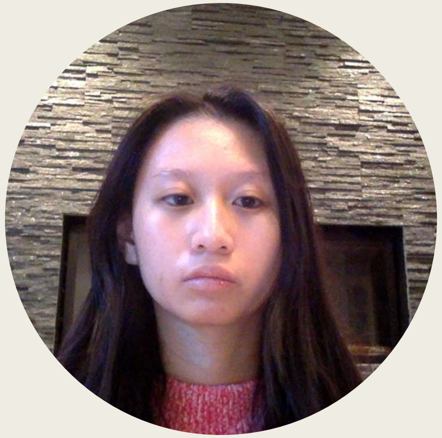
# CIS731 FINAL PROJECT: FINE GRAINED DEMAND FORECASTING & PREDICTIVE ANALYTICS USING XGBOOST & PROPHET

Minh Pham

Presentation 15

# Content

- Introduction/Background
- Data Overview
- Data Cleaning
- Data Exploration
- Machine Learning Models
- Evaluation
- Results
- Conclusion



# Introduction

**Problem Statement:**

- Accurate demand forecasting is crucial for optimizing inventory and improving supply chain management
- Need a scalable solution for performing store-item level predictions

**Objective:**

- Compare and evaluate machine learning models for fine-grained demand forecasting at the store-item level
    - *Using Kaggle's Store Item Demand Forecasting historical sales dataset [1]*
- Use Databricks' Fine-Grained Demand Forecasting Accelerator with Prophet as a baseline model and compare it against XGBoost [2].
- Assess forecasting performance using evaluation metrics: MAE, RMSE, MAPE, and one sided t-test

[1] https://www.kaggle.com/competitions/demand-forecasting-kernels-only/data

[2]https://notebooks.databricks.com/notebooks/RCG/Fine_Grained_Demand_Forecasting/index.html#Fine_Grained_Demand_Forecasting_1.html

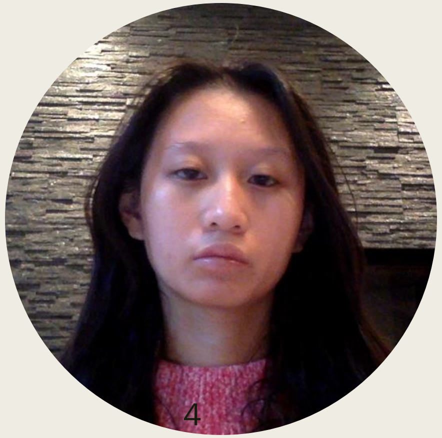
# Background

**Prophet**

- Designed specifically for time-series forecasting
- Automatically handles seasonality, holidays, and trend changes
- Requires minimal parameter tuning and works well with missing data
- Ideal for business forecasting with built-in interpretability

- **XGBoost**
- A powerful gradient boosting model typically for classification or regression
- Known for high performance and is a top choice for Kaggle time series prediction competitions
- Requires custom feature engineering for time-series data
- Optimized for scalability and efficient training using boosting techniques

[1] https://facebook.github.io/prophet/docs/multiplicative_seasonality.html
[2] https://machinelearningmastery.com/xgboost-for-time-series-forecasting/
[3] ChatGPT Prompt: “I am doing a project on fine grained demand forecasting. I am following a databricks solution that uses Prophet, what are some other alternatives to this forecasting model that are simple but still accurate?”



# Dataset Overview

- **Dataset:** Kaggle Store Item Demand Forecasting Training Set [1]
  - *5 years of historical sales data 2013-2017*
  - *10 stores, 50 items*
  - *includes dates, store IDs, item IDs, and #of sales*
  - *913,000 rows of data*
- **Granularity:** Store-item combinations across multiple years, enabling trend and seasonality analysis.

