1. Problem Specification

This project addresses the critical business problem of customer churn within the telecommunications industry. With increasing market competition and the ease of switching providers, accurately predicting which customers are likely to discontinue their service is essential for targeted retention efforts. This investigation formulates churn prediction as a binary classification task, where the objective is to predict whether a customer will churn (1) or not (0). Given the moderate class imbalance in the dataset (approximately 28.8% churn), the primary evaluation metrics will be the PR AUC , recall, and the F1-Score, supplemented by precision and accuracy for a comprehensive assessment.

2. Data Collection and Preprocessing

The analysis is based on the "Cell2Cell: The Churn Dataset" from the Teradata Center for Customer Relationship Management at Duke University, available on Kaggle. The initial dataset contained 71,047 customer records, of which a holdout set of approximately 20,000 unlabeled records was excluded. After removing duplicates, the working dataset consists of 51,047 observations. A detailed data dictionary is provided in the Appendix.

To prepare the data for modeling, a preprocessing pipeline was established. Key steps included:

- Feature Engineering: A new feature, InactiveSubs, was created to reduce multicollinearity between UniqueSubs and ActiveSubs. Redundant or uninformative columns were removed.
- Data Cleaning: Duplicate records were removed. Invalid values, such as negative values in CurrentEquipmentDays, were treated as missing and imputed using the training set's median.
- Encoding & Scaling: The high-cardinality ServiceArea feature was managed with smoothed target mean encoding to capture geographic risk without inflating dimensionality. All features were standardized using StandardScaler.
- Data Splitting: The data was stratified into training (60%), validation (20%), and test (20%) sets before any transformations to prevent data leakage.

3. Exploratory Data Analysis (EDA)

The EDA revealed several key insights that guided the modeling strategy. The target variable, Churn, is moderately imbalanced, with 28.8% of customers having churned, reinforcing the choice of PR AUC , recall, and the F1-Score as primary metrics. (See Appendix for visualisations)

Key findings include:

- Equipment Tenure: A strong positive correlation was observed between CurrentEquipmentDays and churn, suggesting that customers with older devices are more likely to leave.
- Customer Service Interactions: The number of RetentionCalls also showed a positive correlation with churn, indicating that such calls are often a lagging indicator of dissatisfaction.
- Usage Patterns: High-engagement users, as measured by MonthlyMinutes and TotalRecurringCharge, were less likely to churn.
- U-Shaped Churn by Tenure: Churn rates were lower for very new customers but spiked around the 10-12 month mark, likely coinciding with the end of promotional contracts.

Exploratory Data Analysis revealed key non-linear relationships justifying the use of deep learning. For instance, churn risk was highest for customers with long equipment tenure and for those at the 10-12 month service mark, likely corresponding to contract expirations.

4. Benchmark Model: Logistic Regression

To establish a reference point for evaluating deep learning models, I first trained a logistic regression model—a widely used, interpretable method for binary classification. Given the class imbalance (28.8% churn), I applied class weighting and standardized numeric features to ensure fair treatment of both classes and stable convergence. On the test set, the model achieved around **60% accuracy**, with **recall of 54%**, **F1-score of 0.43**, and a **ROC AUC of 0.609**. While recall was relatively strong, precision remained low, reflecting challenges in identifying true churners without excessive false positives. Overall, the model demonstrated modest discriminatory power. Despite its simplicity, the logistic regression model serves as a solid benchmark. However, its limited ability to capture complex, non-linear churn patterns underscores the motivation for deploying more expressive deep learning models. (Full metrics, confusion matrices, and precision-recall curves are available in **Appendix A**, alongside direct performance comparisons with deep learning architectures.)

5. Deep Learning Approaches and Hyperparameter Tuning

To move beyond the linear limitations of the baseline model, two deep learning architectures were developed, tuned, and compared: a Dense Feedforward Neural Network (FNN) and a Wide & Deep Network. The selection and design of these models were guided by their ability to capture complex, non-linear patterns in tabular data. This section describes the methodology used for hyperparameter tuning and the final, optimized architecture for each model, with specific details on the best hyperparameter combinations provided in the appendix.

A systematic hyperparameter tuning process was conducted using the Keras Tuner library with a Bayesian Optimization strategy. This approach intelligently explores a predefined search space to efficiently identify high-performing parameter combinations. The tuning was performed over 30 trials for each architecture, with the objective of maximizing the Area Under the Curve (AUC) on the validation set. An EarlyStopping callback was implemented to monitor validation performance, halting training after 10 epochs without improvement and restoring the model's best weights to prevent overfitting.

5.1 Architecture 1: Dense Feedforward Neural Network (FNN)

The first model is a standard Dense Feedforward Neural Network (FNN), a powerful and versatile architecture for tabular data problems. Built using the Keras Sequential API, the network consists of a stack of fully connected layers, allowing it to learn increasingly abstract representations of the input features. (The full model Architecture is in Appendix)

The final, optimized Dense FNN architecture featured a deep branch with four hidden layers ([256, 512, 128, 64] units) utilizing the swish activation function. Dropout was applied after each hidden layer to regularize the network, with rates ranging from 0.3 to 0.5. The model was trained with the Nadam optimizer and a learning rate of 0.00024. The full summary of the hyperparameter search space and the best combination is provided in Appendix A1.

This architecture is designed to be a strong general-purpose classifier, capable of modeling intricate relationships between customer attributes and churn behavior.

5.2 Architecture 2: Wide & Deep Network

The second model employs a more advanced Wide & Deep architecture. This hybrid model, built using the Keras *Functional* API, is designed to leverage the strengths of two distinct learning pathways simultaneously: (The full model Architecture is in Appendix)

- **The "Deep" Path:** This component is a multi-layer feedforward network, similar in principle to the Dense FNN. Its purpose is to achieve **generalization** by transforming the input features through a series of non-linear hidden layers, allowing it to identify complex and previously unseen feature combinations. The optimal deep path for this model consists of six hidden layers with varying numbers of units.
- **The "Wide" Path:** This component is a simple, linear path that directly connects the original input features to the final output layer. Its purpose is to achieve **memorization** by learning the direct, first-order effects and interactions of features. This is particularly useful for capturing simple, interpretable rules that a deep network might otherwise overlook.

The outputs from both paths are concatenated before being passed to a final output neuron. The tuning process revealed that a more complex architecture was optimal for this model. The final Wide & Deep network features a six-layer deep path with units ranging from 64 to 512. Like the FNN, it utilized the swish activation function. It is optimized using the 'nadam' algorithm with a more conservative learning rate of 0.000254 and a larger batch size of 128. L2 regularization was applied alongside dropout to further control overfitting.(See Table1)

This structure allows the model to learn a wide array of both simple and intricate patterns from the data. The complete set of optimal hyperparameters for this model can be found in Table A2 in the Appendix. The outputs from both the wide and deep paths are concatenated and then fed into a final sigmoid output neuron. By combining memorization and generalization, the Wide & Deep model can learn both simple rules and complex patterns, often leading to superior performance on tabular datasets.

