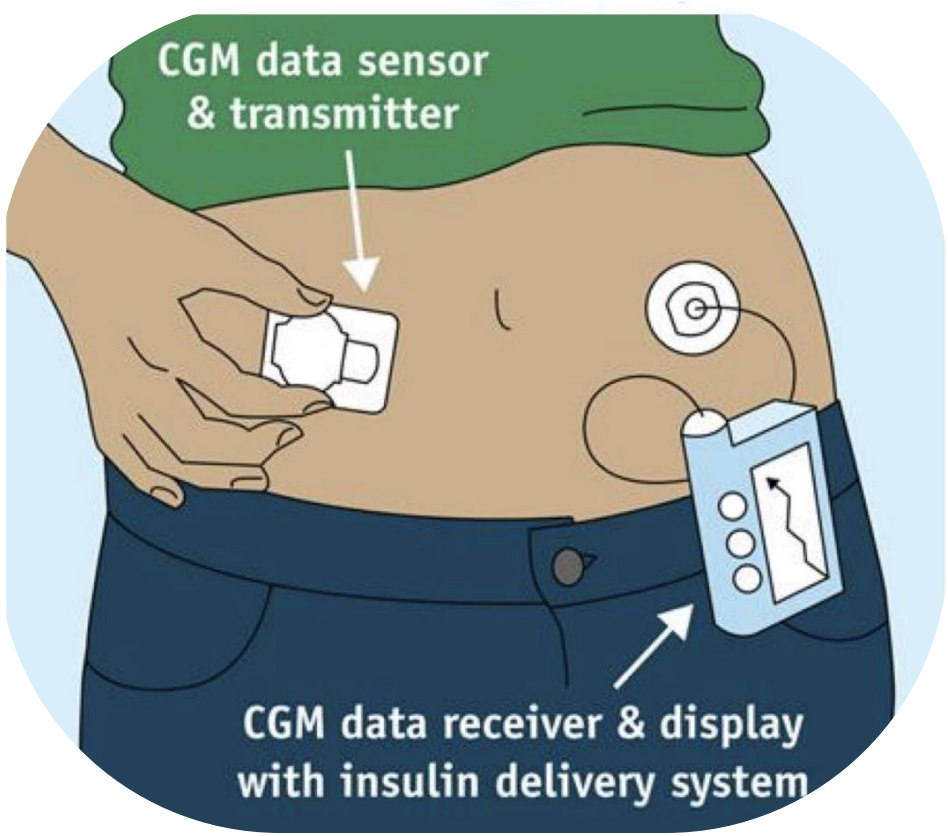
Anna Dahlhaus, Christopher Kunze,
Tim Oldörp, Leo Simak

OVERVIEW

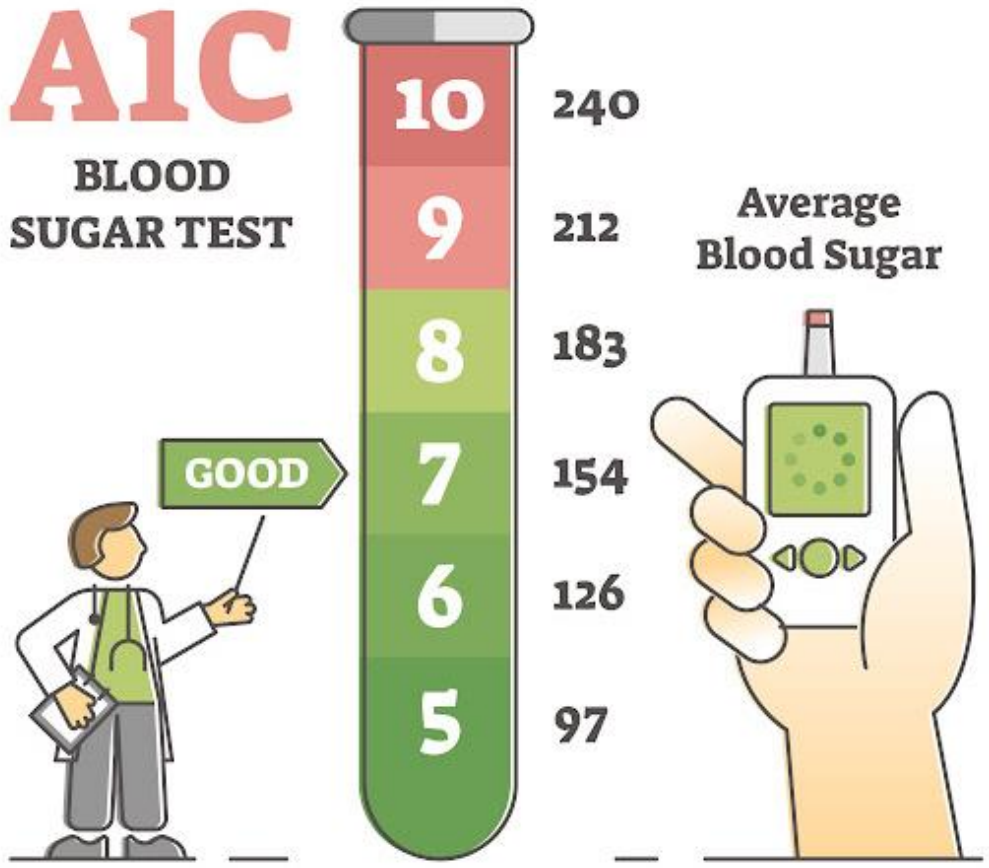
- Introduction & domain background
- HUPA-UCM Diabetes Dataset
  - EDA
  - Problems and solutions
  - Feature engineering
- Models
  - Baseline
  - ARIMA
  - Tree models: Random Forest & XGBoost
  - Neural networks: GRU, LSTM, N-Hits, Chronos
- Result overview
- Discussion & Outlook

INTRODUCTION

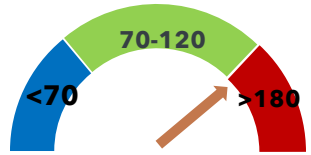
- **Task**:
  - Predict future blood glucose values
- **Dataset**: HUPA-UCM Diabetes Dataset:
  - 25 people with Type 1 Diabetes Mellitus (T1DM)
  - Continuous Glucose Monitoring (CGM) data over several days (>7d)
  - Fitness tracker data: Steps, Heart rate, Burned calories, (sleep)
  - Carbohydrate intake & Insulin dosages (Basis + Bolus)



BASIC DOMAIN KNOWLEDGE
BLOOD GLUCOSE

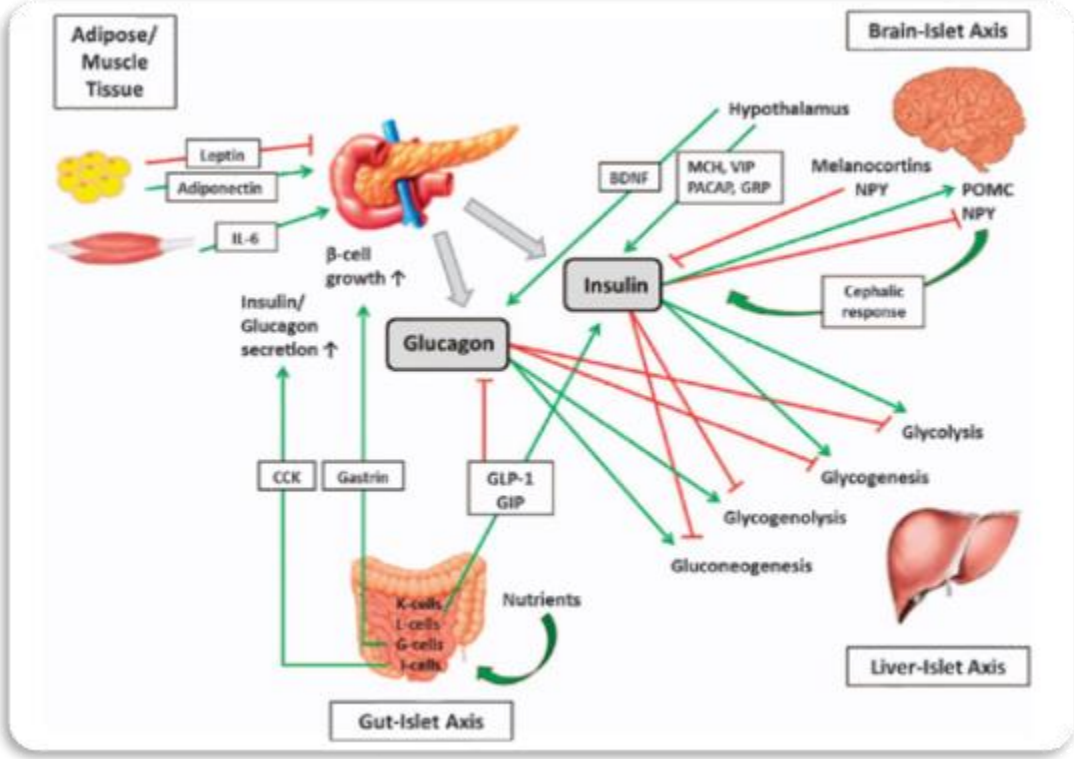


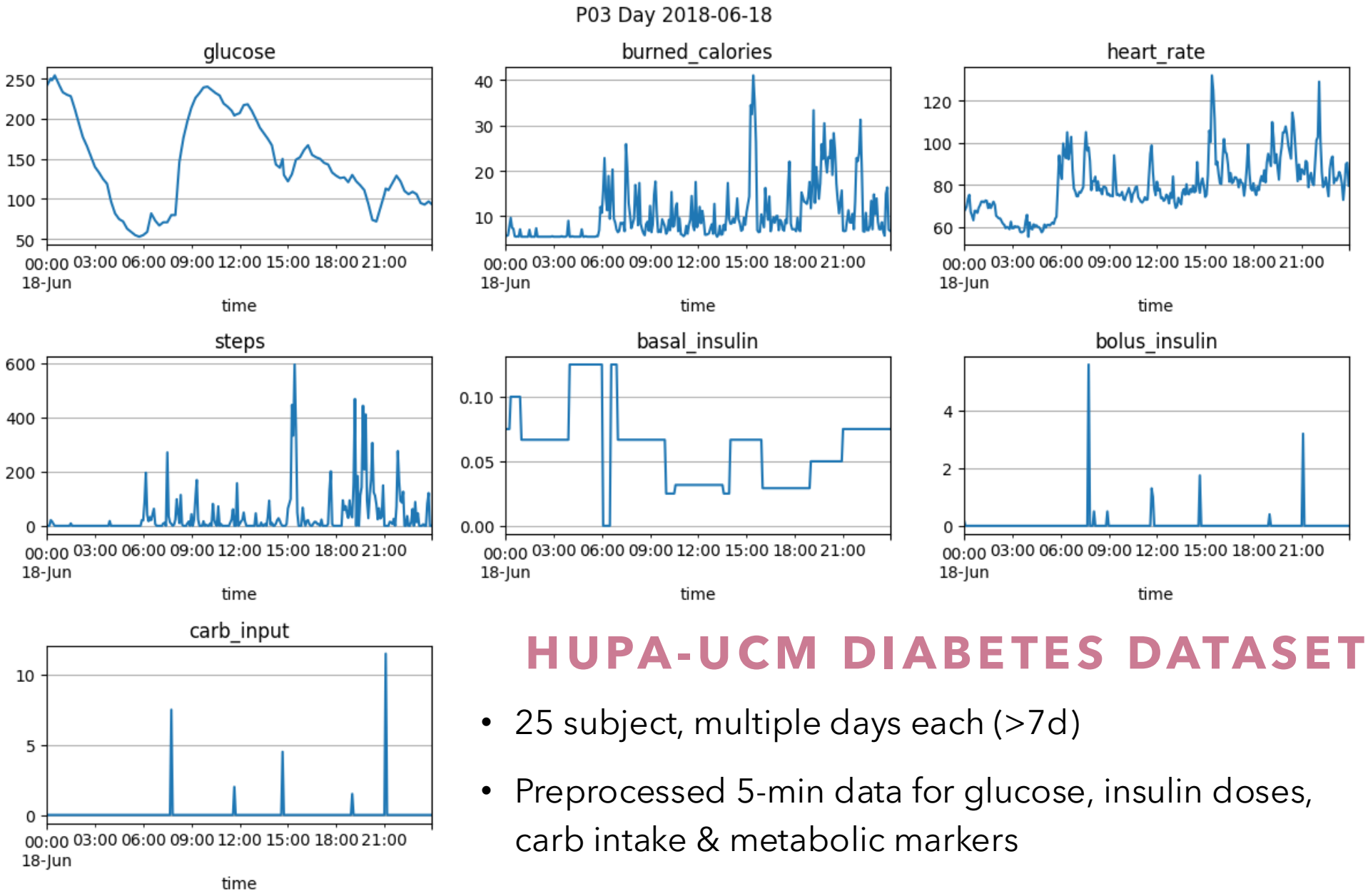
- **Blood Glucose** (BG)
  - Glucose ($C_6H_{12}O_6$.) is a simple sugar (monosaccharide) that serves as a primary source of energy for the body's cells.
  - Levels of glucose in the bloodstream is regulated by the two hormones Insulin (BG↓) and Glucagon (BG ↑) released in the pancreas
- **Blood Glucose Levels in T1D**
  - Normal fasting blood glucose:
    - 70–100 mg/dL (3.9–5.6 mmol/L).
  - *Hyperglycemia* (high blood sugar):
    - >180 mg/dL (10 mmol/L).
  - *Hypoglycemia* (low blood sugar):
    - <70 mg/dL (3.9 mmol/L).



