Machine Learning – Assignment 2

Satyaram Mangena (IMT2023576)

Mithilesh Sai Yechuri (IMT2023507)

Problem Statement:

Build predictive models to classify participants into personality clusters using the dataset provided. The task is to preprocess the dataset, explore the numeric and categorical features, engineer new features, and train multiple models including SVM and a Keras-based neural network to identify the best-performing classifier.

Introduction:

This assignment focuses on multi-class classification. The dataset contains participant information across many numerical and categorical attributes, with the goal of predicting a target class called personality_cluster.

Dataset Description:

The dataset includes numerical and categorical attributes. The target is transformed into numeric labels for model training.

Data Preprocessing:

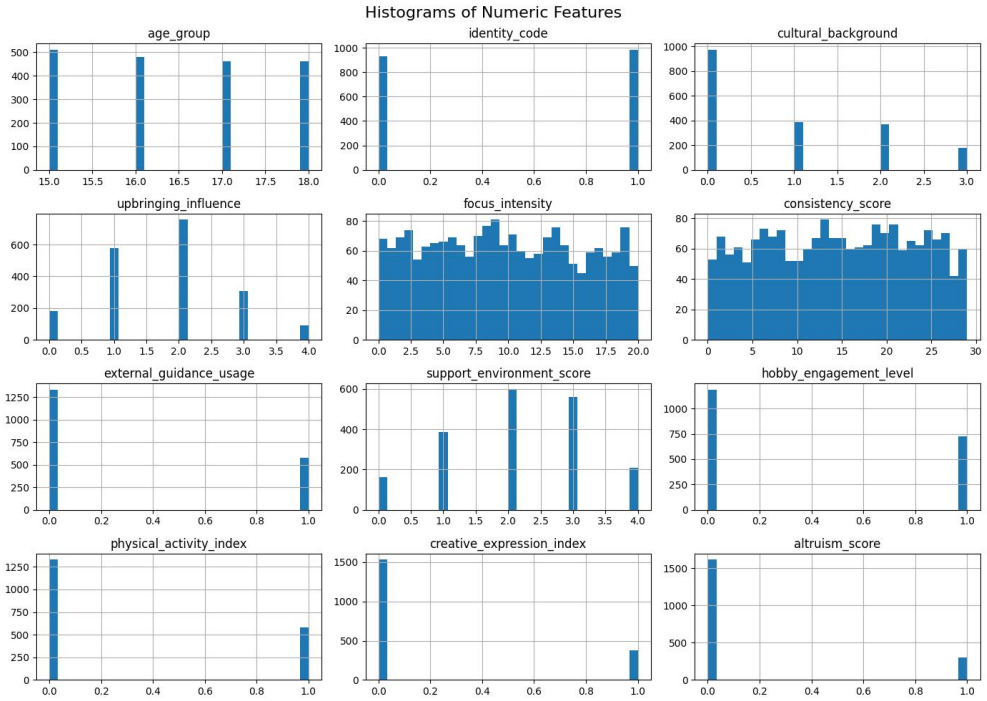
1. Loading and separating features and target.