Hyperparameter	Search Space	Best Value (Dense NN)	Best Value (Wide & Deep NN)
num_layers	2 to 6	5	6
units_i	[32, 64, 128, 256, 512]	[128, 128, 128, 64, 32]	[128, 512, 512, 128, 512, 64]
activation	["relu", "tanh", "swish"]	"swish"	"swish"
dropout_i	[0.3, 0.4, 0.5]	[0.3, 0.3, 0.5, 0.3, 0.4]	[0.3, 0.4, 0.5, 0.5, 0.5, 0.5]
use_bn_i	[True, False]	[True, True, False, False, True]	[True, True, True, True, False, False]
l2_i	[0.0, 1e-5, 1e-4, 1e-3]	[1e-5, 1e-4, 0.0, 0.001, 1e-5]	[1e-5, 1e-5, 0.0001, 0.0, 0.001, 0.001]
optimizer	["adam", "rmsprop", "sgd", "nadam"]	"adam"	"nadam"
learning_rate	[1e-4, 1e-2] (log sampling)	0.00124	0.000254
batch_size	[32, 64, 128]	32	128
loss_fn	["binary_crossentropy", "focal"]	-	"binary_crossentropy"

Table1:Hyperparameter Search Space and Best Values

6. Final Fitting Results and Discussion

After training the final models with the best hyperparameters, they were evaluated on the unseen test set. The results, as summarised in the table, demonstrate a clear performance improvement from the deep learning models over the baseline.

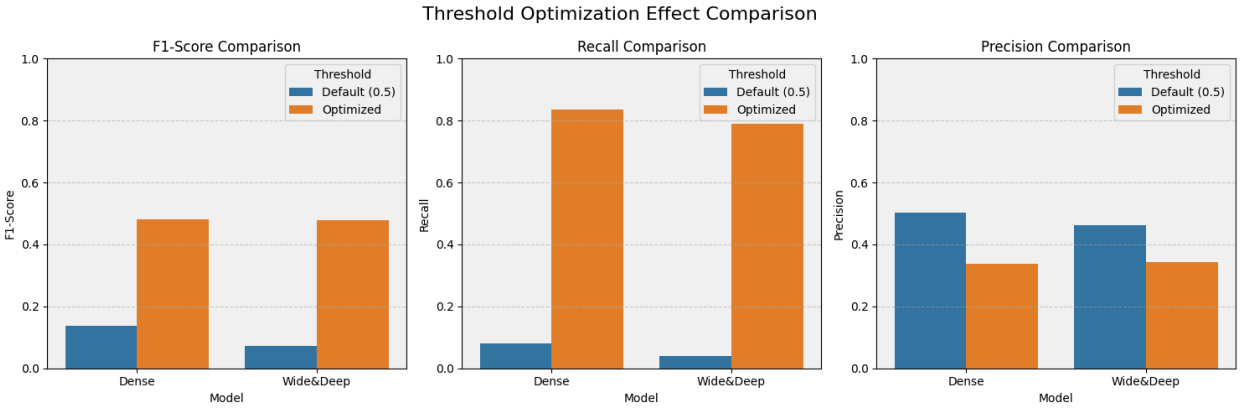
Model Metrics on Test Set (Default vs Optimized Threshold)

	Accuracy	Loss	Precision	Recall	F1-Score	ROC AUC	PR AUC
Baseline	0.5960	0.6797	0.3647	0.5418	0.4359	0.6090	0.3660
Dense	0.7118	0.5825	0.4989	0.0795	0.1372	0.6384	0.3985
Wide&Deep	0.7101	0.5807	0.4634	0.0387	0.0715	0.6316	0.3902
Dense_Optimized	0.5221	0.5825	0.3516	0.7801	0.4847	0.6384	0.3985
Wide&Deep_Optimized	0.5044	0.5807	0.3433	0.7886	0.4784	0.6316	0.3902

Table 2: Final model performance on the test set. Deep learning model metrics are based on an optimized classification threshold.

The **Dense FNN emerged as the best-performing model**, achieving the highest **AUC-ROC (0.6384)** and **F1-Score (0.4847)**. Both deep learning models substantially outperformed the logistic regression baseline in AUC, indicating a superior ability to distinguish churners from non-churners.

A critical insight from this analysis was the importance of **classification threshold optimization**. Using the default 0.5 threshold, the deep learning models achieved high accuracy but had extremely poor recall (below 10%), making them practically useless for identifying at-risk customers. By analyzing the precision-recall trade-off, an optimal threshold was identified for each model to maximize the F1-score. For the Dense FNN, shifting the threshold to **0.24** dramatically increased recall to **78%**. This transformed the model from a passive classifier into an actionable business tool capable of flagging a large majority of potential churners, which is often the primary goal of such a model, even at the cost of lower precision.



The learning curves, such as the validation loss and AUC plots (please refer to the Appendix), reveal that while the models continued to improve on the training data, performance on the validation set plateaued and eventually began to show signs of decline. This is a classic indicator of overfitting. However, as our

training procedure included EarlyStopping with the restore_best_weights=True option, the final model saved for each architecture is not the one from the last epoch, but rather the version that achieved the highest performance on the validation set. This guarantees that the models presented in this report are the optimal versions discovered during the training process, effectively mitigating the observed overfitting.

7. Ethical Concerns

Deploying churn prediction models raises several ethical risks:

- **Bias and Fairness:** The model may reflect and reinforce existing biases in historical data, such as disproportionately flagging customers from certain regions or income groups, leading to unfair treatment.
- **Transparency and Explainability:** Deep learning models often lack interpretability, making it difficult to justify decisions or audit for fairness.
- **Privacy and Potential Misuse:** Customer data is sensitive and must be handled securely. Churn scores could be misused—for example, by offering better deals only to those likely to leave, unintentionally penalizing loyal customers.

To mitigate these risks, organizations should establish clear ethical guidelines, conduct regular bias audits, and use model insights to enhance customer relationships—not harm them.

8. Conclusion

This project successfully demonstrated that deep learning provides a significant advantage over traditional linear models for predicting customer churn. Through systematic hyperparameter tuning and architecture design, both a Dense Feedforward Network and a Wide & Deep network were developed, with both outperforming a logistic regression baseline. The Dense FNN proved to be the superior model, delivering the highest PR AUC, Recall, ROC-AUC and F1-Score. The analysis also underscored a crucial operational step: the optimisation of the classification threshold is essential to turn a performant but passive model into a practically valuable and actionable tool for churn mitigation.

