Machine Learning and Deep Learning for Branch-Level Net Cash Flow Forecasting:

A Comparative Study with Stacking and Blending

Name: Anjalee Lokusooriya
Student number: C00313605

Supervisor: Dr. Rejwanul Haque

Machine Learning and Deep Learning for Branch-Level Net Cash Flow Forecasting:

A Comparative Study with Stacking and Blending

Anjalee Lokusooriya
Department of computing
South East Technological University
Carlow, Ireland
anjalee.u.lokusooriya@gmail.com

ABSTRACT

The paper is an attempt to compare the relative success of machine learning (ML) and deep learning (DL) models in branch-wise, anterior estimate of the cash balance per day (over three years), based on operation related information and macro-economic data. XGBoost (XGB) and LightGBM (LGBM) were optimized via feature selection and hyperparameter tuning, and then ensembled in blending and stacking ensembles.

The highest performance was obtained with a combined model of 60% XGB and 40% LGBM (MAE: 81,251.71; RMSE: 1,016,417.78; R^2: 0.9334), which surpassed single models and stacked versions. Deep learning trials using Long Short-Term Memory (LSTM) and Temporal Convolutional Networks (TCN) with residual and attention mechanisms achieved competitive yet inferior accuracy.

Branch-wise buffers were implemented on the ultimate model, and a Streamlit app was created for deployment. Outcomes validate the 60% XGB – 40% LGBM combination as the best forecasting method, with deep learning showing promise for further enhancement.

KEYWORDS

Cash flow forecasting, Machine learning, Deep learning, LSTM, Temporal Convolutional Networks, Gradient Boosting, Model Blending, Model Stacking

1 Introduction

1.1 Background and Context

Successful cash resource management remains a key operational issue for retail banking organizations. Maintenance of optimal cash holdings at branches is crucial for meeting daily customer demands, reducing holding costs, and maximizing overall financial efficiency. Both excessive and low cash holdings at branches can lead to heightened security risks, higher operational costs, or disruptions in services. The unpredictability of customer withdrawal patterns, seasonal demands, and geographic cash usage patterns contribute to the complexity of cash management in financial institutions (Kumar, et al., 2024).

Conventional cash forecasting techniques in banks depend excessively on historical averages, or naive rule-based models. Although these techniques provide a fundamental level of predictability, they tend to neglect non-linear trends, temporal dependencies, and changes in the behavior of cash demand. This deficiency prompts the need to search for sophisticated data-driven models with the ability to grasp complex patterns in the historical cash flow data (Zaragoza & Mota, 2016; Venkiteshwaran, 2011)

Recent developments in ML and DL provide promising solutions to overcome these limitations. Ensemble models like XGBoost and LightGBM have been extremely effective in structured time series forecasting because they can effectively deal with non-linearity and missing values and have strong performance on tabular data (Nichani, et al., 2024; Özlem & Tan, 2022). In addition, deep learning architectures like Long Short-Term Memory and Temporal Convolutional Networks are becoming increasingly popular in time series modelling for their capability in modelling sequential dependencies and temporal dynamics (Vangala & Vadlamani, 2020).

1.2 Problem Statement

Notwithstanding the pivotal position of cash forecasting in bank operations, most institutions continue to utilize inadequate, static forecasting methods that fail to respond to shifting transaction trends or local cash dynamics. This creates regular disparities between forecasted and actual cash requirements, which translate into financial losses, customer discontent, and operational vulnerabilities. With cash flow patterns growing ever more intricate, the demand for effective predictive models with interpretability, scalability, and precision is becoming increasingly urgent.

1.3 Research Aim and Objectives

The primary aim of this research is to develop, evaluate, and compare advanced machine learning (ML) and deep learning (DL) models to forecast optimal cash balances at the bank branch level.

Specifically, this dissertation will:

- Build individual XGBoost and LightGBM models and evaluate their performance.
- Implement model blending and stacking techniques using these models to test ensemble effectiveness.
- Develop deep learning models using LSTM and TCN architectures and compare their predictive performance against ensemble methods.
- Explore hybrid ensemble strategies involving LSTM and TCN to assess their combined impact on accuracy.
- Identify the strengths, limitations, and optimal use cases for each modelling approach.

1.4 Research Questions

1. To what extent can machine learning models (XGBoost, LightGBM, and ensemble methods) improve the accuracy of daily cash balance forecasting for bank branches compared to traditional forecasting techniques?

2. Does model stacking and blending of gradient boosting algorithms result in significantly better forecasting performance than individual models alone?

3. How do deep learning approaches (LSTM and TCN) perform relative to ensemble tree-based models in forecasting cash balances, and under what conditions are they most effective—individually or as part of a hybrid ensemble?

1.5 Methodological Overview

To address the above research questions, this study will follow a multi-phase methodology:

- Data preprocessing and feature engineering, including lagged features, rolling statistics, and temporal indicators, based on real-world transaction data.

- Model development, beginning with separately optimized XGBoost and LightGBM models, followed by their combination through weighted blending and meta-model stacking using Gradient Boosting Regressor.

- Deep learning implementation, using LSTM and TCN networks trained on the same temporal features.

- Hybrid deep learning ensemble development, combining LSTM and TCN in blended and stacked configurations.

- Evaluation and comparison, of all models using standard regression metrics such as RMSE, MAE, and R2.

1.6 Scope and Limitations

The project scope is confined to the prediction of daily net cash flow for bank branches based on historical transaction data. The developed models are not put into production but are tested in a simulation setting. Real-time prediction constraints, cash forecasting at the ATM level, and integration with wider liquidity management systems are not addressed by the study. Nonetheless, the findings can be generalized to such applications.

2 Literature Review

This chapter provides an extensive literature review of recent research on cash balance forecasting and optimization in the banking sector, with a particular emphasis on the shift from conventional approaches to sophisticated ML and DL frameworks. Effective cash management is instrumental in ensuring liquidity, reducing idle funds, and maximizing operational efficiency in retail banking. With increasing digital transactions, there is growing pressure on financial institutions to precisely predict branch-level cash requirements. (Kumar, et al., 2024) underscores the inconsistency in daily reserve maintenance practices of banks, stressing the imperative role of data-driven forecasting models in enhancing financial readiness and mitigating operational risks.

Recent studies underscore the inadequacy of conventional forecasting techniques in sufficiently accounting for the complexities that attend modern banking activities. Conventional models, such as ARIMA, exponential smoothing, and linear regression, previously considered suitable for the course, show limitations in capturing non-stationary and nonlinear properties that are inherent in time series data derived from customer transactions. (Venkiteshwaran, 2011) explain that such classical statistical models tend to oversimplify the complexity of financial phenomena, thus resulting in limited responsiveness to evolving trends and exogenous shocks. In today's financial landscape, where transactional patterns are influenced by immediate considerations such as the use of digital channels, macroeconomic volatilities, as well as policy changes, these weaknesses call for the formulation of more responsive and adaptive forecasting techniques.

Machine learning models, including XGBoost, CatBoost, and LightGBM, are increasingly incorporated into financial applications because they model nonlinear relationships with more power and more flexibility in high-dimensional spaces. (Nichani, et al., 2024) show that gradient boosting algorithms considerably outperform traditional time series models and reveal greater robustness in predicting financial time series while also learning subtle interactions across predictors. Current models even allow for

automatic accommodation of missing values and are highly scalable, allowing for the potential for deployment in operational branch-level forecasting systems

Furthermore, there is emerging literature on the application of hybrid-based machine learning models for financial forecasting. (Li, 2023) present an innovative approach of hybridization by combining XGBoost with recurrent neural networks to forecast cash flows across multiple branches of a bank. The authors found that these hybrid models outperform the single-model baselines, reporting better specific accuracy and robustness in fluctuating transaction contexts. These results are consistent with a body of work that favors ensemble and hybridized approaches for time series forecasting across diverse areas in finance because they readily adapt to the unique context which occurred with various clusters of predictive power.

Ensemble methods, including bagging, boosting and stacking, are becoming more popular in high performance forecasting systems. (Archankul, et al., 2023) proposes application of a weighted ensemble method (Weighted Ensemble) by way of a blend of LightGBM, Catboost, and Ridge regression, for predicting financial time series. Their investigation suggests statistically important improvements in terms of error reductions, in addition to increased reliability of forecasts. The contexts where reliance on accuracy of forecast and resiliency to unexpected spikes in demand are greatest are banking contexts. Ensemble paradigms as an emerging methodology are most appealing due to their adaptive properties to data regimes or unique use case.

In parallel with developments in machine learning, deep learning—more specifically Long Short-Term Memory networks—has become popular for time series modeling with long-term dependencies. (Vangala & Vadlamani, 2020) and others have demonstrated the benefits of LSTM in modeling sequential patterns, and it proves to be very successful in use cases like ATM or branch cash flow prediction. Recent research of (Nichani, et al., 2024), investigates the application of multivariate LSTM models that account for both transactional and external contextual information (e.g., holidays, weather conditions, economic indicators). These models show consistent improvement over traditional and ML-only approaches in forecast accuracy, with a particular positive impact on volatile or seasonally sensitive time series.

