Predicting lifetime value of users using machine learning

W207: Applied Machine Learning
Fall 2023
Spencer Hodapp, Robert Greer, Nick Luong, Gilbert Wong

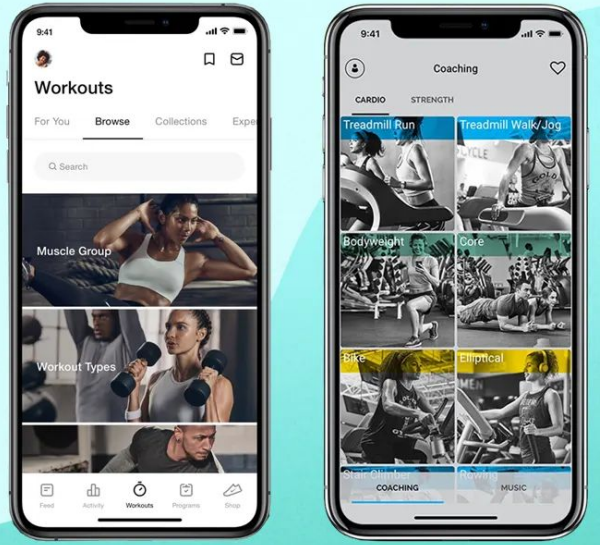
Motivation - Background

Small, local elite fitness business that recently made the investment to build a mobile app, transitioning the business from selling pdf files of workouts.

- Geared towards NCAA Div 1 & Div 2, Professional, and Olympic Athletes

The Fitness App - Freemium model

- Sport specific
- Custom workout programming
- User entry for trend analysis, notifications, progress reporting



Note: this is not the actual screenshot

Motivation - Question

The main question to answer is:

How can we predict the Lifetime value per user at any given time to inform our ongoing business strategy.

Business environment considerations:

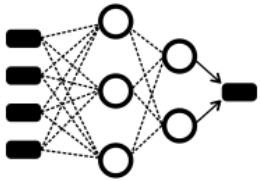
- App is only marketed through in house content, primarily through Youtube presence and website funnel
- Held a 6 month content campaign to boost initial app debut through Youtube feature videos and #shorts.

Motivation - what has been done in this space

- Customer lifetime value (CLV) - Recency, Frequency, Monetary (RFM) probability models:
  - Pareto/NBD model - probability model of customer behaviors with dropout process (Schmittlein, Morrison, Colombo, 1987)
  - BG/NBD model - only account for frequent purchase customers dropout (Fader, Hardie, Lee, 2004)
  - Contractual/Non-contractual customers and continuous/discrete transactions - (Fader, Hardie, 2009)
- Machine learning models
  - Use clustering, regression and classification models to predict app user engagement (Barbo, Grua, Malavolta, Sterevic, Weusthof, van den Hoven, 2020)

Motivation - our plan

Total Download, New Download, Number of Paying Customers, Lifetime Value/Customer, Number of YouTube posted, Last time YouTube posted, Season



Lifetime value per user

Models (Single/Multiple Steps)

- Linear Regression
- Feed Forward Neural Network
- CNN
- LSTM

Key Metric

- Mean Absolute Error

Motivation - results summary

We can predict the daily lifetime value per user with 1% - 7%error rate using linear regression, feed forward neural network, CNN and LSTM models.

Data - Description and Source

Company provided two sets of data:

355 days of time series app data:

- Number of downloads
- Number of users in free trial
- Number of paying users
- Lifetime value per user
- Revenue generated
- Number of users leaving the app
- Refund rate
- Annual recurring revenue

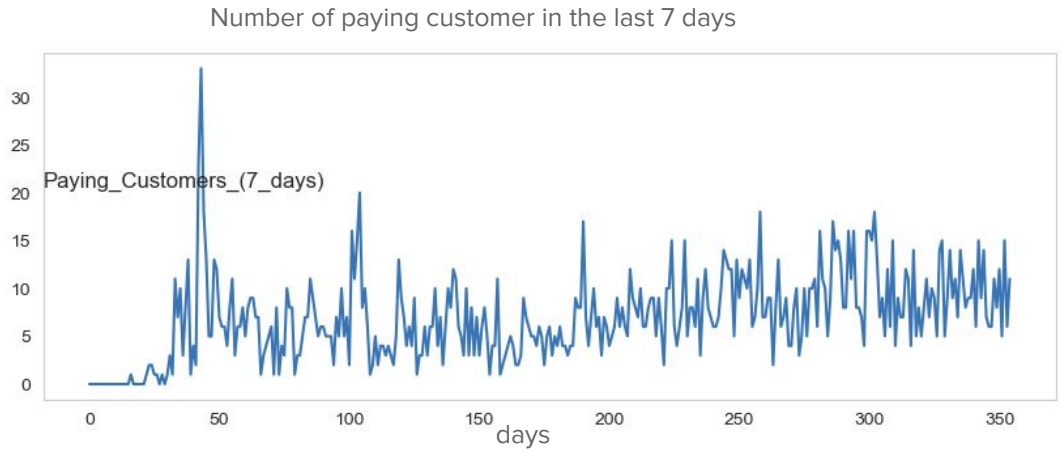
YouTube

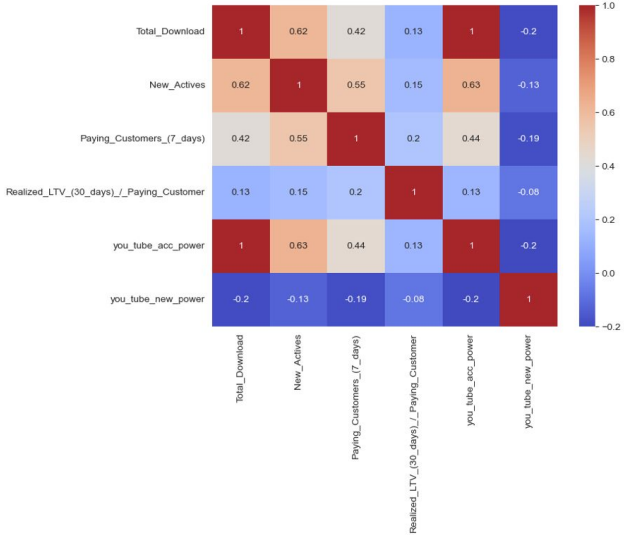
Youtube Videos:

- All videos released during the same 1 year period with total view, like comment and favorite counts.