References

jpacse. (2020). *Datasets for churn telecom* [Data set]. Kaggle. https://www.kaggle.com/datasets/jpacse/datasets-for-churn-telecom
S. Saleh, S. Saha, Customer retention and churn prediction in the telecommunication industry: a case study on a Danish University, Soc. Netw. Anal. Appl. Sci. 5 (2023) 173, https://link.springer.com/article/10.1007/s42452-023-05389-6

Poudel, S. S., Pokharel, S., & Timilsina, M. (2024). *Explaining customer churn prediction in telecom industry using tabular machine learning models*. Machine Learning with Applications, 17, 100567. https://doi.org/10.1016/j.mlwa.2024.100567

Beeharry, Y., & Tsokizep Fokone, R. (2022). *Hybrid approach using machine learning algorithms for customers' churn prediction in the telecommunications industry*. Concurrency and Computation: Practice and Experience, 34(4), e6627. https://doi.org/10.1002/cpe.6627

Wagh, S. K., Andhale, A. A., Wagh, K. S., Pansare, J. R., Ambadekar, S. P., & Gawande, S. H. (2024). *Customer churn prediction in telecom sector using machine learning techniques*. Results in Control and Optimization, 14, 100342. https://doi.org/10.1016/j.rico.2023.100342

Lalwani, P., Mishra, M. K., Chadha, J. S., & Sethi, P. (2022). Customer churn prediction system: A machine learning approach. *Computing, 104*(2), 271–294. https://doi.org/10.1007/s00607-021-00908-y

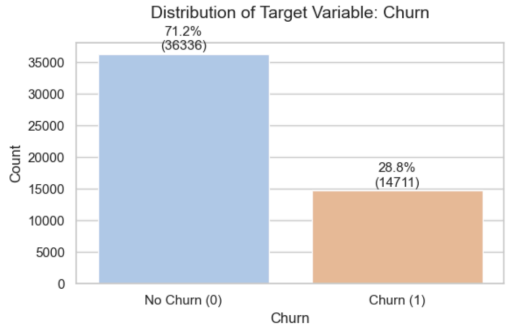
Asif, D., Arif, M. S., & Mukheimer, A. (2025). A data-driven approach with explainable artificial intelligence for customer churn prediction in the telecommunications industry. *Results in Engineering, 26*, 104629. https://doi.org/10.1016/j.rineng.2025.104629

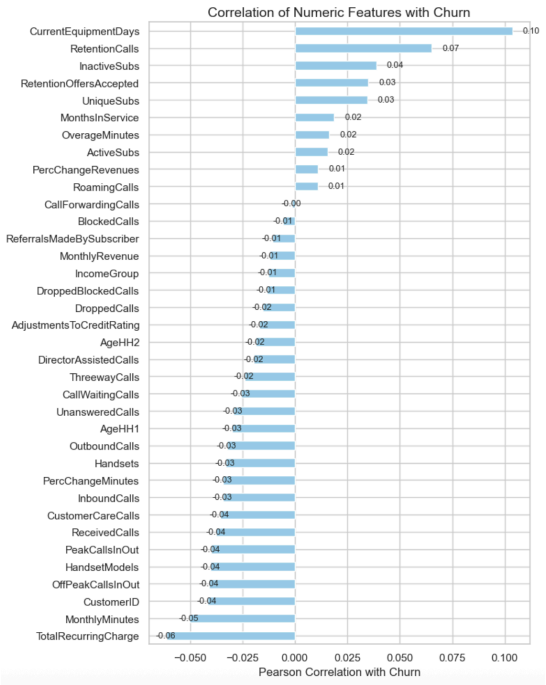
Huang, B., Kechadi, M. T., & Buckley, B. (2012). Customer churn prediction in telecommunications. *Expert Systems with Applications, 39*(1), 1414–1425. https://doi.org/10.1016/j.eswa.2011.08.024

Appendix A

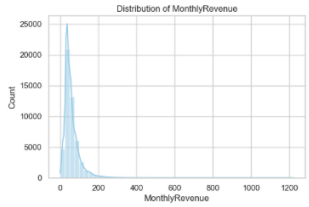
3. EDA

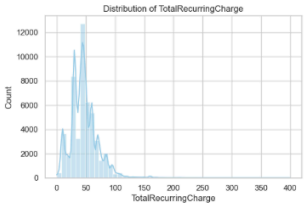
3.5 Additional Observations

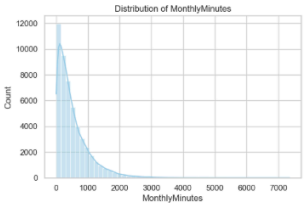


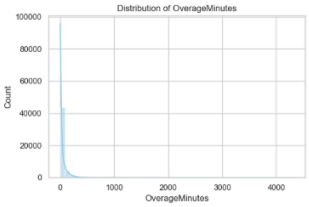


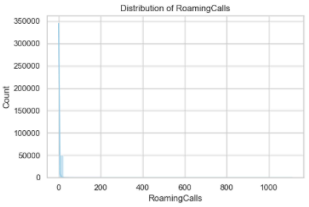
- **Outliers**:

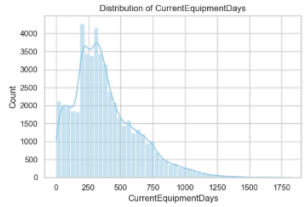

Distribution of MonthlyRevenue
Count
MonthlyRevenue

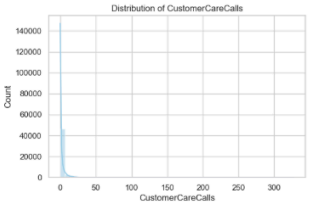

Distribution of TotalRecurringCharge
Count
TotalRecurringCharge

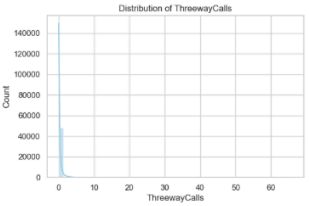

Distribution of MonthlyMinutes
Count
MonthlyMinutes

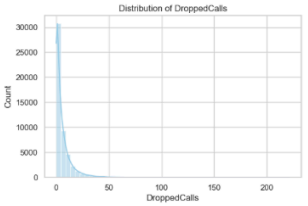

Distribution of OverageMinutes
Count
OverageMinutes

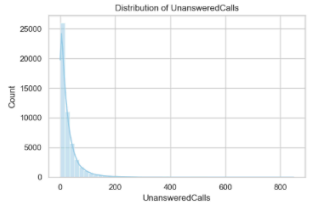

Distribution of RoamingCalls
Count
RoamingCalls

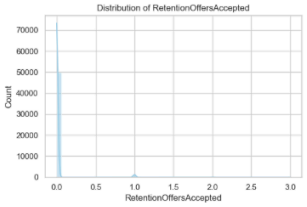

Distribution of CurrentEquipmentDays
Count
CurrentEquipmentDays

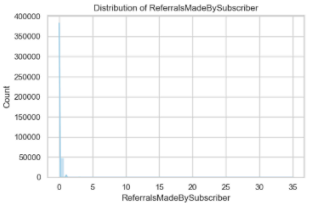

Distribution of CustomerCareCalls
Count
CustomerCareCalls

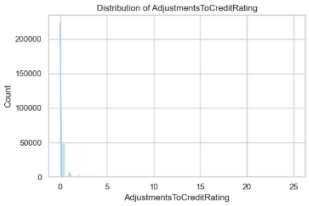

Distribution of ThreewayCalls
Count
ThreewayCalls

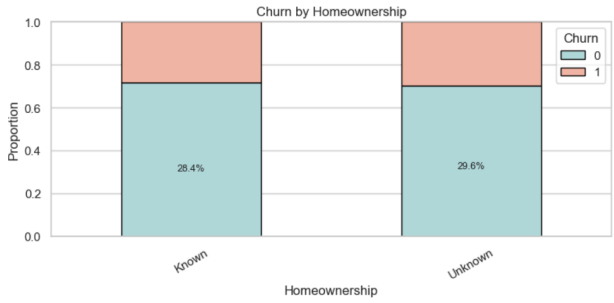

Distribution of DroppedCalls
Count
DroppedCalls

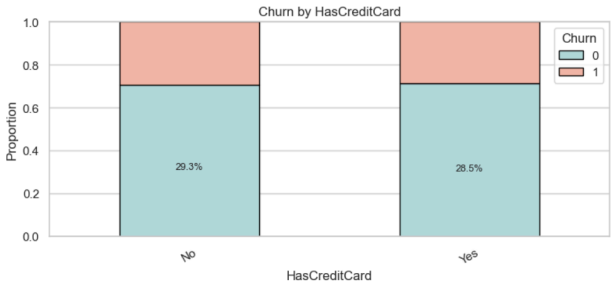

Distribution of UnansweredCalls
Count
UnansweredCalls

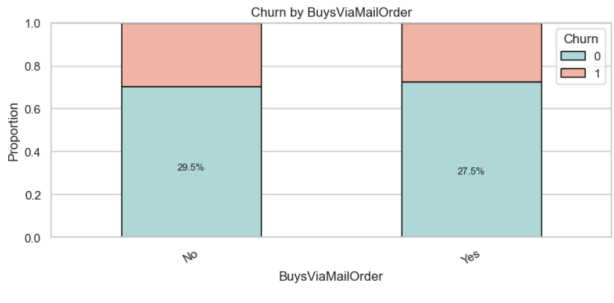

Distribution of RetentionOffersAccepted
Count
RetentionOffersAccepted


Distribution of ReferralsMadeBySubscriber
Count
ReferralsMadeBySubscriber

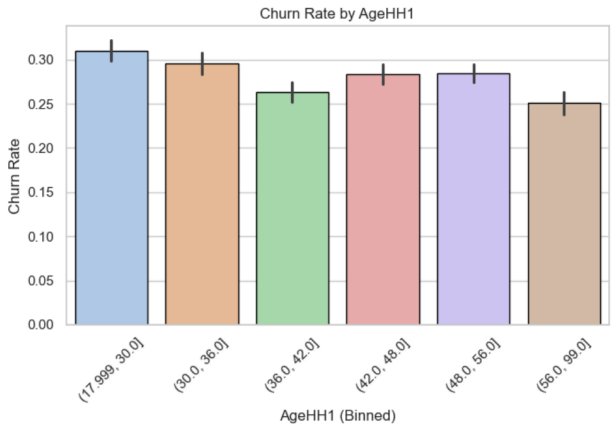

Distribution of AdjustmentsToCreditRating
Count
AdjustmentsToCreditRating

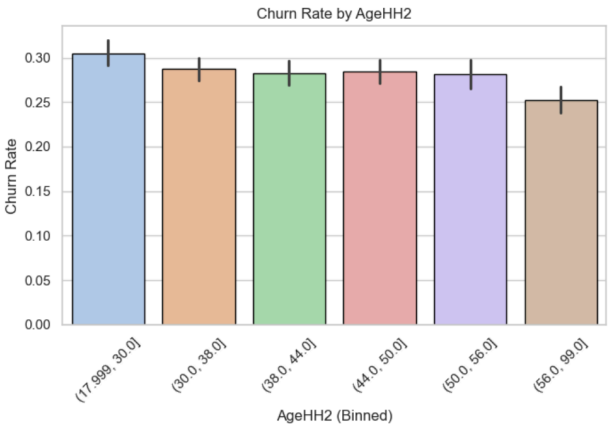

Churn by Homeownership
Proportion
Churn
0
1
28.4%
29.6%
Known
Unknown
Homeownership

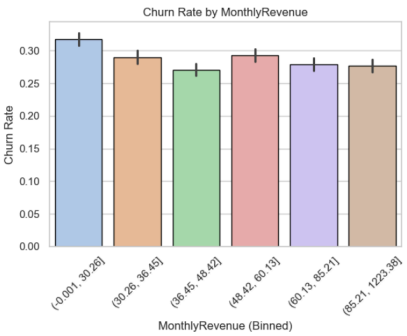

Churn by HasCreditCard
Proportion
Churn
0
1
29.3%
28.5%
No
Yes
HasCreditCard

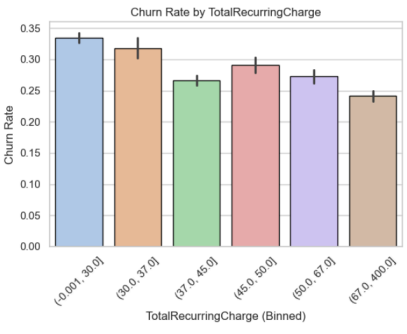

Churn by BuysViaMailOrder
Proportion
Churn
0
1
29.5%
27.5%
No
Yes
BuysViaMailOrder

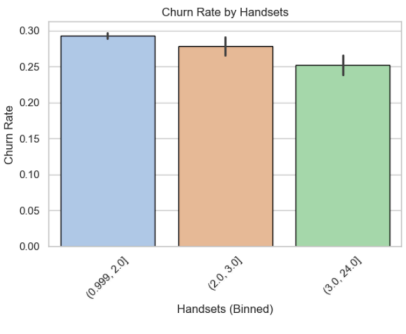

Churn by PrizmCode
Proportion
Churn
0
1
28.6%
31.7%
28.1%
30.0%
Other
Rural
Suburban
Town
PrizmCode