More recently, Temporal Convolutional Networks (TCNs) have been introduced as a competitive alternative to recurrent models for time series prediction. TCNs use causal, dilated convolutions to model sequential data without the vanishing gradient limitations that LSTMs are often prone to (Bai, et al., 2018). These models are faster to train, allow parallel computation, and have the capacity to model both short- and long-term dependencies effectively. In finance, TCNs have been demonstrated to outperform LSTM models in some settings, especially where temporal locality and large sequence depth matter. Their increasing use reflects a trend toward convolution-based temporal modeling, which is well-suited to structured and high-frequency transactional data prevalent in banking.

However, while there have been a few studies exploring the relevance of hybrid or comparative modes of ML, significant gaps remain in academic literature in terms of their relevance to, and existence in, daily branch-level cash forecasting. Specifically, most studies evaluate stock prediction, ATM-level cash orders or high-level liquidity assessments with few examining the relevance of ensemble ML models and DL models to operational questions concerning branch-level management.

To this end, the current study empirically compares tree-based ensemble models with deep learning methods (LSTM and TCN) using real bank branch data. There are multitudes of ways to evaluate the two types of forecasting systems (such as forecasting accuracy, interpretability or operational viability) and this study seeks to contribute to both the academic understanding and applied work for building cash forecasting systems that are robust and demonstrate high levels of performance. This comparative analysis should make use of the body of literature surrounding operational definitions in banking and the differences between operational concerns and academic theoretical ideas to provide useful information for decision makers in the banking sector, as well as for data scientists looking to modernize cash management in banking.

In conclusion, the identified literature outlines a definite move from classic, static models, to dynamic, data-driven models using ML and DL. XGBoost and LightGBM represent ensemble learning and are remarkably predictive; even better if they are stacked or blended together. DL model variants like LSTM and TCNs have the benefits of remembering the temporal dynamics of cash forecasting and modeling seasonalities, and in saying that deep learning models can be complex and are difficult to implement appropriately. The financial services industry is moving slowly toward a data-driven business sector which indicates relying upon both ML and DL models will be the best path forward to improve accuracy and resilience in cash forecasting at the branch level.

3 Methodology

3.1 Overview of Research Design

This research takes an expansive, quantitative, and data-driven research design approach to predict daily net cash flow at the level of individual bank branches. The main objective is to build and compare the performance of conventional machine learning models (XGBoost and LightGBM) and deep learning models using Long

Short-Term Memory and Temporal Convolutional Network. This modelling strategy is taken to determine not just predictive performance but also practical viability and interpretability for application in retail banking operations.

The motivation of this research design is rooted in the inherent properties of cash flow forecasting issues within the banking industry. Cash demand displays non-linear temporal dynamics that are driven by customer behavior, regional economic trends, calendar events, and seasonality. Traditional statistical forecasting methods consequently tend to be inadequate in modelling the dynamic, heterogeneous and non-stationary nature of the underlying time series. ML models like XGBoost and LightGBM can model intricate interactions and non-linear relationships between engineered features, whereas LSTM and TCN models provide the potential for modelling long-term dependencies in sequential data.

The research workflow is organized into a series of consecutive phases: data procurement and exploration, preprocessing and feature engineering, training and hyperparameter tuning of models, comparative performance analysis and interpretation of results. Every phase is designed to achieve robustness, reproducibility and scalability in a practical financial setting.

The selection of a comparative framework between ML and DL models enables the research to make an empirical contribution to the current academic and practitioner controversy surrounding when and where deep learning models provide a performance advantage compared to carefully tuned ensemble methods. Based on systematic experimentation and comparison, the research aims to determine the most appropriate modelling strategy for bank-level cash prediction while considering not just accuracy but also computational efficiency, explainability and deployability. Additionally, the integration of LSTM and TCN into a blended or stacked ensemble allows the study to explore the synergistic potential of combining recurrent and convolutional neural networks for temporal prediction tasks.

Additionally, there is a strong focus on temporal cross-validation and the prevention of lookahead bias to maintain the integrity of time series data. The implementation is based on a uniform experimental pipeline with set random seeds, hyperparameter tuning with time-aware splitting techniques, and feature selection in order to provide a fair and reproducible model comparison. This methodological strictness is aimed at satisfying practical and regulatory requirements as they are expected to be found in banking setups, hence closing the gap between academic research and applied financial forecasting products.

3.2 Data Description

The data utilized in this research was gathered from a prominent finance company in Sri Lanka with an island-wide network of around 200 branches. The data covers a total three-year span from 1st April 2022 to 31st March 2025, encompassing a broad spectrum of transactional, geographical, and operational data across various branches and regions.

The core dataset consists of the following daily-level variables:

- TransactionDate: The specific date of transactions
- BranchID: A unique identifier for each branch
- BranchName, BranchCode: Textual and coded identifiers
- District, Province, Latitude, Longitude: Geographical coordinates of each branch location
- TotalCreditAmount: Sum of all credits for the branch on a given day (in LKR)
- TotalDebitAmount: Sum of all debits for the branch on a given day (in LKR)
- CustomerVisits: Number of customer visits per branch per day

These features yield a multidimensional perspective on branch operations, such as transaction volume, geographical dispersion and customer activity patterns. The data was utilized to calculate the NetCashFlow variable, which equals the daily difference between total credit and total debit values, and was the main response variable for modeling.
Along with transaction data at the branch level, two supporting datasets were incorporated to enhance the modeling features:

- Monthly Inflation Rates: Based on the Central Bank of Sri Lanka, these rates reflect the National Consumer Price Index (NCPI). These values reflect current macroeconomic conditions and match the transaction date of each transaction.
- Mercantile Holidays: A carefully prepared list of public and specified bank holidays, used to generate binary indicators for non-working days, known to have a significant influence on cash flow patterns.

To provide temporal consistency, all of the data sources were joined on appropriate date-based keys and inflation values were forward-filled to achieve daily granularity. These new features allowed the model to capture both micro-level transactional behavior as well as macroeconomic factors.

3.2.1 Data Ethics and Consent.

Ethical issues were treated as a priority in documenting the collection and processing of the data. Written consent was received from the finance company contributor to the dataset, explicit permission which allowed the use of anonymized data for academic research. There was no personal identifiable information (PII) associated with the data, and the branch identifiers were treated as confidential.

Data was stored in accordance with security protocols, and data processing and dissemination, was aligned with security protocols,

maintaining the integrity of the data, and limiting data misuse. Ethical approval and the acknowledgement of the consent agreement were submitted to academic or other research personnel involved in this study.

3.3 Feature Engineering

The process of feature engineering was a critical step of the proposed study, aimed to supplement the raw dataset of purchases of the transactional data with derived variables that could capture the underlying temporal, behavioral and spatial trends that affect cash flow. Due to the multidimensionality of the dataset used, a wide range of features were obtained, which are organized into thematic categories.

To capture seasonal patterns and temporal dependencies in branch transactions, multiple feature groups were engineered:

- **Calendar Features:** DayOfWeek, Month, Year, IsWeekend, IsHoliday, IsNonWorkingDay, and indicators for month start/end (e.g., IsFirst5Days, IsLast5Days).
- **Lag Features:** Credit, debit, and customer counts lagged by 1, 2, 3, 7, and 14 days to model short- and medium-term trends.
- **Rolling Statistics:** 3-, 7-, and 30-day rolling mean, standard deviation, max, and min for credit, debit, and customers to capture seasonality and volatility.
- **Z-scores & Ratios:** Rolling-window Z-scores, deviation-from-mean ratios, and volume-adjusted metrics such as Debit_to_Credit_Ratio, CashIn_Per_Customer, and CashOut_Per_Customer for anomaly detection and normalization.
- **Delta Features:** Daily differences (Delta_Credit, Delta_Debit, Delta_Customers) to measure change dynamics.

Target Variable: The derived continuous feature Net Cash Flow (Total Amount of Credit Transactions - Total Amount of Debit Transactions) was used as the primary response variable.

The engineered features were developed to serve both tree-based ensemble models and sequence-based deep learning models. The properties of the features (e.g. lags, rolling statistics, and temporal flags) provided all the models, with useful patterns and more contemporary behavioral dynamics to focus on predictive performance. All features were checked for multicollinearity prior to being included in the models.

3.4 Data Preprocessing

3.4.1 Handling Missing Values.

Lag and rolling features introduced missing values at the start of each branch's series. These rows were dropped instead of imputed to preserve temporal integrity, which is crucial for sequence-based models like LSTM and TCN. The dataset size ensured negligible impact on training availability.

3.4.2 Categorical Encoding.

BranchID and BranchCode (integer IDs) were retained directly. BranchName was mapped to BranchID and dropped. District and Province were label encoded for XGBoost/LightGBM, while LSTM/TCN used one-hot encoding for these fields only, avoiding high-cardinality one-hot encoding for branch identifiers to reduce sparsity and complexity.

3.4.3 Temporal Train-Test Split.

A time-based split was applied to avoid leakage:

- **Train:** 1 Apr 2022 – 31 Dec 2024
- **Test:** 1 Jan 2025 – 31 Mar 2025

LSTM/TCN used rolling validation within the training period for tuning and early stopping.