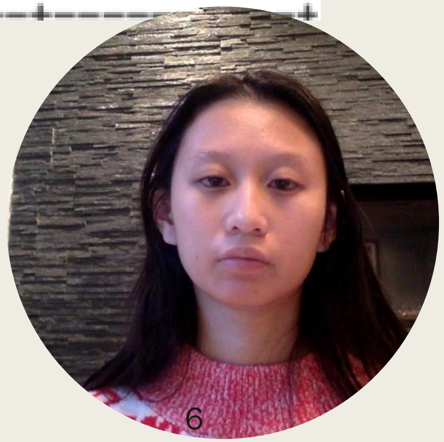
| | date | store | item | sales |
|---|---|---|---|---|
| 1 | 2013-01-01 | 1 | 1 | 13 |
| 2 | 2013-01-02 | 1 | 1 | 11 |
| 3 | 2013-01-03 | 1 | 1 | 14 |
| 4 | 2013-01-04 | 1 | 1 | 13 |
| 5 | 2013-01-05 | 1 | 1 | 10 |
| 6 | 2013-01-06 | 1 | 1 | 12 |


- [1]https://www.kaggle.com/competitions/demand-forecasting-kernels-only/data

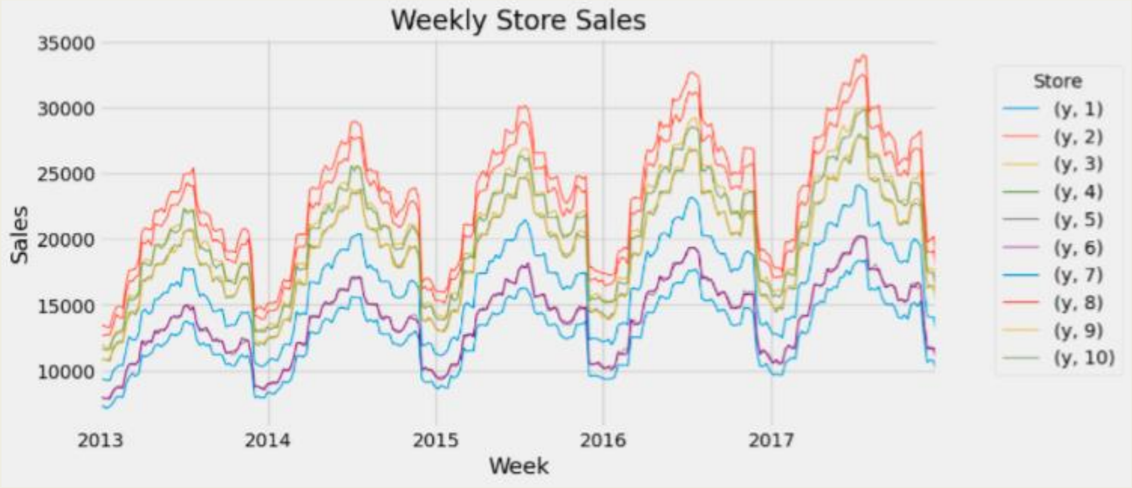

# Data Cleaning

- Overall, a clean dataset
  - Aggregate to ensure dataset is at store item level
- Standardized datetime
- Created column with month and weekday names with spark.sql functions

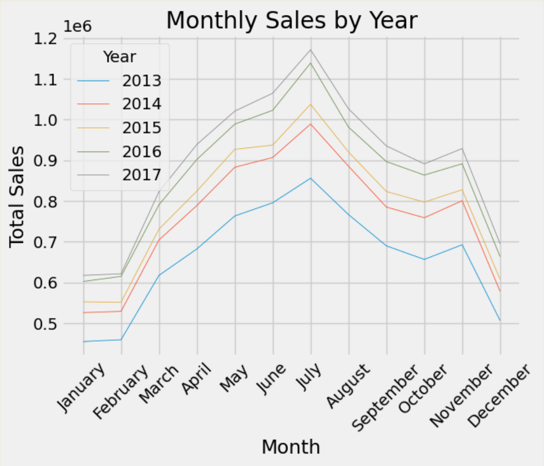
| Day | month | day_of_week | day_of_week_name | month_name |
|---|---|---|---|---|
| Tuesday | 7 | 3 | Tuesday | July |
| Tuesday | 8 | 3 | Tuesday | August |
| Saturday | 12 | 7 | Saturday | December |
| Thursday | 5 | 5 | Thursday | May |
| Monday | 8 | 2 | Monday | August |
| Tuesday | 12 | 3 | Tuesday | December |
| Sunday | 4 | 1 | Sunday | April |
| Wednesday | 3 | 4 | Wednesday | March |
| Wednesday | 5 | 4 | Wednesday | May |
| Tuesday | 6 | 3 | Tuesday | June |