TYPE 1 DIABETES MELLITUS (T1DM)

- Autoimmune disease : Immune system destroys the insulin-producing beta cells in the pancreas
- Without insulin, the body cannot regulate blood glucose levels effectively
- Therapy:
  - Administering insulin (via injections or pumps):
    - Basal insulin: longterm basis
    - Bolus insulin: short term before meals
  - Monitoring blood glucose levels
  - Balancing carbohydrate intake, physical activity, and medication
- Regular monitoring helps prevent complications like heart disease, nerve damage, and kidney problems.





HUPA-UCM DIABETES DATASET

- 25 subject, multiple days each (>7d)
- Preprocessed 5-min data for glucose, insulin doses, carb intake & metabolic markers

EXPLORATORY DATA ANALYSIS

```
P01:  14 days (1.3%)
P02:  11 days (1.0%)
P03:  13 days (1.2%)
P04:  11 days (1.0%)
P05:  13 days (1.2%)
P06:   7 days (0.7%)
P07:  13 days (1.2%)
P09:  13 days (1.2%)
P10:  10 days (0.9%)
P11:  13 days (1.2%)
P14:  13 days (1.2%)
P15:  13 days (1.2%)
P16:  13 days (1.2%)
P17:  12 days (1.1%)
P18:  13 days (1.2%)
P19:  12 days (1.1%)
P20:   9 days (0.8%)
P21:   8 days (0.8%)
P22:  13 days (1.2%)
P23:  13 days (1.2%)
P24:  10 days (0.9%)
P25:  13 days (1.2%)
P26: 140 days (13.2%)
P27: 573 days (54.0%)
P28:  89 days (8.4%)
```

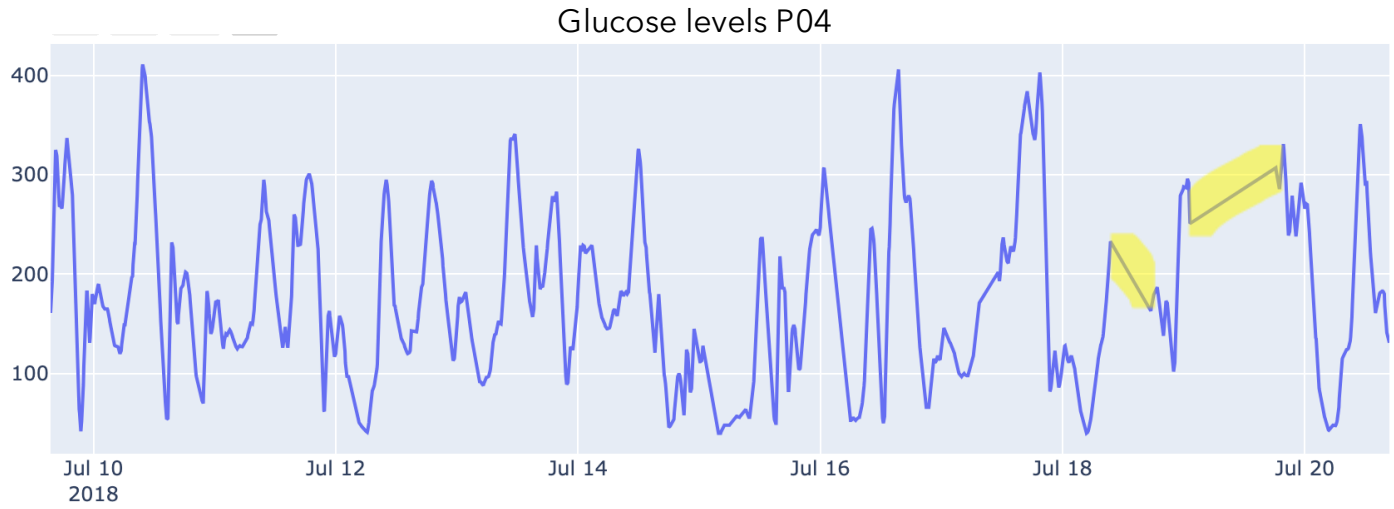
- Vastly imbalanced data between subjects!
  - > 50% of the dataset is just one subject: P27
  - ~75% is from P26, P27 or P28
  - All other just contribute ~1% each

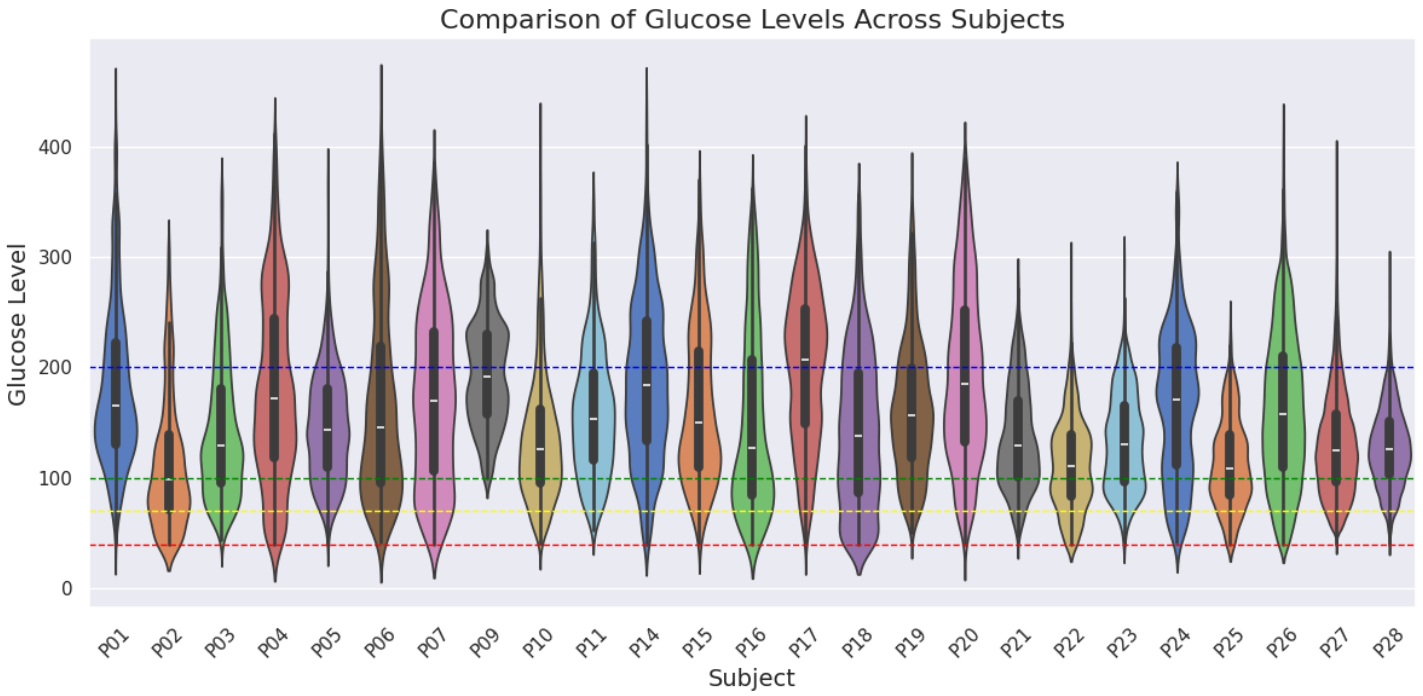
- Data quality is mixed 😕
  - Missing data:
    e.g. P15 has no tracking of carb_input nor bolus_insulin
  - Carb_input has different scales between subjects (μ = 8 vs. μ=281)

	carb_input			bolus_insulin		
	mean	min	max	mean	min	max
subj						
P01	8.0	4.0	12.0	11.0	4.0	15.0
P02	53.0	0.0	135.0	19.0	8.0	34.0
P04	168.0	0.0	320.0	40.0	14.0	49.0
P15	0.0	0.0	0.0	0.0	0.0	0.0
P22	281.0	271.0	301.0	28.0	22.0	37.0

TARGET: GLUCOSE

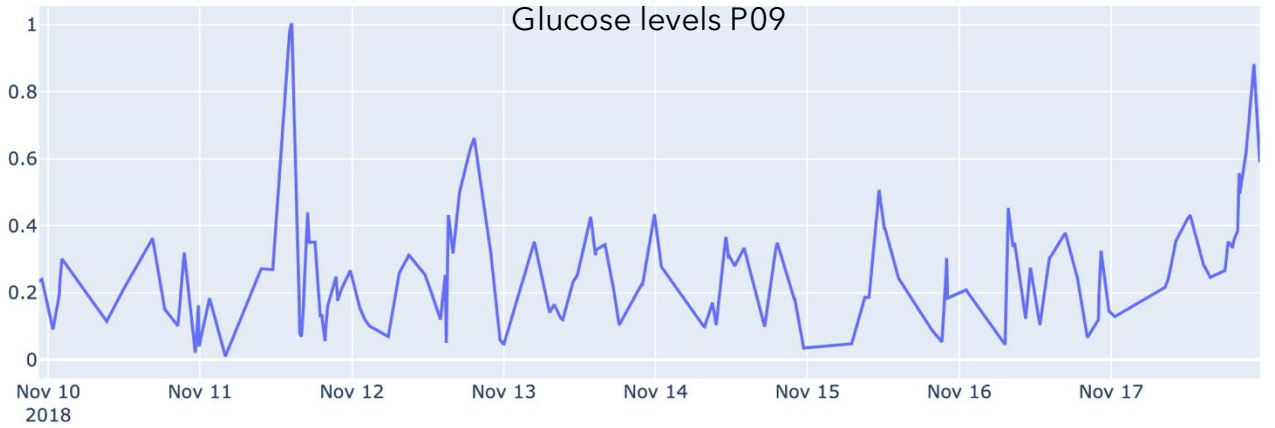
- Severe cases of hypoglycemia and hyperglycemia present
- Missing data was linearly interpolated
- Big interindividual differences in therapeutic success

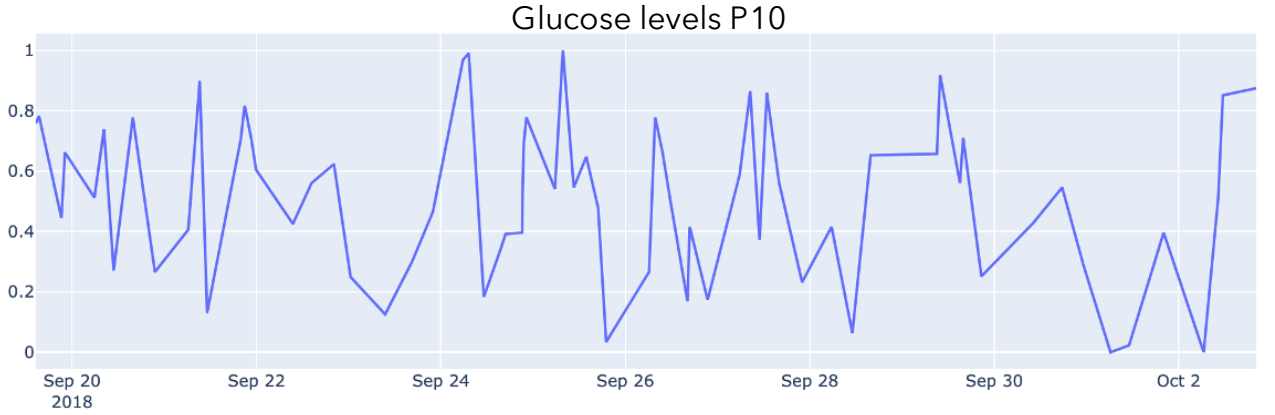


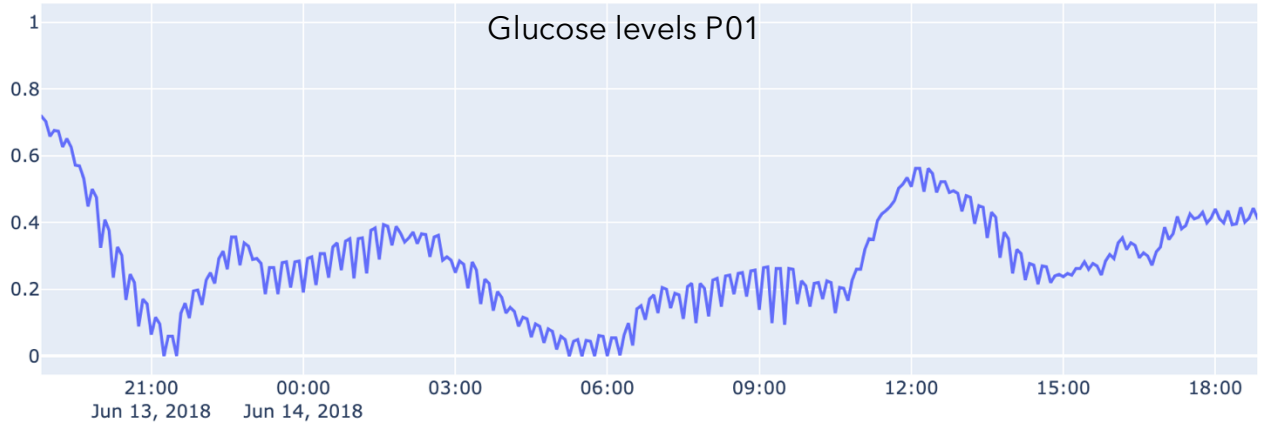


PROBLEMS FOUND

- 2 subjects (P09 & P10) have extremely low measurement frequency
  - → Bad quality!
  - ➔ Delete from dataset
- Subject P01 has weird ripple effect
  - → interpolation artifact?
  - ➔ MA-smoothed