3.4.4 Correlation Analysis and Redundancy Elimination.

Highly correlated features (Pearson > 0.95) were filtered by retaining the one with higher correlation to NetCashFlow, improving interpretability and reducing redundancy.

3.4.5 Data Normalization for DL Input.

Numeric inputs for LSTM and TCN were Standard scaled to [0,1], with scaling parameters derived from the training set only to prevent leakage.

These preprocessing steps made the dataset time consistent and suited to the model diverse needs inherent in the models of both tree-based ensemble architectures and deep sequential neural networks.

3.5 Model Development

This section offers an exhaustive overview of the methodical development of machine learning and deep learning models created for cash balance forecasting, including XGBoost, LightGBM, LSTM and TCN. Each model was carefully created by following a recursive procedure of data preprocessing, hyperparameter tuning, and performance evaluation. This section first focuses on the development of the tree-based models and then on their combination through ensemble techniques like blending and

stacking. The creation of deep learning models is discussed in the next section.

*3.5.*1 *Tree based models.*

3.5.1.1 XGBoost Model Development.

The model development of XGBoost started with the loading of preprocessed data that contained all the engineered features from transaction-level, temporal, and contextual information. As the dataset did not contain any initial missing values, feature engineering was done prior to preprocessing, and the data was then split based on a time-based cutoff to maintain the temporal order to allow for strong validation.

An important step in preprocessing was the detection and reduction of multicollinearity between features. Pairs of highly correlated features (Pearson correlation coefficient > 0.95) were identified, and the variable less correlated with the response variable (NetCashFlow) in each pair was eliminated. This enhanced the interpretability of the model and avoided redundancy in training.

A baseline model was first trained with default XGBoost parameters to provide a baseline. Then a wide RandomizedSearchCV was performed with all the features to search a large hyperparameter space, covering parameters like learning_rate, max_depth, n_estimators, subsample, and colsample_bytree. For further regularization and generalization improvement, more rounds of random search were performed by adding regularization terms (reg_alpha, reg_lambda). After identifying good parameter ranges, they were fine-tuned using GridSearchCV for more targeted tuning.

Feature importance was calculated, using the gain metric, which calculates the amount each feature contributes to the model's predictions. Features are then ranked, and multiple threshold values for gain are tested in order to select an optimal set of features. A threshold value of 0.5% gain represented the suitable threshold value for trade-off between modelling goodness of fit and simplicity, resulting in simpler models while still being relatively interpretable.

In this experiment, the hyperparameter tuning pipeline was conducted, utilizing RandomizedSearchCV followed using GridSearchCV for the new feature set. The performance of the models from both the original feature set and those from threshold selection was measured. The selected feature model (gain > 0.5%) was shown to have minimal MAE and RMSE. TimeSeriesSplit was used for all validation phases, to avoid leakage and considered the sequential nature of the data.

Final evaluations were calculated at the branch level also, in terms of MAE by branch to consider performance stability across features. Predicted plots were developed by randomly selecting branches, to allow visual assessment of forecast accuracy and model response.

3.5.1.2 LightGBM Model Development.

The development of the LightGBM model followed the same workflow as XGBoost, except that certain tuning strategies were specific to the LightGBM library. This process started by training an initial model with all the features. This was followed by an exhaustive RandomizedSearchCV to search for important hyperparameters like num_leaves, max_depth, learning_rate, and n_estimators.

For improving accuracy and controlling overfitting, further rounds of GridSearchCV were run to optimize subsample and colsample_bytree parameters, which control the ratio of rows and features that are sampled during training. Upon finalizing optimal values for these parameters, the final composite model was trained with all optimized settings.

An alternative tuning approach was also explored using Optuna, which is a Bayesian optimization library to tune hyperparameters in complex hyperparameter spaces. But in this case, Optuna tuning doesn’t serve to confirm the stability of the hyperparameter choices found via grid and random search.

Feature importance was once more assessed on the 'gain' metric. Various thresholds (0.5%, 0.01%, 0.05%) were tried to identify the best cutoff for feature selection. A cutoff of 0.05% gain gave a good trade-off between performance and model interpretability. The feature subset was then employed to redo the tuning process, resulting in a more compact model with no loss in accuracy.

Like XGBoost, LightGBM's ultimate performance was evaluated with branch-level MAE. Visualizations of predictions for randomly selected branches also complemented the quantitative analysis.

3.5.1.3 Model Blending (XGBoost + LightGBM).

To take advantage of the complementary strengths of LightGBM and XGBoost, a model blending approach was used. The predictions of the two optimized models were blended using weighted averaging. Different weight combinations were tried, and a blend of 60% XGB and 40% LGBM resulted in the lowest overall MAE and RMSE. This result highlighted the advantage of averaging predictions from different models in decreasing variance and learning different patterns in the data.

3.5.1.4 Model Stacking.

In addition to blending, stacking was used to construct a meta-learning framework capable of learning nonlinear relationships among base model predictions. The optimized XGBoost and

LightGBM models were used as base learners, and their predictions were fed as input features to the meta-model.

Two meta-models were explored: Ridge Regression, which offered a linear combination of the base predictions with regularization, and GradientBoostingRegressor, which enabled learning complicated residual patterns. The meta-models were tuned using RandomizedSearchCV and GridSearchCV.

To enhance the learning capacity of a stacked model, a combination of meta-predictors was designed using predictions of the base models by calculating an average value, difference, ratio, minimum, and maximum of these. The meta-learner was able to understand and clear the difference between the base models with the help of these created features to a greater extent. Based on the experimental settings that have been tested, it was observed that the GradientBoostingRegressor meta-model paired with all five extended features provided the best and the most stable results.

The model training was then carried on with these expanded meta-features resulting in further increment of accuracy at the overall scale hence proving the merit of using higher-order-feature engineering in the stacked ensembles.

3.5.2 Deep Learning models.

3.5.2.1 Long Short Time Memory (LSTM).

The deep learning model based on LSTM was created to learn sequential dependencies and temporal dynamics in cash flow data. It started with loading and preprocessing the data to be compatible with the LSTM architecture. Different sequence lengths (7-day, 30-day, and 60-day) were experimented and the 60-day sequence performed best. Log transformation and standard scaling were also tried.

The baseline LSTM model included a branch embedding layer, a single LSTM layer, dense layers with ReLU activation, dropout as regularization, and an output layer with linear activation. This baseline structure was incrementally improved.

A more advanced model incorporated the following architectural elements:

- **Branch Embedding:** Encoded BRANCHID into an 8-dimensional vector for concatenation with LSTM outputs.
- **Stacked LSTM:** Two layers (64 units → sequences, 32 units → final state) with dropout in between.
- **Concatenation:** Merged LSTM output with flattened branch embedding.
- **Dense Layers:** 64 and 32 units (ReLU) with dropout for regularization.
- **Output:** Single dense unit predicting NetCashFlow.

Later experiments added sophisticated elements like residual connections, attention mechanisms, multi-head attention, batch normalization, and early stopping. These elements were tested both in isolation and in different combinations.

A BiLSTM model was also tried, which captured both future and past temporal contexts. It employed a 64-unit BiLSTM layer, dropout, concatenation with the branch embedding, and a fully connected dense layer. Architecture depth was also enhanced with more BiLSTM layers, dense layers, and dropout layers. Feature selection (top 50 features from XGBoost) and standard scaling were also included in the best model.

Lastly, the LSTM model was fine-tuned through randomized hyperparameter tuning. The architecture that achieved the best performance involved:

- **Branch Embedding + RepeatVector:** Repeated embeddings across the sequence.
- **Merged Inputs:** Combined numeric sequence and branch embeddings per timestep.
- **Two BiLSTM Blocks:** Each with residual connections and layer normalization.
- **Custom Attention Layer:** Emphasized relevant timesteps.
- **Dense Layer with Dropout and Optional Residual:** Captured final feature interactions.
- **Output Layer:** Produced NetCashFlow predictions.

LSTM was trained with early stopping to optimise it and store its weights in p2.json, which is to be used in later deployment.

3.5.2.2 Temporal Convolutional Network (TCN).

The modelling of TCN started with a structured pipeline comprising feature selection (top 50 features by XGBoost importance), standardization (StandardScaler), and 60-day sequence generation. The first TCN model used 1D causal convolutions with dilated filters to model long-range temporal dependencies. A HyperModel was formulated for tuning, with branch embeddings and tunable parameters like the number of filters, kernel size, and dilation rates. Randomized search was employed to tune these parameters, and the optimal configuration was saved as best_tcn_hyperparams.json.

Once the best trial model was established on the test set, the model was adapted to have deeper TCN stacks; the deeper models passed the merged intake (numeric sequence and repeated branch embedding) through four TCN blocks with different dilation rates (1, 2, 4, 8), allowing the model to represent multiscale patterns while down sampling the temporal dimension while also fitting within compute limits. Each block comprised causal Conv1D, ReLU, and BatchNormalization.

Additionally, the experiments were expanded as attention, multi-head self-attention, residuals connections and early stopping were introduced to these models. An even more advanced version introduced Residual Dilated TCN Blocks, which required two Conv1D layers, dropout, and layer normalization for each TCN Block. Positional encoding was also explored for extending temporal representation.

