Churn Prediction with a Deep Neural Network on Telco Customer Data: A Comprehensive Tutorial

1. Introduction

Since the market is competitive, it makes it much more cost effective to retain existing customers rather than acquiring new ones. The rate at which customers discontinue the product or service from the company will have a huge impact on the revenue of the company. The ability to accurately predict churn models allows businesses to target at risk customers with customized incentives and diminish the probability of churn and enhance the resulting length of customer value.

This tutorial tries to introduce how to use a Deep Neural Network (DNN) for churn prediction using the Telco Customer Churn dataset. We cover from data preprocessing to exploration, model building, evaluation, visualization in the whole pipeline. Our objective is to offer a rigorous and educational tutor that also matches academic and business criteria. (Pedregosa, 2011)

Key Topics Covered:

1. **Dataset Overview** and **Preprocessing**
2. **DNN Construction** with Keras
3. **Training** (with early stopping)
4. **Evaluation Metrics** (accuracy, precision, recall, F1, ROC AUC)
5. **Advanced Visuals** (Interactive ROC, Radar Chart for Feature Importance, Violin Plot, 3D Scatter, Smoothed Learning Curve, Correlation Heatmap)

By following these steps, you'll gain both theoretical and practical insights into building and interpreting deep learning models for churn prediction.

2. Why Churn Prediction Matters

1. **Revenue Preservation**: It is estimated that it costs 5 times — or more — to acquire a new customer than to retain one. Churn model allows companies to intervene by finding customers susceptible to leaving.
2. **Targeted Retention Strategies**: With a reliable churn score, marketing teams can design special offers, loyalty programs, or personalized services to keep at-risk customers engaged.
3. **Data-Driven Insights**: By analyzing which features are most predictive of churn, businesses can refine their products, pricing, and support processes.

3. Dataset Overview

3.1. Telco Customer Churn Dataset

The dataset consists of **7043 records** (rows) and **21 features**, each representing a customer's demographic, account, and usage information. Key columns include:

- **customerID**: Unique identifier for each customer.
- **gender**: Whether the customer is male or female.
- **SeniorCitizen**: Indicator if the customer is 65+.
- **Partner, Dependents**: Family status.
- **tenure**: Number of months the customer has stayed.
- **PhoneService, MultipleLines**: Telephone usage details.
- **InternetService**: DSL, Fiber optic, or No internet.
- **OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport**: Additional internet-based services.
- **StreamingTV, StreamingMovies**: Entertainment service usage.
- **Contract**: Month-to-month, one-year, or two-year plan.
- **PaperlessBilling**: Whether billing is paperless.
- **PaymentMethod**: e.g., Electronic check, Mailed check.
- **MonthlyCharges, TotalCharges**: Monthly and total billing amounts.
- **Churn**: Target variable (Yes = churn, No = no churn).

3.2. Basic Statistics

- **Shape**: (7043, 21).
- **Target Distribution**: 5174 "No" (approx. 73.4%) and 1869 "Yes" (approx. 26.6%). This indicates a moderate class imbalance.
- **Data Types**: 2 integer, 1 float, 18 object. Some columns (e.g., TotalCharges) may need conversion from string to numeric.

3.3. Data Splitting

After encoding, the dataset grows to 31 columns, with the final shapes:

- **Train**: (5634, 30)
- **Test**: (1409, 30)

We used an **80:20** split, stratifying on Churn to maintain the proportion of yes/no churn across sets.

4. Data Preprocessing

1. **Convert TotalCharges:** We convert from string to float. Missing or invalid values can be set to the median.

2. **Encode Categorical Variables:** gender, Partner, Dependents, and so forth are turned into binary or multi-class dummy variables. For instance, gender becomes gender_Male, InternetService might become [InternetService_Fiber optic, InternetService_No].
3. **Scale Numerical Features:** MonthlyCharges, TotalCharges, and tenure are scaled via StandardScaler. This ensures the DNN's gradient-based optimization converges more smoothly.

By carefully handling each column, we create a clean dataset for modeling.