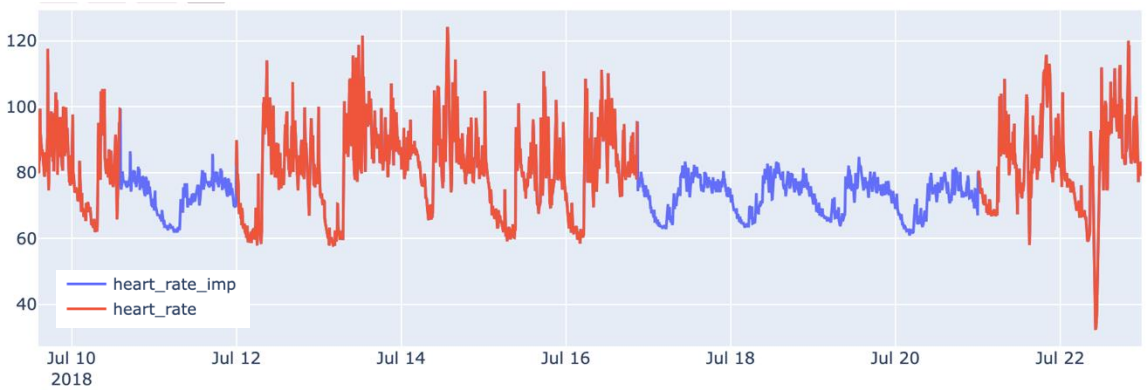


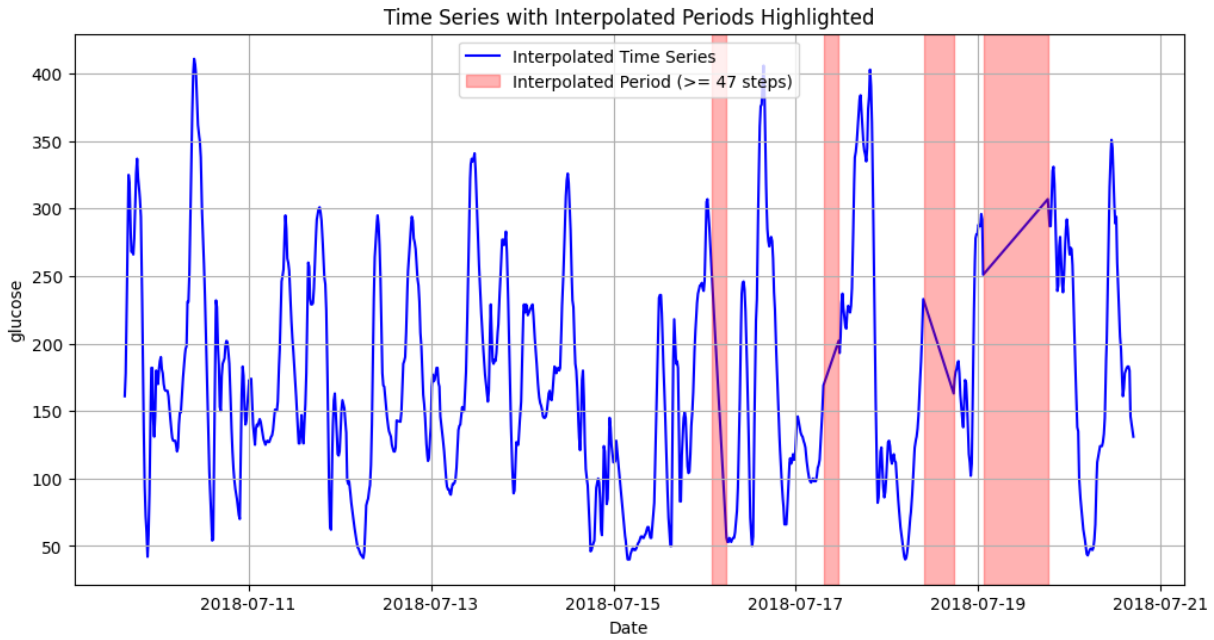


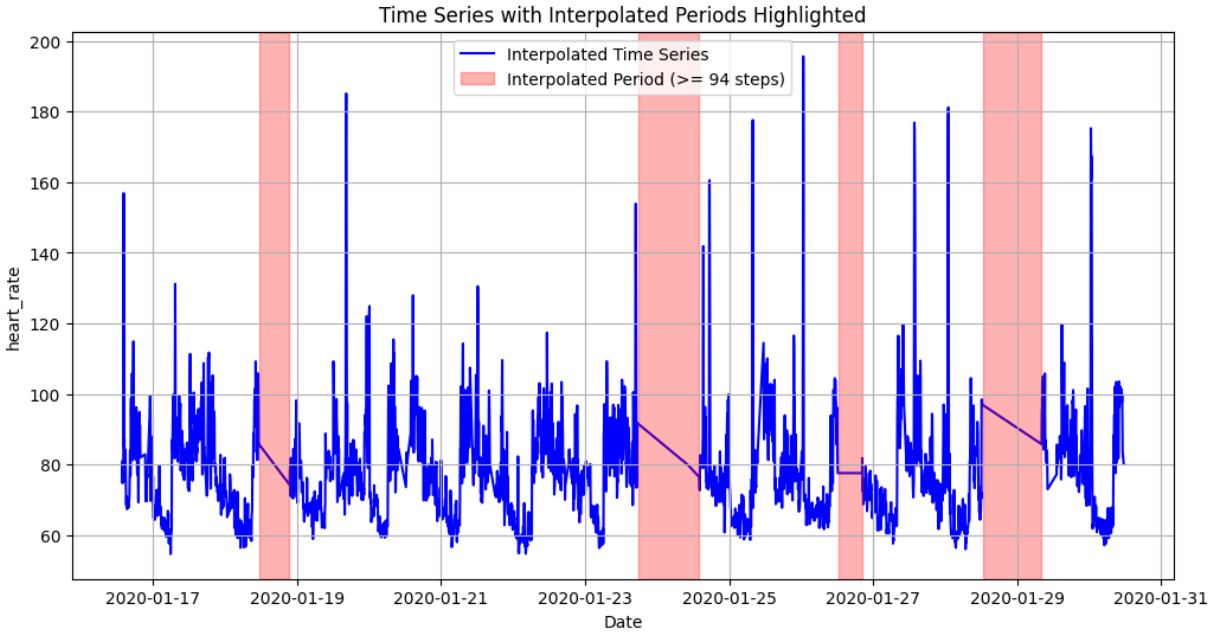


INTERPOLATED DATA

- Missing data were not marked but either set to zero (carb, insulin, steps, calories) or linearly interpolated (heart rate, glucose)
- Detected using smoothing threshold (2nd derivative)
- *"Multiple Imputation with Chained Equations"* was used to impute missing heart rate measurements
- Glucose imputation: Mean value

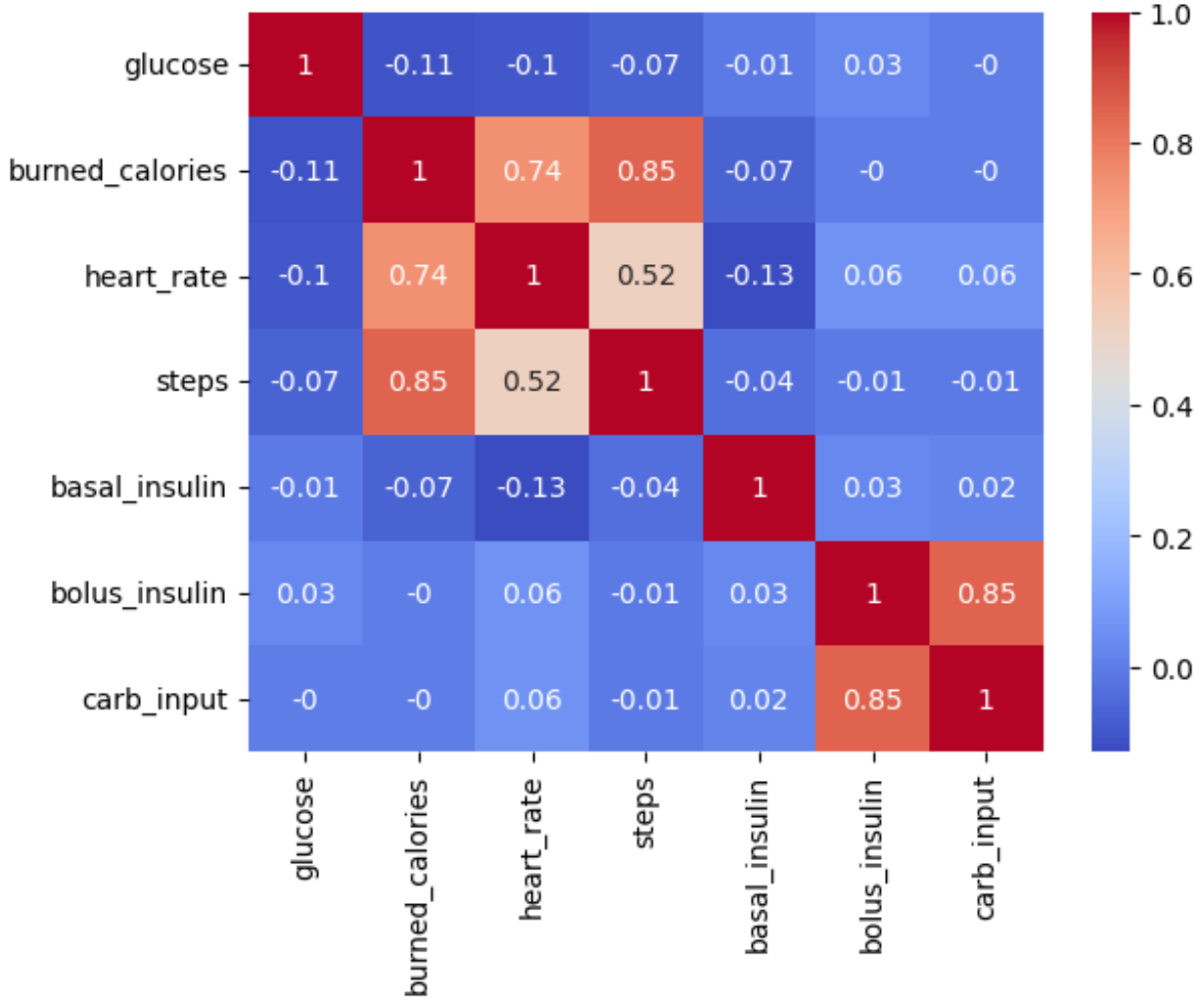






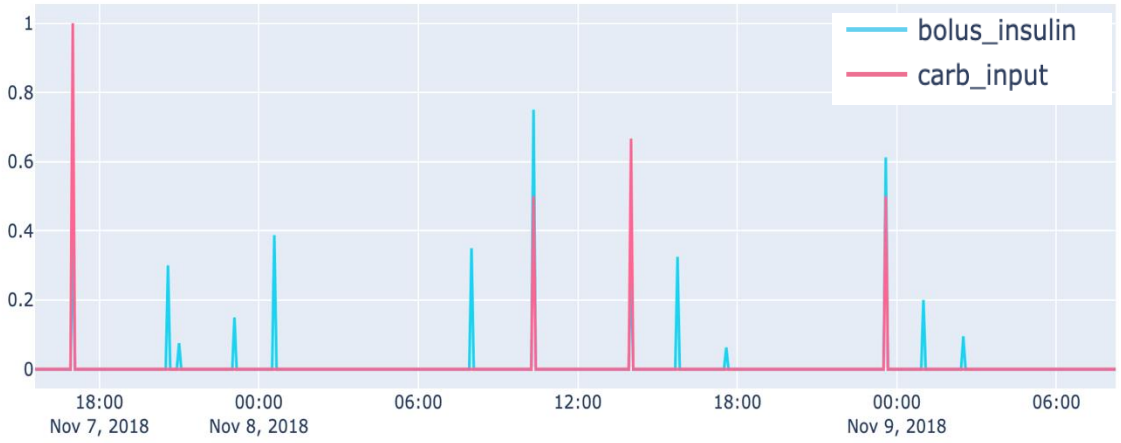
FEATURE CORRELATION

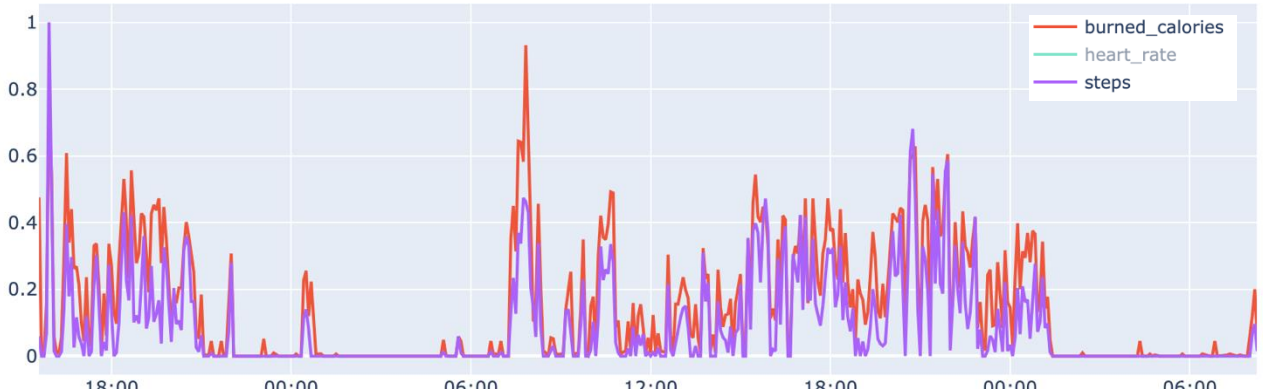
- Some features are strongly correlated
  - Bolus insulin is given before meals and is thus dependent on carb input
  - Burned calories, heart rate and steps were all measured using a fitness tracker. Burned calories were properly approximated using heart rate and steps
- Only linear dependence captured



