The CRISP-DM Process Model

Project: Human Activity Recognition Using Smartphones

Project Group 16

Students: Netta Yaakobi, Dorin Hazan and Yuval Tzur

Modeling

4.1 Select Modeling Technique

Task Select Modeling Technique

As the first step in modeling, select the actual modeling technique that is to be used initially. If multiple techniques are applied, perform this task for each technique separately.

Output Modeling Technique

This output refers to the actual modeling technique that is used.

We began with a basic Logistic Regression model. Logistic Regression is a simple, interpretable classifier that estimates probabilities using a linear combination of features.

After evaluating its performance, we moved on to a Deep Neural Network model. The Neural Network can capture more complex, nonlinear patterns in the data by stacking multiple layers of neurons.

Output Modeling Assumptions

Many modeling techniques make specific assumptions about the data, or the data format

Modeling Assumptions

Random Forest Classifier

- **Feature Format & Consistency**
  - All 561 input features are numeric and aligned between training and test sets.
  - No duplicate or misordered feature columns.
- **Data Completeness & Quality**
  - No missing values; any gaps have been imputed or removed.
  - Extreme outliers have been capped or filtered so they do not dominate tree splits.
- **Sampling & Independence**
  - Each sample (time window) is independent.
  - Training, validation, and test sets come from the same underlying distribution (no concept drift).
- **Class Representation**
  - Each activity class has enough examples to allow reliable vote aggregation across trees.
- **No Leakage**
  - No subject IDs or temporal information are used as features, preventing indirect leakage.

Deep Neural Network

- **Feature Scaling & Format**
  - All input features are numeric and have been standardized or normalized.
  - Feature distributions in training and test sets match closely.

- **Data Completeness & Quality**
  - No missing values; outliers have been treated to avoid exploding gradients.
- **Sampling & Independence**
  - Samples are independent draws; no overlap of time windows that would leak future data.
  - Training, validation, and test splits are IID samples from the same distribution.
- **Capacity vs. Data Size**
  - The network's depth and width are appropriate for the available sample size to avoid severe overfitting.
- **Reproducibility**
  - Random seed fixed for weight initialization and data shuffling to ensure consistent results.
- **No Feature Leakage**
  - Only sensor-derived features are used; no subject or session identifiers are fed into the network.

4.2 Generate Test Design

Task Generate Test Design

Prior to building a model, a procedure needs to be defined to test the model's quality and validity. For example, in supervised data mining tasks such as classification, it is common to use error rates as quality measures for data mining models. Therefore the test design will specify that the data set should be separated into training and test set, the model will be built on the training set, and its quality estimated on the test set.

Output Test Design

This deliverable describes the intended plan for training, testing and evaluating the models. A primary component of the plan is to decide how to divide the available data set into training data, test data and validation test sets.

We started by separating out the feature columns from the labels and subject IDs in both the training and test files.

We have two files - `train.csv` and `test.csv` - we splitted the training set into two parts:

- **Training subset:** 80% Training set
- **Validation subset:** 20% Validation set

We used stratified sampling on `y_train` so that each activity appears in the validation subset in roughly the same proportion as in the full training set. After splitting, the proportions of each activity remain nearly the same in both the training subset and the validation subset when compared to the original training set.

We also fix a random seed (42) to ensure that this split can be reproduced exactly.

Finally, the test set (`X_test`, `y_test`) remains completely untouched until the end, when we apply our selected model and report final accuracy.

4.3 Build Model

Task Build Model

Run the modeling tool on the prepared dataset to create one or more models.

Output Parameter Settings

With any modeling tool, there are often a large number of parameters that can be adjusted. This report lists the parameters and their chosen values, along with the rationale for the choice.

Output Parameter Settings
With any modeling tool, there are often a large number of parameters that can be adjusted. This report lists the parameters and their chosen values, along with the rationale for the choice.

Random Forest Classifier

- **n_estimators:** 200

  Provides enough trees to reduce variance while keeping training and inference time reasonable.