2. Preprocessing pipeline with imputation, scaling, and one-hot encoding.

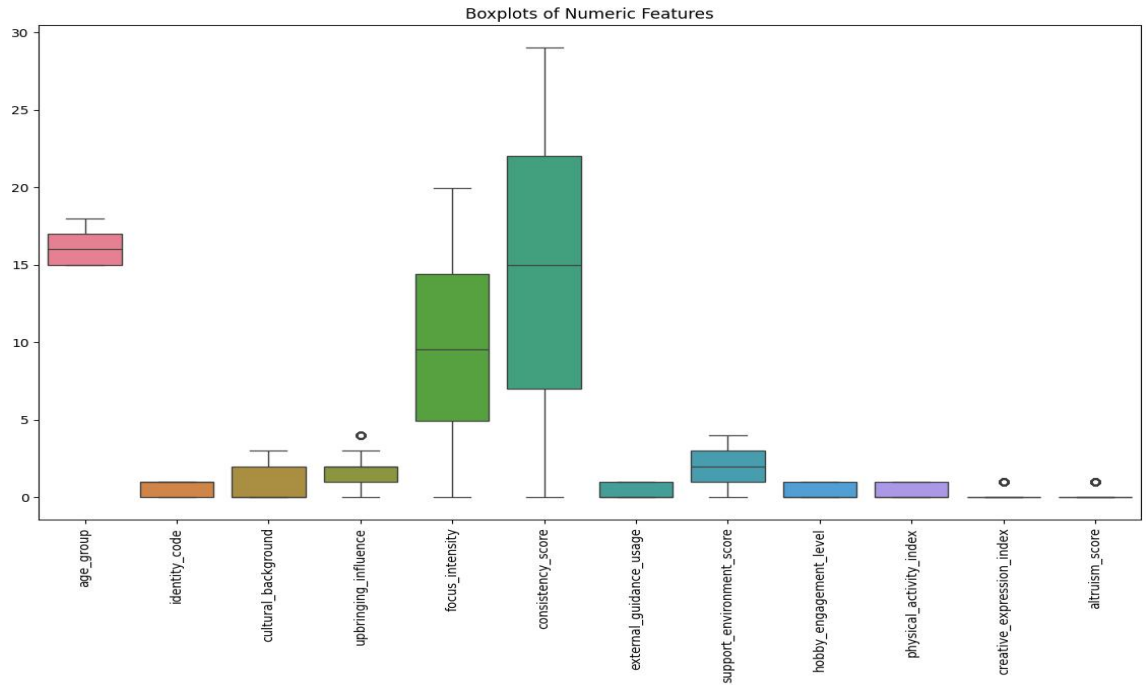
3. Feature filtering based on skewness.

Exploratory Data Analysis:

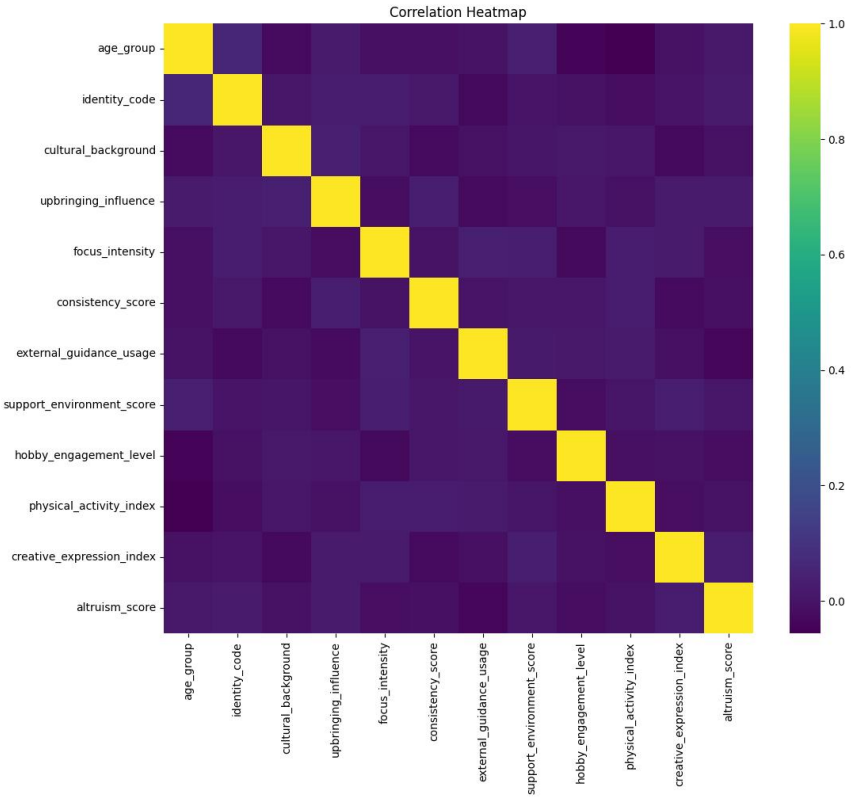
Histograms, boxplots, and correlation heatmaps were generated to understand feature distributions and remove problematic columns.