FEATURE SELECTION

- Carb input removed
  → Strong correlation with bolus insulin but worse data quality
- Steps removed
  → Strong correlation with burned calories (basis of approximation)
- New feature time_of_day: sin/cos encoded
  → To capture circadian rhythm





EVALUATION

Task 1 - Predict the glucose level 1h in advance for one subject

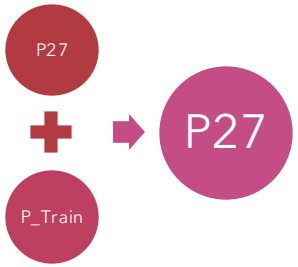
- 1a: Training on the same subject's data
- 1b: Training augmented with other subjects' data

Task 2 - Predict the glucose level 1h in advance for multiple subject

- 2a: Training on one other subject's data
- 2b: Training on other subjects' data

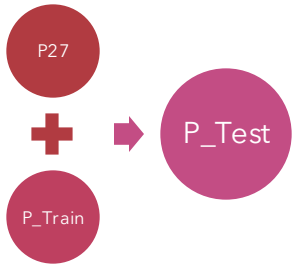
```
P01:  14 days (1.3%)
P02:  11 days (1.0%)
P03:  13 days (1.2%)
P04:  11 days (1.0%)
P05:  13 days (1.2%)
P06:   7 days (0.7%)
P07:  13 days (1.2%)
P09:  13 days (1.2%)
P10:  10 days (0.9%)
P11:  13 days (1.2%)
P14:  13 days (1.2%)
P15:  13 days (1.2%)
P16:  13 days (1.2%)
P17:  12 days (1.1%)
P18:  13 days (1.2%)
P19:  12 days (1.1%)
P20:   9 days (0.8%)
P21:   8 days (0.8%)
P22:  13 days (1.2%)
P23:  13 days (1.2%)
P24:  10 days (0.9%)
P25:  13 days (1.2%)
P26: 140 days (13.2%)
P27: 573 days (54.0%)
P28:  89 days (8.4%)
```

Task 1a



Task 1b

Task 2a



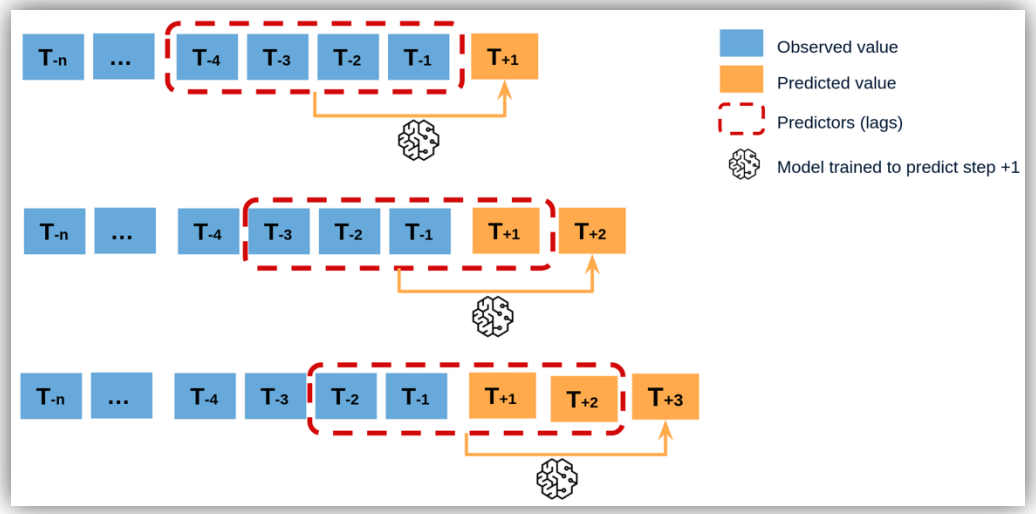
Task 2b

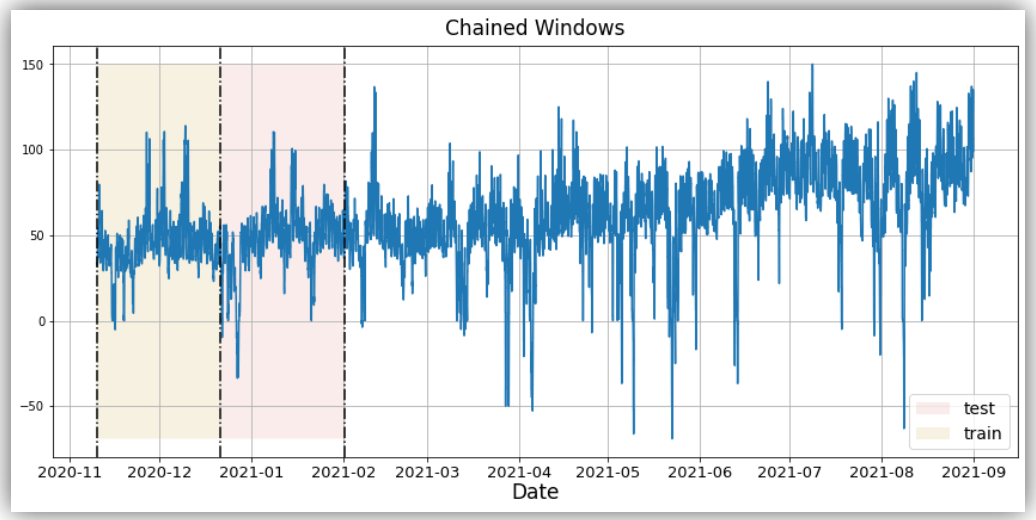
EVALUATION RULES

1. Single evaluation on test set
2. Evaluation using chained windows / rolling windows / autoregression
3. No further parameter updates
4. Metrics:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}} \qquad MAPE = 100\frac{1}{N}\sum_{t=1}^{N}\left|\frac{(x_i - \hat{x}_i)}{\hat{x}_i}\right|$$

∞





BASELINE MODEL: (SEASONAL)NAIVE

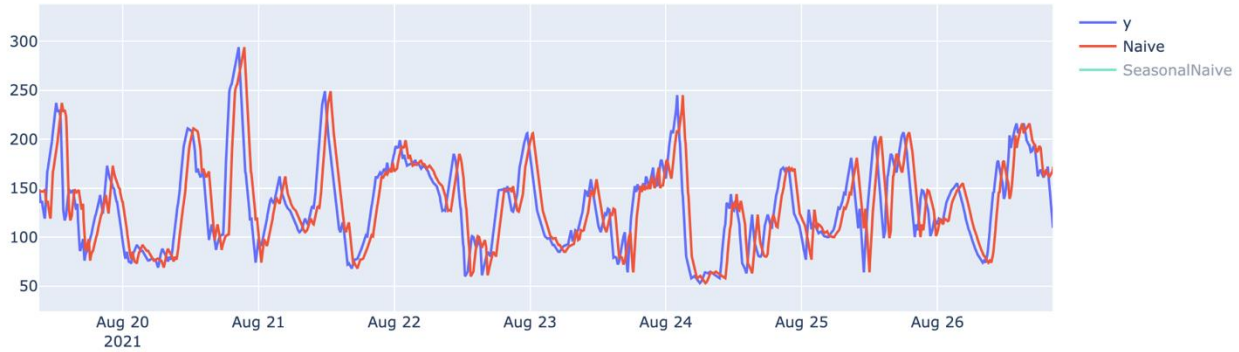
- Naïve model - All forecasts have the value of the last observation (aka shift data by 1h)
- Season naïve - All forecasts have the value shifted by one season (e.g. 24h)

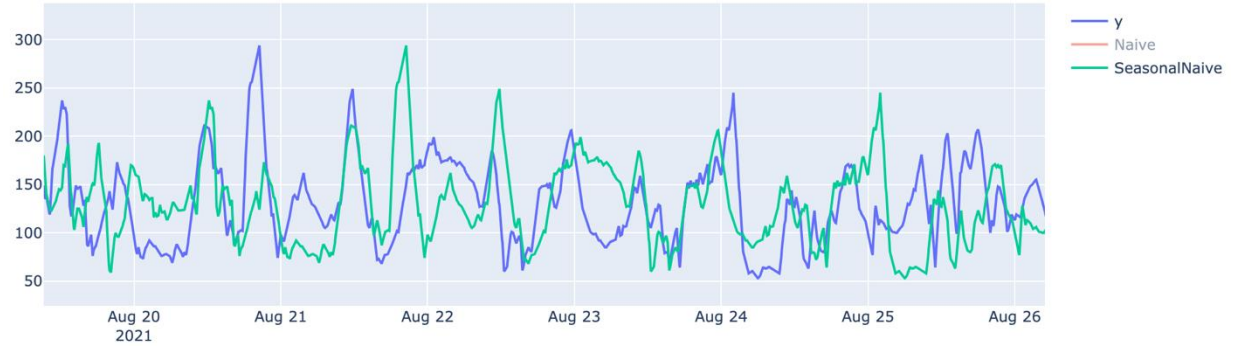
Naive	RMSE	MAPE
Task 1*	28.2	16.9%
Task 2*	48.2	25.2%

*no difference between a and b

S-Naive	RMSE	MAPE
Task 1*	63.0	42.5%
Task 2*	86.0	51.2%

*no difference between a and b





NEXT STEP

Maybe using several previous values?

Different weighting?

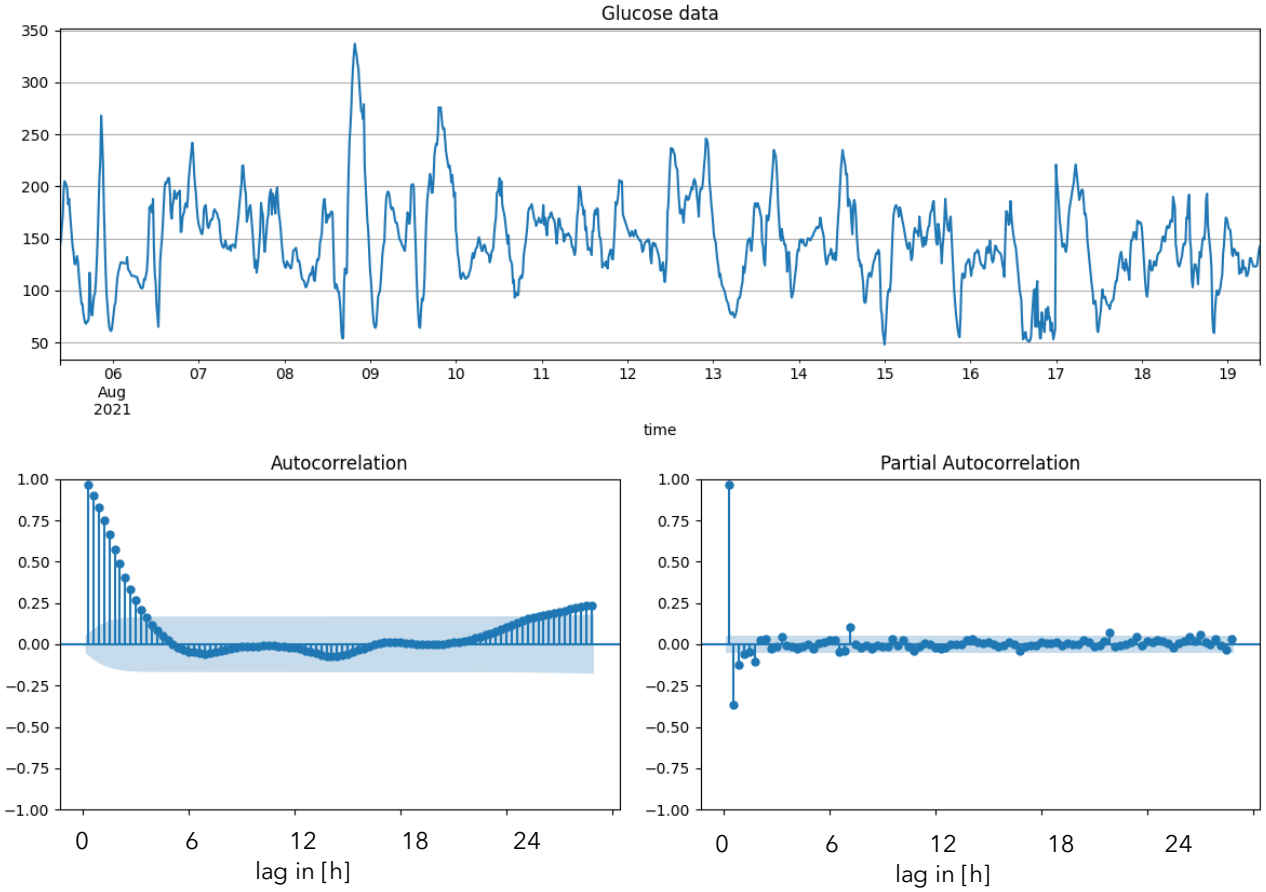
Smoothing?