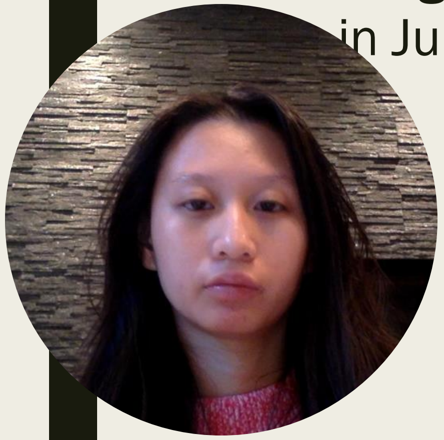


# Data Exploration




- **Visualizations Conducted:**
  - *Weekly sales trends by store and item*
  - *Monthly and weekly sales peaks.*
  - *Year over year analysis*
- **Insights:** Identified peak sales in July and weekends.

# Methodology Overview

**Prophet Model (Baseline):**

- Selected for its ability to handle seasonality and long-term trends automatically
- Requires minimal manual feature engineering, making it user-friendly for time-series forecasting

**XGBoost Model (Comparative):**

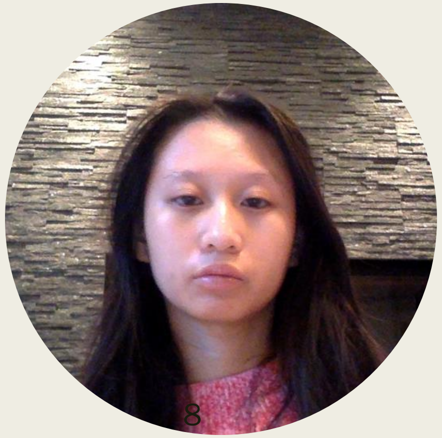
- Chosen for its ability to incorporate custom-engineered features
- Supports gradient boosting, providing high accuracy in regression tasks

**Processing Framework:**

- **PySpark:** Used for scalable and efficient data handling, enabling fast data processing for the large dataset of 913,000 rows.

**Statistical Hypothesis Testing:**

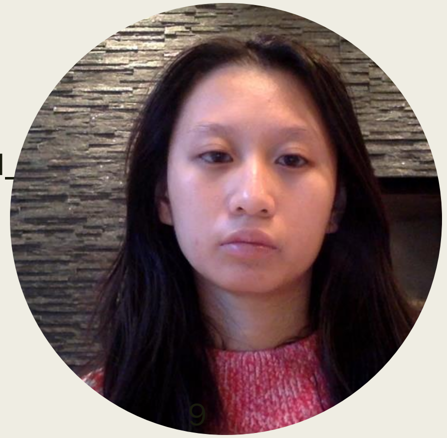
- **Null Hypothesis ($H_0$):** Prophet performs better or equally as well as XGBoost
- **Alternative Hypothesis ($H_1$):** XGBoost performs better than Prophet



# Building Prophet Model

- **Import Prophet Model using pip install prophet**
- **Configuration: followed the Databricks solution accelerator's set parameters [1]**
  - *Interval width: 0.95*
  - *Seasonality: Weekly and yearly enabled*
  - *Growth: Linear*
- **Training:** Full dataset, including 2017.
- **Forecast:** Automatically predicts for all historical and future dates.
- **Strengths:** Automated feature handling, trend and seasonality detection.