- **max_features:** `sqrt`

  Considers $\sqrt{561} \approx 24$ features per split, balancing split quality with tree diversity.

- **max_depth:** `None`

  Lets trees grow fully to capture complex patterns, with the ensemble itself guarding against overfitting.

- **min_samples_split:** 2

  Splits nodes as long as at least two samples exist, ensuring that trees explore meaningful splits.

- **random_state:** 42

  Fixes bootstrap sampling and feature-selection randomness for reproducibility.

Deep Neural Network (DNN)

- **Input Dimension:** 436 (PCA components)

  Reduces 561 raw features to those capturing ≥90 % variance.

- **Hidden Layers:**

  1024 → 512 → 256 → 128 neurons (each with ReLU)

  Decreasing widths enable hierarchical pattern learning while controlling parameter growth.

- **Kernel Regularizer:** L2 (1e-4) on all dense layers

  Penalizes large weights to improve generalization.

- **Batch Normalization:** after each dense layer, stabilizes and accelerates training by normalizing intermediate activations.
- **Dropout Rates:** 0.5 → 0.4 → 0.3 → 0.2

  Prevents neuron co-adaptation and reduces overfitting at each depth.

- **Optimizer:** Adam with ExponentialDecay
  - **Initial LR:** 1×10^{-3}
  - **Decay:** ×0.9 every 1,000 steps

  Adaptive learning rates speed convergence and reduce manual LR tuning.

- **Loss Function:** Categorical cross-entropy

  Standard choice for multi-class classification with softmax output.

- **Batch Size:** 64

  Balances gradient-estimate stability and computational efficiency.

- **Epochs:** up to 100 with EarlyStopping (patience = 10)

  Allows sufficient learning while halting before overfitting.

- **random_seed:** 42

  Ensures reproducible weight initialization and data shuffling.

Output Models

Run the modeling tool on the prepared data set to create one or more models

We ran our chosen modeling tools on the prepared training subset to produce three final models:

- **Neural Network Model**
  Trained on the same 80 % training subset with two hidden layers (128 → 64 neurons, ReLU) and a six-unit Softmax output. It was optimized using Adam, with early stopping on validation loss, and is now ready to predict on new data.
- **Random Forest Model**
  Trained on the 80 % training subset using 100 decision trees of maximum depth 10, selecting 561 features at each split. The ensemble aggregates votes from all trees to assign each sample to one of the six activity classes.

Each of these models has been saved after training on the prepared data and is ready for evaluation on the untouched 20 % test set.

Output Model Description

Describe the resultant model and assess its expected accuracy, robustness, and possible shortcomings. Report on the interpretation of the models and any difficulties encountered

Output Model Description
After training on the prepared data, we produced three models: Random Forestand a Neural Network. Below is a summary of each:

Random Forest

- **Expected Accuracy:** high accuracy (abouve 0.9) on the hold-out test set.
- **Robustness:**
  - Handles noisy or irrelevant features gracefully.
  - Resistant to outliers and moderate class imbalance.
- **Shortcomings:**
  - Large model size (200 trees) leads to slower inference and higher memory use.
  - Limited transparency: feature-importance rankings indicate which variables matter most, but do not reveal how they combine in each split.
- **Interpretation:**
  - We ranked features by mean decrease in impurity and by permutation importance to identify the most predictive sensor signals.
  - Detailed per-tree decision logic remains opaque.
- **Difficulties Encountered:**
  - Tuning tree depth and number of trees to balance bias vs. variance required multiple runs.

Deep Neural Network (Feedforward)

- **Expected Accuracy:** 0.87 on the hold-out test set.
- **Robustness:**
  - Capable of modeling complex, nonlinear feature interactions when properly regularized.
  - Sensitive to hyperparameter choices and prone to overfitting without early stopping.
- **Shortcomings:**
  - Acts as a “black box”; individual weight matrices and activations are hard to map back to meaningful sensor patterns.
  - Training is computationally intensive - limited by CPU throughput in our environment.
- **Interpretation:**
  - We relied on training/validation accuracy and loss curves to monitor learning.
  - Simple masking experiments suggested that frequency-domain features carry strong predictive signal, but direct weight inspection was not feasible.
- **Difficulties Encountered:**
  - Finding the right network depth, layer widths, dropout rates, and learning-rate schedule involved extensive trial and error.
  - Without early stopping (patience = 10), validation loss began rising around epoch 30.
  - CPU-only training forced us to cap epochs at 100 and batch size at 64 for reasonable turnaround.