5. Building a Deep Neural Network with Keras

5.1. Model Architecture

We use a **sequential** architecture with several dense layers:

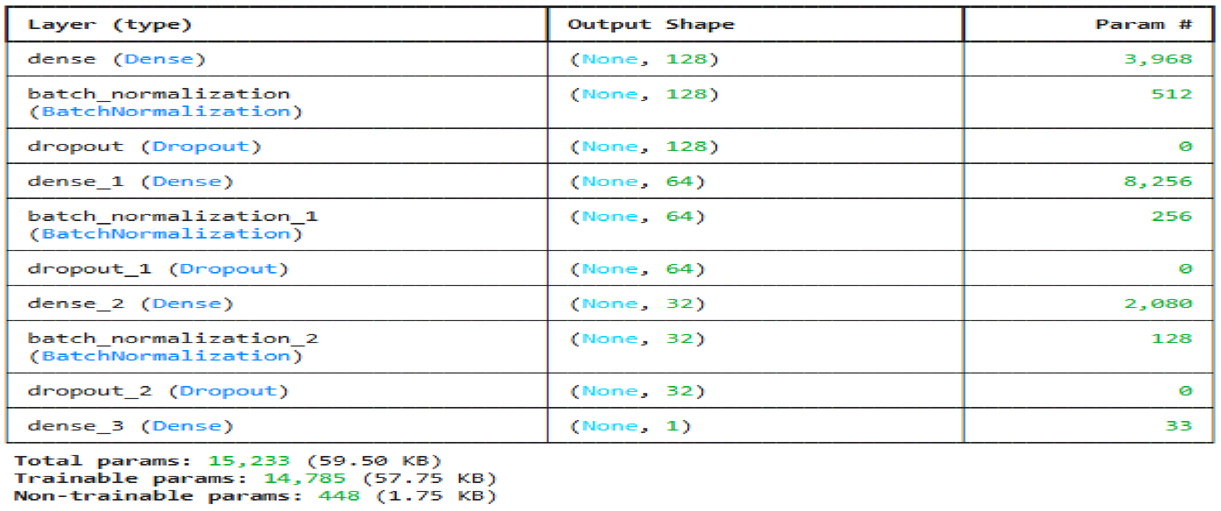
1. **Input Layer**: Accepts 30 features (after encoding).
2. **Dense(128, ReLU)**: Large hidden layer to learn complex interactions.
3. **BatchNormalization**: Normalizes activations, improving stability. (Lundberg, 2017)
4. **Dropout(0.3)**: Randomly zeroes 30% of neurons, reducing overfitting.
5. **Dense(64, ReLU)**: Additional hidden layer for refined feature extraction.
6. **BatchNormalization** and **Dropout(0.3)** repeated for further regularization.
7. **Dense(32, ReLU)** + **BatchNormalization** + **Dropout(0.2)** for final transformations.
8. **Dense(1, Sigmoid)**: Outputs a single probability for churn (Yes=1, No=0).

Why This Setup?

- **Multiple Layers**: Depth helps capture subtle patterns in user behavior and billing data.
- **Dropout**: Vital for controlling overfitting, especially with a modest dataset (~7000 rows).
- **Batch Normalization**: Speeds up training and reduces sensitivity to initial weights.

5.2. Model Summary

This indicates a moderate network size, suitable for a dataset with a few thousand samples.



Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	3,968
batch_normalization (BatchNormalization)	(None, 128)	512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
batch_normalization_1 (BatchNormalization)	(None, 64)	256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
batch_normalization_2 (BatchNormalization)	(None, 32)	128
dropout_2 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 1)	33

```
Total params: 15,233 (59.50 KB)
Trainable params: 14,785 (57.75 KB)
Non-trainable params: 448 (1.75 KB)
```

6. Training and Early Stopping

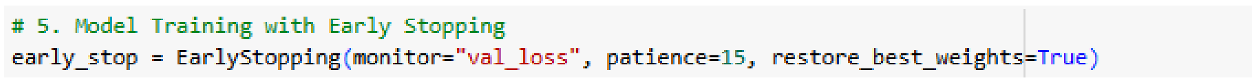
6.1. Configuration

We compile the model with:

- **optimizer='adam'**: Adaptive moment estimation for gradient updates.
- **loss='binary_crossentropy'**: Standard for binary classification.
- **metrics=['accuracy']**: Basic performance measure.