Data - EDA

- **Total records:** 355 days of app data, 238 youtube publish data, 4 seasons of a year
- **Observation Frequency:** daily intervals
- **Features:** Total gross number of app downloads, daily number of new download, number of paying customers in last 7 days, realized lifetime value per paying customer in their first 30 days, cumulative number of youtube posted, number days from the last youtube posted, seasons
- **Time range:** August 16, 2022 to August 5, 2023
- **Source**: Composite app store analytics for the app, YouTube API





Data - Feature Engineering

- Created two features to account for the YouTube videos, a primary marketing channel, posted by the company:
  1. Total number of YouTube Videos Posted - we anticipate that an increased number of YouTube video uploads will draw more app users. This feature considers the cumulative number of YouTube videos posted to boost downloads and enhance the app's popularity gradually over time
  2. Number of days since the last YouTube Video Posted - since new YouTube videos can also attract new audiences, this feature is crafted to monitor the release timing of each new YouTube video. It aims to capture the days when these videos attract more viewers, potentially leading to increased downloads

- Implemented a new feature ("season") to track the date seasonality, anticipating its potential impact on download numbers during seasonal sport periods

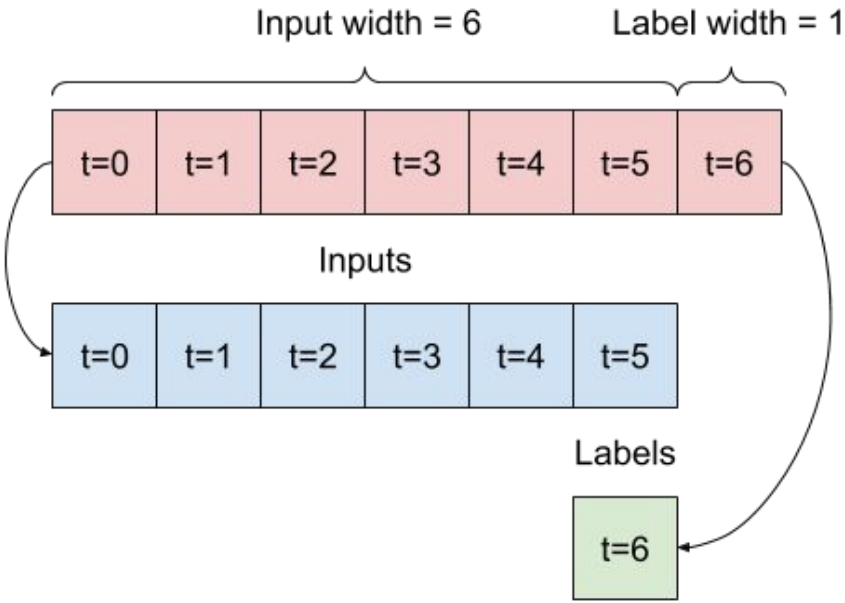
Data - Normalization of Time Series

- Given that we were using time-series data, we wanted to avoid using normalization methods that looked into the future (such as the z-score).
- Therefore, we chose to normalize our data by using a rolling average function
- We had to experiment the number of days for our lookback on the rolling average,
  - One thing to consider is that the greater the number of days for the lookback, the more data will become NaN. This is because the first X number of days will become unavailable due to not having enough days to do a moving average
- We ultimately chose a 30 day rolling average because it gave us enough of an average to capture the variation of the data while also giving us enough data to work with
  - Using a 30 day lookback period, we had 326 data points available now with our dataset

Data Split

Outcome Variable: Lifetime Value per Customer

Data - Create batched windows of normalized



1. Create windows

2. Split windows by inputs and labels

Modelling - List of models used

Single Step: 1 day input, 1 day output
- Baseline (previous value)
- Linear Regression
- Feed Forward Neural Network

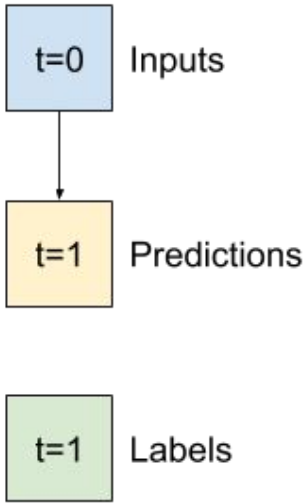
Multi-step: 6 days input, 1 day output
- Baseline (previous value)
- Linear Regression
- CNN
- LSTM

Multi-step Forecast: 18 days input, 18 day output
- Baseline (both previous value for all future values , and repeating previous values)
- Linear Regression
- FFNN
- CNN
- LSTM

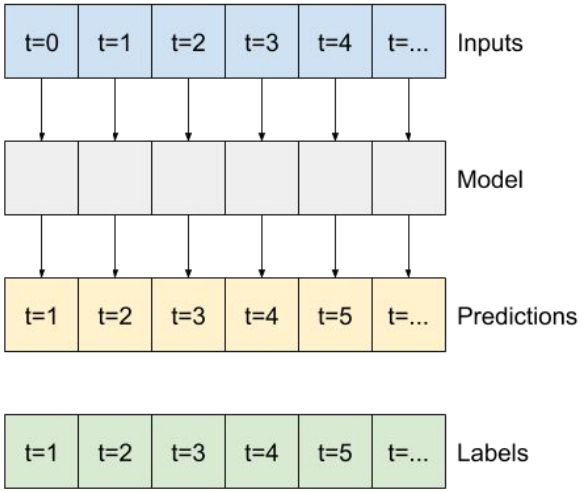
Single Step Models

Take 1 day of input, predict the next day

Baseline
Use the previous day

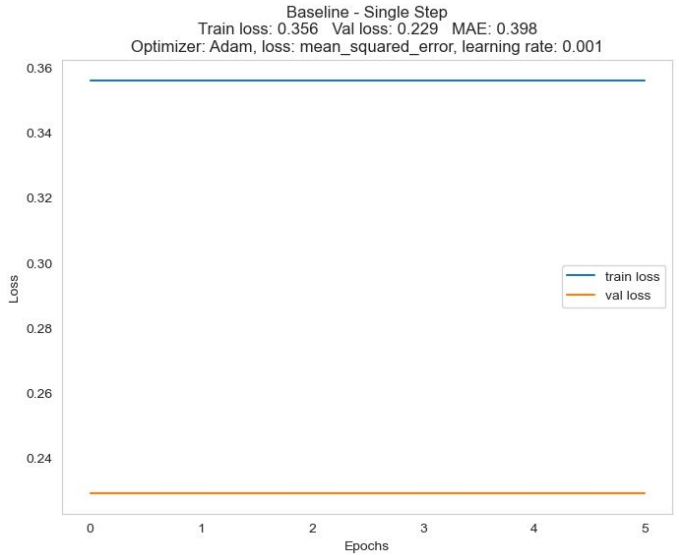


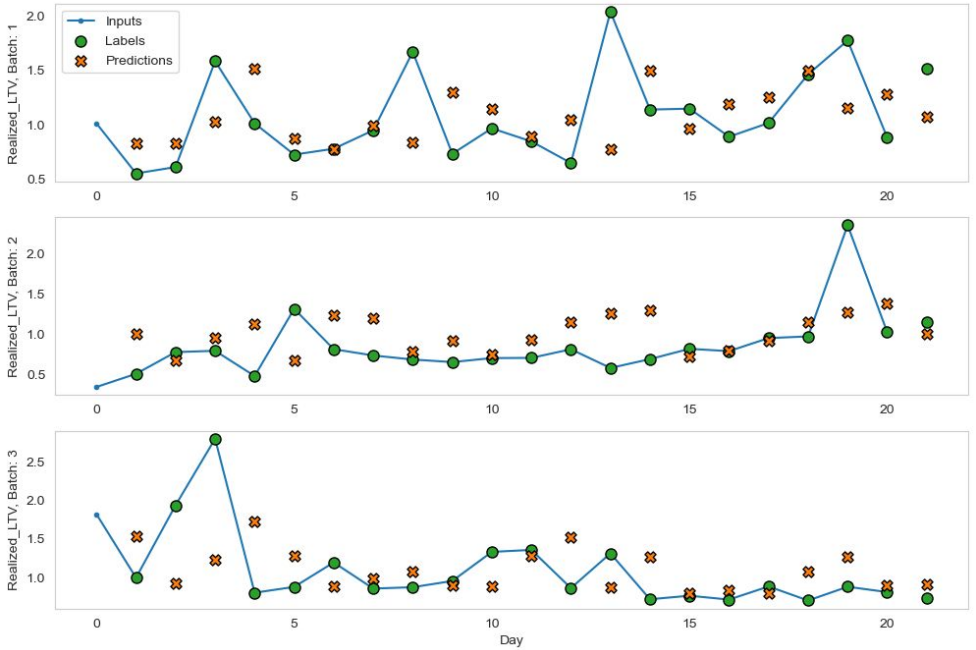
Linear and FFNN



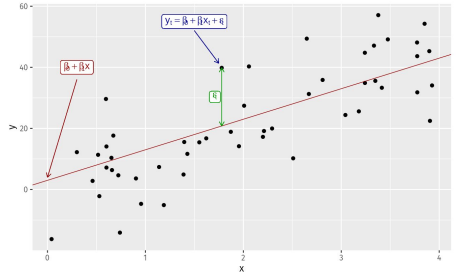
Experiments - Baseline

Model	Description	Hyperparameters
Baseline	Use previous day to predict following day	N/A