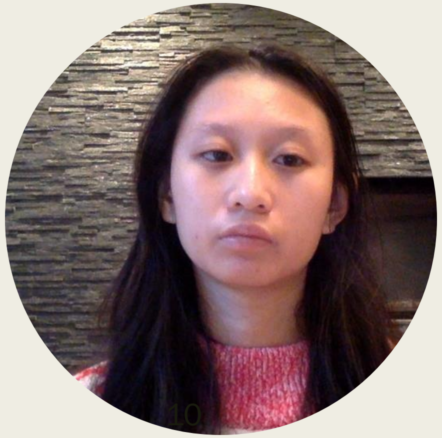
- [1]https://notebooks.databricks.com/notebooks/RCG/Fine_Grained_Demand_Forecasting/index.html#Fine_Grained_Demand_
_1.html



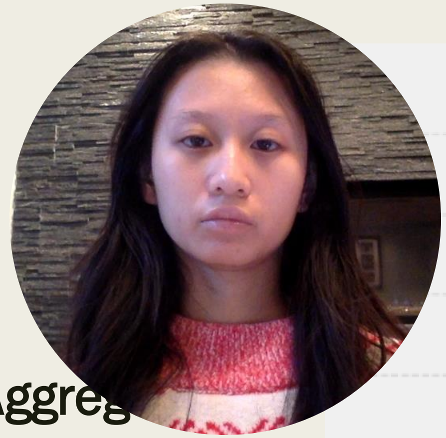
# Building XGBoost Model

- **Features Engineered:**
  - ***Temporal:*** *Year, month, day, day of the week, is_weekend.*
  - ***Sales History:*** *Lag-1,2,3 , rolling mean (window=3).*
- **Training Process: 80/20 split**
  - *Training set: Data before 2017.*
  - *Test set: Data from 2017.*
- **Forecast:** Predicted values for 2017 and next 90 days.
- **Strengths:** Custom feature selection, works well for irregular patterns