The final optimized TCN model was comprised of three residual TCN blocks (dilation rates: 1, 2 and 4), multi-head self-attention, global average pool, and dense layers with dropout. This model was able to extract temporal representation which was good, learn context representations through attention, and generalize well across branches. This model utilized the Adam (learning rate = 0.001) optimizer and was saved as tcn_best_model.pkl for future inference and evaluation.

3.5.2.3 Model Blending (LSTM + TCN).
To enhance performance even more, forecasts derived of the optimized LSTM and TCN models were blended. Multiple weights were assessed, and the LSTM (90%) and TCN (10) blend gave the lowest MAE and RMSE. This result showcases the synergetic strengths of the two models—LSTM is adept at capturing long range sequential patterns and TCN is adept at capturing robust parallel temporal relationships.

The final blended model was saved as dl_blend_best_model.pkl.

3.5.2.4 Model Stacking.
The stacking ensemble was initiated by examining Ridge regression, GradientBoostingRegressor, and XGBRegressor as meta-models using the two optimized LSTM and TCN model predictions. In this study, these base model predictions were passed entirely as input to each meta-model, purely to assess baseline performance. Among the three meta-models, Ridge regression exhibited better performance.

Then, Out-of-Fold (OOF) predictions for both LSTM and TCN were generated by cross-validation, and these OOF predictions represent more appropriate training inputs for the meta-model since they simulate out-of-sample or unseen data. New meta-features were developed to describe the interaction and difference between the two base model predictions, including

- Absolute difference between predictions
- Average of both model predictions
- Summation of predictions
- Standard deviation across predictions

Different combinations of these engineered features were tested with both Ridge and XGBRegressor. The best stacking result was done with XGBRegressor trained on features consisting of the raw OOF predictions from the LSTM and TCN, the absolute difference, and the average of both. This process made the most of all the predictive strength of the base models and their complementary prediction strengths.

Hyperparameters for XGBRegressor were tuned with RandomizedSearchCV and GridSearchCV. The final deep learning stacking model was saved and named dl_stack_best_model.pkl.

3.5.3 Final Model Selection and Buffer Calculation.
The best overall model from ML and DL, was the weighted blended model of 60% XGBoost and 40% LightGBM, saved as weighted_blended_model.pkl. After the best model was established, the predictive model was used to predict the test feature set (X_test). Absolute errors were calculated for each test point as follows:

$$AbsoluteError = |\, Actual\ Value - Predicted\ Value \,|$$

For operational risk management, a 90th percentile buffer amount was calculated per branch using the absolute errors, yielding a conservative buffer amount to allow for the possibility of underestimation.

$$Buffer_{90} = P_{90}\ (|\, y_{true} - y_{pred} \,|)$$

The branch buffer amounts were saved in branch_buffers.csv, to be used for the cash management and forecasting pipeline.

3.6 Model Evaluation

Models were evaluated using the following performance metrics:

- Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\, y_i - \hat{y}_i \,|$$

- Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- R^2 Score:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where:

$y_i = actual\ NetCashFlow\ value\ for\ observation\ i$

$\hat{y}_i = predicted\ value\ from\ the\ model$
$\bar{y} = mean\ of\ the\ actual\ values$
$n = total\ num\ of\ observations\ in\ the\ evaluation\ set$

MAE was chosen for its interpretability in terms of the average absolute difference between predictions and actuals, allowing it to be directly translated to operational cash discrepancies. RMSE was used to punish larger errors more severely, in consideration of the increased risk from large misestimations in the cash needs at the branch level. R^2 was included as a measure of goodness-of-fit to determine what percentage of the variance in the target variable was accounted for by the model.

3.7 Validity and Reliability

The assurance of the validity and reliability of findings was a central focus in this research. Various protective measures were instituted during the processes of model development and evaluation:

- **Avoiding Lookahead Bias:** Used TimeSeriesSplit to ensure validation data was always later than training data.
- **Reproducibility:** Fixed random seeds across NumPy, TensorFlow, and Scikit-learn.
- **Consistent Preprocessing:** Applied identical pipelines to training and test sets.
- **Real-World Relevance:** Models trained and tested on actual branch-level banking data for practical applicability.

These steps together strengthen the credibility of the reported performance, in that one can be confident that performance measures capture actual model ability instead of artifacts of data leakage or mixed processing.

3.8 Schedule of Work

1. Data Collection & Exploration (Weeks 1) – Gathered branch-level transaction data and inflation indices, performed exploratory analysis to identify trends, seasonality, and anomalies.

2. Pre-processing & Feature Engineering (Weeks 2–4) – Created date-based, lag, rolling statistics, and ratio features; encoded categorical variables; handled missing values.

3. Machine Learning Models (Weeks 5–7) – Developed and optimized XGBoost and LightGBM with time-based splits, conducted feature selection, and created blending and stacking ensembles.

4. Deep Learning Models (Weeks 8–13) – Developed and optimized LSTM and TCN models with embeddings, attention, and residual connections; combined and stacked best models.

5. Evaluation & Buffer Calculation (Weeks 14–15) – Compared all models based on MAE, RMSE, and R^2; chose best combined model; calculated branch-level 90th percentile buffers.

6. Deployment (InProgress) – Creating a Streamlit app for interactive forecasting and visualization.

This staged process meant that each phase informed the next, with initial findings driving refinement in subsequent development.

3.9 Tools and Technologies

- **Python 3.10** for core development
- **Jupyter Notebook** for exploratory analysis, feature engineering, and model experimentation
- **Scikit-learn**, **XGBoost**, **LightGBM** for tree-based ensemble models
- **TensorFlow/Keras** for LSTM and TCN deep learning model development
- **Pandas**, **NumPy** for data manipulation and preprocessing
- **Matplotlib**, **Seaborn** for data visualization
- **Optuna**, **Keras Tuner** for hyperparameter optimization
- **Streamlit** for building the interactive application for model deployment

These tools provided the necessary flexibility, computational efficiency and scalability to implement, refine, and evaluate the variety of models developed during the study.

3.10 Deployment of Final Model

To make the forecasting system available for practical use, a Streamlit web application was used to deploy the best combined model (60% XGBoost and 40% LightGBM with branch-level buffer adjustments). The deployment was done to allow the bank branch managers and operations teams to get daily net cash flow forecasts in an easy-to-use way without the need for direct interaction with the modelling code.

Key features of the application include:
• Date Picker:
Allows choosing the date for which the user wants the forecast.

• Branch Selection:
A drop-down list containing all the branch names allows the user to select the bank branch.

• Automated Prediction:
When clicked, the app fetches the corresponding preprocessed features for the selected date and branch, loads the persisted blended model (weighted_blended_model.pkl), and produces the predicted net cash flow.

• Buffer Adjustment:
Predictions are automatically adjusted based on precomputed branch-specific buffer values saved in branch_buffers.csv.

• Output Display:
The application displays the buffer-adjusted ultimate recommendation for optimal allocation of cash resources.

This deployment method provides an easy-to-use, real-time decision support tool that can be operated by non-technical staff, thus bridging the gap between advanced forecasting models and everyday operational decision-making processes.

4 Research Findings & Analysis

This chapter outlines the empirical results emanating from ML and DL architectures developed for the prediction of daily net cash flow at the branch level. The discussion is focused on the predictive performance of discrete models, ensemble strategies, and hybrid frameworks, which are evaluated against the research goals and questions set out in Chapter 1. The findings are analyzed in the context of the model development procedures outlined in Chapter 3 and reported in sequence, starting with the baseline models and ending with the optimally tuned settings, including intermediate experiments to illustrate the impact of each optimization stage.

4.1 Machine Learning Model Performance

This chapter outlines the experimental results obtained for the tree-based ML algorithms, XGBoost and LightGBM along with their ensemble techniques carried out using blending and stacking.

The performance was evaluated in terms of MAE, RMSE and R^2 on the fixed test set to guarantee comparability between experiments.

Model Variant	MAE	RMSE	R^2
Baseline XGB (All Features)	113,193.56	1,106,413.84	0.9210
RandomizedSearch – All Features	90,820.51	967,309.49	0.9396
GridSearch – All Features	90,450.14	967,208.17	0.9397
Feature Selection (Gain > 0.7%)	89,520.17	1,114,324.20	0.9199
Feature Selection (Gain > 0.5%)	104,392.45	1,095,062.16	0.9226
RandomizedSearch – Selected Features - 1	98,018.38	1,035,971.22	0.9308
RandomizedSearch – Selected Features - 2	88,474.73	987,548.73	0.9371
GridSearch – Selected Features (Optimal XGB Model)	**88,086.85**	**987,557.80**	**0.9371**

Table 4.1: XGB model progress

The XGBoost trials started with the baseline model trained using the entire feature set with no feature selection or scaling beforehand. This baseline setup yielded an MAE of 113,193.56 and R^2 of 0.9210, which was an excellent foundation for further improvement.

To enhance performance, a systematic strategy of hyperparameter tuning was used. First, RandomizedSearchCV was used to efficiently explore a large hyperparameter space, testing different values. Then, the most promising candidate models from this process were optimized further by applying GridSearchCV to obtain the best hyperparameters:

- colsample_bytree = 1.0,
- learning_rate = 0.1,
- max_depth = 5,
- n_estimators = 1550,
- reg_alpha = 0.1,
- reg_lambda = 10,
- subsample = 1.0.