Churn Rate by AgeHH1
Churn Rate
(17.999, 30.0]
(30.0, 36.0]
(36.0, 42.0]
(42.0, 48.0]
(48.0, 56.0]
(56.0, 99.0]
AgeHH1 (Binned)


Churn Rate by AgeHH2
Churn Rate
(17.999, 30.0]
(30.0, 38.0]
(38.0, 44.0]
(44.0, 50.0]
(50.0, 56.0]
(56.0, 99.0]
AgeHH2 (Binned)


Churn Rate by MonthlyRevenue
Churn Rate
(-0.001, 30.26]
(30.26, 36.45]
(36.45, 48.42]
(48.42, 60.13]
(60.13, 85.21]
(85.21, 1223.38]
MonthlyRevenue (Binned)


Churn Rate by TotalRecurringCharge
Churn Rate
(-0.001, 30.0]
(30.0, 37.0]
(37.0, 45.0]
(45.0, 50.0]
(50.0, 67.0]
(67.0, 400.0]
TotalRecurringCharge (Binned)


Churn Rate by Handsets
Churn Rate
(0.999, 2.0]
(2.0, 3.0]
(3.0, 24.0]
Handsets (Binned)

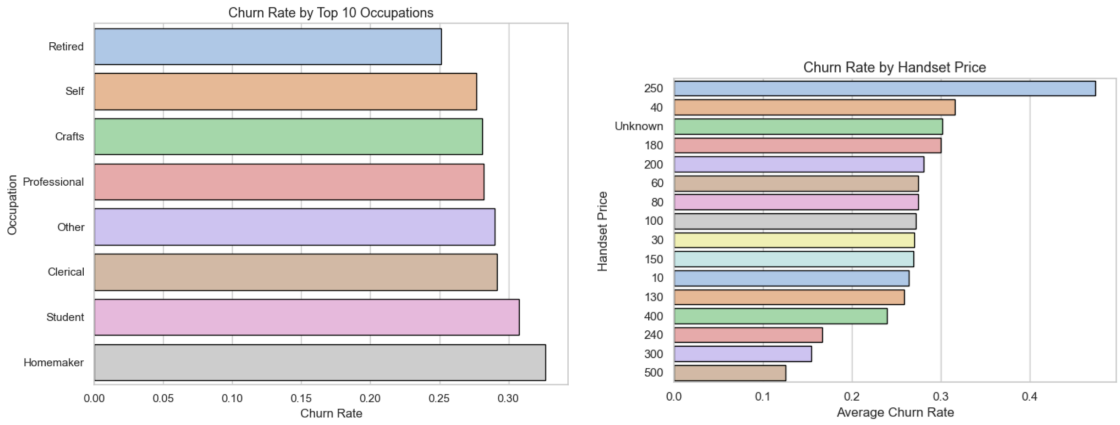
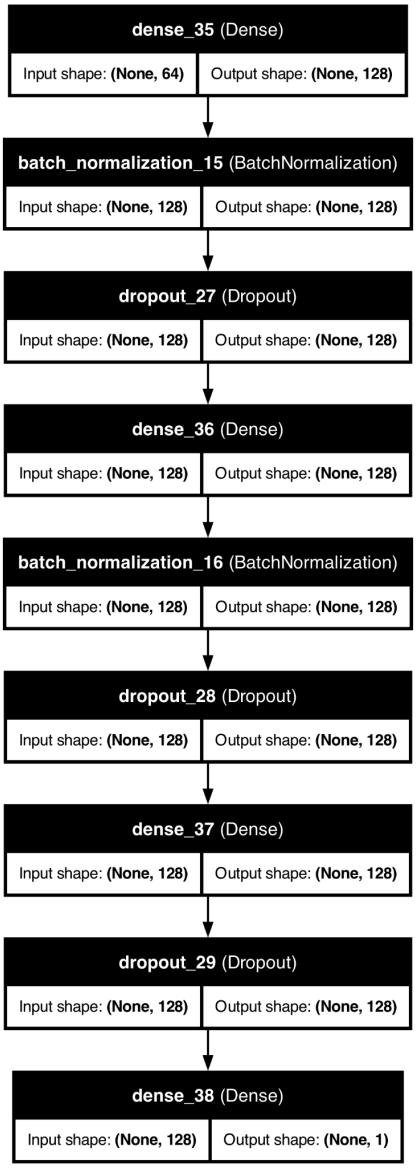


Table A1: Dense FNN – Hyperparameter Tuning Summary

Hyperparameter	Search Space	Best Value Found
batch_size	[32, 64, 128]	32
num_layers	2 to 6	3
units_i	[32, 64, 128, 256, 512]	[128, 128, 128]
activation	["relu", "tanh", "swish"]	"swish"
l2_i	[0.0, 1e-5, 1e-4, 1e-3]	[1e-5, 0.0001, 0.0]
use_bn_i	[True, False]	[True, True, False]
dropout_i	[0.3, 0.4, 0.5]	[0.3, 0.3, 0.5]
optimizer	["adam", "rmsprop", "sgd", "nadam"]	"adam"
learning_rate	[1e-4, 1e-2] (log)	0.00124



Layer (type)	Output Shape	Param #
dense_35 (Dense)	(None, 128)	8,320
batch_normalization_15 (BatchNormalization)	(None, 128)	512
dropout_27 (Dropout)	(None, 128)	0
dense_36 (Dense)	(None, 128)	16,512
batch_normalization_16 (BatchNormalization)	(None, 128)	512
dropout_28 (Dropout)	(None, 128)	0
dense_37 (Dense)	(None, 128)	16,512
dropout_29 (Dropout)	(None, 128)	0
dense_38 (Dense)	(None, 1)	129

Total params: 126,469 (494.02 KB)

Trainable params: 41,985 (164.00 KB)