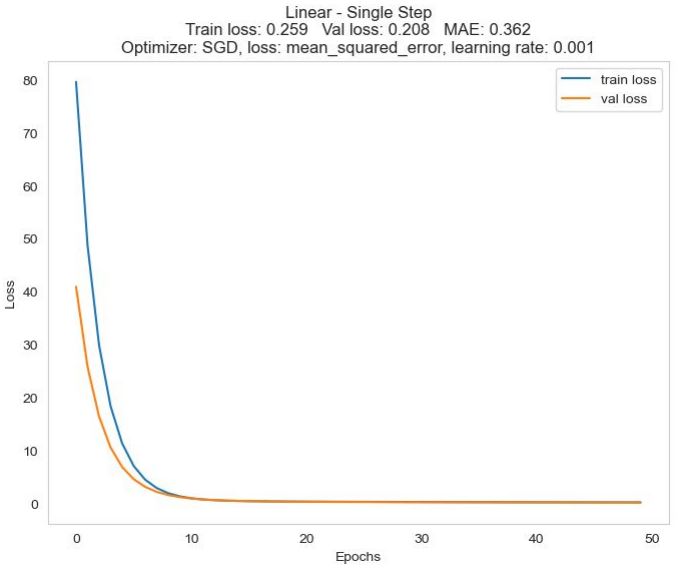


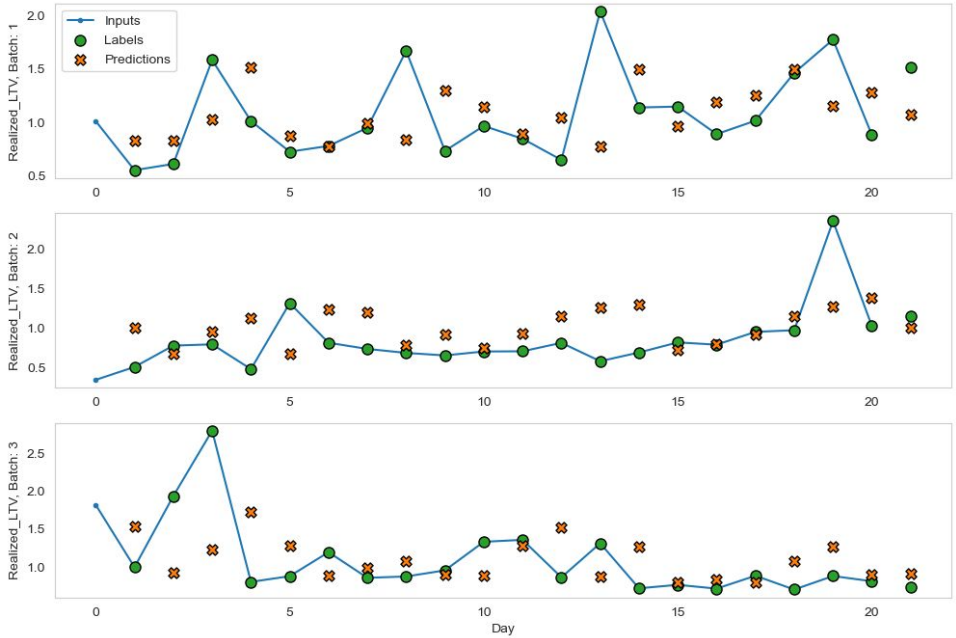


Experiments - Linear Regression



Model	Description	Hyperparameters
Linear	Linear Regression with bias. Kernel and bias initializer set to ones	Optimizer: SGD Loss Function: MSE Learning Rate: 0.001



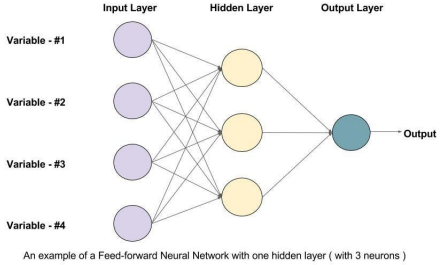


Tuning - Linear

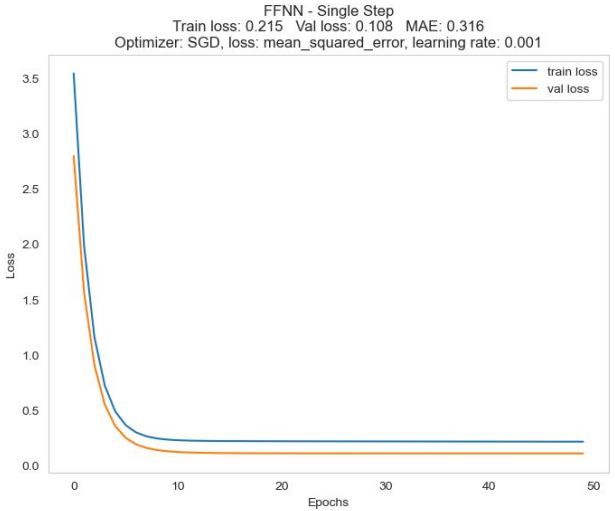
optimizer	loss_type	learning_rate	loss	val_loss	mean_absolute_error
Adam	mean_squared_error	0.100	0.205	0.118	0.301
SGD	mean_squared_error	0.010	0.206	0.120	0.301
		0.001	0.210	0.133	0.302
Adam	mean_absolute_error	0.010	0.286	0.201	0.286
SGD	mean_absolute_error	0.010	0.290	0.207	0.290
Adam	mean_squared_error	0.010	1.071	0.218	0.610
	mean_absolute_error	0.001	0.371	0.226	0.371
		0.100	0.372	0.226	0.372
SGD	mean_absolute_error	0.001	0.298	0.229	0.298
		0.100	0.725	0.561	0.725
Adam	mean_squared_error	0.001	3.411	0.789	1.136
	mean_absolute_percentage_error	0.001	20782328.000	38.559	0.780