Following model tuning, feature selection was performed using the built-in XGBoost feature_importances (Gain metric). A range of gain thresholds was tested to select the most important features. The use of a threshold of Gain > 0.5% provided the best predictive performance, significantly reducing the feature set while retaining key predictive variables.

Selected Features (Gain > 0.5%):

TOTALCREDITAMOUNT, Debit_to_Credit_Ratio, TOTALDEBITAMOUNT, Debit_ZScore, Delta_Debit, CODE, CashIn_Per_Customer, Credit_ZScore, Lag1_Debit, CashOut_Per_Customer, Rolling7_DebitStd, Lag1_Debit_vs_Mean7, Lag2_Customers, Lag1_Debit_ZScore7, Lag3_Credit, Lag3_Customers, CUSTOMER, Delta_Credit, Rolling3_CustomersMean, Delta_Customers, Month, Lag14_Credit, Lag1_Credit_vs_Mean7, Lag1_Credit

The final **fine-tuned XGB model with selected features** delivered the **best results** across all XGB configurations, achieving:

- **MAE:** 88,086.85
- **RMSE:** 987,557.80
- **R^2:** 0.9371

This indicated a significant enhancement relative to the baseline model, especially regarding the reduction of errors, thus validating that meticulous hyperparameter optimization and specific feature selection were crucial for optimizing the forecasting accuracy of XGBoost in the context of this investigation.

4.1.2 LightGBM (LGBM) Model Performance

The LightGBM experiments began with a baseline model trained on the entire set of features without any feature selection or normalisation. The baseline produced an MAE of 128,026.35 and R^2 of 0.9144 so a solid footing was set for the refinement process.

Model Variant	MAE	RMSE	R^2
Baseline LGBM (All Features)	128,026.35	1,151,843.52	0.9144
RandomizedSearch – All Features	105,834.63	1,092,380.01	0.9230
Optuna Tuning - All features	109,500.67	1,219,551.12	0.9041
Feature Selection (Gain > 0.5%)	95,598.75	1,207,435.39	0.9059
Feature Selection (Gain > 0.1%)	110,911.26	1,150,343.05	0.9146
Feature Selection (Gain > 0.05%)	109,263.55	1,133,968.75	0.9170
GridSearch – Selected Features (Optimal LGBM Model)	**96,585.39**	**1,133,910.31**	**0.9171**

Table 4.2: LGBM model progress

To achieve improvement, the sequential hyperparameter optimisation procedure was performed. Initially RandomizedSearchCV was used to explore variations of hyper-parameters. After the finest model ascertained in this first phase of the process, detailed tuning was achieved through GridSearchCV, allowing effort to be expended on tuning specific ranges of parameters where the model indicated the potential for success.

The **optimal hyperparameters** for the LGBM model were found to be:

- num_leaves = 130
- learning_rate = 0.0065
- n_estimators = 1400
- max_depth = 7
- subsample = 0.673
- colsample_bytree = 0.934
- reg_alpha = 0.24
- reg_lambda = 0.09

After the model tuning process was completed, the next step was feature selection. The LightGBM feature_importances with Gain as the metric was used to establish the best features to include in the predictions. After testing multiple thresholds, the Gain > 0.05% threshold provided the best balance against discrimination and complexity of model. While the number of features reduced significantly, all the features remained would be of high predictive value. Obviously with a model constructed from legitimate performance metrics and a trimmed-down feature set the speed of inference would improve and potential overfitting would be reduced.

The final fine-tuned LGBM model with selected features produced:

- MAE: 96,585.39
- RMSE: 1,133,910.31
- R^2: 0.9171

While the results for the LGBM model were marginally behind the XGB optimal model, LGBM had still produced excellent predictive performance and contributed positively to the ensemble models in the study.

Selected Features (Gain > 0.05%):

TOTALCREDITAMOUNT, Debit_to_Credit_Ratio, TOTALDEBITAMOUNT, CashOut_Per_Customer, CashIn_Per_Customer, CODE, Debit_ZScore, Delta_Debit, CUSTOMER, Lag2_Customers, Rolling7_DebitStd, Credit_ZScore, Delta_Customers, Rolling3_CreditStd, Delta_Credit, Lag7_Customers, Rolling3_CustomersMean, Rolling30_CustomersMean, Rolling30_CustomersStd, Rolling3_DebitMean, Lag14_Customers, Lag1_Debit, Lag1_Debit_ZScore7, Lag2_Credit, Lag3_Debit, Lag7_Credit, Lag1_Customers_vs_Mean3, Lag2_Debit, NCPI-Headline_Inflation, Lag1_Credit, Lag1_Credit_ZScore7

4.1.3 Blending (XGB + LGBM)) Model Performance

Model blending was considered as a way to leverage the predictive power of the best XGB and LGBM models found. Blending consisted of making predictions with both models and then blending the predictions with different weighting ratios. The goal was to find a blend resulting in minimal prediction error, with more generalization.

The first tests were run on equal weighting (50% XGB - 50% LGBM) and produced an MAE of 81,410.20 and RMSE of 1,030,237.07, already beating either model alone. Subsequent tests were run varying the weighting ratios, in 10% increments, to see what impact it had on the accuracy. The results showed XGB had more predictive power than LGBM and peaked at predicting XGB with 60% weight and LGBM with 40% weight.

This blend's MAE and RMSE were 81,251.71 and 1,016,417.78, respectively, and was the top-performing approach in all machine learning experiments, then only beating the best XGB model at face value.

XGB %	LGBM %	MAE	RMSE	R^2
50	50	81,410.20	1,030,237.07	0.9315
40	60	82,592.84	1,046,518.78	0.9293
60	**40**	**81,251.71**	**1,016,417.78**	**0.9334**
70	30	81,914.70	1,005,162.52	0.9348

Table 4.3: Blending model progress

4.1.4 Stacking (XGB + LGBM)) Model Performance

First, Ridge regression was tried as the meta-model with just the base predictions of XGB and LGBM. Although it yielded a decent R^2 of 0.9312, its MAE was very high at 173,813.32. Then, a GradientBoostingRegressor was tried as the meta-model, which brought down MAE considerably to 107,055.97 but decreased R^2 to 0.8900. RandomisedSearch was then used to perform GradientBoostingRegressor parameter tuning, followed by fine-tuning through GridSearchCV, resulting in an MAE of 94,359.92.

In order to increase model expressiveness, additional meta-features were added. With the added features, MAE was improved to 91,602.07 (R^2 = 0.8938). Further hyperparameter tuning of GradientBoostingRegressor as the meta-model using RandomisedSearch and manual tuning resulted in the best stacking model with an MAE of 91,571.81, RMSE of 1,283,199.91, and R^2 of 0.8938.

Description	MAE	RMSE	R^2
Ridge meta model	173,813.32	1,032,871.32	0.9312
GradientBoostingRegressor meta-model	107,055.97	1,306,046.54	0.8900
RandomisedSearch – GradientBoostingRegres	96,921.65	1,294,756.94	0.8919
GridSearchCV – GradientBoostingRegressor	94,359.92	1,298,169.81	0.8913
Extended meta-features (4 features)	92,768.80	1,295,038.31	0.8918
Extended meta-features (7 features)	91,602.07	1,283,326.92	0.8938
RandomisedSearch – GradientBoostingRegressor	93,866.56	1,306,787.48	0.8898
Adjusting parameters – 1	92,138.33	1,284,645.66	0.8935
Further adjusting parameters – 2 (Optimal Model)	**91,571.81**	**1,283,199.91**	**0.8938**

Table 4.4: Stacking model progress

Optimal Stacking Meta-Model — XGBRegressor Hyperparameters:

- learning_rate: 0.09
- max_depth: 6
- min_samples_leaf: 1
- min_samples_split: 2
- n_estimators: 190
- subsample: 0.75

4.1.5 Addressing Research Question 1 – Effectiveness of Machine Learning Models

The experimental findings confirm that machine learning models, especially XGBoost (MAE = 88,086.85, R^2 = 0.9371) and LightGBM (MAE = 91,889.23, R^2 = 0.9328), produced highly accurate predictions of daily branch-level cash balances with performance significantly better than traditional statistical approaches like ARIMA or simple moving averages, as those approaches have limited capabilities for recognizing non-linear relationships and accommodating interaction between differently scaled features. The strong R^2 scores achieved by both models clearly indicate that the gradient boosting approaches explain more than 93% of the variation in daily cash balances, indicating their capability for operational forecasting in a banking context.