Combinging lags and seasonality?

➔ That's an (S)ARIMA model ;)

ARIMA & SARIMA

- ADF-Test: p<0.000 → no unit-root
- 2 big PAC-components
- ACF properly mainly propagation of the autocorrelation at lag-1 and lag-2
- Check different orders:
  - e.g. (2,0,0) -- (2,0,1) -- (3,0,0) -- (1,1,2)
- Try to include a seasonality:
  - One day (24h) or half a day(12h)



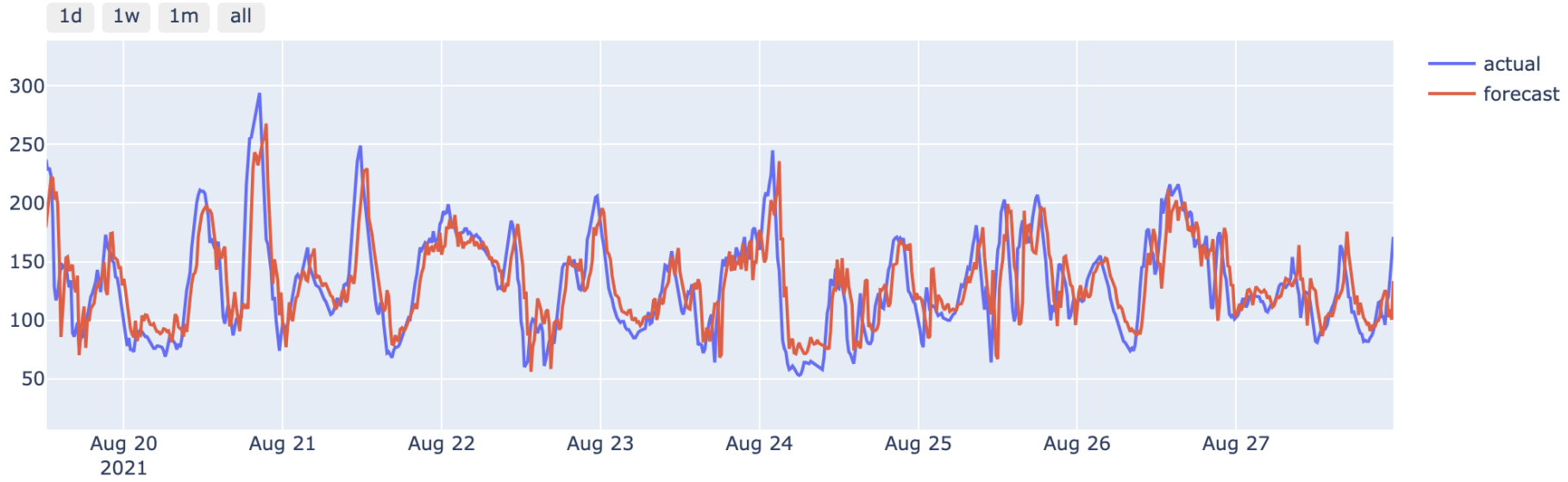
ARIMA - RESULTS

- Best model:
  - Order: (2,0,1) x (0,0,1,12h)
- Worse than naïve?!

SARIMAX Results

Dep. Variable:	y	No. Observations:	2688
Model:	ARIMA(2, 0, 1)x(0, 0, 1, 48)	Log Likelihood	-10078.754
Date:	Tue, 28 Jan 2025	AIC	20169.507
Time:	10:24:25	BIC	20204.887
Sample:	0	HQIC	20182.304
	- 2688		

Naive	RMSE	MAPE
Task 1*	25.9	17.4%
Task 2*		



ARIMA - DETAILS

- Not an optimal fit:
  - Non-normally distributed error
  - Seasonality not significant
  - Big spikes result of interpolation?
  - Does not include exogenous factors / covariates

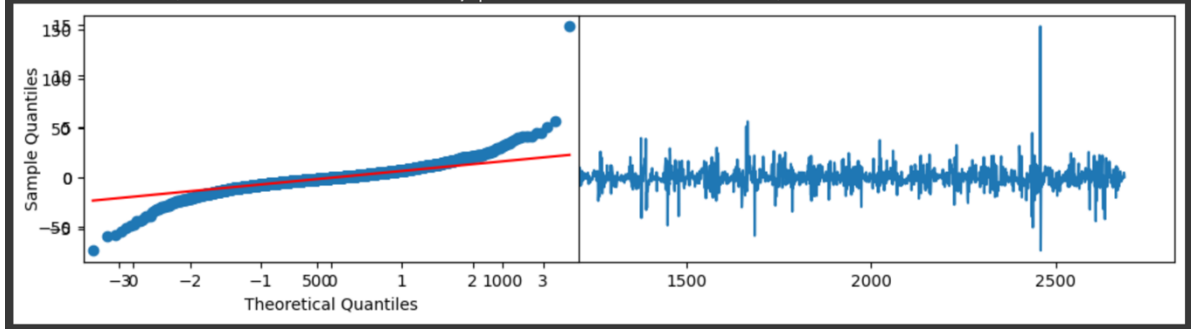
SARIMAX Results

Dep. Variable:	y	No. Observations:	2688
Model:	ARIMA(2, 0, 1)x(0, 0, 1, 48)	Log Likelihood	-10078.754
Date:	Tue, 28 Jan 2025	AIC	20169.507
Time:	10:24:25	BIC	20204.887
Sample:	0 - 2688	HQIC	20182.304
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	139.1276	4.942	28.153	0.000	129.442	148.813
ar.L1	1.5012	0.034	44.445	0.000	1.435	1.567
ar.L2	-0.5367	0.033	-16.276	0.000	-0.601	-0.472
ma.L1	-0.1644	0.036	-4.591	0.000	-0.235	-0.094
ma.S.L48	0.0146	0.021	0.697	0.486	-0.027	0.056
sigma2	105.6341	0.998	105.825	0.000	103.678	107.591

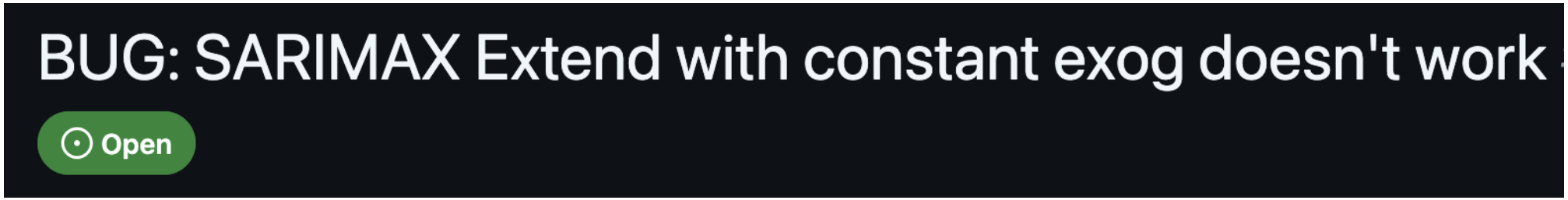
Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	57575.31
Prob(Q):	0.96	Prob(JB):	0.00
Heteroskedasticity (H):	1.00	Skew:	1.11
Prob(H) (two-sided):	0.98	Kurtosis:	25.56

Durbin-Watson: 1.9898946165980989
NormaltestResult(statistic=1044.4683996606436, pvalue=1.5724207004064284e-227)



ARIMA-X: USING EXOGENOUS FEATURES?

- Some technical problems....



ARIMA - CODE SNIPPETS

```python
def rolling_forecast_eval(model, y, n_steps=1, refit=False):
    forecasts = []
    # Save initial forecast
    forecasts.extend(model.forecast(steps=n_steps))

    # Step through the rest of the sample
    for t in range(len(y) - n_steps):
        # Update the results by appending the next observation
        if refit:
            model = model.append(y.values[t:t+1], refit=True)
        else:
            model = model.extend(y.values[t:t+1])

        # Save the next forecast
        forecasts.append(model.forecast(steps=n_steps, )[-1])

    # Combine all forecasts into a Series
    forecasts = pd.Series(forecasts, index=y.index, name=y.name)
    scores = {
        'rmse': mean_squared_error(y[n_steps-1:], forecasts[n_steps-1:]),
        'mape': mean_absolute_percentage_error(y[n_steps-1:], forecasts[n_steps-1:]),
    }
    return forecasts, scores
```

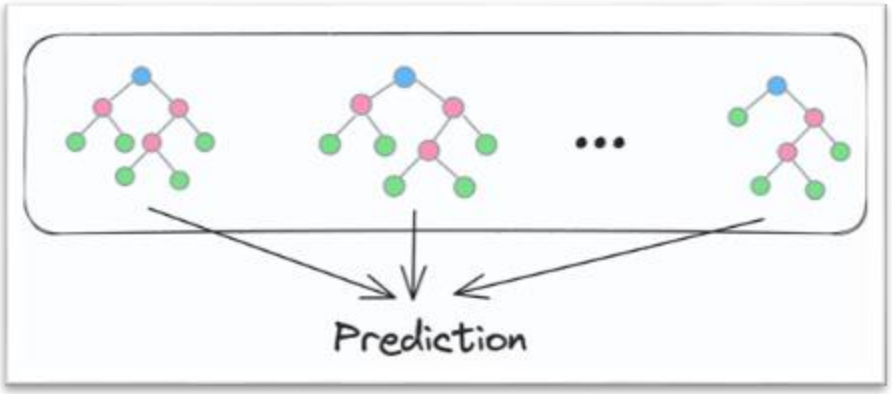
```python
#Check for non-existence of a unit-root
glucose = df_train['glucose']
print(f'ADF test(glucose): p-Value={sm.tsa.stattools.adfuller(glucose)[1]:.4f}')

# plot ACF & PCF
plt.figure(figsize=(12, 8))
plt.subplot(2, 1, 1)
glucose.plot(grid=True)
plt.title('Glucose data')
ax = plt.subplot(2, 2, 3)
sm.graphics.tsa.plot_acf(glucose, zero=False, lags= one_day, ax=ax);
ax = plt.subplot(2, 2, 4)
sm.graphics.tsa.plot_pacf(glucose, zero=False, lags= one_day, ax=ax);
```

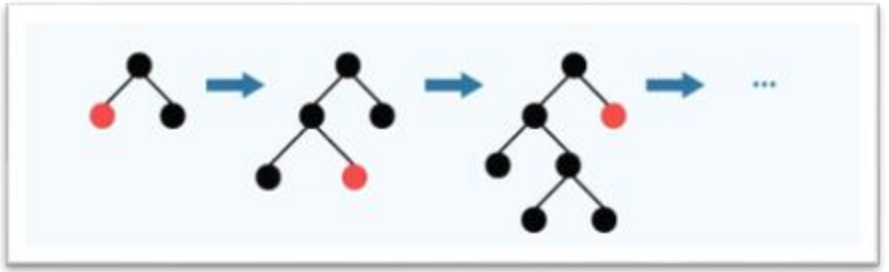
```python
X = df_train['glucose'].iloc[-4*one_week:]
y = df_test['glucose']

order = (2,0,1)
model = ARIMA(X.values, order=order, seasonal_order=(0,0,1,one_day))
model_fit = model.fit()

forecasts, scores = rolling_forecast_eval(model_fit, y, n_steps=one_hour)

display(scores)
print(model_fit.summary())
```

- Implemented using statsmodels

REGRESSION TREE ENSEMBLES BOOSTING & BAGGGING

- Why?
  1. Handle non-linear relationships well
  2. Provide feature importance rankings
  3. Less prone to overfitting due to ensemble nature
  4. Work well with numerical time series data
- Feature engineering:
  - Time features:
    - time of day (sin/cos), day of week, month
  - Lag features:
    - 6h lag
    - Glucose level, heart rate, insulin





