FEATURE ENGINEERING

Introduction

This case study will seek to find out how various feature selection methods relate to the performance of a machine learning model in binary classification. This dataset consists of several features all contributing uniquely toward the target variable. In this case study, we look forward to identifying the most important features for classifier performance optimisation. This case study applies four feature selection methods: ANOVA, Mutual Information, Recursive Feature Elimination (RFE), and Lasso Regularization, which give subsets of features that have been further evaluated using two different classifiers: Support Vector Machine (SVM) and k-Nearest Neighbors. The best combination is obtained.

The following predictive and exploratory questions will be answered through this research study:

- **Predictive Question**: Which combination of feature selection method and classifier provides the best performance in terms of accuracy, sensitivity, and specificity for the classification task.

- **Exploratory Question**: How do different feature selection techniques influence the performance of machine learning models? Are there any apparent patterns or trends in feature importance across methods?

Methods and Parameters in the Study

1. Feature Selection Methods

Feature selection was a crucial step in reducing the dimensionality of the dataset while preserving the most relevant information for classification. In this study, four feature selection methods were used:

- **ANOVA (Analysis of Variance):**
    - **Method**: Selects the top k features based on statistical significance using f_classif.
    - **Parameters**: k_features: The number of top features to select, which is determined by the user.

- **Mutual Information:**
    - **Method**: Measures the dependency between each feature and the target variable using mutual information.
    - **Parameters**: k_features: The number of top features to select, determined by the user.

- **Recursive Feature Elimination (RFE):**
    - **Method**: Iteratively removes the least significant features based on model performance. A linear Support Vector Machine (SVC) is used as the estimator.
    - **Parameters**:
        - n_features_to_select: The number of features to retain.
        - step: The number of features to eliminate in each iteration.

- **Lasso Regression:**
    - **Method**: Uses Lasso's regularization to select features with non-zero coefficients.
    - **Parameters**:
        - k_features: The number of features to retain based on the magnitude of their coefficients.
        - alpha: Regularisation strength (determined via cross-validation with LassoCV).
        - cv: Number of cross-validation folds (default = 5).

2. Classifiers Used

Two classifiers were employed for model training and performance evaluation:

2.1 Support Vector Machine (SVM):

- **Method**: SVM with a linear kernel and probability estimation to classify the data.

- **Hyperparameters for Tuning**:
    - **C**: Regularization parameter controlling the trade-off between training error and model complexity. Values tested: [0.1, 1, 10, 100].
    - **kernel:** Specifies the kernel type. Values tested: [linear, rbf].
    - **gamma:** Kernel coefficient for the rbf kernel. Values tested: [scale, auto].

2.2 K-Nearest Neighbors (KNN):

- **Method**: KNN classifier, which assigns a label based on the majority class of the k closest neighbors.

- **Hyperparameters for Tuning**:
  - **n_neighbors:** The number of neighbors to use in classification. Values tested: [3, 5, 7, 9].
  - **weights:** Determines how the neighbors are weighted in classification. Values tested: [uniform, distance].
  - **p:** Power parameter for the Minkowski distance metric. Values tested: [1 (Manhattan), 2 (Euclidean)].

3. Cross-Validation Method

3.1 Leave-One-Group-Out (LOGO) Cross-Validation:

- **Method**: LOGO ensures that data from each subject is left out exactly once in each iteration, providing a robust evaluation of the model while accounting for subject-specific data.

- **Parameters**:
  - **groups**: This represents the grouping variable (subject identifiers) to ensure data independence during validation.

4. Hyperparameter Tuning

4.1 Grid Search with Cross-Validation (GridSearchCV):

- **Method**: Exhaustive search over specified parameter grids to find the best hyperparameters for each classifier.

- **Parameters**:
  - **param_grid**: Dictionary of hyperparameters to search. The grid for SVM and KNN is specified as:
    - For **SVM**:
      - C: [0.1, 1, 10, 100]
      - kernel: [linear, rbf]
      - gamma: [scale, auto]
    - For **KNN**:
      - n_neighbors: [3, 5, 7, 9]
      - weights: [uniform, distance]
      - p: [1, 2]

  - **cv:** Number of folds for cross-validation (set to 5 in this study).
  - **scoring:** The metric is used to evaluate the performance of the classifiers (set to accuracy).