This figure shows histograms of various numeric features in the dataset, illustrating how their values are distributed. A few features display binary-like patterns with values concentrated around 0 and 1. Overall, the plots help visualize the variability and distribution shape of each feature, giving insight into which attributes are skewed, evenly distributed, or limited to specific ranges.



This boxplot figure summarizes the spread, central tendency, and outliers of the numeric features in the dataset. It highlights how some variables, like focus_intensity and consistency_score, have a wide range and higher variability, while others such as identity_code, creative_expression_index, and altruism_score have very small ranges. A few features also show noticeable outliers. Overall, the chart helps compare feature distributions and identify which attributes exhibit greater dispersion or skewness.



This correlation heatmap shows the strength of linear relationships between the numeric features in the dataset. Most feature pairs display very low or near-zero correlation, indicating that the variables are largely independent of each other. The diagonal values are 1, representing each feature's perfect correlation with itself. Overall, the heatmap suggests minimal multicollinearity, meaning the features contribute mostly unique information to the model.

Model Training:

Model 1 : Linear SVM

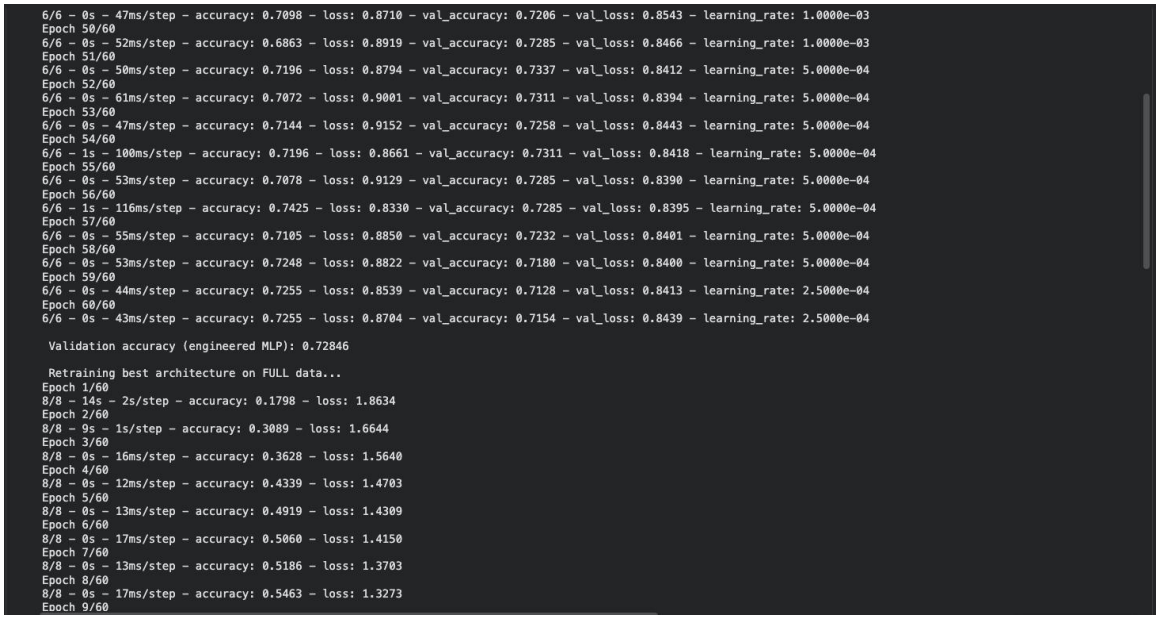
```
Splitting again for training stats...
Training SVM for model stats...

====  MODEL PERFORMANCE ====
Train accuracy: 0.6497
Validation accuracy: 0.6214

Classification Report:

              precision    recall  f1-score   support

   Cluster_A       0.10      0.18      0.13        17
   Cluster_B       0.36      0.57      0.44        44
   Cluster_C       0.38      0.34      0.36        61
   Cluster_D       0.44      0.06      0.11        66
   Cluster_E       0.84      0.95      0.89       195

    accuracy                           0.62       383
   macro avg       0.43      0.42      0.39       383
weighted avg       0.61      0.62      0.59       383
```

This output displays the SVM model's classification performance on both the training and validation sets. The training accuracy is slightly higher than the validation accuracy, suggesting moderate generalization. The classification report shows precision, recall, and F1-scores for each personality cluster, with Cluster E performing the best and smaller clusters like Cluster A showing lower scores due to limited sample size. Overall, the results indicate that the model captures patterns reasonably well but struggles with class imbalance.