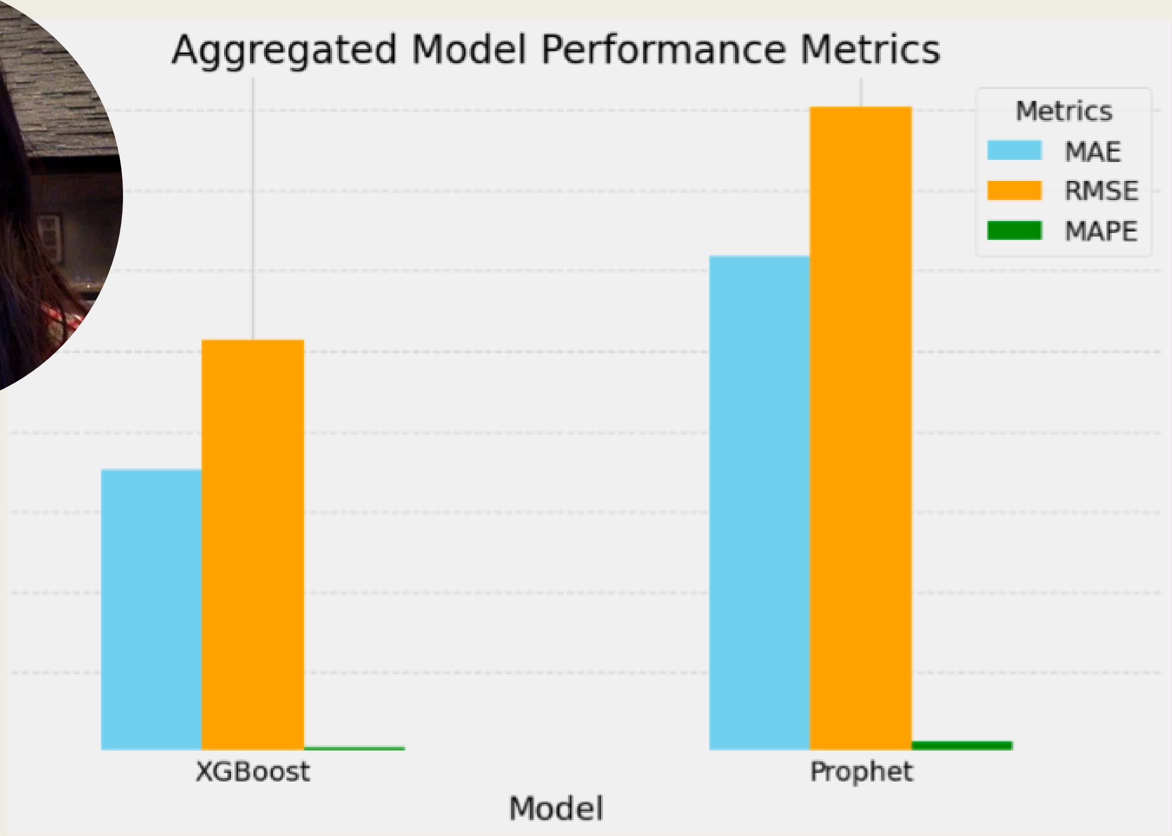

- [1] https://www.kaggle.com/code/enolac5/time-series-arima-dnn-xgboost-comparison#Findings-and-Steps-Forward
- [2] https://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost/notebook#Train/Test-Split

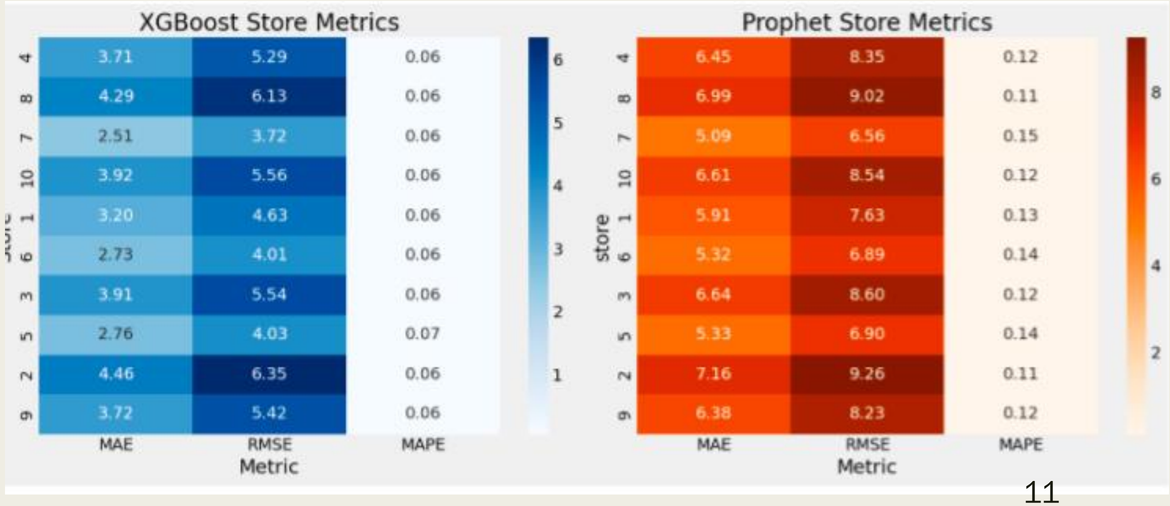



# Evaluation



- **Model Evaluation Results (Aggreg Metrics):**
- **XGBoost Model:**
  - *MAE: 3.52*
  - *RMSE: 5.14*
  - *MAPE: 6.33%*
- **Prophet Model:**
  - *MAE: 6.19*
  - *RMSE: 8.05*
  - *MAPE: 12.73%*
- Below picture is a comparison at store level