6.2. EarlyStopping Callback

We monitor val_loss with a patience of 15 epochs:



```
# 5. Model Training with Early Stopping
early_stop = EarlyStopping(monitor="val_loss", patience=15, restore_best_weights=True)
```

This halts training when the validation loss stops improving, reverting to the best weights. Our logs show training often stabilizes around 40–50 epochs.

6.3. Observed Training Logs

- **Training accuracy**: ~0.82
- **Validation accuracy**: ~0.78–0.79
- **Loss**: Decreases steadily, but the gap between training and validation suggests mild overfitting. (Lundberg, 2017)

7. Model Evaluation

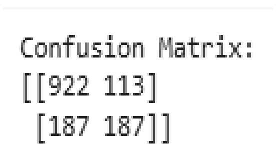
7.1. Final Test Performance

On the test set (1,409 samples), we observe:

Interpretation:

- The model is correct ~79% of the time.
- Class 0 (no churn) has high recall (0.89), meaning we rarely misclassify stable customers.
- Class 1 (churn) recall is 0.50, meaning we capture about half the churners. A bank or telecom might want a higher recall if preventing churn is a priority.

7.2. Confusion Matrix



```
Confusion Matrix:
[[922 113]
 [187 187]]
```

- **TN (922)**: Correctly predicted no churn.
- **FP (113)**: Predicted churn but actually no churn.
- **FN (187)**: Missed churners.
- **TP (187)**: Correctly predicted churners.

7.3. ROC AUC Score

AUC = 0.835
A higher AUC (clser to 1.0) implies better discrimination between churners and anti-churners. The model can separate the two classes fairly well at 0.835.

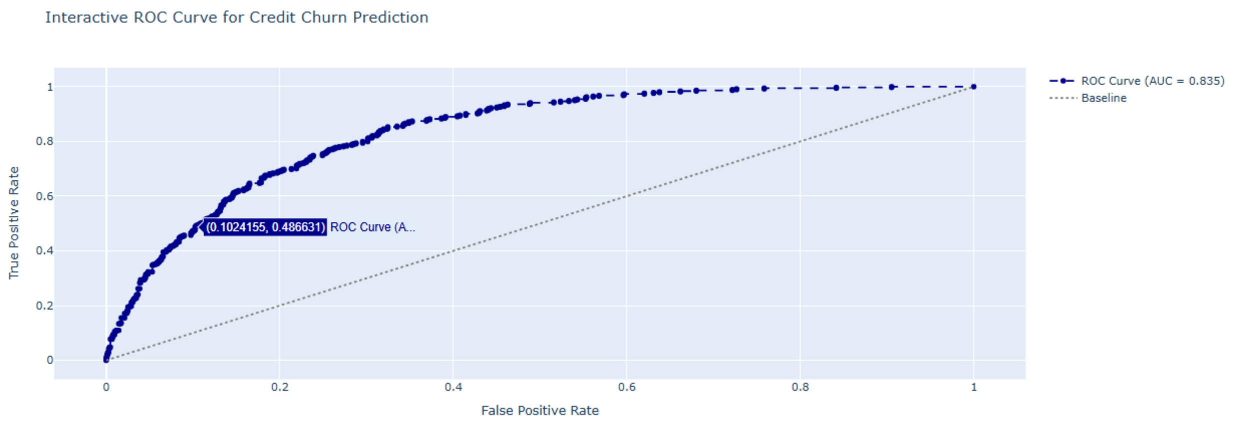
8. Advanced Visual Explorations

8.1. Interactive Plotly ROC Curve

This curve plots TPR (y-axis) vs. FPR (x-axis) at various thresholds. The AUC ~ 0.835. Hovering over the points in the interactive chart reveals exact TPR-FPR pairs.

Why It Matters:

- We can identify thresholds that yield an acceptable trade-off between recall (TPR) and fallout (FPR).
- In churn scenarios, businesses might shift the threshold to catch more churners at the expense of some false alarms.