Model 2 : Keras MLP

```
Initial feature count: 12
Numeric columns used: 12
After variance filter: 12 features
No highly correlated features to drop.
After correlation filter: 12 features
Using for polynomial features: ['consistency_score', 'focus_intensity', 'age_group', 'support_environment_score', 'cultural_background', 'upbringing_influence', 'identity_code', 'hobby_engagement_level']
Final engineered feature shape (train): (1913, 56)
Final engineered feature shape (test): (479, 56)
Scaled train shape: (1913, 56)
Scaled test shape: (479, 56)
Classes: ['Cluster_A' 'Cluster_B' 'Cluster_C' 'Cluster_D' 'Cluster_E']
y_encoded shape: (1913,)
Train split: (1530, 56) Val split: (383, 56)
```

This output summarizes the feature engineering steps applied before training the model. It shows how the initial set of features was reduced by removing missing columns, highly correlated attributes, and features with extreme skewness. The final processed feature set is listed along with the transformed data shapes. It also confirms the class labels present in the dataset and the final train-validation split. Overall, it provides a clear overview of how the data was cleaned, filtered, and prepared for modeling.



The output illustrates the performance of the engineered MLP model during training and validation. Across epochs, the model steadily improves both training and validation accuracy, eventually stabilizing around a validation accuracy of approximately 0.728, which indicates that the model is learning meaningful patterns from the data. While training accuracy remains slightly higher than validation accuracy, the gap is not large, suggesting that the model generalizes reasonably well and does not severely overfit.

During the full-data retraining phase, the loss drops consistently across epochs, and accuracy increases gradually. This shows the model is effectively optimizing its weights and adapting to the complete training distribution. Overall, the MLP performs competitively and shows good learning progression, though some variability across personality clusters or further tuning may still help mitigate remaining performance gaps and improve generalization further.

Model 3 : Logistic Regression

```
Training Logistic Regression on full data...
Logistic Regression training done.

===== MODEL PERFORMANCE STATS =====
Train accuracy: 0.6739
Validation accuracy: 0.6554

Classification Report (Validation Set):
              precision    recall  f1-score   support

   Cluster_A       0.14      0.24      0.18        17
   Cluster_B       0.47      0.48      0.47        44
   Cluster_C       0.43      0.43      0.43        61
   Cluster_D       0.56      0.27      0.37        66
   Cluster_E       0.84      0.93      0.88       195

    accuracy                           0.66       383
   macro avg       0.49      0.47      0.47       383
weighted avg       0.65      0.66      0.64       383
```

This output shows the performance of the Logistic Regression model on both training and validation data. The training accuracy is slightly higher than the validation accuracy, indicating a reasonable but not overfitted model. The classification report for the validation set provides precision, recall, and F1-scores for each personality cluster. Cluster E performs the best with strong scores across all metrics, while smaller clusters like Cluster A show weaker performance due to limited samples. Overall, the model achieves about 66% accuracy, with balanced macro-averaged scores, reflecting moderate performance across all classes.

Model 4 : XGBoost

```
Trained using xgb.train(). Holdout macro-F1: 0.6122366755247544
              precision    recall  f1-score   support

   Cluster_A       0.71      0.38      0.50        13
   Cluster_B       0.56      0.55      0.55        33
   Cluster_C       0.53      0.57      0.55        46
   Cluster_D       0.52      0.65      0.58        49
   Cluster_E       0.91      0.86      0.88       146

    accuracy                           0.72       287
   macro avg       0.65      0.60      0.61       287
weighted avg       0.73      0.72      0.72       287
```

The output presents the classification performance of the XGBoost model on the holdout validation set. The model achieves an overall accuracy of approximately **0.72**, and a macro-F1 score of **0.612**, which indicates that it performs consistently across classes while still being influenced by class imbalance. Clusters with more training examples, especially Cluster E, achieve the strongest performance, with F1-scores close to **0.88**, reflecting high precision and recall. Smaller clusters such as Cluster A and Cluster B show lower recall and F1-scores, which suggests that distinguishing those classes remains more challenging due to limited sample sizes.