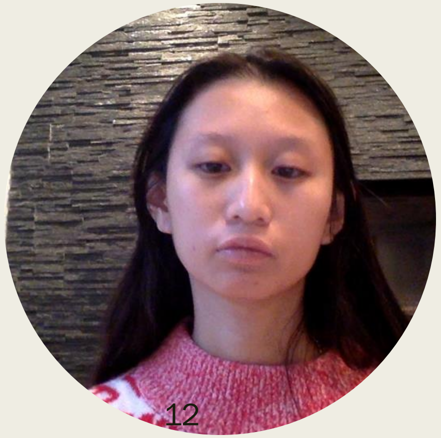




| store | XGBoost MAE | XGBoost RMSE | XGBoost MAPE | Prophet MAE | Prophet RMSE | Prophet MAPE |
|---|---|---|---|---|---|---|
| 4 | 3.71 | 5.29 | 0.06 | 6.45 | 8.35 | 0.12 |
| 8 | 4.29 | 6.13 | 0.06 | 6.99 | 9.02 | 0.11 |
| 7 | 2.51 | 3.72 | 0.06 | 5.09 | 6.56 | 0.15 |
| 10 | 3.92 | 5.56 | 0.06 | 6.61 | 8.54 | 0.12 |
| 1 | 3.20 | 4.63 | 0.06 | 5.91 | 7.63 | 0.13 |
| 6 | 2.73 | 4.01 | 0.06 | 5.32 | 6.89 | 0.14 |
| 3 | 3.91 | 5.54 | 0.06 | 6.64 | 8.60 | 0.12 |
| 5 | 2.76 | 4.03 | 0.07 | 5.33 | 6.90 | 0.14 |
| 2 | 4.46 | 6.35 | 0.06 | 7.16 | 9.26 | 0.11 |
| 9 | 3.72 | 5.42 | 0.06 | 6.38 | 8.23 | 0.12 |

# Evaluation

- Our null hypothesis: (Prophet's performance is equal to or better than XGBoost's) is rejected
- XGBoost is significantly more accurate

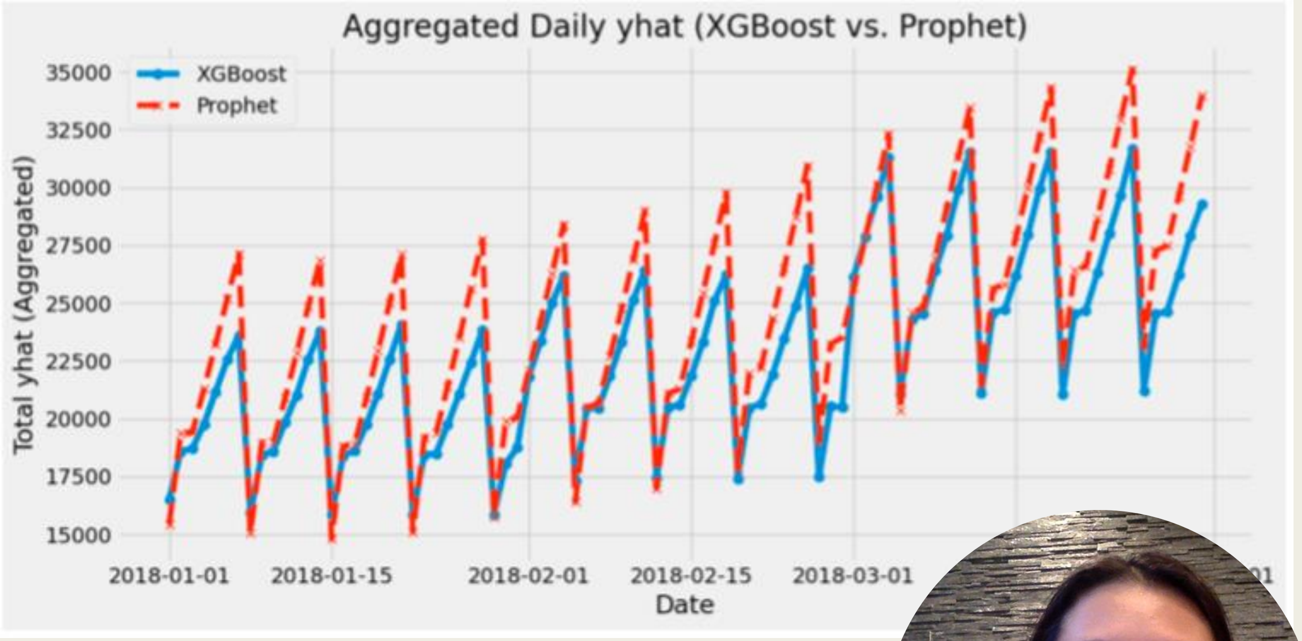
```
MAE T-Statistic: -134.7186
MAE P-Value: 1.7382e-16
Reject the null hypothesis for MAE: XGBoost performs significantly better.

RMSE T-Statistic: -104.8531
RMSE P-Value: 1.6564e-15
Reject the null hypothesis for RMSE: XGBoost performs significantly better.

MAPE T-Statistic: -15.4742
MAPE P-Value: 4.3032e-08
Reject the null hypothesis for MAPE: XGBoost performs significantly better.
```