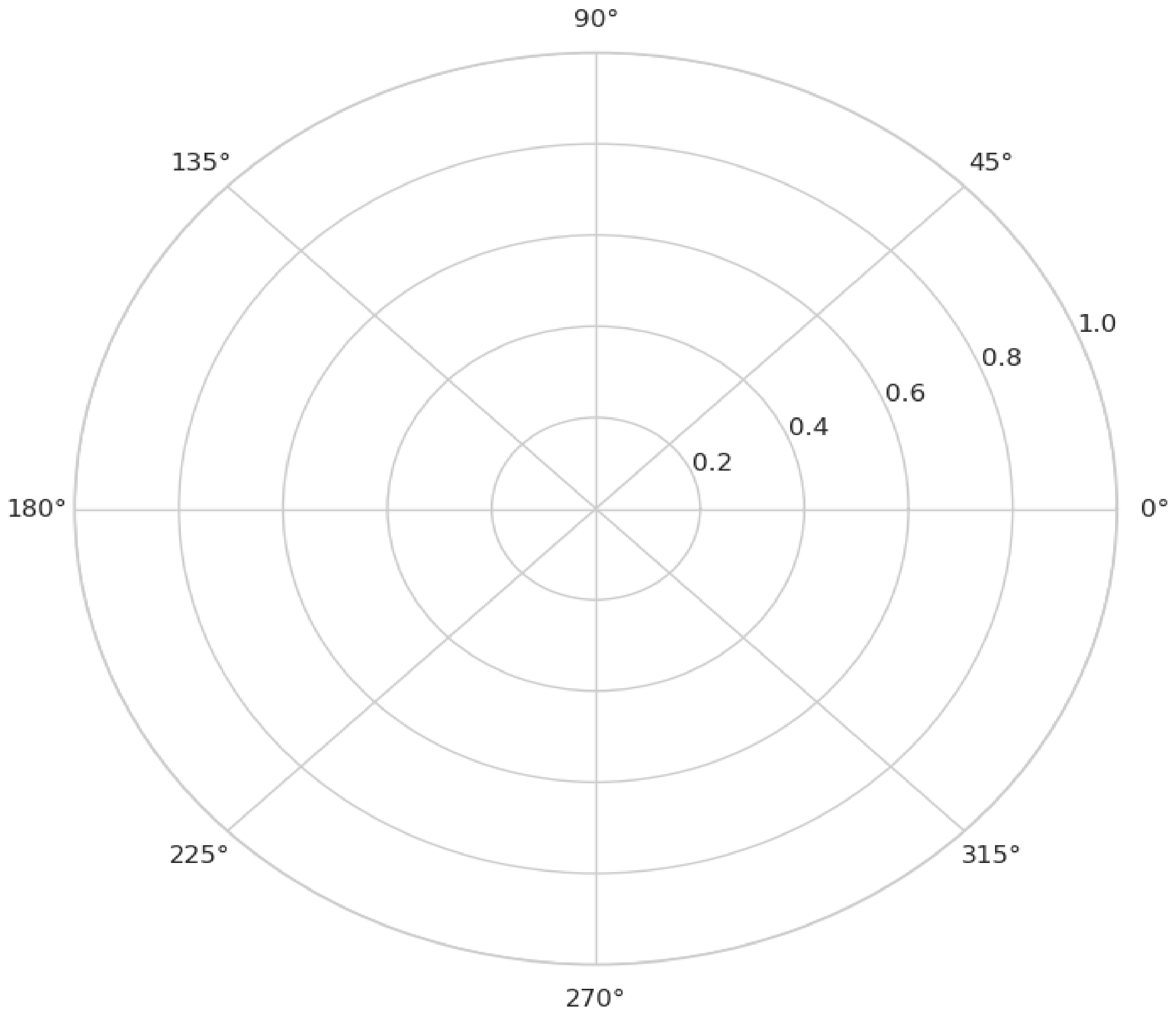
8.2. Radar Chart for Permutation Feature Importance

Using permutation_importance, we measure how each feature’s random shuffling affects accuracy. A radar (spider) chart normalizes these importances (range 0–1). Features with large radial extents are the most predictive.

Possible Key Features:

- **tenure**: Long-standing customers are less likely to churn.