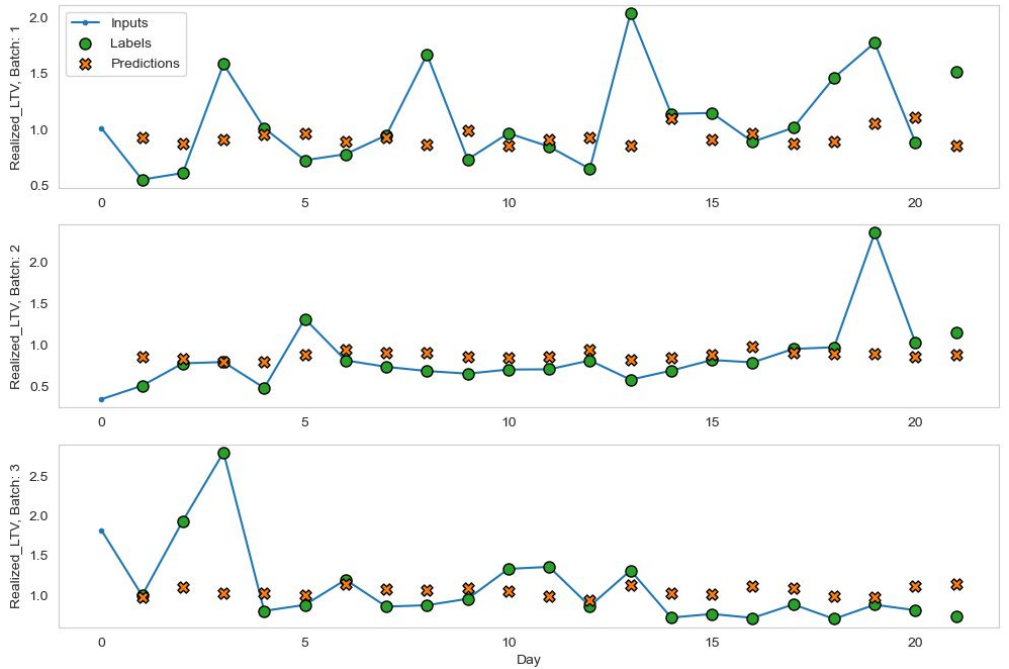
Experiments - Feed Forward NN

Model	Description	Hyperparameters
FFNN	2 hidden layers with no dropout	Optimizer: SGD Loss Function: MSE Learning Rate: 0.001 Activation: tanh Units in 2 layers: 32, 16



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

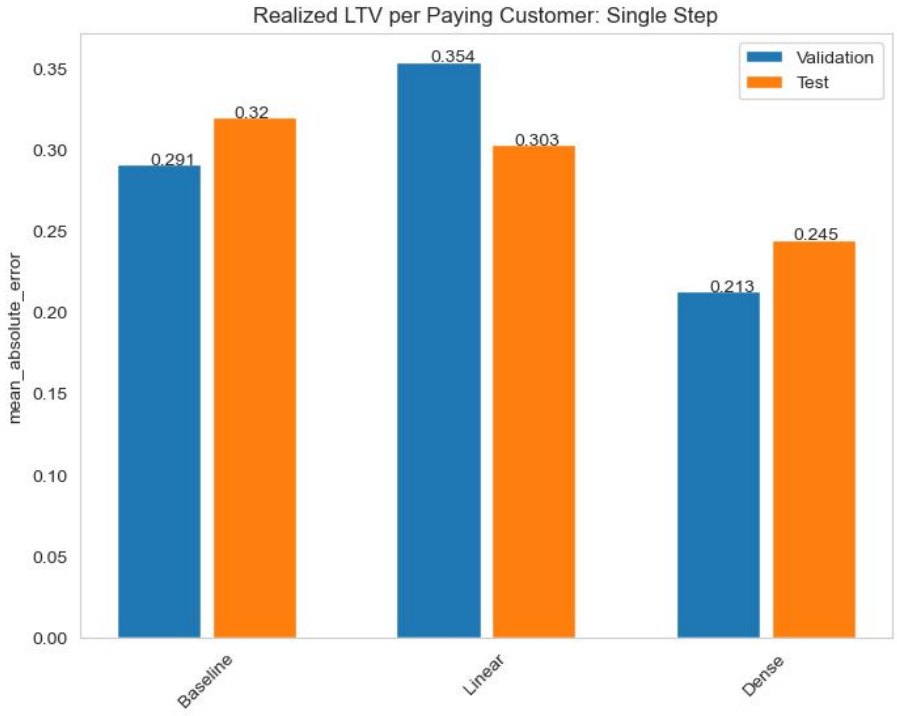




Tuning - Feed Forward NN

optimizer	loss_type	activation	layer_1_units	layer_2_units	learning_rate	loss	val_loss	mean_absolute_error
SGD	mean_squared_error	tanh	32	16	0.01	0.235	0.114	0.337
Adam	mean_squared_error	tanh	32	16	0.01	0.223	0.114	0.323
			32	32	0.01	0.206	0.115	0.304
			32	64	0.01	0.203	0.116	0.303
			64	16	0.01	0.201	0.120	0.303
			64	32	0.01	0.203	0.120	0.300
			64	64	0.01	0.206	0.123	0.302
SGD	mean_squared_error	tanh	64	64	0.01	0.205	0.128	0.299
			16	64	0.01	0.229	0.129	0.323
			64	32	0.01	0.206	0.132	0.306

Experiments - Single Step

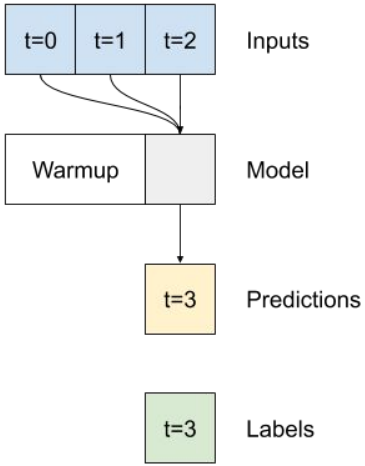


Model	Description	Hyperparameters Considered
Baseline	Use previous day to predict following day	N/A
Linear	Linear Regression with bias and kernel and bias initializer set to ones	Optimizer: SGD Loss Function: MSE Learning Rate: 0.001
FFNN	2 hidden layers with no dropout	Optimizer: SGD Loss Function: MSE Learning Rate: 0.001 Activation: tanh Units in 2 layers: 32, 16

