SCHOLARLY REVIEW JOURNAL

E-ISSN: 2996-8380
Published by Leadership & Innovation Lab

SUMMER 2026

APPENDIX 7

Code for optimised XGBoost model

```
# Model w/ optuna optimisation

def objective(trial):

    """

    Defines the objective function for Optuna hyperparameter tuning of
the XGBRegressor model.


    Parameters:

    - trial: An Optuna Trial object which suggests values for the
hyperparameters.


    Returns:

    - rmse: The Root Mean Squared Error (RMSE) on the validation set,
used as the objective to minimize.

    """


    # Define the hyperparameters with search spaces for Optuna to
optimize

    param = {

        "objective": 'reg:tweedie',

        "n_estimators": 100,

        "early_stopping_rounds": 10,

        "learning_rate": trial.suggest_float("learning_rate", 1e-3,
0.3, log=True),
```

SCHOLARLY REVIEW JOURNAL

E-ISSN: 2996-8380
Published by Leadership & Innovation Lab

SUMMER 2026

```
        "max_depth": trial.suggest_int("max_depth", 3, 10),
        "min_child_weight": trial.suggest_int("min_child_weight", 1,
10),
        "subsample": trial.suggest_float("subsample", 0.5, 1.0),
        "colsample_bytree": trial.suggest_float("colsample_bytree",
0.5, 1.0),
        "gamma": trial.suggest_float("gamma", 1e-8, 1.0, log=True),
        "lambda": trial.suggest_float("lambda", 1e-8, 10.0, log=True),
        "alpha": trial.suggest_float("alpha", 1e-8, 10.0, log=True),
        "tweedie_variance_power":
trial.suggest_categorical("tweedie_variance_power", [i/10 for i in
range(11, 20)]),
        "eval_metric": "rmse",
        "random_state": 67,

    }

    model = xgb.XGBRegressor(**param)

    model.fit(
        X_train, y_train,
        eval_set=[(X_test, y_test)],
        verbose=False,
    )

    preds = np.abs(model.predict(X_test))
    rmse = np.sqrt(mean_squared_error(y_test, preds))
```

SCHOLARLY REVIEW

SCHOLARLY REVIEW JOURNAL

E-ISSN: 2996-8380
Published by Leadership & Innovation Lab

SUMMER 2026

```
    return rmse


study = optuna.create_study(direction="minimize")
study.optimize(objective, n_trials=50)


# Train final model with best parameters
best_params = study.best_params
best_params["objective"] = 'reg:tweedie'
best_params["n_estimators"] = 100
best_params["early_stopping_rounds"] = 10
best_params["eval_metric"] = "rmse"
best_params["random_state"] = 67


# Initialize and train the final model using the best-found
hyperparameters
model = xgb.XGBRegressor(**best_params)
model.fit(
    X_train, y_train,
    eval_set=[(X_test, y_test)],
    verbose=False,
)
```