RANDOM FOREST

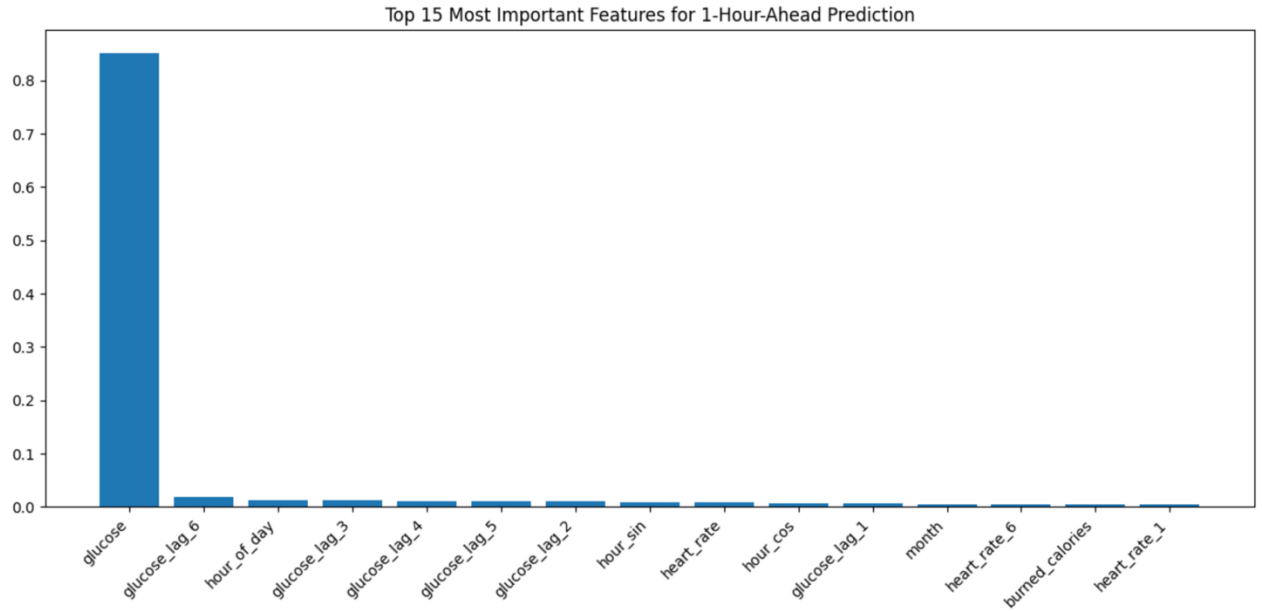
- GridsearchCV:

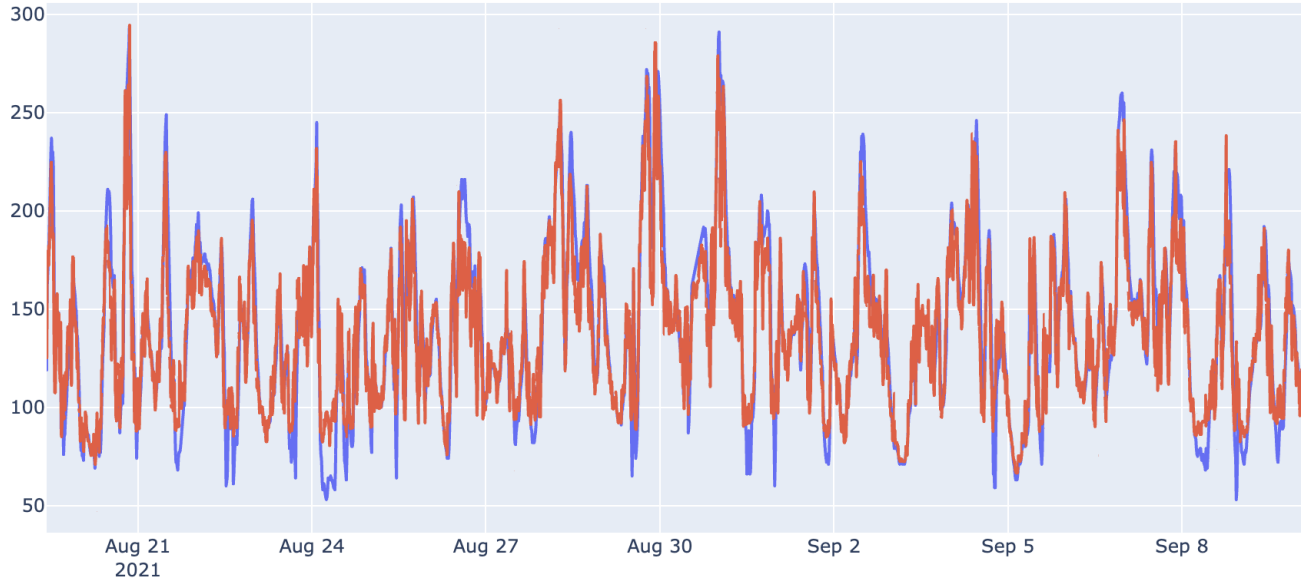
```
param_grid = {
    'n_estimators': [25, 50, 100],
    'max_depth': [5, 10],
}
```

- Best parameters:

{'max_depth': 10, 'n_estimators': 50}

RF	RMSE	MAPE
Task 1a	16.8	11.2%





RANDOM FOREST -CODE SNIPPETS

- Feature engineering via pandas
- Ordinary sklearn pipeline for training

```
df['month'] = df['old_time'].dt.month
df['day_of_week'] = df['old_time'].dt.dayofweek

df['total_insulin'] = df['basal_insulin']+df['bolus_insulin']

# Add lag features
for k in range(one_hour//2):
    df[f'glucose_lag_{k+1}'] = df.groupby('subj')['glucose'].shift(k+1)
    df[f'heart_rate_{k+1}'] = df.groupby('subj')['heart_rate'].shift(k+1)
    df[f'insulin_{k+1}'] = df.groupby('subj')['total_insulin'].shift(k+1)
```

```
1 from sklearn.ensemble import RandomForestRegressor
2
3 # Define parameter grid for Random Forest
4 param_grid = {
5     'n_estimators': [25, 50, 100],
6     'max_depth': [5, 10],
7 }
8
9 # Grid Search
10 rf_model = RandomForestRegressor(random_state=42)
11 cv = TimeSeriesSplit(n_splits=5)
12 grid_search = GridSearchCV(rf_model, param_grid, cv=cv, scoring='neg_mean_squared_error')
13 grid_search.fit(X, y)
14
15 print("\nBest parameters:", grid_search.best_params_)
16 print("Best MSE:", -grid_search.best_score_)
17
18
19 # Get the best model
20 final_model = grid_search.best_estimator_
```

XGBOOST

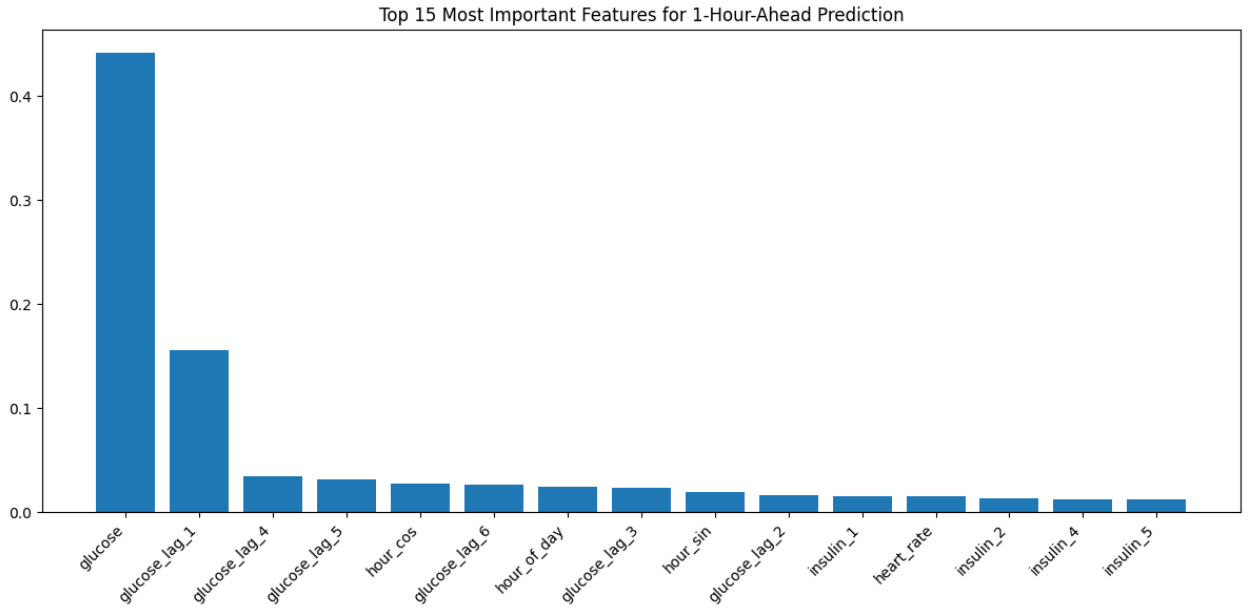
- GridsearchCV:

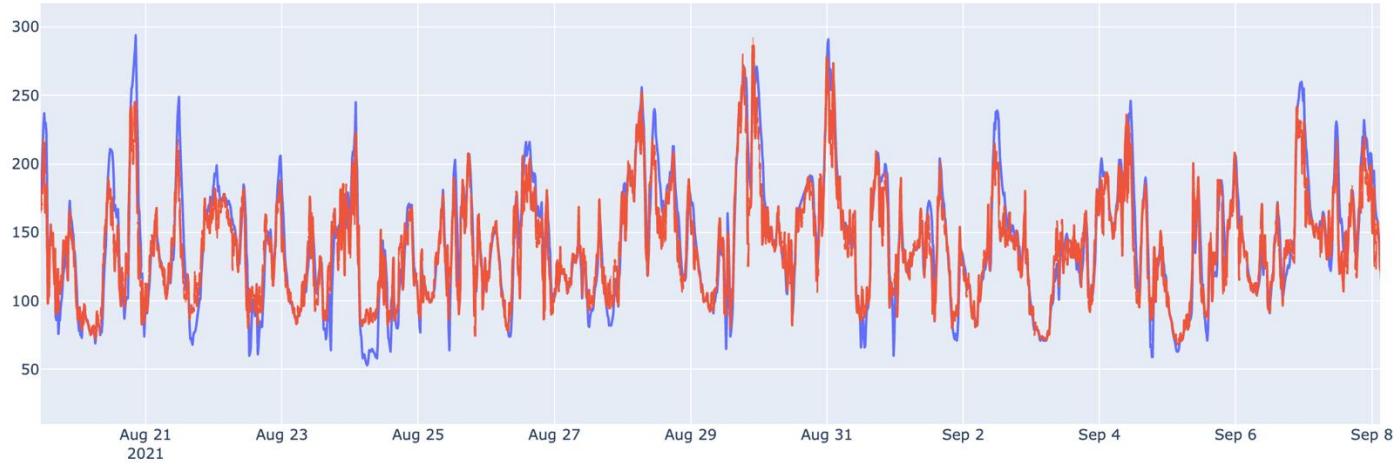
```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
```

- Best parameters:

{'colsample_bytree': 0.8, 'max_depth': 5, 'n_estimators': 200, 'subsample': 0.8}

RF	RMSE	MAPE
Task 1a	17.6	11.9%





XGBOOST – CODE SNIPPETS

- xgboost library → sklearn model

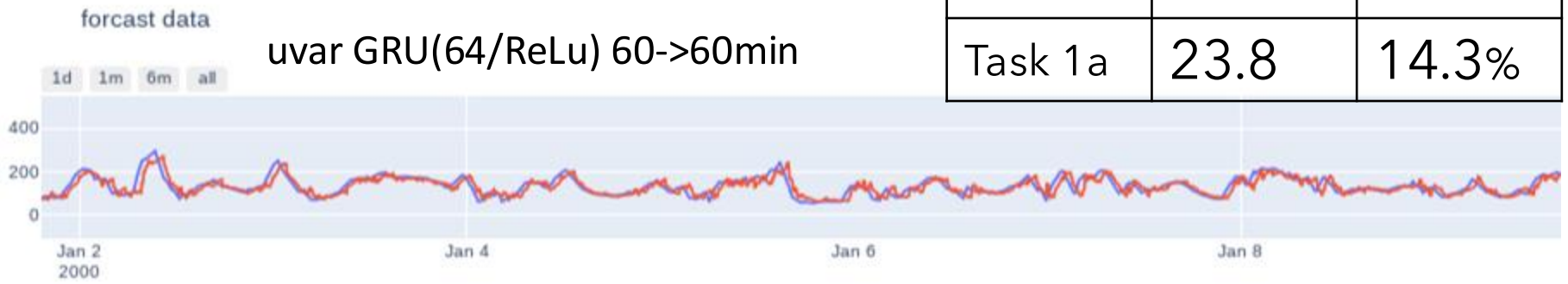
```python
from xgboost.sklearn import XGBRegressor

xgb_model = XGBRegressor(
    objective='reg:squarederror',
    n_estimators=100,
    learning_rate=0.1,
    max_depth=5,
    random_state=42
)

# Define parameter grid for XGBoost
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

# Perform Grid Search
cv = TimeSeriesSplit(n_splits=5)
grid_search = GridSearchCV(
    estimator=xgb_model,
    param_grid=param_grid,
    cv=cv,
    scoring='neg_mean_squared_error',
    verbose=2,
    n_jobs=-1
)
# Fit the model
grid_search.fit(X, y)

# Get the best model
final_model = grid_search.best_estimator_

print("Best parameters:", grid_search.best_params_)
print("Best score:", np.sqrt(-grid_search.best_score_))
```