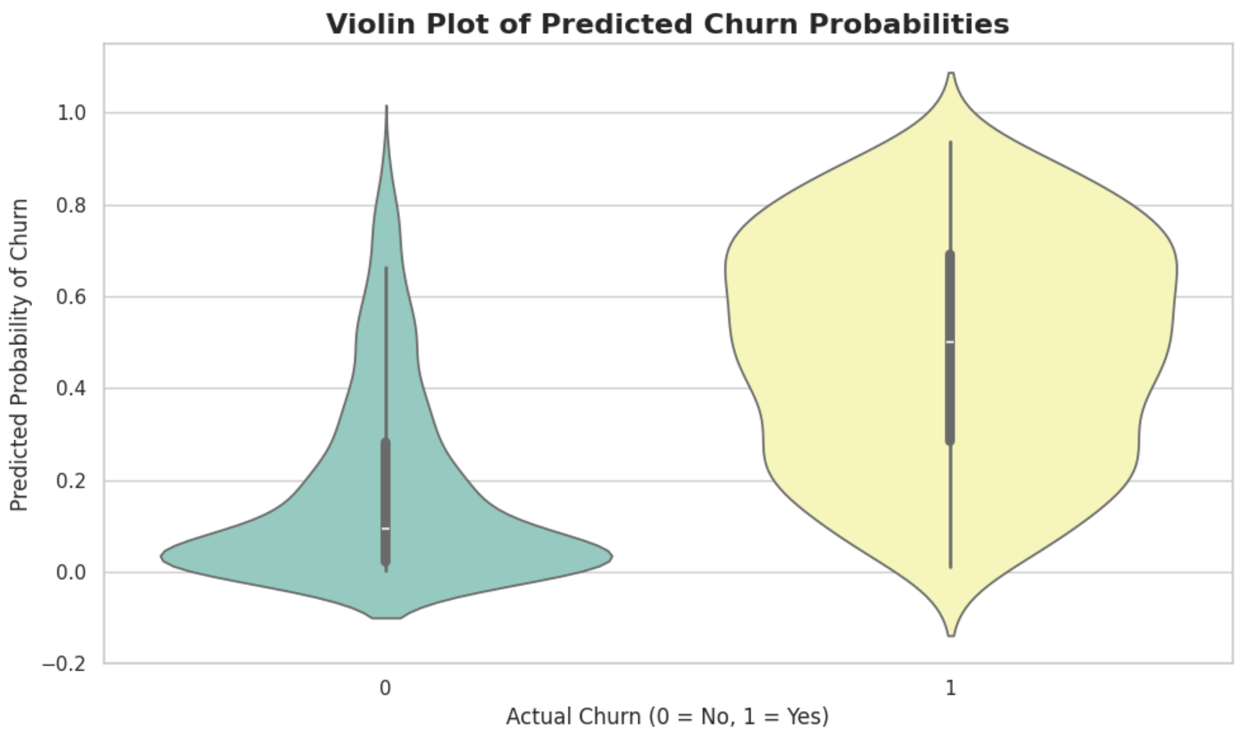
- **Contract_Two year**: Multi-year contracts typically reduce churn.
- **MonthlyCharges**: Higher monthly fees might correlate with dissatisfaction, increasing churn risk.



8.3. Violin Plot of Predicted Churn Probabilities

The violin plot is used to plot the predicted churn probability distributions for the actual classes (0 vs. 1). The “no churn” distribution clusters near lower probabilities and the “yes churn” near higher probabilities. Some overlap indicates borderline predictions.

Violin plots are a teaching tip and a combination of a boxplot with a kernel density estimate for a fuller picture of the distribution than a standard boxplot.