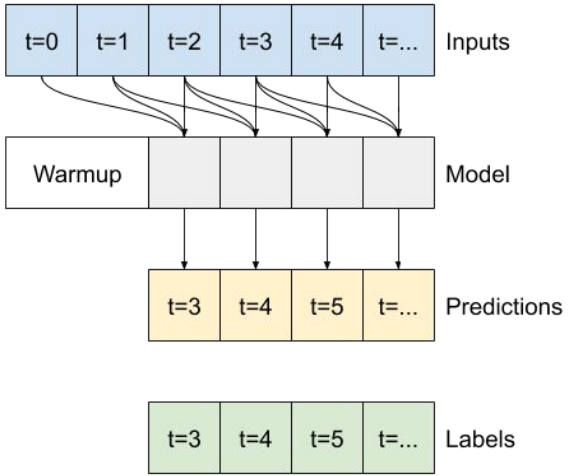
Multi-step

Take multiple days of input, and predict the next day

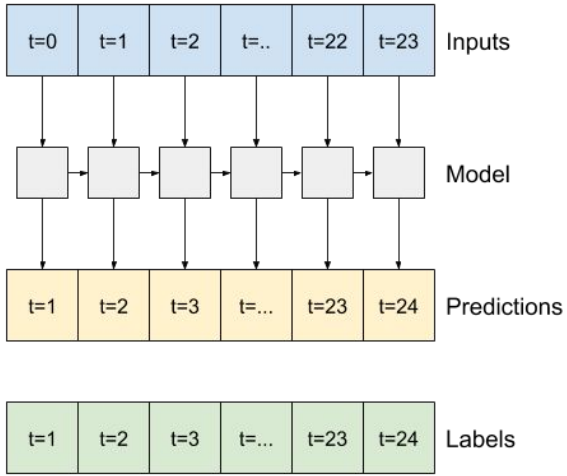
Linear, FFNN



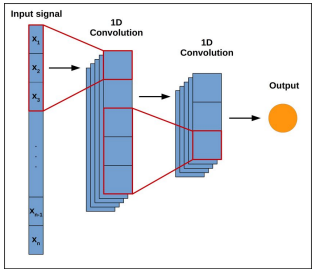
Convolutional NN



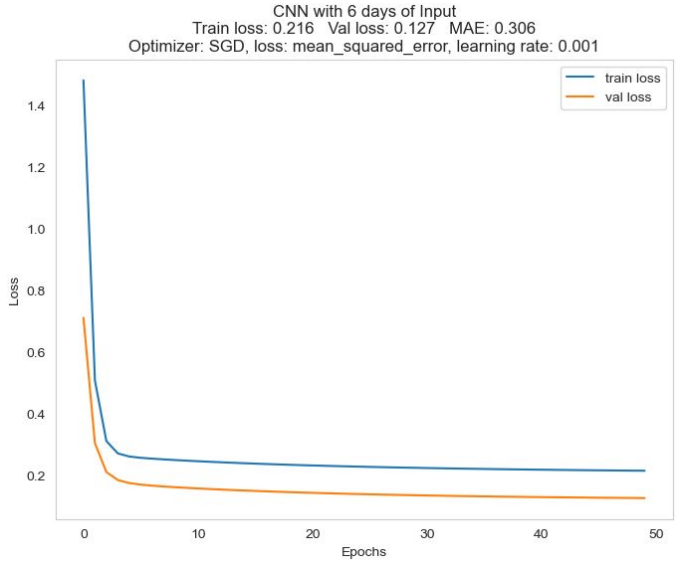
LSTM

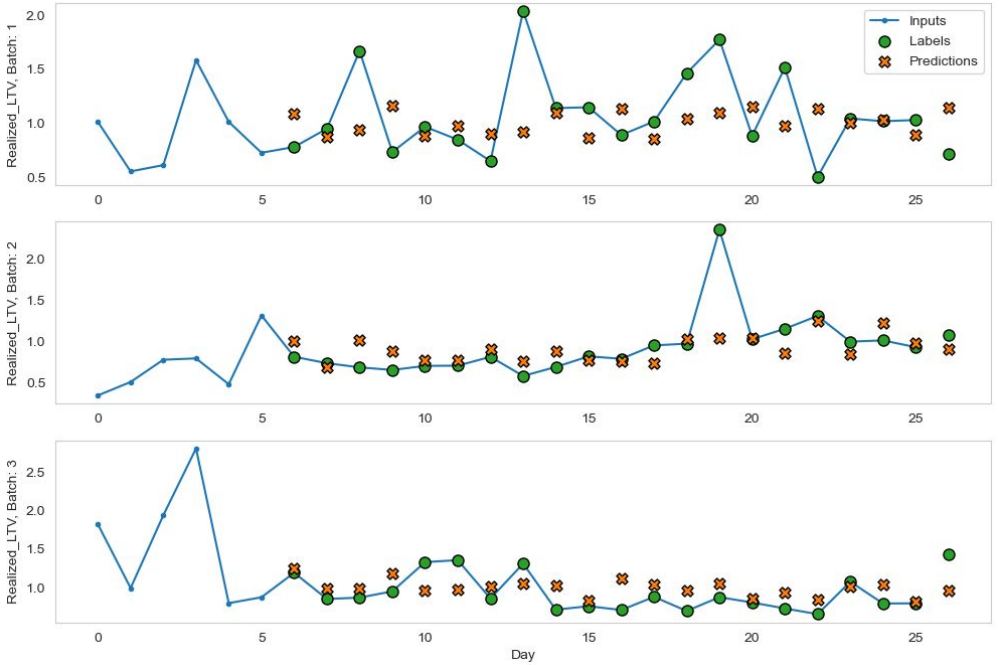


Experiments - Convolutional NN



Model	Description	Hyperparameters
CNN	Conv1d layer 1 dense hidden layer	Optimizer: SGD Loss Function: MSE Learning Rate: 0.001 Activation: tanh Filters and units: 64, 32 Kernel size: 4