RECURRENT NEURAL NETWORKS

- GRU

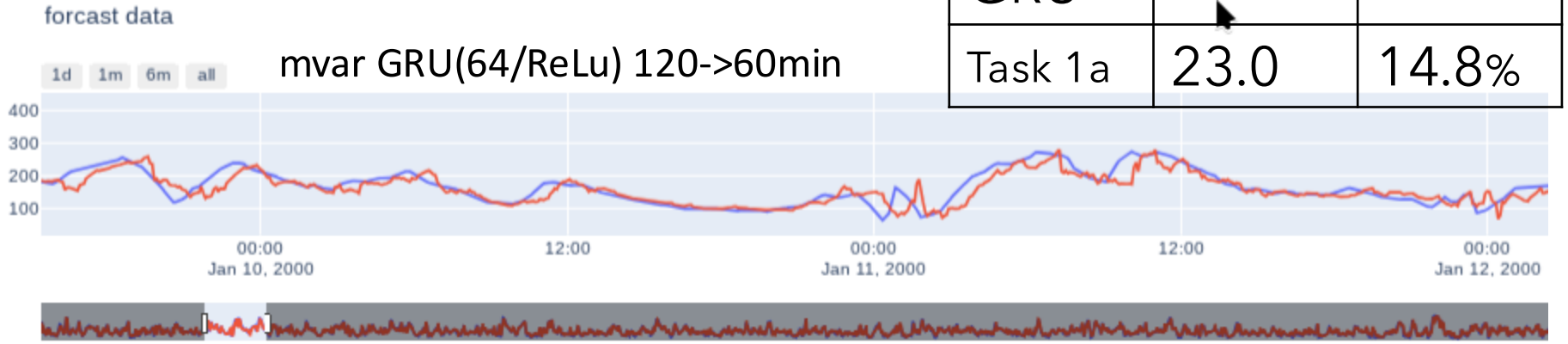
uvar GRU(64/ReLu) 60->60min

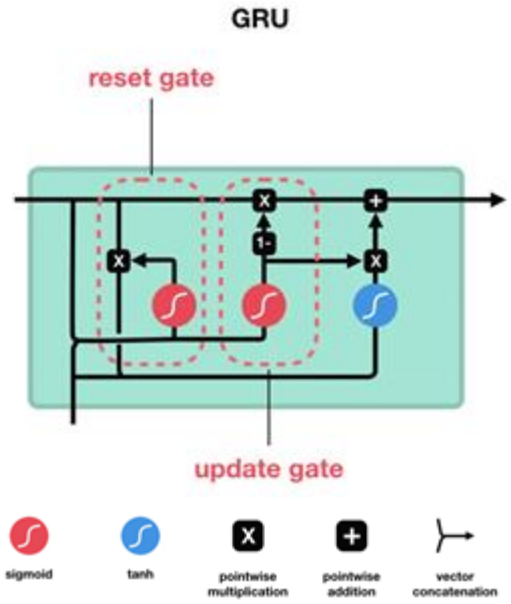
GRU	RMSE	MAPE
Task 1a	23.8	14.3%



mvar GRU(64/ReLu) 120->60min

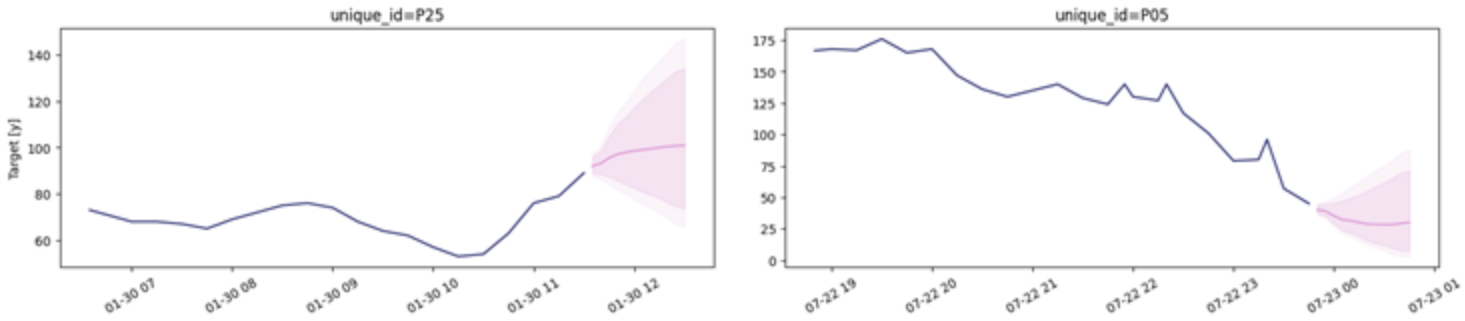
GRU	RMSE	MAPE
Task 1a	23.0	14.8%

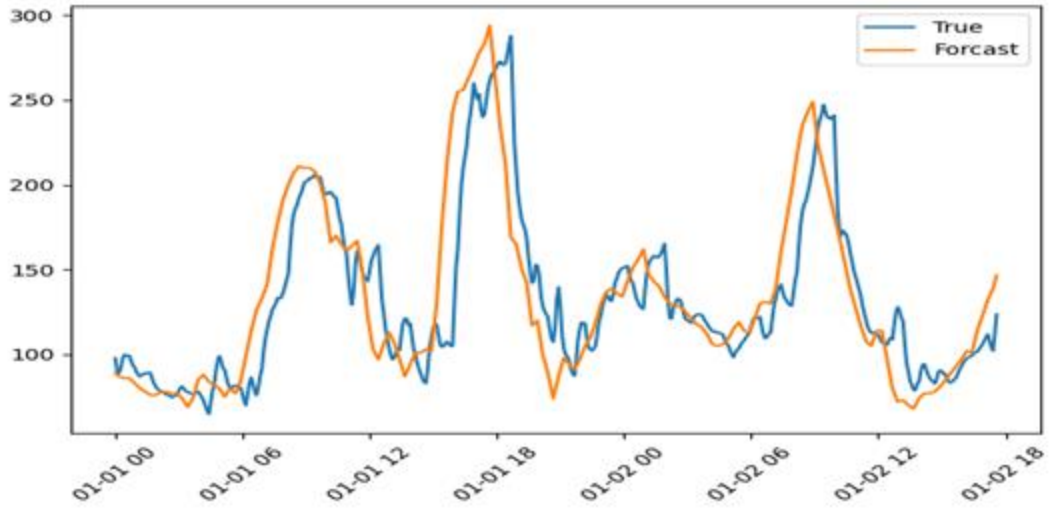




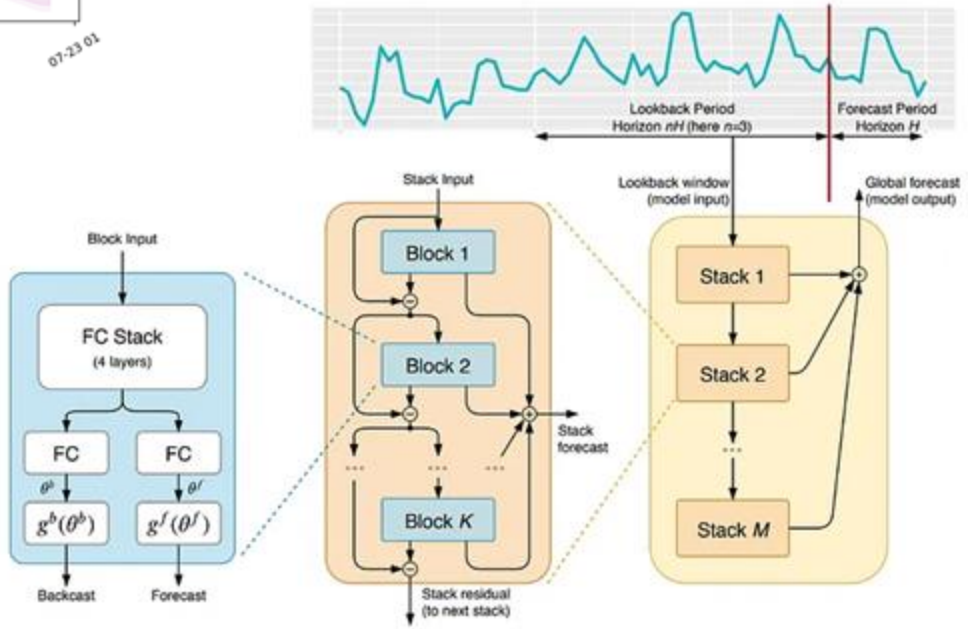
MODERN DEEP LEARNING ARCHITECTURES

- N-Hits





N-Hits	RMSE	MAPE
Task 1a	23.8	14.9%



LSTM

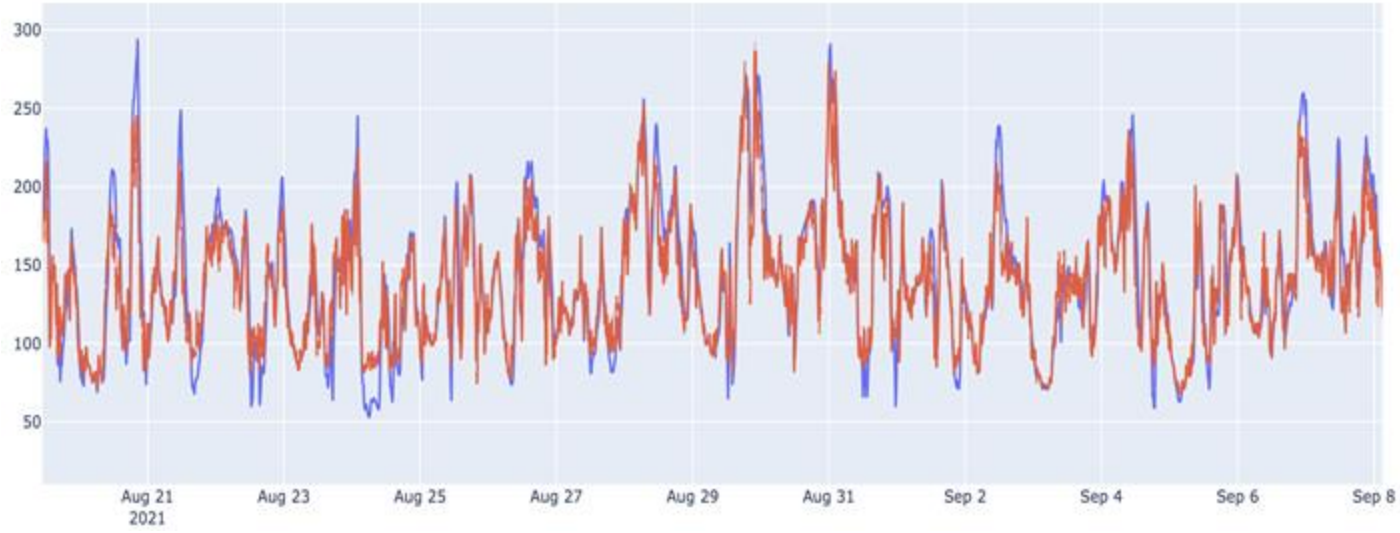
Capturing Long-Term Trends

LSTMs can model long-term dependencies in blood sugar trends by understanding the impact of past glucose levels, meals, medications, and activities. This is crucial for managing blood sugar, as patterns often depend on events hours or even days earlier.

Handling Irregular and Noisy Data

Blood sugar data can be noisy (e.g., due to sensor inaccuracies) and may have gaps or irregular intervals. LSTMs can process such sequences effectively, preserving temporal relationships and learning meaningful patterns despite variability in the data.

LSTM	RMSE	MAPE
Task 1a	18.3	12.7%



LSTM
– CODE SNIPPETS

- optuna → hyperparam opt.

```
study = optuna.create_study(direction='minimize')
study.optimize(lambda trial: objective(trial, window_size, num_features, x_train, y_train, scaler), n_trials=10)

best_params = study.best_params
model = build_lstm_model(window_size, num_features, best_params)
history = train_lstm_model(model, x_train, y_train, epochs, best_params['batch_size'])

x_test = prepare_test_data(close_values, scaler, window_size)
predictions = make_predictions(model, x_test)
```

```python
# Function to build LSTM model
def build_lstm_model(window_size, num_features, best_params):
    units = best_params['units']
    dropout_rate = best_params['dropout_rate']
    learning_rate = best_params['learning_rate']
    batch_size = best_params['batch_size']

    print("Building LSTM model...")
    model = Sequential()
    model.add(LSTM(units=units, return_sequences=True, input_shape=(window_size, num_features)))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(units=units, return_sequences=True))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(units=units))
    model.add(Dropout(dropout_rate))
    model.add(Dense(units=1))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='mean_squared_error')
    print("Model built successfully!")
    return model


# Function to train LSTM model
def train_lstm_model(model, x_train, y_train, epochs, batch_size):
    print("Training LSTM model...")
    early_stopping = EarlyStopping(monitor='loss', patience=10, restore_best_weights=True)
    reduce_lr = ReduceLROnPlateau(monitor='loss', factor=0.2, patience=5, min_lr=0.001)
    history = model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1, callbacks=[early_stopping, reduce_lr])
    print("Model trained successfully!")
    return history
```