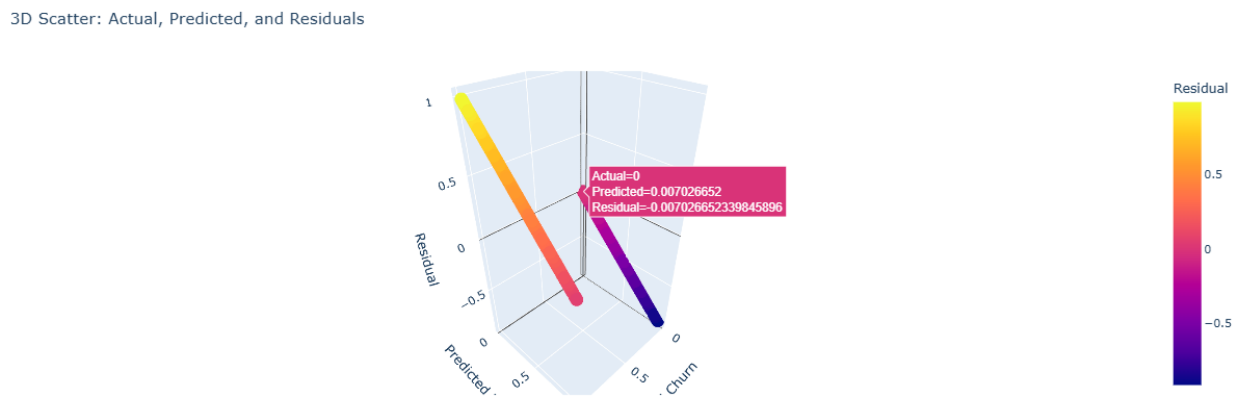


8.4. 3D Scatter Plot: Actual vs. Predicted vs. Residual

We create an interactive 3D plot:

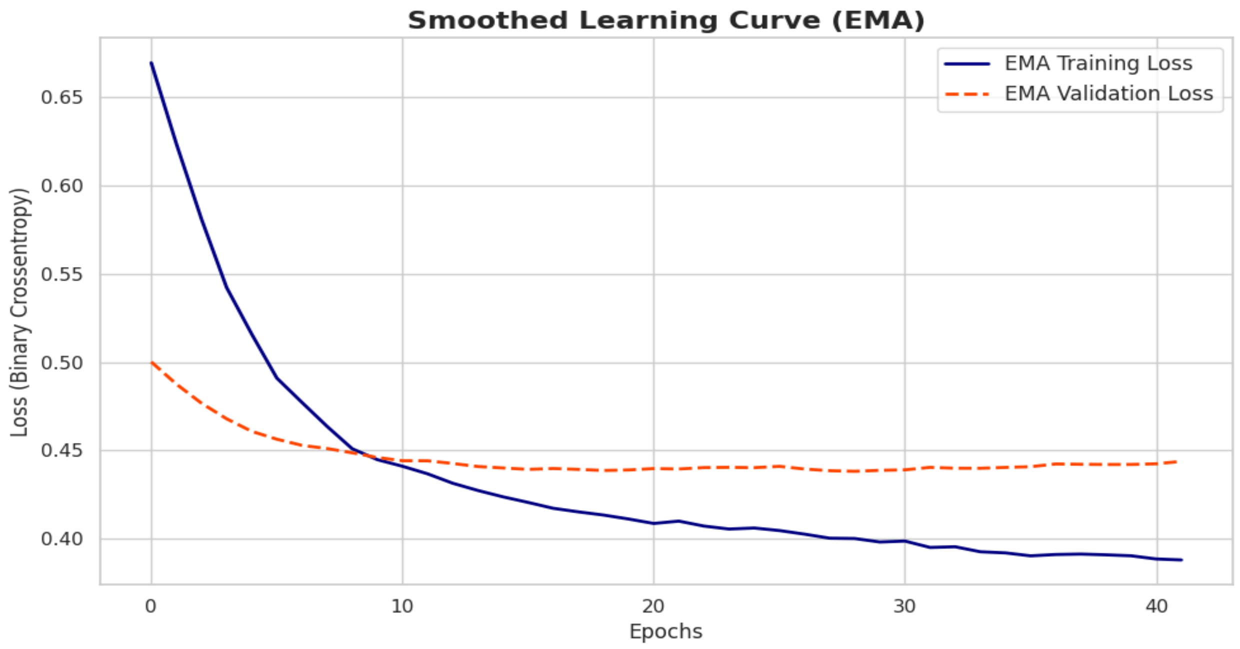
- **x-axis** = actual churn label (0 or 1)
- **y-axis** = predicted churn probability
- **z-axis** = residual (actual – predicted)

Color-coded by residual magnitude, points near z=0 are accurate predictions. Large positive z-values indicate actual=1 but predicted<0.5, i.e., missed churners.



8.5. Smoothed Learning Curve (EMA)

We take an exponential moving average of training and validation loss on epochs. Validation decrements off around 0.44-0.45, with training curve dipping below 0.40, indicating a mild overfitting. Early stopping prevents divergence further.