4.4 Assess Model

Task Assess Model

The model should now be assessed to ensure that it meets the data mining success criteria and the passes the desired test criteria. This is a purely technical assessment based on the outcome of the modeling tasks.

Output Model Assessment

Summarizes results of this task, lists qualities of generated models (e.g. in terms of accuracy), and ranks their quality in relation to each other.

We now evaluate each trained model against our untouched test set, using the actual summary metrics obtained.

Random Forest:

Accuracy: 0.925

Macro Avg

Precision: 0.926

Recall: 0.921

F1-score: 0.922

Weighted Avg

Precision: 0.926

Recall: 0.925

F1-score: 0.924

Strengths: Robust to noisy or irrelevant features; handles complex feature interactions without extensive preprocessing.

Weaknesses: Large model size (200 trees) leads to slower inference and higher memory usage; internal decision logic is not easily interpretable beyond feature-importance rankings.

Neural Network (DNN)

Accuracy: 0.876

Macro Avg

Precision: 0.881

Recall: 0.872

F1-score: 0.873

Weighted Avg

Precision: 0.881

Recall: 0.876

F1-score: 0.875

Strengths: Capable of modeling complex, nonlinear relationships.

Weaknesses: Acts as a "black box"; requires careful hyperparameter tuning and is prone to overfitting without strong regularization and early stopping.

Ranking of Models (by test-set accuracy)

1. **Random Forest:** 0.925
2. **Neural Network (DNN):** 0.876

Random Forest clearly leads in both accuracy and balanced performance across classes. The DNN, while flexible, trails in raw accuracy and demands more tuning and compute to approach similar performance.

Output Revised Parameter Settings

According to the model assessment, revise parameter settings and tune them for the next run in task 'Build Model'. Iterate model building and assessment until you find the best model.

Output Revised Parameter Settings
Based on the test-set results, we propose the following updates for the next iteration:

Random Forest

- **n_estimators:** 200 → 300
  More trees reduce variance and often improve generalization.
- **max_depth:** 10 → 15
  Deeper splits capture richer feature interactions without overfitting too quickly.
- **max_features:** `sqrt` → `log2`
  Sampling fewer features per split decorrelates trees and boosts robustness.
- **min_samples_leaf:** 1 → 2
  Enforcing at least two samples per leaf prevents overly specific splits on noise.
- **random_state:** 42 (unchanged)
  Ensures reproducible bootstrap samples and feature splits.

Deep Neural Network (DNN)

- **Hidden Layers:** [128 → 64] → [256 → 128 → 64]
  A deeper, funnel-shaped network learns more abstract feature hierarchies.
- **Dropout:** 0.3 after each hidden layer
  Randomly dropping 30 % of neurons combats overfitting in a deeper architecture.
- **Learning Rate:** 0.001 → 0.0005
  A lower rate enables more stable, fine-grained convergence on small gradients.

- **Batch Size:** 32 → 64
  Larger batches smooth gradient estimates and improve throughput.
- **Epochs & Early Stopping:** Up to 100 epochs with patience = 10
  More training time with automatic stopping once validation loss plateaus.
- **Weight Initialization:** He normal (unchanged)
  Suits ReLU activations and maintains healthy gradient scales.
- **L2 Regularization:** Add 0.001 penalty on all dense layers
  Penalizes large weights to improve generalization in a deeper model.
- **random_seed:** 42 (unchanged)
  Keeps weight initialization and data shuffling consistent.

We will retrain both models on the 80% main training split, monitor performance on the 20 % validation hold-out, and compare updated test-set accuracy before selecting the final configuration.