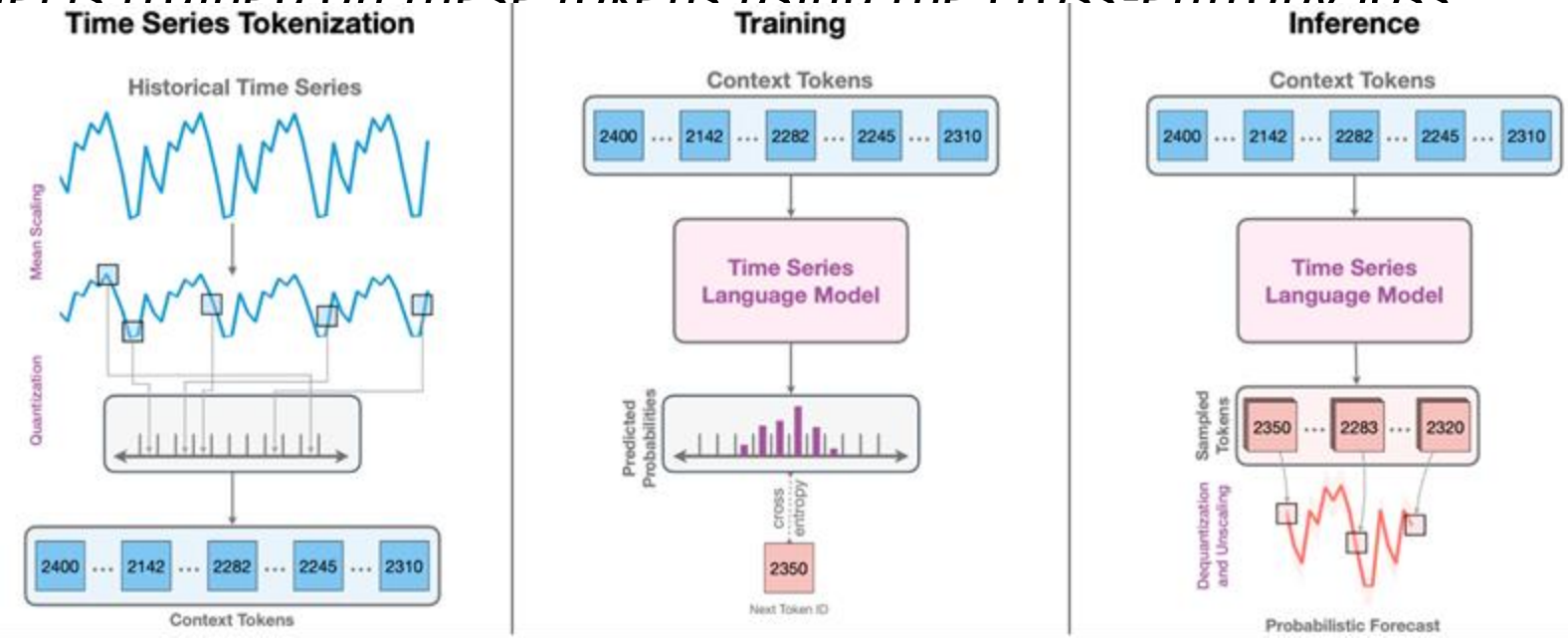
CHRONOS FOUNDATION MODEL

- Pretrained time series forecasting models based on language model architectures

  "A time series is transformed into a sequence of tokens via scaling and quantization,

  and a language model is trained on these tokens using the cross-entropy loss"

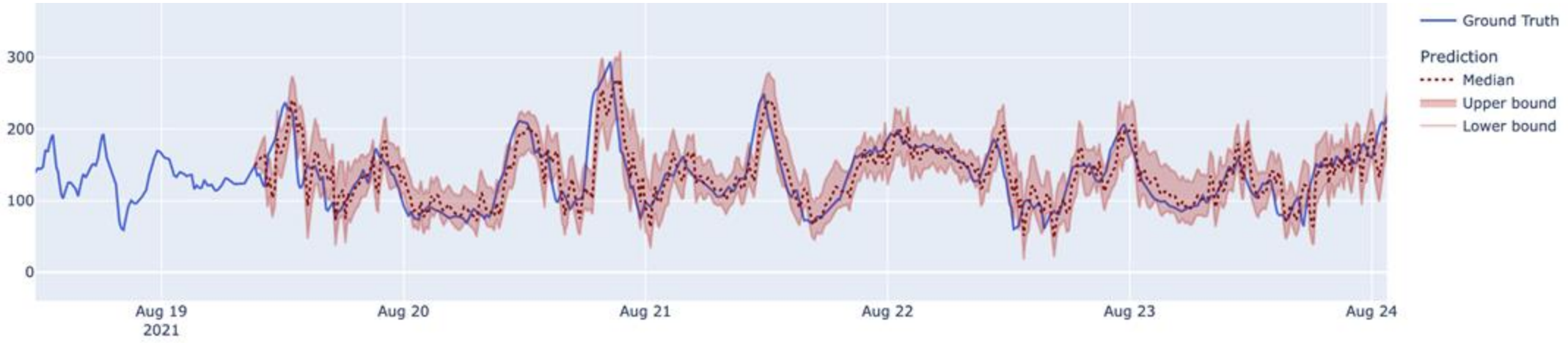
- Zero-shot application
- Fine-Tuning



CHRONOS - ZERO SHOT RESULTS

- Good performance for zero shot
- Only glucose levels used no other features!
- Just the *'bolt-small'* model
  → bigger = better ?

Zero-shot	RMSE	MAPE
Task 1a	25.0	15.3%



CHRONOS – CODE SNIPPETS

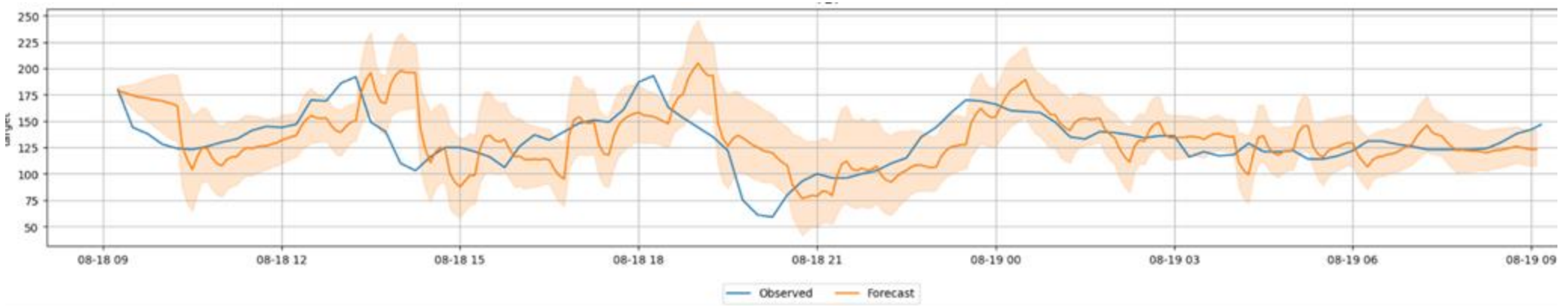
- chronos-forecasting library for zero-shot evaluation

```python
one_hour = 12
one_day = 24 * one_hour
one_week = 7 * one_day

train_df = dfs['train_p27_no_gaps'].set_index('time')
test_df = dfs['test_p27_no_gaps'].set_index('time')
window = one_hour
horizon = one_day

X = train_df['glucose_imp']
y = test_df['glucose_imp']

mean_preds = []
quantile_preds = []
for t in range(len(y)-window+1):
    bg_hist = pd.concat([X, y[:t]]).values

    quantiles, mean = pipeline.predict_quantiles(
        context = torch.tensor(bg_hist),
        prediction_length = one_hour,
        quantile_levels=[0.1, 0.5, 0.9],
    )
    if t == 0:
        mean_preds.extend(mean.squeeze().tolist())
        quantile_preds.append(quantiles.squeeze().numpy())
    else:
        mean_preds.append(mean.squeeze().numpy()[-1])
        quantile_preds.append(quantiles.squeeze().numpy()[[-1]])

mean = np.array(mean_preds)
quantiles = np.vstack(quantile_preds)
preds = np.hstack([mean[:, None], quantiles])
preds_df = pd.DataFrame(preds, columns=['mean', 'low', 'median', 'high'], index=y.index)

_df = pd.concat([X.iloc[-one_day:], y])
plot_results(_df, preds_df)
```

CHRONOS - COVARIATES & FINE TUNING RESULTS

- No full evaluation yet…
- But first results:
  Not better than zero-shot :(

Zero-shot	RMSE	MAPE
Task 1a	27.0	17.0 %



CHRONOS – CODE SNIPPETS

```python
from autogluon.timeseries import TimeSeriesDataFrame, TimeSeriesPredictor

train_data = TimeSeriesDataFrame.from_data_frame(
    train_df,
    id_column="subj",
    timestamp_column="time"
)
test_data = TimeSeriesDataFrame.from_data_frame(
    test_df,
    id_column="subj",
    timestamp_column="time"
)

n_steps = one_hour

## fit the model / fine tune it on train data
predictor = TimeSeriesPredictor(prediction_length = n_steps)
predictor = predictor.fit(
        train_data=train_data,
        hyperparameters={
            "Chronos": [
                {"model_path": "bolt_small", "ag_args": {"name_suffix": "ZeroShot"}},
                {"model_path": "bolt_small", "fine_tune": True, "ag_args": {"name_suffix": "FineTuned"}},
            ]
        },
        time_limit=600,  # time limit in seconds
        enable_ensemble=False,
)
```

```python
## evaluate on test data
predictions = []
context_data = train_data[-one_day:] # we don't have to feed all into it every time..
y = test_data['target']

model = 'ChronosZeroShot[bolt_small]' # None

for t in range(len(y)-n_steps):
    eval_data, _ = pd.concat([context_data, test_data[:t]])
    pred = predictor.predict(eval_data, model=model)
    predictions.append(pred.iloc[-1:])
predictions = pd.concat(predictions)

predictor.plot(
    data=context_data,
    predictions=predictions,
    max_history_length=one_day,
);
```

- AutoGluon for customization and finetuning

RESULT OVERVIEW

Models	RMSE	MAPE
Naive	27.0	17.0%
ARIMA	25.9	17.4%
RandomForest	16.8	11.2%
XGBoost	17.6	11.9%
LSTM	18.3	12.7%
GRU	23.8	14.3%
N-Hits	23.8	14.9%
Chronos	27.0	17.0%

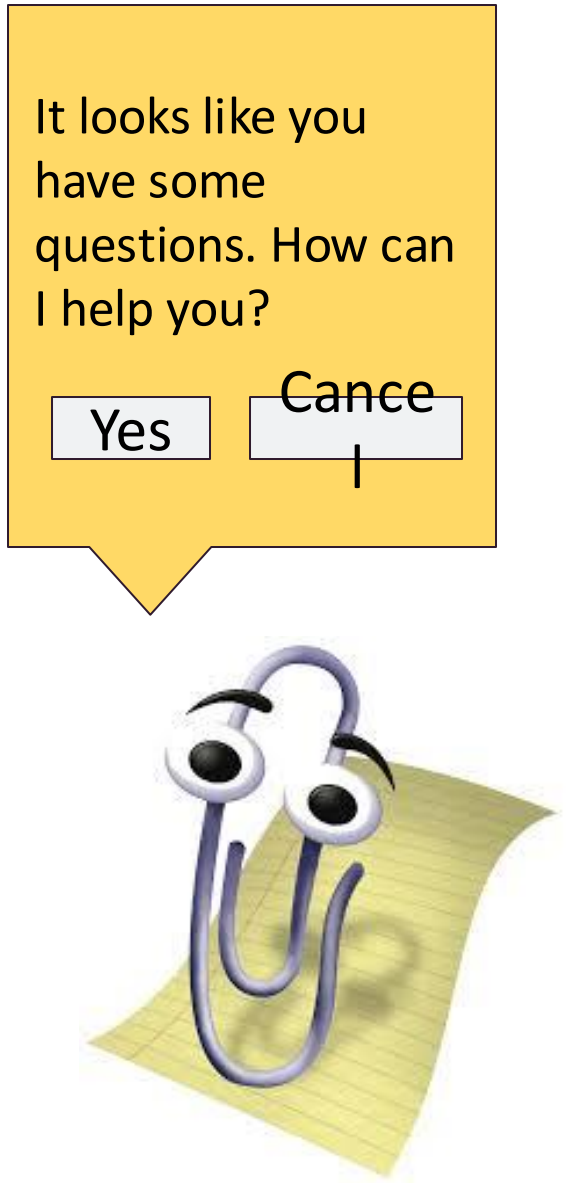
DISCUSSION

- Most models not much better than the naive baseline :/
- Tree based models most successful

OUTLOOK

- Additional feature engineering
  - Anthropometrics (age, gender, height, etc.)
  - Sleep features
  - Time since last meal, avg activity level, etc.

Q & A


It looks like you have some questions. How can I help you?
Yes
Cancel

LITERATURE

- Hidalgo et al., Alvarado, J.,Botella, M., et al. (2024): HUPA-UCM diabetes dataset. Data in Brief, 55, 110559. https://doi.org/10.1016/j.dib.2024.110559
- Azur, M. J., Stuart, E. A., Frangakis, C., & Leaf, P. J. (2011). *Multiple imputation by chained equations: what is it and how does it work?*. International journal of methods in psychiatric research, *20*(1), 40–49. https://doi.org/10.1002/mpr.329
- Ansari A. F., Stella L., Turkmen, C, et al.. (2024). *Chronos: Learning the Language of Time Series*. *Transactions on Machine Learning Research*. https://doi.org/10.48550/arXiv.2403.07815