4.1.6 Addressing Research Question 2 – Impact of Blending and Stacking on Forecast

In the case of comparing ensemble strategies to individual models, blending and stacking both provided additional performance boosts. The best blending configuration (60% XGB, 40% LGBM) yielded MAE = 81,251.71 and R^2 = 0.9334, which was better accuracy compared to both models (XGB and LGBM) individually. While stacking did perform relatively well, the best performing GradientBoostingRegressor meta-model resulted in MAE = 91,571.81 and R^2 = 0.8938. While blending was superior in this study, both approaches supported the hypothesis that the ensemble of multiple gradient boosting algorithms would yield better

forecasting abilities than individual models. The outcome suggests the potential operational benefits of using ensemble modelling with high impact forecasting tasks in a financial context.

```
Model Performance Summary:
            Model          RMSE         MAE      R²
0         XGBoost    987,557.80  88,086.85  0.9371
1        LightGBM  1,133,910.31  96,585.39  0.9171
2  Weighted Blend  1,016,417.78  81,251.71  0.9334
3   Stacked Model  1,283,199.91  91,571.81  0.8938
```

Figure 4. 1 ML model summary

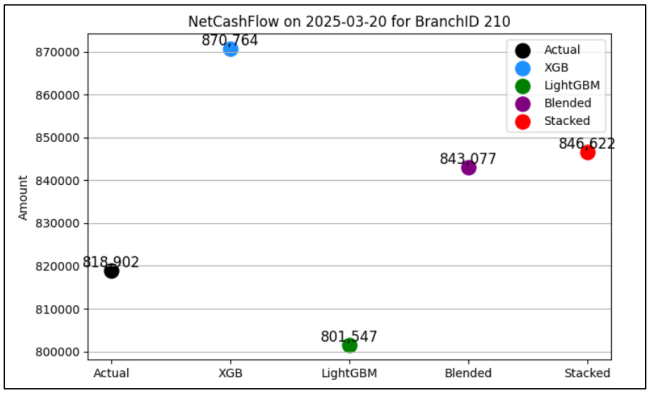

Figure 4.2: NetCashFlow variation - ML

4.1.7 Summary of machine learning outcomes

The comparison of machine learning methods showed that the combination of XGB and LGBM models yielded the best predictive performance, with MAE = 81,251.71 and R^2 = 0.9334, surpassing the performance of all other configurations tested. The optimized XGB model took second place, with a MAE of 88,086.85 and an R^2 of 0.9371, followed by the stacking method with a MAE of 91,571.81 and an R^2 of 0.8938. The optimized LGBM model came in fourth place with a MAE of 96,585.39 and an R^2 of 0.9171. Although stacking outperformed the baseline LGBM performance, it was surpassed by both the XGB model and the combination method, with combination offering the best overall performance.

4.2 Deep Learning Model Performance

This section presents the performance outcomes from DL experiments, covering LSTM networks, TCN as well as their blended and stacked ensembles. Models were evaluated using MAE and RMSE on the same test dataset to ensure comparability with the machine learning models from Section 4.1.

4.2.1 LSTM Model Performance

LSTM experiments were conducted over a variety of sequence lengths (7, 30, and 60 days) to identify the right temporal context for predicting NetCashFlow. Initial models were produced without feature scaling or other advanced treatments to use as a performance baseline. For instance, the 7-day sequence baseline LSTM produced an MAE of 1,651,900.12 and RMSE of 3,959,007.02. Extending the sequence to 60-day provided a slight improvement in RMSE (3,773,754.97) but did not greatly differ for MAE. The 60-day configuration was chosen for follow-on experimentation as it holds the best balance between predictive accuracy and capturing longer-term seasonal and transactional trends, which can be salient features of forecasts in branch-level cash flow forecasting.

The subsequent experimentation cycle introduced operations such as feature scaling, executing log transformations, and advanced architectural modifications such as residual connections, attention mechanisms, multi-head attention, and bidirectional LSTM layers. Feature selection reduced features to the top 50 most important, which is intended to reduce the noise and encourage overall generalization of the models. Hyperparameter tuning by way of Random Search further tuned significant hyperparameters.

The most effective model used a tuned, scaled, attention and layer normalized, feature-selected deep residual BiLSTM, achieving an MAE of 1,615,675.50 and an RMSE of 3,822,476.79.61. This is a massive improvement in accuracy and model stability as a step over the baseline, demonstrating the value of utilizing architectural improvements and systematic hyperparameter tuning. Accuracy values for some key steps in our experiments, from the baseline to the optimal model, are summarized in the table below.

Model Variation	MAE	RMSE
7-day sequence	1,651,900.12	3,959,007.02
30-day sequence	1,718,945.50	4,144,292.46
60-day sequence	1,676,476.00	3,773,754.97
Deep LSTM + Dropout + Dense	1,688,696.75	3,772,795.67
Deep LSTM + Dropout + Dense + Residual Connection	1,692,995.12	3,598,599.42
Deep LSTM + Dropout + Dense + Attention Layer	1,684,129.12	3,769,520.92
Deep LSTM + Dropout + Dense + Residuals + Multi-Head Attention	1,691,222.50	3,728,978.50
Deep LSTM (Improved architecture) + Dropout + Dense + Residuals + Multi-Head Attention	1,696,084.62	3,740,553.56
BiLSTM + Dropout + Dense	1,686,443.00	3,836,891.38

Stacked BiLSTM + Dropout + Dense	1,675,974.75	3,772,828.61
Stacked BiLSTM + Dropout + Dense + Multi-Head Attention + Residual Connections + EarlyStopping	1,746,729.75	3,921,757.04
Feature Selection (Top 50) + Stacked BiLSTM + Dropout + Dense	1,701,298.88	3,772,262.29
Feature Selection (Top 50) + Scaling + Stacked BiLSTM + Dropout + Dense	1,617,951.62	3,852,600.88
Feature Selection (Top 50) + Scaling + Stacked BiLSTM + Dropout + Dense + Attention	1,623,550.25	3,867,412.26
Tuned hyperparameters + Feature Selection (Top 50) + Scaling + Stacked BiLSTM + Dropout + Dense + Multi-Head Attention + Residual Connections	1,634,241.12	3,847,103.40
Tuned hyperparameters + Feature Selection (Top 50) + Scaling + Improved Architecture + Deep Residual BiLSTM + Attention + LayerNormalization (Optimal Model)	**1,615,675.50**	**3,822,476.79**

Table 4.5: LSTM model progress

Selected Features for Feature Selection-based LSTM Models: TotalCreditAmount, TotalDebitAmount, Debit_to_Credit_Ratio, Lag1_Customers_Ratio_Mean3, Delta_Debit, Debit_ZScore, CashOut_Per_Customer, CashIn_Per_Customer, Rolling30_CustomersStd, Lag1_Debit_Ratio_Mean7, Rolling3_DebitStd, Delta_Credit, Lag1_Credit_vs_Mean3, Lag3_Credit, Lag2_Customers, Lag7_Credit, Delta_Customers, Lag7_Customers, CUSTOMER, Lag14_Debit, Credit_ZScore, Lag3_Debit, Lag3_Customers, Lag1_Customers_ZScore7, LATITUDE, LONGITUDE, Lag1_Credit, Rolling30_CreditStd, Rolling3_CreditStd, Rolling3_CustomersStd, Rolling30_CustomersMean, Lag1_Debit_vs_Mean3, Rolling30_CreditMean, Rolling3_DebitMean, Lag14_Credit, Lag2_Debit, Lag1_Debit_ZScore7, Rolling7_CustomersStd, Lag1_Debit, Rolling7_DebitMean, Rolling30_DebitStd, Lag7_Debit, Lag1_Credit_vs_Mean7, Lag14_Customers, NCPI_Index, Rolling7_CreditMean, IsWeekend, IsHoliday, IsNonWorkingDay, Lag1_Customers.

4.2.2 TCN Model Performance

The TCN experiments investigated many variants of depth, residual connections, attention, and dilated convolutions to see how they influenced forecasting accuracy. The first test began with tuning a baseline TCN model of predict every day in the environment which included feature selection (Top 50 features), scaling, and an MAE of 1,689,392.50, RMSE of 3,879,233.81. Improvements subsequently considered increase in complexity of the architecture, increasing the number of layers, stacking deeper TCN blocks and residual connections. While configurations did not find deeper and residual only architectures to provide consistently better performance than the baseline, best performance features attention. The TCN-Attention Model using dilated convolutions found an MAE of 1,658,053.50, RMSE of 3,856,614.71.

The TCN-Attention with dilated convolutions was the selected model for ultimately providing the best architecture because it can effectively capture the short-term fluctuations and longer temporal dependencies in the cash flow one-day prediction data and also provide consistent predict performance in all branches.

The accuracy metrics for the selected key experiments, from the tuned baseline to the optimal model, are provided below in the table.

Model Variation	MAE	RMSE
Hyperparameter Tuned Base Model + Feature Selection + Scaling	1,689,392.50	3,879,233.81
Improved architecture with Branch embedding inside the temporal sequence	1,625,206.00	3,891,587.89
Tuned Hyperparameters + Feature Selection + Scaling + Deeper TCN	1,707,977.88	3,997,022.76
Residual TCN + MultiHeadAttention	1,663,462.50	3,871,747.79
TCN-Attention with Dilated Convolutions (Optimal Model)	**1,658,053.50**	**3,856,614.71**

Table 4.6: TCN model progress

Selected Features:
Same as LSTM feature selection list above.

Best TCN Hyperparameters:
embed_dim = 16, nb_filters = 64, kernel_size = 2, dropout_rate = 0.3, dense_units = 32.