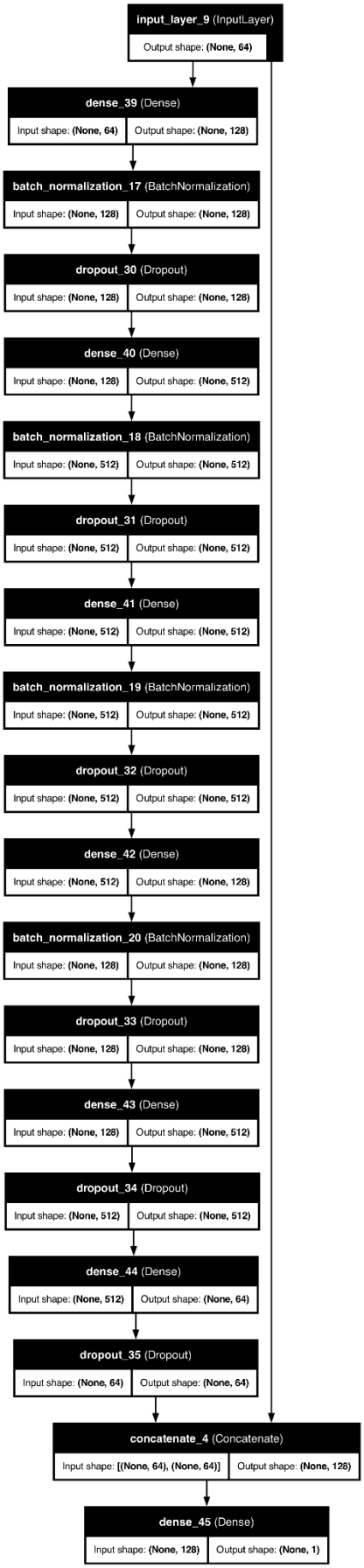
Non-trainable params: 512 (2.00 KB)

Optimizer params: 83,972 (328.02 KB)

Table A2: Wide & Deep NN – Hyperparameter Tuning Summary

Hyperparameter	Search Space	Best Value Found
batch_size	[32, 64, 128]	128
num_layers (Deep Path)	2 to 6	6
units_i	[64, 128, 256, 512]	[128, 512, 512, 128, 512, 64]

activation	["relu", "tanh", "swish"]	"swish"
l2_i	[0.0, 1e-5, 1e-4, 1e-3]	[1e-5, 1e-5, 1e-4, 0.0, 1e-3, 1e-3]
use_bn_i	[True, False]	[True, True, True, True, False, False]
dropout_i	[0.3, 0.4, 0.5]	[0.3, 0.4, 0.5, 0.5, 0.5, 0.5]
optimizer	["adam", "rmsprop", "sgd", "nadam"]	"nadam"
learning_rate	[1e-5, 1e-2] (log)	0.000254
loss_fn	["binary_crossentropy", "focal"]	"binary_crossentropy"



Layer (type)	Output Shape	Param #	Connected to
input_layer_9 (InputLayer)	(None, 64)	0	–
dense_39 (Dense)	(None, 128)	8,320	input_layer_9[0]…
batch_normalizatio… (BatchNormalizatio…	(None, 128)	512	dense_39[0][0]
dropout_30 (Dropout)	(None, 128)	0	batch_normalizat…
dense_40 (Dense)	(None, 512)	66,048	dropout_30[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 512)	2,048	dense_40[0][0]
dropout_31 (Dropout)	(None, 512)	0	batch_normalizat…
dense_41 (Dense)	(None, 512)	262,656	dropout_31[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 512)	2,048	dense_41[0][0]
dropout_32 (Dropout)	(None, 512)	0	batch_normalizat…
dense_42 (Dense)	(None, 128)	65,664	dropout_32[0][0]
batch_normalizatio… (BatchNormalizatio…	(None, 128)	512	dense_42[0][0]
dropout_33 (Dropout)	(None, 128)	0	batch_normalizat…
dense_43 (Dense)	(None, 512)	66,048	dropout_33[0][0]
dropout_34 (Dropout)	(None, 512)	0	dense_43[0][0]
dense_44 (Dense)	(None, 64)	32,832	dropout_34[0][0]
dropout_35 (Dropout)	(None, 64)	0	dense_44[0][0]
concatenate_4 (Concatenate)	(None, 128)	0	input_layer_9[0]… dropout_35[0][0]
dense_45 (Dense)	(None, 1)	129	concatenate_4[0]…

Total params: 1,515,334 (5.78 MB)

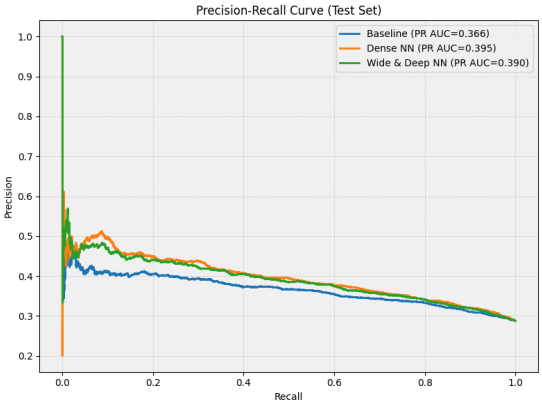
Trainable params: 504,257 (1.92 MB)

Non-trainable params: 2,560 (10.00 KB)

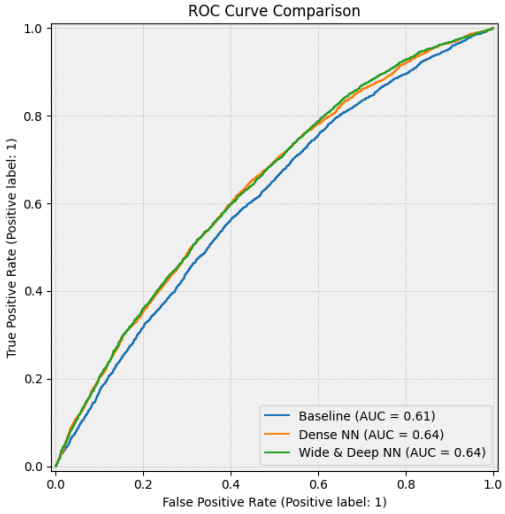
Optimizer params: 1,008,517 (3.85 MB)