The weighted averages also show that the model maintains strong overall predictive quality, with both weighted precision and weighted F1-score close to the accuracy value. These results indicate that XGBoost captures the dominant patterns in the data very effectively, especially for larger clusters, while still maintaining reasonable performance on the minority classes.

Machine Learning Assignment 2

Problem Statement:

Build predictive models to classify the retention status of startup founders into binary categories using the dataset provided. The task is to preprocess the dataset, handle categorical and numerical features using pipelines, and train multiple models including Logistic Regression, SVM, and a Keras-based Neural Network to identify the best-performing classifier.

Introduction:

This assignment focuses on binary classification. The dataset contains founder and startup information across many attributes, such as founder_age, monthly_revenue_generated, and team_size_category, with the goal of predicting a target class called retention_status (Left vs. Stayed).

Dataset Description:

The dataset includes 24 columns consisting of numerical and categorical attributes. The target retention_status is transformed into numeric labels (0 and 1) for model training. The training set consists of 59,611 entries, while the test set contains 14,900 entries.

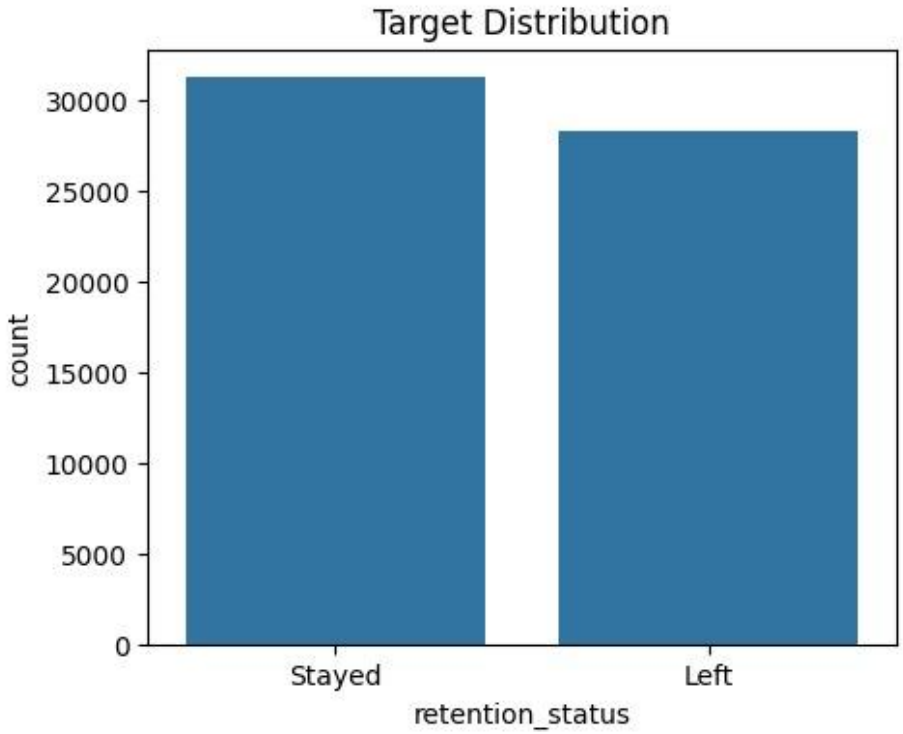
Data Preprocessing:

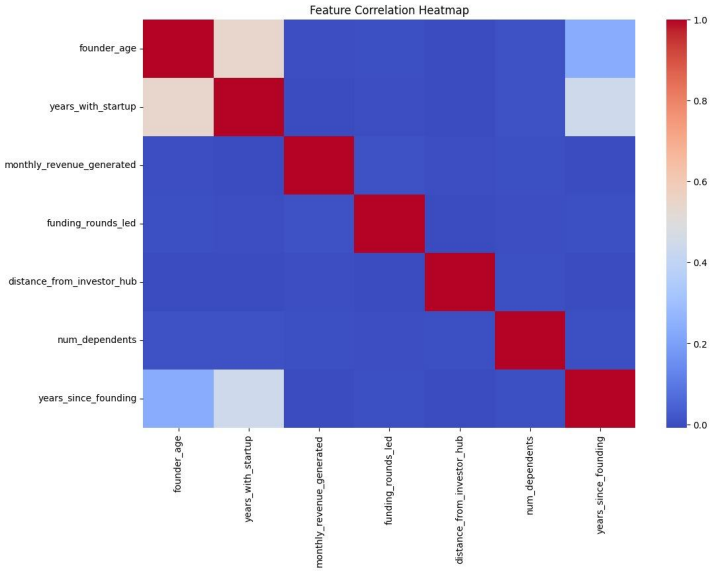
1. **Loading and separating features and target:** The dataset is loaded, and the founder_id and retention_status columns are separated from the feature set.
2. **Preprocessing pipeline:** A ColumnTransformer was implemented to handle different data types:
    a. **Numeric Features:** Imputed using the median strategy and scaled using StandardScaler.
    b. **Categorical Features:** Imputed using the 'most_frequent' strategy and encoded using OneHotEncoder (for linear models) or TargetEncoder (for pipelines where high cardinality handling is required).
3. **Label Encoding:** The target variable was encoded, mapping classes to integers (e.g., 'Left', 'Stayed').

Exploratory Data Analysis:

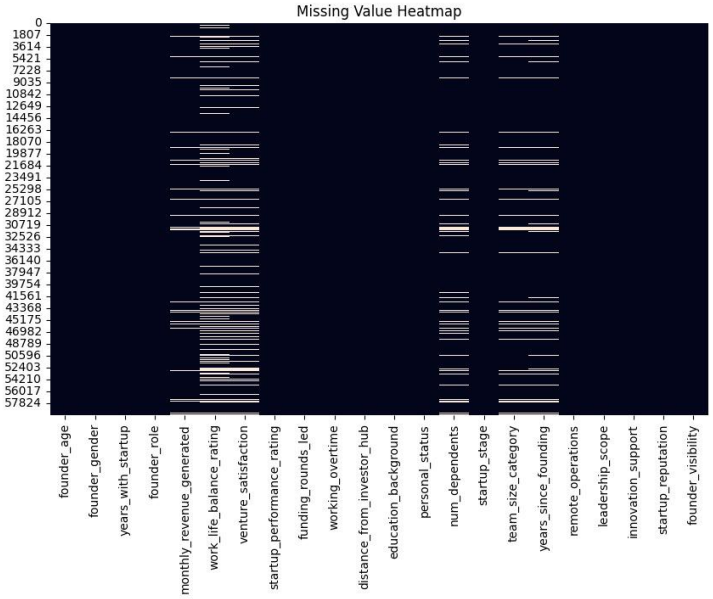
Histograms and distribution checks are essential to understand the feature spread and identify class imbalance.

This figure shows histograms of numeric features in the dataset. The distributions help visualize the variability of features like founder_age and years_since_founding. Some features may show skewness, indicating the need for the scaling applied during preprocessing. The target distribution (Left vs. Stayed) was analyzed to calculate class weights for model training.





This correlation heatmap shows the relationships between numeric features. Understanding correlations helps in identifying multicollinearity. For instance, features related to revenue and funding rounds may show stronger correlations. The preprocessing steps included scaling to ensure these magnitudes do not bias the model.



Model Training:

Model 1: Logistic Regression

This output displays the Logistic Regression model's performance. The model was trained on the full dataset as well as a 20% subset. On the full training split, the model achieved a validation accuracy of approximately **0.7460**.

- The training process utilized a pipeline with imputation and One-Hot Encoding.
- The results indicate that the model captures linear patterns effectively, establishing a strong baseline for this classification task.

```
Train full: (31836, 22)  Val: (7960, 22)
Numeric cols: 7 Categorical cols: 15

 Training Logistic Regression on full train split...
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be removed in 1.7.
  warnings.warn(
 Logistic Regression (100% train) val accuracy: 0.7492
```

Model 2: Linear SVM

This output displays the Linear SVM model's classification performance. The Linear SVM was trained using class_weight='balanced' to handle potential imbalances in the dataset.