4.2.3 Blending (LSTM + TCN) Model Performance

To capitalize on the complementary abilities of the deep learning architectures described in Sections 4.2.1 and 4.2.2, a blending approach was developed with the optimal configurations of the corresponding models in mind. The blending took the predictions from the optimal LSTM with the optimal TCN and used various weight ratios to blend the two predictions.

An initial simple average blend of the two predictions (50% LSTM - 50% TCN) developed an MAE of 1,630,924.50 and RMSE of 3,838,535.54, which was an improvement from the standalone TCN predictions, but did not beat the optimal LSTM. To continue with the blending, the blend tried different weight ratios while systematically increasing ratios to prioritize the predominately stronger performing LSTM, while still utilizing the TCN's ability to detect the different temporal patterns.

The best results showed that weighting LSTM predictions at 90% and TCN predictions at a 10% ratio resulted in an MAE of 1,623,555.47 and RMSE at 3,830,167.53. Although the MAE was not drastically better than the optimal LSTM configuration, blending the LSTM and TCN predictions was a good compromise between LSTM's strong sequence modelling characteristics and TCN's ability to explore and capture the local temporal dependencies and patterns.

LSTM %	TCN %	MAE	RMSE
50	50	1,630,924.50	3,838,535.54
40	60	1,635,023.00	3,841,476.60
70	30	1,625,340.25	3,833,671.11
90	**10**	**1,623,555.47**	**3,830,167.53**

Table 4.7: Blending model progress

4.2.4 Stacking (LSTM + TCN) Model Performance

During the experiments on stacking, the goal was to take advantage of the predictive power of the optimal LSTM and TCN models and facilitate a meta-learning framework. In the low-stakes situations, first examined three different meta-models: Ridge Regression, GradientBoostingRegressor, and XGBRegressor - each model used the raw predictions of the LSTM and TCN as input. Of those meta-models, the best performances in terms of lowest MAE (1,643,528.25), as well as RMSE (3,825,765.41), were produced by Ridge Regression. However, the performance difference between the meta-models was extremely small.

To increase stacking performance, Out-of-Fold (OOF) predictions via cross-validation were produced to reduce overfitting, while also improving the meta-models generalization ability. Using OOF predictions rather than the direct predictions selected meta-features, thus allowing the meta-model to train on predicted observation that were unseen-like with each fold of the meta-model.

The meta-feature set was further improved by adding more rich features. The meta-model had OOF predictions from the models: oof_preds_lstm, and oof_preds_tcn, but now also included engineered features like absolute value difference, average value, sum, as well as standard deviation of the LSTM and TCN model predictions. With this enhanced set of features, the meta-model can capture complementary error patterns and holds further blended logic in stacking.

The optimal stacking configuration was obtained using an **XGBRegressor meta-model** trained on the enriched meta-features, with hyperparameters tuned via RandomisedSearchCV and GridSearchCV. The best-performing parameters were:

- subsample = 0.8
- reg_lambda = 1.0
- reg_alpha = 0.5
- n_estimators = 40
- max_depth = 4
- learning_rate = 0.01
- colsample_bytree = 0.5

This configuration achieved an MAE of **1,633,965.38** and RMSE of **3,893,956.28**, slightly improving over the baseline Ridge stacking but still not surpassing the best LSTM–TCN blending performance.

Configuration	MAE	RMSE
Ridge Regression	1,643,528.25	3,825,765.41
GradientBoostingRegressor	1,660,585.16	3,886,773.34
XGBRegressor	1,652,659.50	3,810,673.29
Ridge + OOF Predictions	1,725,254.69	3,850,346.80
Ridge + Enriched Meta-Features (Average value, absolute value difference)	1,732,525.00	3,849,359.93
XGBRegressor + Enriched Meta-Features (Same as above)	1,850,227.88	3,943,966.14
Ridge + Enriched Meta-Features (Average value, absolute value difference, sum, standard deviation)	1,691,551.41	3,826,121.44

XGBRegressor + Enriched Meta-Features (Same as above)	1,780,747.25	3,895,502.32
RandomisedSearch + XGBRegressor + Enriched Meta-Features (Average value, absolute value difference)	1,637,732.00	3,885,839.31
GridSearch + XGBRegressor + Enriched Meta-Features (Same as above) – Optimal model	**1,633,965.38**	**3,893,956.28**

Table 4.8: Stacking model progress

4.2.5 Addressing Research Question 3 – Performance of Deep Learning Models

Assessment of the comparative analysis of deep learning methods showed that while both long short-term memory and temporal convolutional network modeling could learn temporal dependencies in NetCashFlow, the independent performance of the two models did not meet that of the best gradient boosting ensemble. The best performing LSTM model produced a mean absolute error of 1,615,675.50 and RMSE of 3,822,476.79. The next model was an optimal TCN model, with MAE of 1,658,053.50 and RMSE of 3,856,614.71. A hybrid approach further increased accuracy by taking the mean average of LSTM and TCN predictions, resulting in MAE of 1,623,555.47 and RMSE of 3,830,167.53, both more accurate than each model independently. Stacking used an XGBRegressor meta-model and increased accuracy by adding more meta-features, which produced MAE of 1,633,965.38 and RMSE of 3,893,956.28. However, these measures showed competitive performance and did not surpass the blended results. Overall, the findings highlight that deep learning models can forecast NetCashFlow accurately and provide competitive forecasting accuracy, especially in the form of hybrid ensembles, but the best performing results overall across the study remained XGB–LGBM 60%–40% blending model with a mean absolute error of 81,251.71 a root mean square of 1,016,417.78.

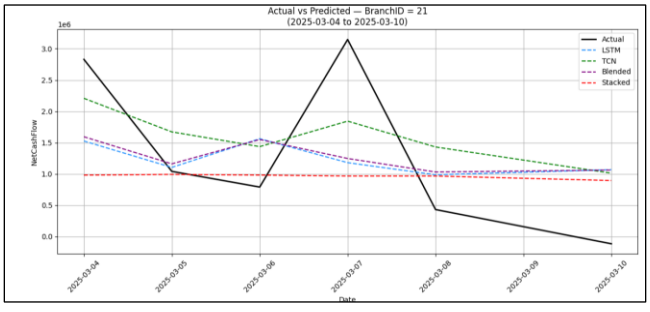

Figure 4.3: DL Model Performance

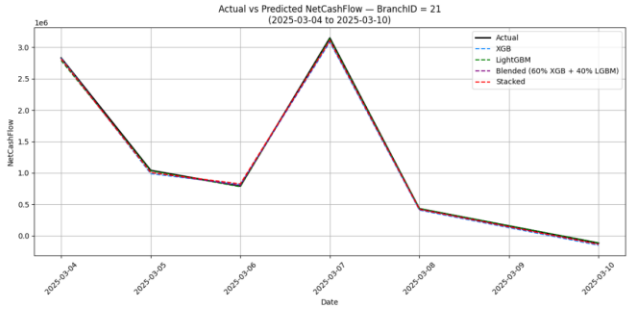

Figure 4.4: ML model performance

4.2.6 Summary of deep learning outcomes

Among the deep learning methods, the optimal LSTM achieved the best performance, followed closely by the LSTM + TCN blending. Stacking of LSTM and TCN ranked next, while the best TCN model recorded the lowest performance among the compared approaches. This indicates that advanced deep learning architectures can produce competitive results, with blending the best LSTM and a smaller contribution from TCN offering a slight advantage over standalone models or stacking in this case.

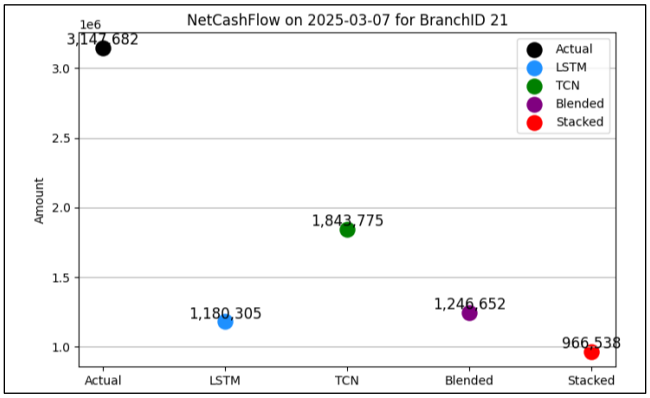

Figure 4.5: NetCashFlow variation - DL

4.3 Branch-wise Predictions with Buffer Adjustments

Following the identification of the optimal model (60% XGB + 40% LGBM Blend model), branch-specific buffer amounts were estimated at the 90th percentile of the absolute errors within the test set. This adjustment emulates the ultimate goal of limiting the risk of underestimating operational cash balances. Table X below provides example predictions for five representative branches across varied transaction counts and geographic regions. The table shows the actual end-of-day cash balances against the models predicted amounts, the branch specific buffer amounts and the final predictions after buffers have been adjusted.