6. Discussion of Results and Ethical Concerns

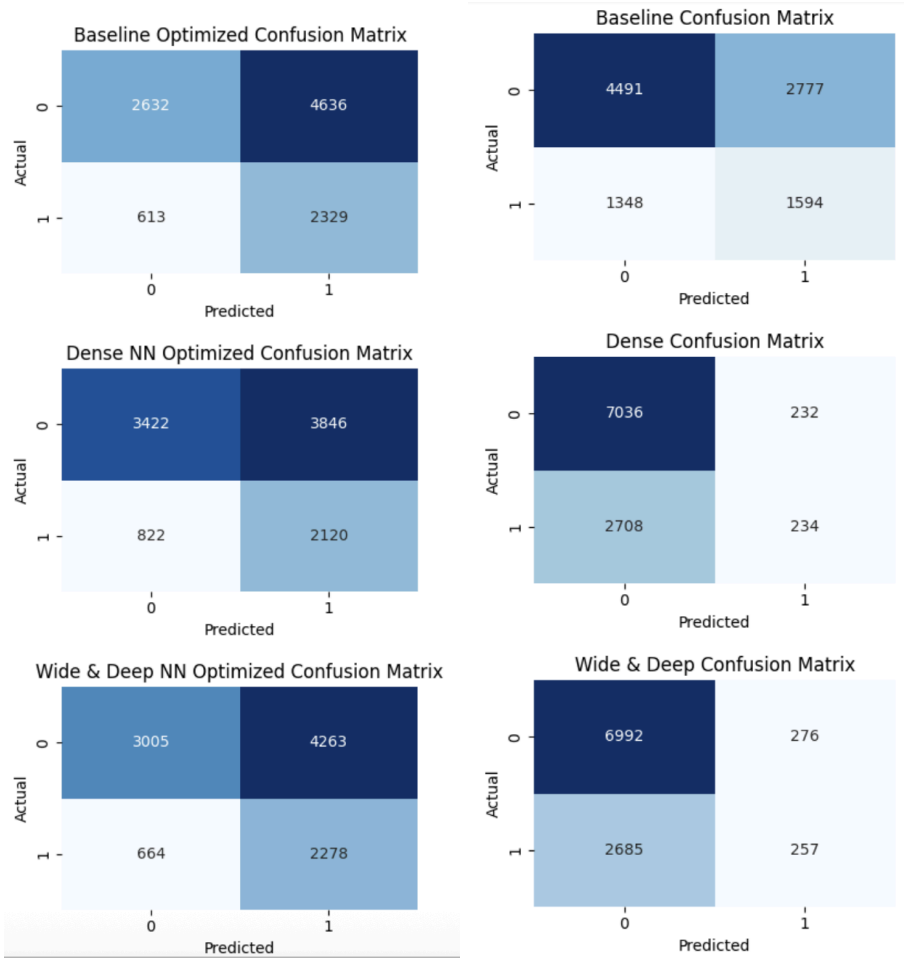
- Precision-Recall Curve



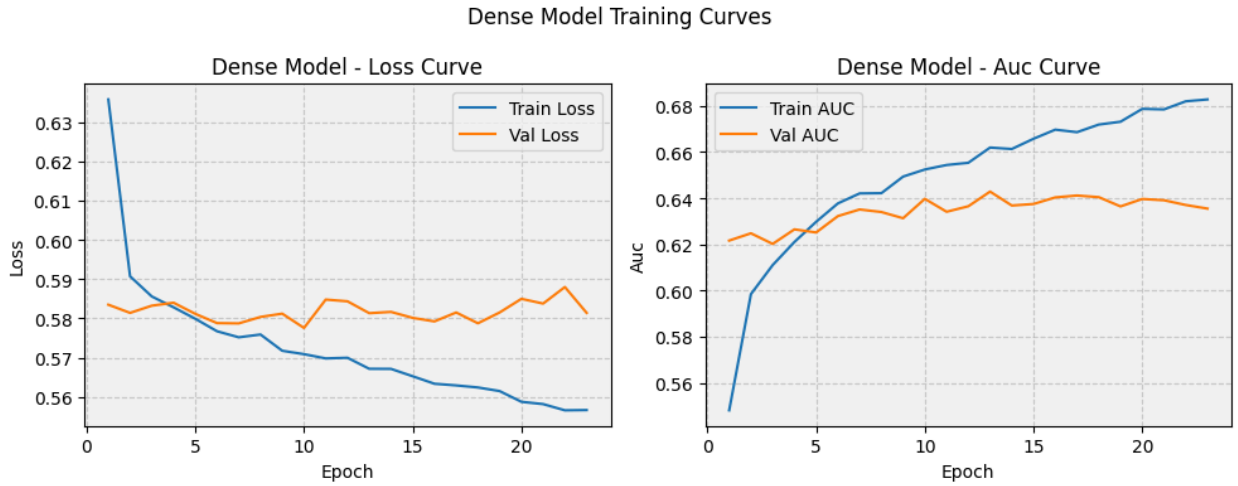
- ROC



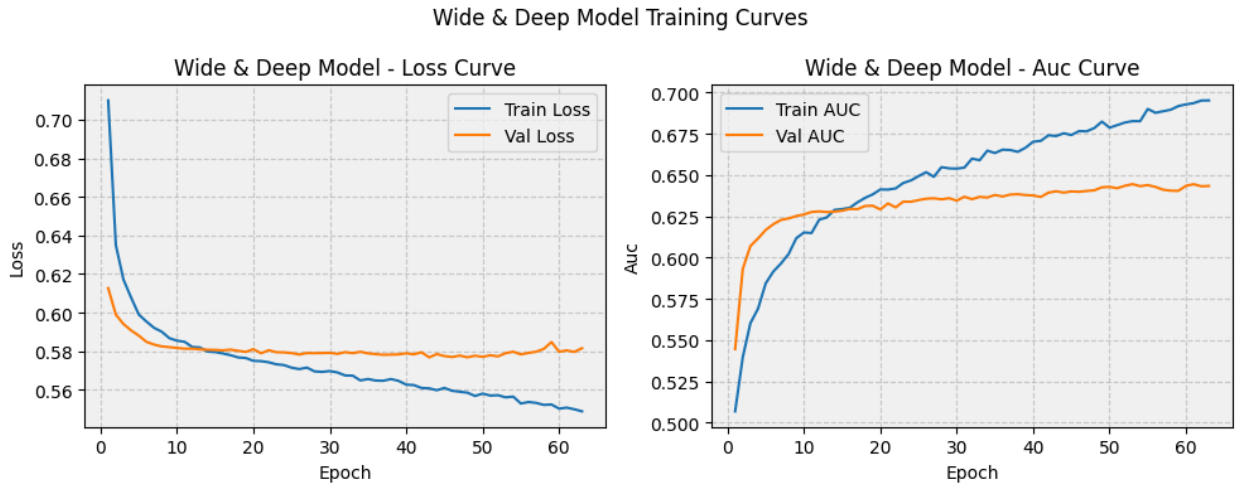
- Confusion matrix(1. optimised Threshold, 2. Default Threshold)



- Dense Model Training Curve



- Wide&Deep Model Training Curve



Data dictionary

Variable Name	Description	Datatype
Churn	Whether the customer has churned (Yes = churned, No = retained)	Numeric (Binary)
InactiveSubs	Engineered feature: Number of inactive subscriptions (UniqueSubs – ActiveSubs)	Numeric
ServiceArea	Customer's geographical service region. This nominal categorical feature was encoded using smoothed target mean encoding (smoothing parameter = 10), replacing each region with its weighted average churn rate. This approach preserves geographic churn patterns while avoiding high dimensionality from one-hot encoding.	Encoded Numeric (Originally Nominal Categorical)
Handsets	Total number of handsets on account	Numeric
HandsetPrice_Clean	Numeric handset price tier after parsing; "Unknown" handled separately.	Numeric
HandsetPrice_Unknown	Flag indicating whether handset price was originally "Unknown".	Numeric
CurrentEquipmentDays	Days current device has been used	Numeric
HandsetRefurbished	Whether the customer's handset is a refurbished model (Yes = refurbished, No = new)	Nominal Categorical
HandsetWebCapable	Whether the handset supports web access	Nominal Categorical
ChildrenInHH	Whether there are children in the household	Nominal Categorical
AgeHH1	Age of primary household member	Numeric
AgeHH2	Age of secondary household member (if applicable)	Numeric
TruckOwner	Whether customer owns a truck	Nominal Categorical
RVOwner	Whether customer owns a recreational vehicle	Nominal Categorical
Homeownership	Whether customer owns their home	Nominal Categorical