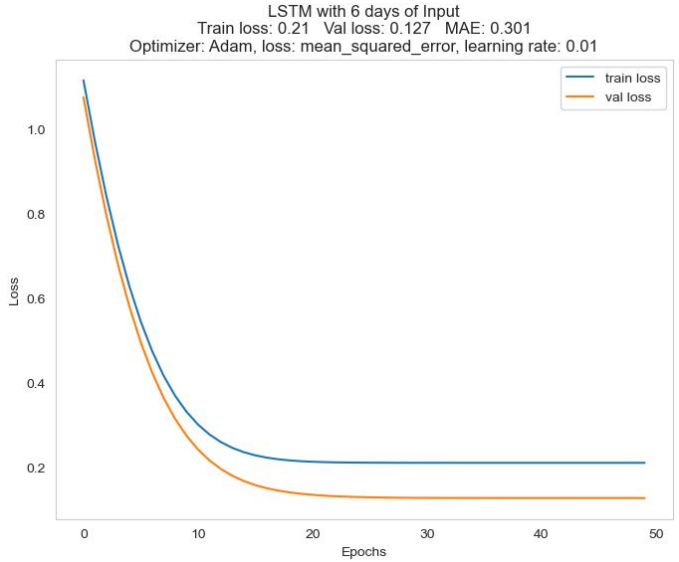


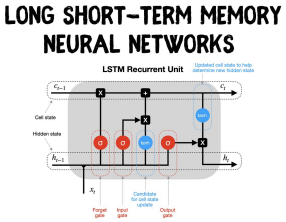
Tuning - CNN Multi-step

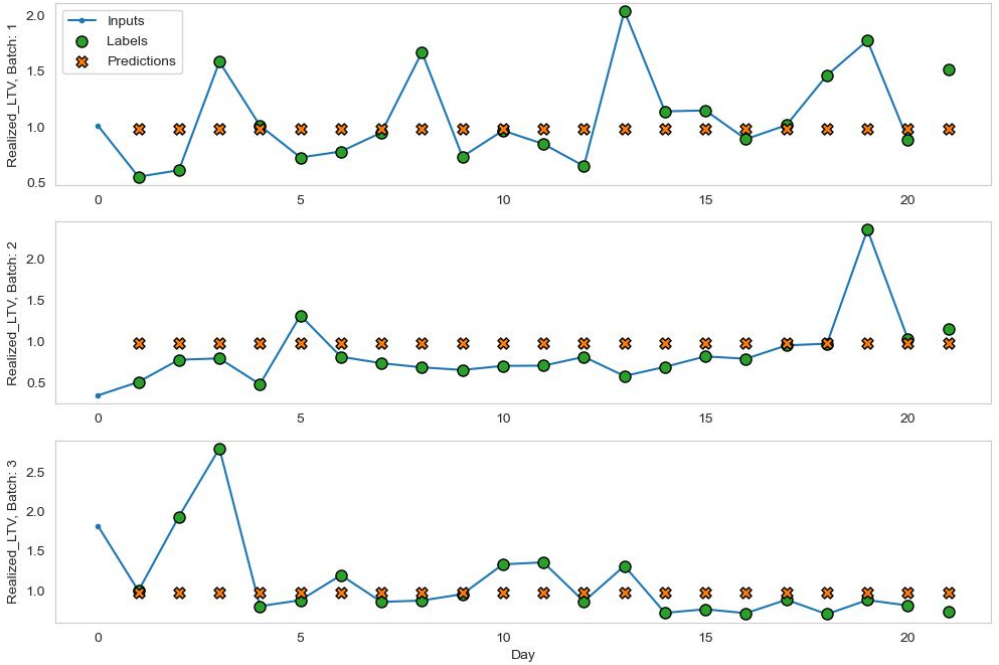
optimizer	loss_type	activation	layer_1_filter	layer_2_units	kernel	learning_rate	loss	val_loss	mean_absolute_error
SGD	mean_squared_error	tanh	64	32	4	0.001	0.217	0.115	0.308
Adam	mean_squared_error	tanh	16	32	3	0.001	0.200	0.120	0.291
			32	32	4	0.001	0.214	0.121	0.310
			32	64	4	0.001	0.197	0.123	0.290
			16	64	4	0.001	0.202	0.123	0.295
SGD	mean_squared_error	tanh	16	64	4	0.001	0.220	0.123	0.311
Adam	mean_squared_error	tanh	64	16	4	0.001	0.189	0.123	0.283

Experiments - LSTM

Model	Description	Hyperparameters
LSTM	Standard LSTM layer with units set to number of features	Optimizer: Adam Loss Function: MSE Learning Rate: 0.01 Activation: relu Return Sequences: True



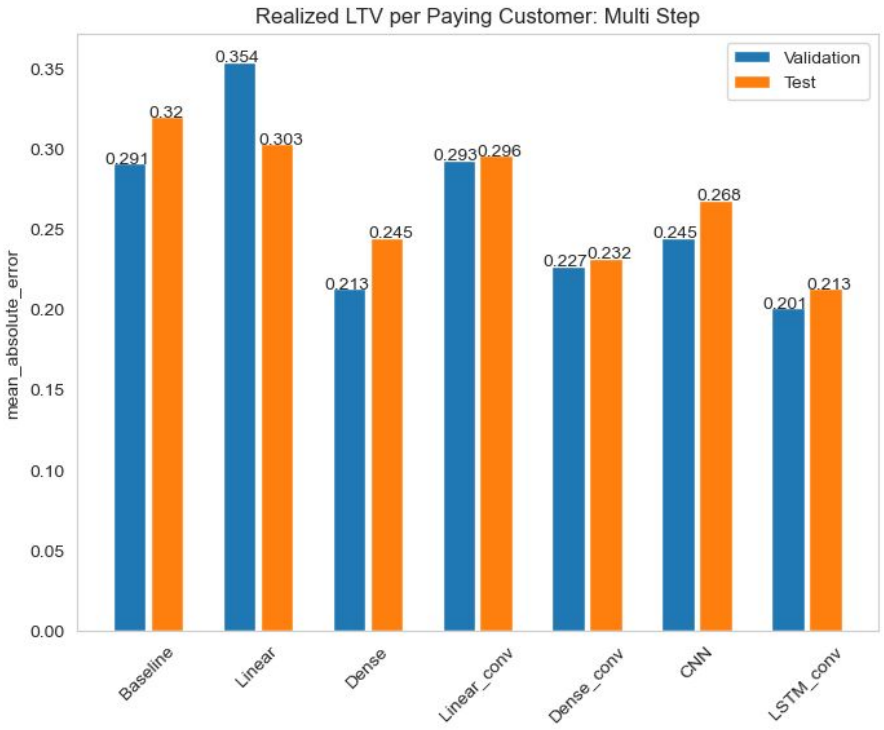




Tuning - LSTM Multi-step

optimizer	loss_type	activation	return_sequences	learning_rate	loss	val_loss	mean_absolute_error
SGD	mean_squared_error	tanh	FALSE	0.01	0.222	0.130	0.303
Adam	mean_squared_error	tanh	FALSE	0.01	0.235	0.134	0.311
			TRUE	0.01	0.232	0.149	0.322
SGD	mean_squared_error	tanh	TRUE	0.01	0.265	0.179	0.348
SGD	mean_absolute_error	tanh	FALSE	0.01	0.299	0.203	0.299
Adam	mean_absolute_error	tanh	FALSE	0.01	0.302	0.208	0.302
			TRUE	0.01	0.300	0.226	0.300
SGD	mean_absolute_error	tanh	TRUE	0.01	0.356	0.293	0.356

Experiments - Multi-step 6 days of Input

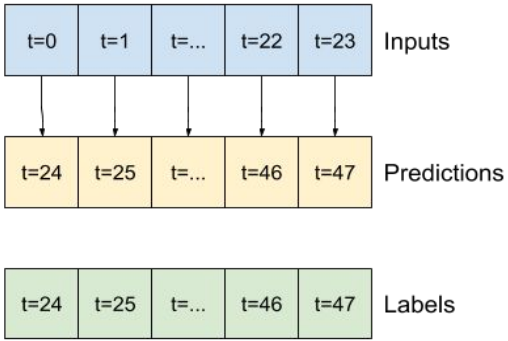


Model	Description	Hyperparameters Considered
Baseline	Use previous day to predict following day	N/A
Linear	Linear Regression with bias and kernel and bias initializer set to ones	Optimizer: SGD Loss Function: MSE Learning Rate: 0.001
FFNN	2 dense hidden layers with no dropout	Optimizer: Adam Loss Function: MSE Learning Rate: 0.001 Activation: tanh Units in 2 layers: 16, 16
CNN	Conv1d layer 1 dense hidden layer	Optimizer: SGD Loss Function: MSE Learning Rate: 0.001 Activation: tanh Filters and units: 64, 32 Kernel size: 4
LSTM	Standard LSTM layer with units set to number of features	Optimizer: Adam Loss Function: MSE Learning Rate: 0.001 Activation: relu Return Sequences: False

