Initial Test Results

Test Size	0.1			
Metric	**Service 1**	**Service 2**	**HashGen**	**RandPw**
CPU R² Train	0.9992	0.9988	0.9992	0.9934
CPU R² Test	0.9963	0.992	0.9943	0.9644
CPU CV Mean R² ± Std	-85.3652 ± 170.3245	0.4644 ± 0.4504	0.7465 ± 0.0321	-0.5609 ± 0.7951
Memory R² Train	0.9587	0.92	0.9339	0.9999
Memory R² Test	0.7567	0.4245	0.5385	0.9993
Memory CV Mean R² ± Std	-4.8495 ± 7.3690	-0.5908 ± 0.3038	-1.0214 ± 0.8745	-1.5927 ± 4.0328
Test Size	**0.2**			
Metric	**Service 1**	**Service 2**	**HashGen**	**RandPw**
CPU R² Train	0.9993	0.999	0.9993	0.9945
CPU R² Test	0.9961	0.9935	0.9949	0.977
CPU CV Mean R² ± Std	-85.3652 ± 170.3245	0.4644 ± 0.4504	0.7465 ± 0.0321	-0.5609 ± 0.7951
Memory R² Train	0.966	0.9244	0.9373	0.9999
Memory R² Test	0.7416	0.4674	0.5964	0.9998
Memory CV Mean R² ± Std	-4.8495 ± 7.3690	-0.5908 ± 0.3038	-1.0214 ± 0.8745	-1.5927 ± 4.0328
Test Size	**0.1**			
Metric	**Service 1**	**Service 2**	**HashGen**	**RandPw**
CPU R² Train	0.9993	0.999	0.9994	0.9952
CPU R² Test	0.9973	0.9941	0.9954	0.9795
CPU CV Mean R² ± Std	-85.3652 ± 170.3245	0.4644 ± 0.4504	0.7465 ± 0.0321	-0.5609 ± 0.7951
Memory R² Train	0.961	0.9268	0.9438	0.9998
Memory R² Test	0.8938	0.5548	0.6285	1
Memory CV Mean R² ± Std	-4.8495 ± 7.3690	-0.5908 ± 0.3038	-1.0214 ± 0.8745	-1.5927 ± 4.0328

CPU Usage Prediction

Consistently High Performance Across All Models

- All models show excellent CPU usage prediction with $R^2 > 0.99$ on both training and test sets.
- This indicates a strong fit and generalization capacity, with little to no overfitting.

Service 1 – Anomaly in Cross-Validation

- Despite very high train/test R^2 values, the CV mean R^2 is extremely negative (~ -85) with high variance.
- This suggests data leakage or severe instability in folds possibly due to:
    - Temporal correlation (e.g., improper time-series split)
    - Inconsistent patterns across folds

Memory Usage Prediction

Mixed Model Performance

- RandPw model shows nearly perfect performance on both train and test sets ($R^2 > 0.99$).
- Other services, especially Service 1 and Service 2, show weaker generalization:
    - Memory R^2 test scores:
        - Service 1: ~0.74–0.89
        - Service 2: ~0.42–0.55
        - HashGen: ~0.53–0.63
- CV scores are negative or near zero, suggesting poor consistency across different folds.

Indicates Underfitting or Unstable Signal

- Memory usage is likely more volatile and less predictable than CPU.
- Current features might not capture memory-related patterns well.

Impact of Test Size

- Reducing test size slightly improves test R^2 for memory predictions, especially for Service 1 and HashGen.
- Suggests models are more stable when more data is allocated for training.
- But CV metrics remain nearly the same: underlying data issues need addressing.

With Grid Search

CPU Usage Predictions

Service	Train R^2	Test R^2	CV R^2	Remarks
Service 1	~0.93	~0.92	-10.75	Good train/test scores but CV R^2 is extremely low potential data leakage or unstable folds.
Service 2	~0.99	~0.99	0.64	Excellent all-around performance. Robust model.
HashGen	~0.96	~0.95	0.82	Strong and stable performance.
RandPw	~0.52	~0.52	0.05	Weak predictive power likely non-linear or noisy patterns.

Memory Usage Predictions

Service	Train R^2	Test R^2	CV R^2	Remarks
Service 1	~0.61–0.64	~0.54–0.64	-3.13	Moderate fit but very poor generalization potential overfitting or poor feature-target relationship.
Service 2	~0.13–0.15	~0.12–0.16	-0.45	Very weak model suggests little signal in features for memory usage.
HashGen	~0.30	~0.28	-0.30	Weak generalization but slightly better than Service 2.
RandPw	~0.99	~0.99	-1.55	Severe overfitting train/test R^2 are misleading due to poor CV score.

With New Features

- Added rolling means and rolling standard deviations for CPU Usage, Memory Usage, and Latency per service.
- Created spike detection features by measuring deviations from rolling means in CPU and Memory usage.
- Introduced a latency trend feature to capture whether latency is increasing, decreasing, or stable over time.
- These features improved model accuracy by adding temporal context and reducing noise.
- Rolling statistics helped reduce overfitting by encouraging better generalization over time-series data.