BuysViaMailOrder	Whether customer shops via mail order	Nominal Categorical
RespondsToMailOffers	Whether customer responds to promotional mailings	Nominal Categorical
OptOutMailings	Whether customer opted out of promotional mailings	Nominal Categorical
NonUSTravel	Whether customer has traveled internationally	Nominal Categorical
OwnsComputer	Whether customer owns a personal computer	Nominal Categorical
HasCreditCard	Whether customer owns a credit card	Nominal Categorical
RetentionOffersAccepted	Number of retention offers accepted	Numeric
NewCellphoneUser	Whether customer is a new cellphone user	Nominal Categorical
ReferralsMadeBySubscriber	Referrals made by the customer	Numeric
IncomeGroup	Encoded household income bracket (1 = lowest income, 9 = highest income). This is an ordinal variable indicating relative income levels.	Ordinal Categorical
CreditRating	Ordinal credit score category where 1 = Highest, 2 = High, 3 = Good, and 4 = Medium. Reflects customer's financial trustworthiness.	Ordinal Categorical
HandsetPrice	Price tier of the current handset	Nominal Categorical
OwnsMotorcycle	Whether customer owns a motorcycle	Nominal Categorical
AdjustmentsToCreditRating	Number of changes to credit rating	Numeric
MadeCallToRetentionTeam	Whether customer made a call to retention	Nominal Categorical
PrizmCode	Customer's residential environment type: Suburban, Town, Rural, or Other. This feature reflects simplified socio-geographic segmentation.	Nominal Categorical
Occupation	Customer occupation category	Nominal Categorical
MaritalStatus	Indicates whether the customer is married (Yes), not married (No), or unknown. Simplified binary marital status with missing values labelled as Unknown.	Nominal Categorical
MonthlyRevenue	Monthly amount paid by the customer	Numeric
MonthlyMinutes	Total number of voice minutes used in the month	Numeric
TotalRecurringCharge	Monthly recurring charges excluding overages	Numeric
DirectorAssistedCalls	Number of calls assisted by a directory operator	Numeric
OverageMinutes	Minutes used beyond the customer's plan limit	Numeric
RoamingCalls	Number of calls made while roaming	Numeric
PercChangeMinutes	Percentage change in minutes used compared to previous month	Numeric
PercChangeRevenues	Percentage change in monthly revenue compared to previous month	Numeric
DroppedCalls	Number of dropped calls	Numeric
UnansweredCalls	Number of calls not answered	Numeric
CustomerCareCalls	Number of calls made to customer care	Numeric
ThreewayCalls	Number of three-way calls made	Numeric
ReceivedCalls	Number of calls received by the customer	Numeric

OutboundCalls	Number of calls made by the customer	Numeric
InboundCalls	Number of calls received from others	Numeric
PeakCallsInOut	Number of calls made or received during peak hours	Numeric
OffPeakCallsInOut	Number of calls made or received during off-peak hours	Numeric
CallWaitingCalls	Number of call waiting events	Numeric
MonthsInService	Number of months customer has been with the provider	Numeric

Appendix B – Generative AI Usage

OpenAI.(2025). *ChatGPT* (Jul 7 version) [Large language model]. https://chat.openai.com/chat

Generative AI tools were used throughout this assignment to support code development, debugging, and documentation. Specifically, **ChatGPT (GPT-4)** was used as a productivity assistant to **accelerate model prototyping, tune neural network architectures**, and **clarify technical concepts**. All outputs were reviewed, edited, and validated before integration into the final code and report. No AI-generated text was copied verbatim.

Some examples of how Generative AI contributed to this project include:

- **Data Preprocessing Advice**: I asked ChatGPT how to handle features with high cardinality. In particular, I sought guidance on encoding techniques suitable for geographic categorical features like ServiceArea, where one-hot encoding would be impractical. Based on this, I applied smoothed target mean encoding to capture regional churn risk without inflating dimensionality.
- **Code Snippet Generation:** AI was used to generate Python code templates for tasks such as plotting binned churn rates, handling missing values, and provided guidance on structuring preprocessing pipelines.
- **Writing and Editing Support:** AI provided suggestions for structuring sections of the report, such as framing the business problem.
- **Model Architecture Design**: Helped explain the difference between fully connected feedforward networks and Wide & Deep networks; supported the Keras Functional API implementation for dual-branch architectures.
- **Focal Loss Implementation**: Guided the creation of a custom `FocalLoss` class in Keras to mitigate class imbalance by focusing more on hard-to-classify churners.
- **Hyperparameter Tuning (Bayesian Optimization)**: Suggested hyperparameter ranges, optimizer options, regularization techniques (e.g., dropout, L2), and explained trade-offs between batch sizes and learning rates for both Dense and Wide & Deep models.
- **Debugging and Refactoring**: Resolved issues such as shape mismatches, early stopping callbacks, and errors with `KerasTuner` configuration; explained when to use validation metrics such as `val_recall` vs `val_auc`.
- Writing and Editing Support: Provided suggestions to improve the clarity and conciseness of report sections (e.g.,model descriptions). Also helped rephrase performance summaries into academic tone.

Examples of prompts include:

- "What is the best way to encode a high-cardinality categorical variable without one-hot encoding all levels?"
- “What is a good way to stratify a train-test split when the target is imbalanced?”
- "Can you help me rephrase this section to be more concise and academic?"
- “Why am I getting ConvergenceWarning when using logistic regression with scikit-learn?”
- *“How to implement focal loss in Keras for binary classification?”*
- *“What’s a good wide & deep architecture for tabular churn prediction data?”*
- *“How to write an early stopping callback that restores the best weights?”*
- *“Help me refactor this Keras model to use the Functional API.”*

All AI-assisted content was cross-verified, independently tested, and modified to suit the project’s specific goals. AI helped accelerate experimentation and improve code readability, but final decisions—including model selection, evaluation strategy, and narrative framing—were made using domain expertise and critical judgment.

This appendix is provided to ensure transparency in the use of generative tools and to acknowledge AI’s role in supporting, but not replacing, the author’s original contribution.