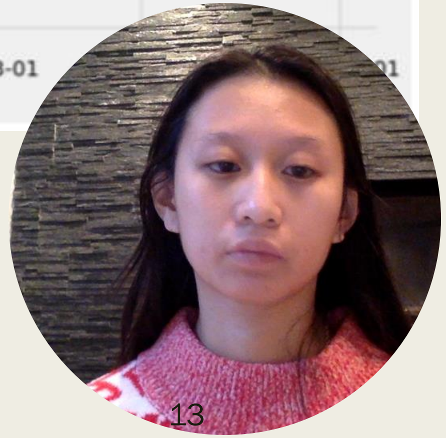


12

# Results: Future Forecast Analysis

- As seen in historical data, sales begin to pick up in the new year
- Similar sales pattern
- Prophet seems to overpredict compared to XGBoost



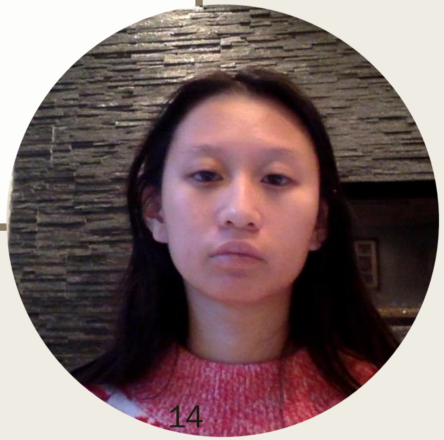
# Conclusion

## Performance Insights:

- Both models are easily scalable,  and quick
- XGBoost has better accuracy and is significantly more accurate.
- Prophet requires less manual work and is sufficient for quick forecasts

## Next Steps:

- Explore additional models (ARIMA, LSTM).
- Improve feature engineering for XGBoost.
- Use Prophet’s custom seasonalities for better fit.



14

# Sources


- Brownlee, J. (2020, August 4). *How to Use XGBoost for Time Series Forecasting*. Machine Learning Mastery. https://machinelearningmastery.com/xgboost-for-time-series-forecasting/
- *ChatGPT*. (2024). Chatgpt.com. https://chatgpt.com/auth/login
- Chui, M., Manyika, J., Miremadi, M., Henke, N., Chung, R., Nel, P., & Malhotra, S. (2018, April 17). *Notes from the AI frontier: Applications and value of deep learning*. McKinsey & Company. https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-applications-and-value-of-deep-learning
- *Demand Forecasting at Scale*. (n.d.). Databricks. https://www.databricks.com/solutions/accelerators/demand-forecasting
- Eno5. (2018). *Blocked*. Kaggle.com. https://www.kaggle.com/code/enolac5/time-series-arima-dnn-xgboost-comparison#Model-(2)---Feed-Forward-Neural-Network-with-Daily-Data
- *Fine_Grained_Demand_Forecasting - Databricks*. (2021). Databricks.com. https://notebooks.databricks.com/notebooks/RCG/Fine_Grained_Demand_Forecasting/index.html#Fine_Grained_Demand_Forecasting_1.html
- *Multiplicative Seasonality*. (2023, October 18). Prophet. https://facebook.github.io/prophet/docs/multiplicative_seasonality.html
- robikscube. (2018, November 9). *[Tutorial] Time Series forecasting with XGBoost*. Kaggle.com; Kaggle. https://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost/notebook#Train/Test-Split
- *Store Item Demand Forecasting Challenge*. (n.d.). Kaggle.com. https://www.kaggle.com/competitions/demand-forecasting-k only/data