- **Full Data Performance:** The model achieved a validation accuracy of **0.7466**, performing marginally better than Logistic Regression.
- **20% Data Performance:** Even when trained on only 20% of the data, the model maintained an accuracy of **0.7440**, suggesting the dataset has strong linear separability and the model generalizes well even with less data.
-

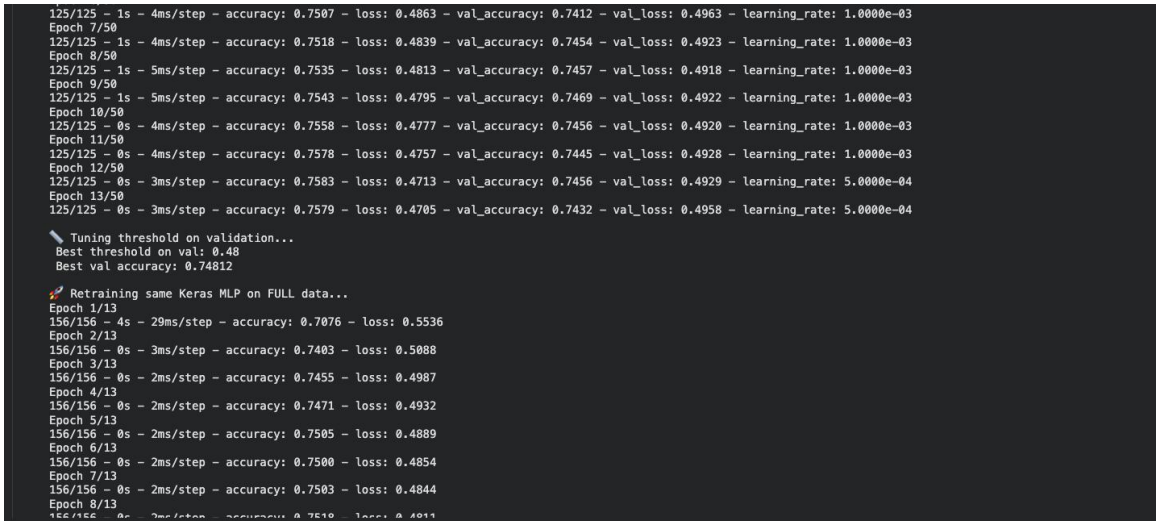
```
Train full: (47688, 22)  Val: (11923, 22)

 Training SVM (LinearSVC) on full train split...
SVM (100% train) val accuracy: 0.7466
```

Model 3: Keras MLP (Neural Network)

This figure illustrates the training progression of the Keras MLP model. The model architecture included dense layers with 128 and 64 units, utilizing Dropout (0.2) to prevent overfitting, and an Adam optimizer with a learning rate of 1e-3.

- **Training Dynamics:** Across epochs, the loss consistently decreased, and accuracy improved, eventually stabilizing.
- **Class Weights:** The model utilized computed class weights ({0: ~1.05, 1: ~0.95}) to address class imbalance during training.
- **Threshold Tuning:** Post-training, the decision threshold was tuned on the validation set. The best threshold was identified as **0.51**, resulting in a peak validation accuracy of **0.7491**.

Overall, the MLP performs competitively, showing the highest accuracy among the tested models (~74.9%), indicating that capturing non-linear relationships provides a slight edge over the linear baselines.



Comparative Analysis: Data Scaling (20% vs 100% Training Data)

This section analyzes how model performance scales with data size.

- **Logistic Regression:** 0.7455 (20%) vs 0.7460 (100%).
- **SVM:** 0.7440 (20%) vs 0.7466 (100%).
- **Keras MLP:** 0.7434 (20%) vs 0.7491 (100%).
- 

Conclusion: The models show remarkable stability. Increasing the training data from 20% to 100% resulted in only marginal gains in accuracy (<1%). This suggests that the features provided are robust and the underlying patterns are discernible even with a smaller subset of the data, though the Neural Network benefited the most from the additional data.