CPU Usage Predictions

Test Size	Service	Train	Test	CV	Best Params
0.3	Service 1	0.9999	0.9998	0.9838	{'max_depth': 10, 'min_samples_leaf':1, 'min_samples_split':5}
	Service 2	1	0.9999	0.9763	{'max_depth': 10, 'min_samples_leaf':2, 'min_samples_split':2}
	HashGen	1	0.9998	0.999	{'max_depth': 10, 'min_samples_leaf':1, 'min_samples_split':2}
	RandPw	0.9989	0.9962	0.9767	{'max_depth': 10, 'min_samples_leaf':2, 'min_samples_split':2}
0.2	Service 1	0.9998	0.9998	0.9838	{'max_depth': 10, 'min_samples_leaf':1, 'min_samples_split':5}
	Service 2	1	0.9999	0.9763	{'max_depth': 10, 'min_samples_leaf':2, 'min_samples_split':2}
	HashGen	1	0.9998	0.999	{'max_depth': 10, 'min_samples_leaf':1, 'min_samples_split':2}
	RandPw	0.9988	0.998	0.9767	{'max_depth': 10, 'min_samples_leaf':2, 'min_samples_split':2}
0.1	Service 1	0.9999	0.9999	0.9838	{'max_depth': 10, 'min_samples_leaf':1, 'min_samples_split':5}
	Service 2	1	0.9999	0.9763	{'max_depth': 10, 'min_samples_leaf':2, 'min_samples_split':2}
	HashGen	1	0.9998	0.999	{'max_depth': 10, 'min_samples_leaf':1, 'min_samples_split':2}
	RandPw	0.999	0.9965	0.9767	{'max_depth': 10, 'min_samples_leaf':2, 'min_samples_split':2}

- The model achieves very high R^2 scores on training, testing, and cross-validation across all test sizes.
- As the test size decreases, training and testing R^2 scores slightly improve or stay nearly perfect.
- Cross-validation scores remain stable and slightly lower than training/testing scores regardless of test size.
- Smaller test sizes give the model more data to train on, which helps improve training and testing scores.
- The consistent cross-validation scores indicate the model's true generalization ability stays reliable.

Memory Usage Predictions

Test Size	Service	Train	Test	CV	Best Params
0.3	Service 1	0.9895	0.5835	0.8593	{'max_depth': 10, 'min_samples_leaf':4, 'min_samples_split':10}
	Service 2	0.999	0.9948	0.9931	{'max_depth': 10, 'min_samples_leaf':1, 'min_samples_split':2}
	HashGen	0.9981	0.9966	0.9779	{'max_depth': 10, 'min_samples_leaf':2, 'min_samples_split':2}
	RandPw	0.9973	0.9946	0.9166	{'max_depth': 10, 'min_samples_leaf':4, 'min_samples_split':10}

0.2	Service 1	0.941	0.8453	0.8593	{'max_depth': 10, 'min_samples_leaf':4, 'min_samples_split':10}
	Service 2	0.9991	0.9946	0.9931	{'max_depth': 10, 'min_samples_leaf':1, 'min_samples_split':2}
	HashGen	0.9986	0.9975	0.9779	{'max_depth': 10, 'min_samples_leaf':2, 'min_samples_split':2}
	RandPw	0.9973	0.9942	0.9166	{'max_depth': 10, 'min_samples_leaf':4, 'min_samples_split':10}
0.1	Service 1	0.9512	0.9963	0.8593	{'max_depth': 10, 'min_samples_leaf':4, 'min_samples_split':10}
	Service 2	0.9991	0.9955	0.9931	{'max_depth': 10, 'min_samples_leaf':1, 'min_samples_split':2}
	HashGen	0.9988	0.9979	0.9779	{'max_depth': 10, 'min_samples_leaf':2, 'min_samples_split':2}
	RandPw	0.9972	0.9927	0.9166	{'max_depth': 10, 'min_samples_leaf':4, 'min_samples_split':10}

- Training R^2 scores vary more across services, with Service 1 showing lower training scores than others.
- Test R^2 scores for Service 1 improve significantly as the test size decreases, from 0.58 (30%) to 0.996 (10%).
- Other services (Service 2, HashGen, RandPw) maintain consistently high test R^2 scores above 0.99 regardless of test size.
- Cross-validation scores for Service 1 remain stable around 0.86 across all test sizes.
- Cross-validation scores for other services are consistently high (above 0.91), indicating good generalization.
- Smaller test sizes help improve test accuracy for Service 1 by giving more training data.
- The model parameters remain stable, suggesting similar model complexity is optimal across test sizes.

Configuration Used

- Model: Random Forest Regressor
- Hyperparameters:
  - `max_depth = 10`
  - `min_samples_split = 2`
  - `min_samples_leaf = 2`
- Test Size: 10%
- Cross-Validation: 5-fold
- Scaling: StandardScaler
- Rolling Features: Enabled (via `add_rolling_features()`)

Service	Target	R^2 Train	R^2 Test	CV Mean R^2	CV Std Dev
Service 1	**CPU Usage**	0.9999	0.9999	0.9835	0.0175
	Memory Usage	0.9935	0.9965	0.8474	0.1926
Service 2	**CPU Usage**	1	0.9999	0.9763	0.0461
	Memory Usage	0.9987	0.9953	0.993	0.0037
HashGen	**CPU Usage**	0.9999	0.9998	0.999	0.0003
	Memory Usage	0.9988	0.9979	0.9779	0.0167
RandPw	**CPU Usage**	0.999	0.9965	0.9767	0.0333
	Memory Usage	0.9985	0.9952	0.9049	0.1308

Conclusions

1. **CPU Usage Predictions:**
    - Excellent performance across all services.
    - CV Mean R^2 consistently above 0.97, peaking at 0.9990 for HashGen.
    - Low standard deviation → Model generalizes well.

2. **Memory Usage Predictions:**
    - Good performance overall, especially on Service 2 and HashGen.
    - Service 1 and RandPw show lower CV Mean R^2 and higher variability — might benefit from more feature engineering or ensemble smoothing.
    - Service 2 Memory had highest consistency (CV Std: 0.0037).

3. **Generalization vs. Overfitting:**
    - Train and test scores are extremely close, suggesting minimal overfitting.
    - However, the high train scores and lower CV scores (especially for memory) could mean:
        - Some patterns are dataset-specific.
        - Additional regularization or data diversity could help.