8.6. Correlation Heatmap of Test Features

A color-coded matrix shows pairwise correlations among the 30 test features. For instance:

- **Tenure** might be negatively correlated with Churn (longer tenure => less churn).
- Some dummy variables (like InternetService_No) can be strongly negatively correlated with others (InternetService_Fiber optic).

Tip: Rotating x-axis labels, adjusting figure size, and controlling annotation font helps clarity, as the chart can be busy with many features.

9. Key Observations and Insights

1. **Overall Accuracy ~79%**: Good for a baseline, but some organizations might prefer higher.
2. **Class 1 (Churn) Recall = 0.50**: We catch half the churners. Depending on cost sensitivity, we might aim for a higher recall.
3. **Potential Overfitting**: The gap between training and validation loss suggests the model memorizes some patterns. Additional regularization or data augmentation might help.
4. **Important Features**: Contract type, tenure, monthly charges, and certain internet service columns stand out as top contributors.
5. **Threshold Adjustments**: If business owners want fewer missed churners, we can shift the decision threshold below 0.5, capturing more churners at the cost of more false positives.

10. Potential Next Steps

1. **Hyperparameter Tuning**: Use advanced search (e.g., KerasTuner or Optuna) to optimize layer sizes, dropout rates, or learning rates. (Pedregosa, 2011)
2. **Cost-Sensitive Learning**: Incorporate a custom cost function that heavily penalizes false negatives (missed churners) if the cost of losing a customer is high.
3. **Explainability Tools**: Leverage LIME or anchors-based methods to produce local, human-readable rules explaining each prediction. (Lundberg, 2017)
4. **Cross-Validation**: Instead of a single train/test split, consider K-fold cross-validation for more robust performance estimates.
5. **Ensemble Approaches**: Combine the DNN with other models (e.g., random forests or gradient boosting) to create a stacked ensemble that might outperform any single model. (Lundberg, 2017)

11. Teaching Emphasis and Best Practices

1. **Proper Data Cleaning**: The dataset had columns needing numeric conversion and missing values. Without correct data handling, the model might degrade.
2. **Feature Engineering**: Additional domain-specific transformations (like average monthly charges per year or a “tenure bucket”) can boost performance.
3. **Regularization**: Dropout and batch normalization are crucial for stable training in moderate datasets.
4. **Early Stopping**: Minimizes overfitting and reduces wasted training epochs.
5. **Visualization**:
    - **Violin plots** highlight distribution of model confidence.
    - **Radar charts** convey permutation-based importance in a visually appealing way.
    - **3D scatter** fosters intuitive understanding of model errors.

12. Conclusion

This tutorial demonstrates a **Deep Neural Network** approach for predicting churn in the **Telco Customer Churn dataset**. The final model achieves:

- **79% accuracy**
- **ROC AUC = 0.835**
- **Recall(Churn=1) ~ 0.50**

Though these metrics are promising, they also offer some opportunity for improvement at capturing more churners. Advanced visualizations such as interactive ROC, radar chart, violin plots, 3D scatter, smoothed learning curve and correlation heatmap allow us understanding the model’s behaviour and providing cues to make further refinements. Through arbitrary changes of thresholds or adoption of cost sensitive learning, the telecom can more aggressively reduce churn.

Core Takeaways:

1. **Data Preparation**: Converting data types, encoding categoricals, scaling numeric columns are essential steps for neural network training.
2. **Model Architecture**: A multi-layer design with ReLU, batch normalization, and dropout balances expressiveness and regularization.
3. **Evaluation**: Standard classification metrics plus ROC AUC reveal performance. The confusion matrix pinpoints misclassifications, especially missed churners.
4. **Insights**: Some features (e.g., contract length, monthly charges, tenure) strongly influence churn risk. The model's limited recall suggests we can tune the threshold to favor capturing churners.
5. **Future Enhancements**: Advanced hyperparameter search, cost-sensitive learning, local interpretability methods, or ensemble techniques can refine performance.

Combining these best practices of robust data processing, good architecture, early stopping and visualization helps to crafts a high quality churn prediction solution. The synergy of deep learning, domain understanding, and efficient evaluation as shown in this tutorial enables others to tackle real world business challenges in customer retention.

11. References

- (Lundberg, 2017) A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NeurIPS)*.
- (Pedregosa, 2011) Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.