Mohammad Samir Rajani

202315691

COMP3202: PROJECT EXPLANATION

1. Data

- **Training Data:** The training dataset contains 71 descriptive features and a target variable as the last column (V72) in the CSV file.

- **Test Data:** The test dataset contains the same 71 features and is used solely to generate predictions.

2. Bottom-line Model Production

- Three baseline classifiers were built using the scikit-learn library with default parameters:

    - **Random Forest:** Implemented with RandomForestClassifier.

    - **k-Nearest Neighbors (k-NN):** Implemented with KNeighborsClassifier.

    - **Support Vector Machine (SVM):** Implemented with SVC.

- *Decisions:*

    - **Random State:**
      A fixed random state of 42 was set in applicable models. This value is arbitrary but widely used in the data science community as a conventional seed to ensure reproducible results. Reproducibility is vital for comparing outcomes and validating experiments.

    - **Pipeline Usage:**
      For k-NN and SVM, which are sensitive to the scale of features, a `Pipeline` is utilized to combine preprocessing (using `StandardScaler`) with model fitting. Pipelines help ensure that the same scaling transformation is consistently applied during both

cross-validation and final predictions, reducing data leakage and making the code cleaner and more maintainable.

3. Tuning Process and Parameter Selection

- *Tuning Objective:*

  The aim was to improve predictive performance by fine-tuning parameters using cross-validation with the F1 score as the evaluation metric.

- *Choice of GridSearchCV:*

  Grid Search was chosen because our parameter space was moderate in size. Being more systematic than Randomized Search, it allows us to explore all combinations of values in the specified grid, ensuring an exhaustive search and reliable identification of the best parameter settings.

- *Parameter Grids:*

  - **Random Forest:** Tuned the number of trees (n_estimators). These values balance model complexity and computation time. More trees can yield better performance but at an increased computational cost. The maximum depth (max_depth) allows for varying depths which helps control overfitting; None represents unbounded depth, while fixed values help find an optimal balance between bias and variance.

  - **k-NN:** Tuned the number of neighbors (n_neighbors). This range covers small to moderately large neighborhood sizes, which affects the smoothness of the decision boundary. The weighting function (weights) tests both options to determine whether giving closer neighbors more influence (distance weighting) improves the predictive performance.

  - **SVM:** The regularization parameter C determines the trade-off between achieving a low error on training data and maintaining a simple decision boundary. This range allows exploration from stronger regularization (lower C) to looser constraints (higher C). The RBF kernel allows modeling non-linear relationships, while the linear kernel checks if a simpler linear decision boundary is sufficient.

- **5-Fold Cross Validation:** A 5-fold cross-validation strategy was used as it provides a balanced compromise between computational efficiency and

reliable performance estimation. By splitting the dataset into five parts, the model is both trained and validated on distinct folds, producing a robust estimate of its performance without excessive computational overhead.

- After exploring the parameters, the cross-validation process revealed that the SVM model achieved the best F1 score compared to the other two classifiers.

4. Feature Engineering

- **Scaling:** For models sensitive to feature scales, such as k-NN and SVM, features were standardized using StandardScaler. This step was integrated into scikit-learn Pipelines to ensure that scaling was consistently applied during both cross-validation and prediction.

5. Final Model Selection and Prediction Generation

- **Final Model Chosen:** The tuned Support Vector Machine (SVM) model was selected because it produced the highest cross-validated F1 score during parameter tuning.

- **Retraining:** The final SVM model was retrained on the complete training dataset using the optimized parameters.

- **Prediction:** The retrained model was then used to predict target values for the test dataset. The resulting predictions were saved in a text file (predictions.txt), with the order of predictions corresponding to the order of instances in the test dataset.