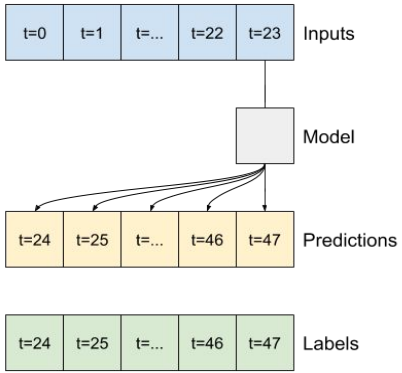
Multi-step forecast

Take 18 days of input, and predict 18 days out

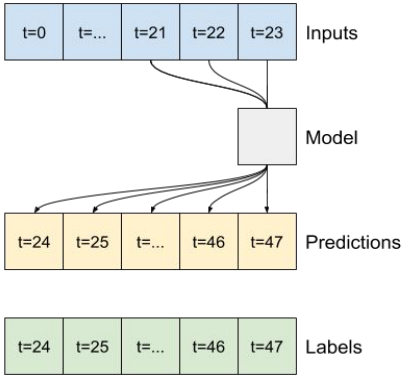
Baseline - Repeat



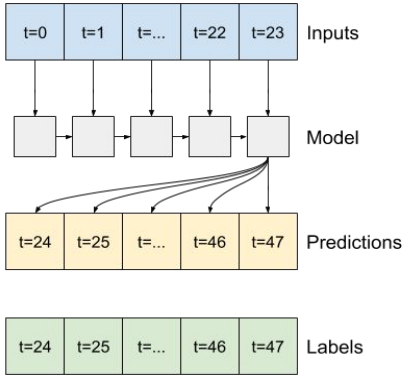
Linear, FFNN



CNN

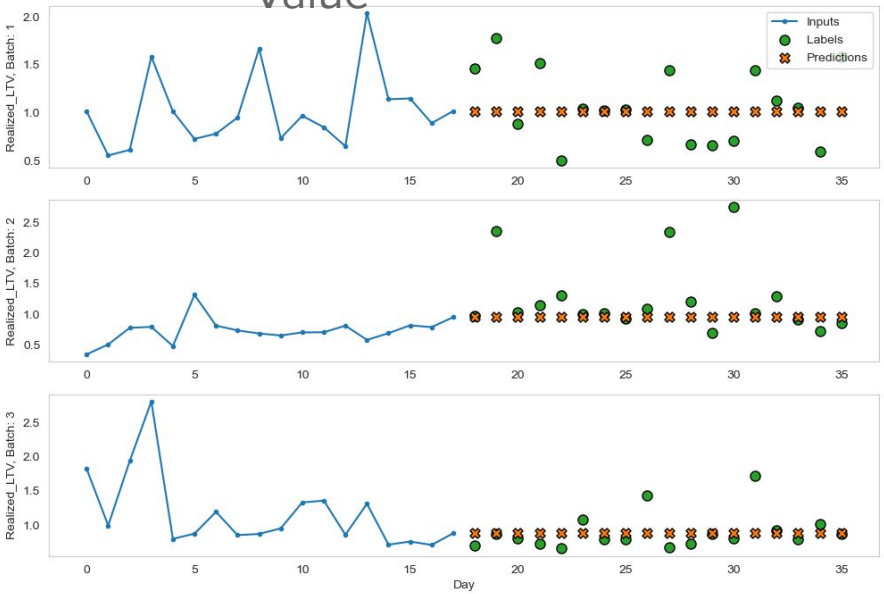


LSTM

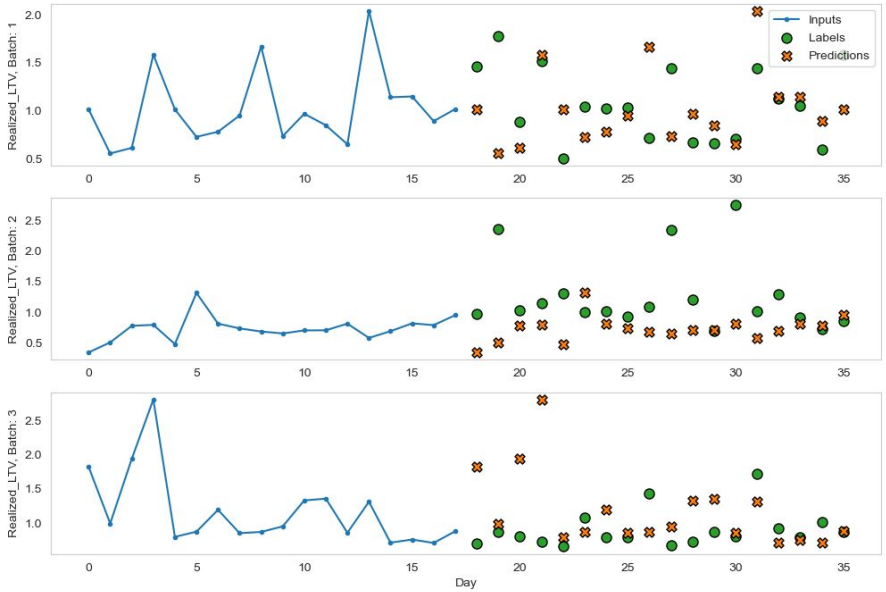


Baselines

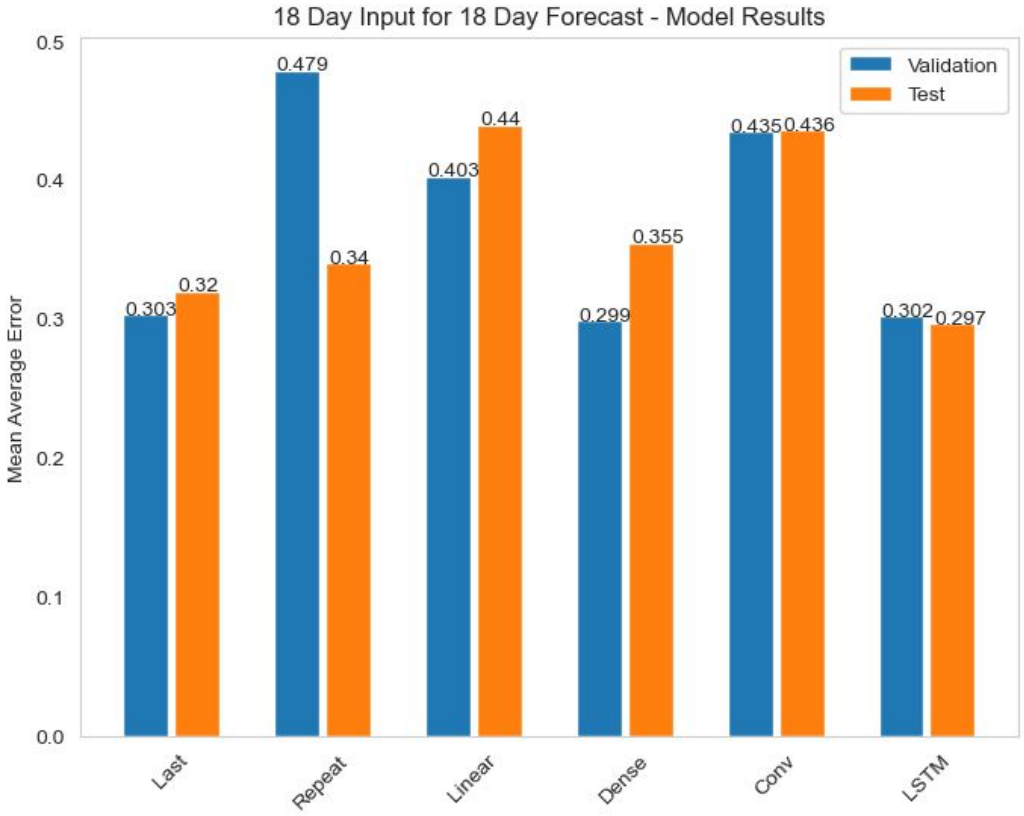
Repeat Last Value



Repeat Previous Pattern



Experiments - Multi-step Forecast

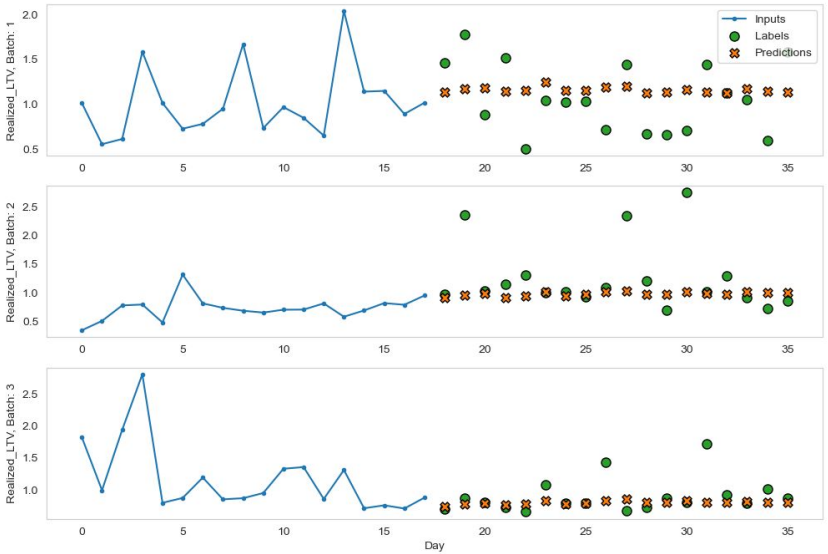


Model	Description	Hyperparameters Considered
Baseline	1. Use previous day to predict following day 2. Use previous input pattern to predict output pattern	N/A
Linear	Linear Regression with bias and kernel and bias initializer set to ones	Optimizer: Adam Loss Function: MSE Learning Rate: 0.001
FFNN	2 dense hidden layers with no dropout	Optimizer: Adam Loss Function: MSE Learning Rate: 0.001 Activation: tanh Units in 2 layers: 16, out_steps*features
CNN	Conv1d layer 1 dense hidden layer	Optimizer: SGD Loss Function: MSE Learning Rate: 0.001 Activation: tanh Filters and units: 128, out_steps*features Kernel size: 4
LSTM	Standard LSTM layer with units set to number of features	Optimizer: Adam Loss Function: MSE Learning Rate: 0.01 Activation: relu Return Sequences: True

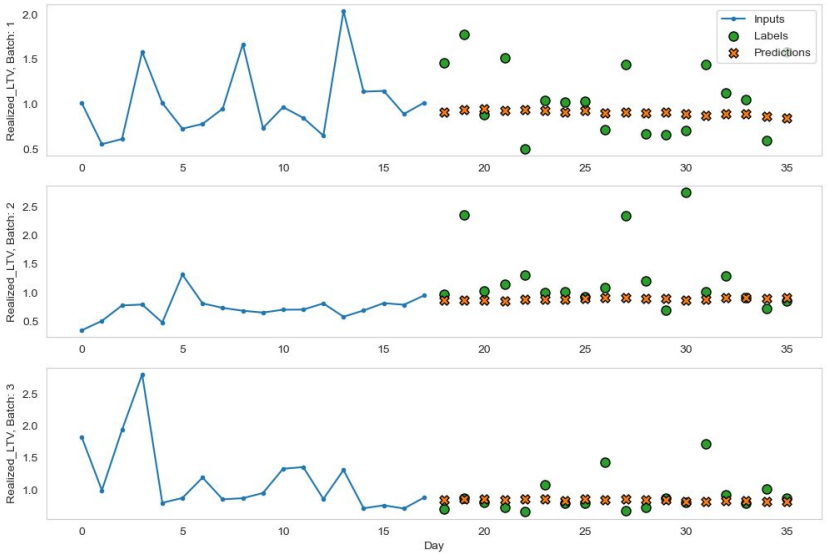
Why skip the tuning?

Because the best performers are predicting the average

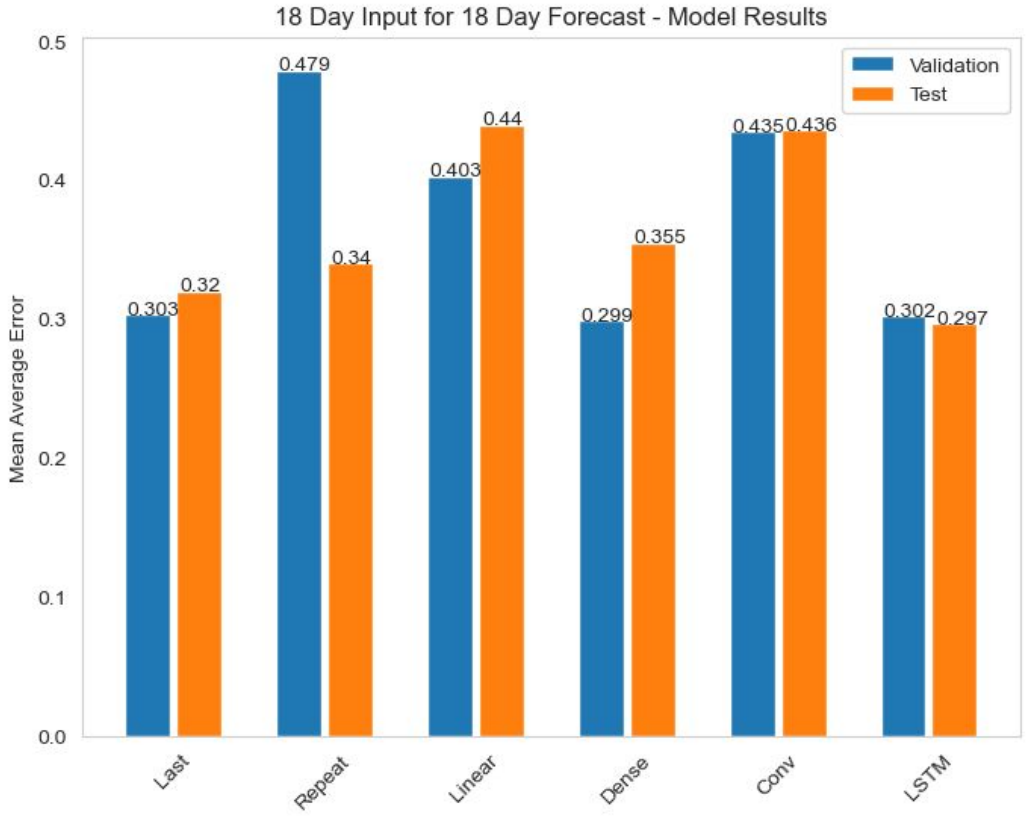
FFNN



LSTM



Conclusions - Results



Conclusions - Lessons Learned

- Initially, it was anticipated that Mean Absolute Error (MAE) would outperform Mean Squared Error (MSE), aiming to mitigate the impact of outliers. However, it became evident that the loss functions of MSE exhibited superior performance
- The training and validation losses have switched, likely attributed to high-variance data. It might have been beneficial to eliminate earlier, more volatile time periods during the initial app release, but the limited data at the outset posed a challenge. Another potential approach is to explore different rolling averages to assess whether they can mitigate the observed volatility
- Time series data presents a unique challenge and we do not believe we fully mined the utility.

Conclusions - Future Work

More user data - would request app specific data to get user behavior items in order to predict how long a user would stay with the app. This would provide much better predictions of the user LTV. Could also be used to predict likelihood of user converting from trial to paid subscription

Normalization - explore different normalization techniques with time series data

Other Models - Try other models like clustering, decision trees

Collect longer period of app data for model training

Thank you and Q&A

Code

https://github.com/rgreer-b/mids_w207_fall_2023_finalproject_GHLW/blob/main/more_code/modeling_v5.ipynb