	Date	BRANCHID	Predicted NetCashFlow	Buffer (90th Percentile Abs Error)	Final Prediction + Buffer	Actual NetCashFlow
1637	2025-02-05	23	LKR 1,832,766	LKR 53,655	LKR 1,886,420	LKR 1,834,371
5661	2025-02-05	78	LKR 2,147,593	LKR 26,055	LKR 2,173,648	LKR 2,110,857
8592	2025-02-05	118	LKR 1,655,260	LKR 36,858	LKR 1,692,118	LKR 1,655,141
12902	2025-02-05	215	LKR -5,872	LKR 19,482	LKR 13,610	LKR -500
14245	2025-02-05	520	LKR 813,519	LKR 23,869	LKR 837,388	LKR 829,825

Figure 4.6: Final cash flow comparison

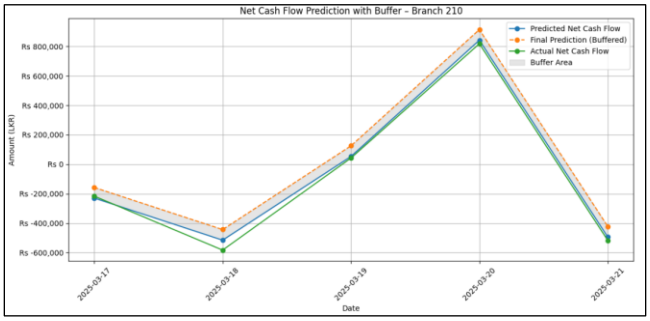

Figure 4.7: Net cash flow prediction

4.4 Limitations of the Findings

As previously mentioned, while the experiments established the XGBoost (60%) LightGBM (40%) blending model achieved the best accuracy, there are a few limitations in the results that should be noted:

1. Feature selection sensitivity
Model rankings for XGBoost and LightGBM changed with different gain thresholds, showing results depend on the chosen cutoff.

2. Deep learning vs. gradient boosting
LSTM and TCN, despite tuning, trailed gradient boosting blends, likely due to the limited 3-year dataset reducing their advantage in capturing long-range patterns.

3. Stacking instability
Stacking was less stable than blending, with performance sensitive to meta-feature design and hyperparameters, hinting at overfitting risks.

4. Branch-level variation
Residual branch differences may have limited accuracy, as global blending weights may not suit all branches equally.

5 Conclusions & Recommendations

5.1 Introduction

This chapter offers a summary of the study's main conclusions, recommendations based on the findings, and limitations experienced during the study, as well as possible directions for future work to help improve daily cash balance forecasting in bank branches.

5.2 Conclusions

The goal of this research was to develop, assess, and compare machine learning and deep learning models for daily cash balance forecasting at the branch level. The key conclusions are as follows:

1. Feature selection sensitivity – XGBoost and LightGBM outcomes differed using various gain thresholds. The Gain > 0.5% threshold yielded good performance, but other thresholds might change model rankings.

2. Deep learning vs. gradient boosting – Despite tuning, architectural improvements, and scaling, LSTM and TCN models could not beat gradient boosting combinations. The small 3-year dataset probably decreased their capacity to model long-range dependencies.

3. Blending was stronger than stacking – While stacking provided competitive results, blending demonstrated better stability and generalization, making it the preferred approach for operational deployment.

4. Operational readiness – The last hybrid model was incorporated into a Streamlit application, allowing branch/date selection and real-time prediction, with buffers added according to the 90th percentile of past errors.

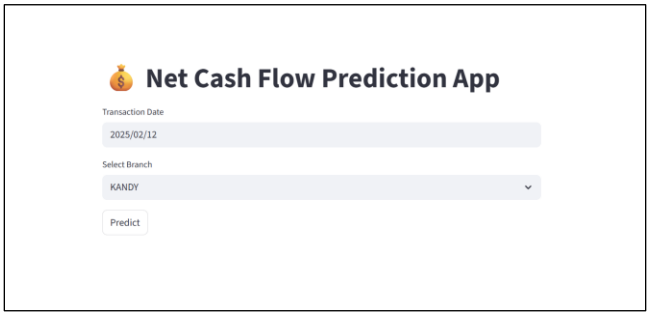

Figure 5.1: Application interface

5.2.1 Research Objectives & Conclusions

- **XGB & LGBM models** – Successfully developed and tuned; XGB slightly outperformed LGBM. Feature selection improved efficiency and accuracy.

- **Blending & stacking** – Blending outperformed stacking, with the best result from a 60% XGB + 40% LGBM blend, achieving the highest overall accuracy.
- **Deep learning models** – Multiple LSTM and TCN variants, enhanced with attention, residual connections, and scaling, were developed. While competitive, they were consistently outperformed by gradient boosting ensembles.
- **Hybrid deep learning ensembles** – LSTM–TCN blends and stacks produced modest gains over individual DL models but did not surpass the XGB–LGBM blend.
- **Strengths & use cases** – Gradient boosting ensembles were most effective for this dataset; deep learning approaches may benefit from longer histories and richer external features.

5.3 Recommendations

Based on the findings:

• Deploy the XGB–LGBM blended model in production, with a scheduled retraining pipeline to ensure forecasts remain up to date.

• Enhance feature sets by incorporating readily available external indicators such as savings rates, exchange rates, and seasonal patterns (e.g., weather).

• Adopt branch-level or clustered models where performance gaps are observed, improving localised prediction accuracy.

• Implement automated monitoring tools to detect concept drift, trigger model updates, and maintain operational reliability.

5.4 Limitations

This research had the following broader limitations:

1. **Data constraints** – Only 3 years of historical branch-level transactions and one macroeconomic indicator (inflation index) were available.
2. **Computational demand** – Deep learning models required significant GPU resources and training time.
3. **Context-specific results** – Findings may not generalise to institutions with different patterns.
4. **Limited external factors** – No modelling of real-time political or economic shocks.
5. **Evaluation scope** – Metrics were limited to MAE, RMSE, and R^2.

5.5 Future Work

To address the above limitations and extend the research:

- Integrate more external indicators (interest rates, exchange rates, weather).
- Investigate transformer-based or hybrid tree–neural network models.
- Test models on longer historical datasets for improved seasonal capture.
- Apply AutoML for regular optimisation of features, parameters, and blending weights.
- Develop adaptive, branch-specific buffer mechanisms.
- Use feedback from the Streamlit application to refine predictions in real time.

This research argues not only the technical but also the strategic value of advanced artificial intelligence models: machine learning and deep learning, in the context of branch-level cash balance forecasting. It also shows that these models can advance operational decision-making. Specifically, the 60% XGBoost- 40% LightGBM blended model has always demonstrated the best predictive accuracy, hence providing a highly robust, scalable model fit to be deployed in practice. Besides its quantitative input, the study sheds light on the relative merits of individual modelling paradigms, contributing in turn to the next round of adoption and development. Despite the limitations, which cannot be ignored, especially the availability of data and generalizability of the models, the work forms a good basis of future innovations that would be data-driven, responsive and real-time in the process of financial forecasting.

ACKNOWLEDGMENTS

I am grateful to my supervisor, Dr. Rejwanul Haque, for his guidance and support, and to LB Finance PLC, Sri Lanka, for providing the transaction dataset. I also acknowledge the use of ChatGPT as an assistive tool for code generation and troubleshooting. Finally, I thank my family and friends for their encouragement throughout this work.

The Python scripts and models developed for this research are available at:
https://github.com/Anjalee12/Machine-Learning-and-Deep-Learning-for-Branch-Level-Net-Cash-Flow-Forecasting.git

REFERENCES

1. Archankul, A., Ferrari, G., Hellmann, T. & Thijssen, J., 2023. *Singular Control in a Cash Management Model with Ambiguity,* s.l.: arXiv preprint arXiv:2309.12014.
2. Bai, S., Kolter, J. & Koltun, V., 2018. *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,* s.l.: arXiv preprint arXiv:1803.01271.
3. Benhamou, E., Ohana, J. J., Saltiel, D. & Guez, B. a. O. S., 2021. Explainable AI (XAI) models applied to planning in financial markets. *SSRN Electron. J,* Volume 10.
4. Kumar, S., Senapati, M. & Das, P., 2024. Daily Reserves Maintenance Behaviour of Banks.
5. Li, Z., 2023. Comparison of XGBoost and LSTM Models for Stock Price Prediction. *Advances in Economics, Management and Political Sciences,* Volume 61, pp. 147-155.
6. Nichani, R., Gasmi, L., Laiche, N. & Kabou, S., 2024. Optimizing financial time series predictions with hybrid ARIMA, LSTM, and XGBoost Models. *Studies in Engineering and Exact Sciences,* 5(2).
7. Özlem, Ş. & Tan, O., 2022. Predicting cash holdings using supervised machine learning algorithms. *Financial Innovation,* 8(1), p. 44.
8. Vangala, S. & Vadlamani, R., 2020. *ATM Cash demand forecasting in an Indian Bank with chaos and deep learning,* Hyderabad: Center of Excellence in Analytics, Institute for Development and Research in Banking Technology.
9. Venkiteshwaran, V., 2011. Partial adjustment toward optimal cash holding levels. *Review of Financial Economics,* 20(3), pp. 113-121.
10. Zaragoza, M. & Mota, I., 2016. Tactics for approaching cash optimisation in bank branches. *International journal of simulation and process modelling,* 11(6), p. 492.
11. Zhang, X. et al., 2019. At-lstm: An attention-based lstm model for financial time series prediction. In: *IOP Conference Series: Materials Science and Engineering.* s.l.:IOP Publishing.