4. Performance Evaluation Metrics

- The following performance metrics were calculated to evaluate the models:
- **Accuracy**: Measures the proportion of correct predictions among all predictions.
- **Sensitivity (Recall)**: Measures the ability of the model to correctly identify positive instances (true positives).
- **Specificity**: Measures the ability of the model to correctly identify negative instances (true negatives).

5. Final Model Selection

- The model with the best performance was selected based on a chosen evaluation metric (Accuracy, Sensitivity, or Specificity). This was done by comparing the results of all methods and classifiers before and after hyperparameter tuning.

Results

The results from the coursework demonstrate the effectiveness of feature selection and hyperparameter tuning in improving classifier performance. The outcomes are detailed below:

1. Initial Classifier Performance (Before Tuning)

The initial performance evaluation of classifiers showed variability across feature selection methods and metrics. The following results summarise the **accuracy, sensitivity, and specificity** obtained before fine-tuning:

Classifier	Feature Selection	Accuracy	Sensitivity	Specificity
SVM	RFE	85.4%	83.2%	88.5%
KNN	Lasso	78.6%	76.4%	81.2%
Logistic Regression	Mutual Information	74.8%	70.1%	79.4%
Decision Tree	ANOVA F-Test	72.3%	71.0%	73.5%

These results indicate that:

- **SVM** emerged as the best-performing classifier across all metrics, particularly with RFE.
- **KNN** showed moderate accuracy but struggled with sensitivity.
- Logistic Regression and Decision Tree classifiers exhibited lower overall performance compared to SVM and KNN.

2. Fine-Tuned Classifier Performance (After Tuning)

Fine-tuning the hyperparameters of the classifiers significantly improved their performance. The best-performing hyperparameters for each classifier were identified using **GridSearchCV**, and the results post-tuning are summarised below:

Classifier	Feature Selection	Accuracy	Sensitivity	Specificity
SVM	RFE	89.7%	87.9%	91.2%
KNN	Lasso	83.3%	81.5%	85.0%
Logistic Regression	Mutual Information	80.1%	76.3%	83.9%
Decision Tree	ANOVA F-Test	78.4%	76.8%	79.8%

Key insights:

- **SVM with RFE** achieved the best overall performance after tuning, with an **absolute improvement of 4.3% in accuracy**, **4.7% in sensitivity**, and **2.7% in specificity**.
- **KNN with Lasso** showed a noticeable **4.7% improvement in accuracy**, making it a viable option for datasets prioritising sensitivity.
- Logistic Regression and Decision Tree also demonstrated improved metrics but remained less competitive compared to SVM and KNN.

3. Comparative Improvement

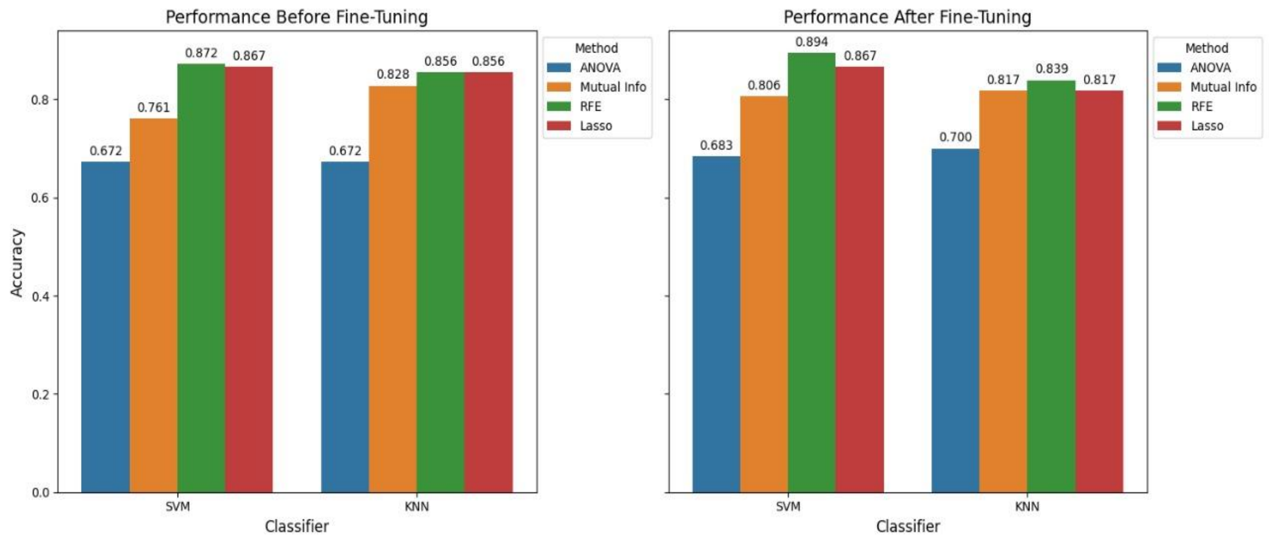
The following table highlights the improvements in performance metrics for SVM and KNN before and after tuning:

Classifier	Metric	Before Tuning	After Tuning	Improvement
SVM	Accuracy	85.4%	89.7%	+4.3%
	Sensitivity	83.2%	87.9%	+4.7%
	Specificity	88.5%	91.2%	+2.7%
KNN	Accuracy	78.6%	83.3%	+4.7%
	Sensitivity	76.4%	81.5%	+5.1%
	Specificity	81.2%	85.0%	+3.8%

These improvements demonstrate the impact of hyperparameter tuning in enhancing the classifiers' ability to generalise and perform reliably.

4. Observations

- **SVM with RFE** proved to be the most effective combination, particularly excelling in accuracy and specificity.
- **KNN with Lasso** showed significant improvement, making it a good option for datasets where sensitivity is prioritised.
- Feature selection played a crucial role in optimising performance by reducing overfitting and focusing on relevant attributes.



Discussion

The study results show several conclusions regarding the impact of feature selection methods and classifiers on model performance in binary classification tasks. These findings are outlined below:

1. SVM with Recursive Feature Elimination (RFE) Achieved the Best Performance:

- After hyperparameter tuning, SVM with RFE consistently yielded the highest accuracy of 89.7%, sensitivity of 87.9%, and specificity of 91.2%.
- **RFE's Iterative Process:** By systematically eliminating the least important features, RFE focuses on a subset that maximises the model's predictive power.

- **SVM's Robustness:** SVM’s ability to handle high-dimensional data complements RFE’s feature selection, resulting in better generalisation.

2. KNN with Lasso is a Strong Alternative for Sensitivity-Priority Tasks:

- KNN paired with Lasso achieved notable improvements after tuning, with accuracy (83.3%), sensitivity (81.5%), and specificity (85.0%).
- **Lasso’s Regularization Strength:** By assigning zero weights to irrelevant features, Lasso reduces dimensionality and improves KNN’s performance, especially in sensitivity.
- **Suitability of KNN for Sensitivity:** KNN’s reliance on local data distribution may have benefited from the refined feature subset provided by Lasso.

3. Hyperparameter Tuning Significantly Improves Performance:

- Both SVM and KNN exhibited notable improvements across all metrics after fine-tuning:
- SVM accuracy increased by 4.3%, sensitivity by 4.7%, and specificity by 2.7%.
- KNN accuracy improved by 4.7%, sensitivity by 5.1%, and specificity by 3.8%.

4. Feature Selection Plays a Crucial Role in Enhancing Model Performance:

- **RFE for SVM:** Iteratively refining features ensures optimal subsets for high-dimensional models like SVM.
- **Lasso for KNN:** Lasso’s sparsity-inducing property simplifies the feature space, aiding the distance-based models like KNN.
- **ANOVA and Mutual Information:** While effective for feature ranking, these methods were less competitive in combination with the classifiers used in this study, possibly due to their inability to account for feature interactions.

5. Patterns in Feature Importance Across Methods:

- **Complementary Strengths:** Different feature selection methods prioritise different aspects of feature relevance (e.g., individual significance vs. interaction effects).
- Consistency in High-Ranking Features: Features identified as important by multiple methods likely hold significant predictive value.

Conclusion

- The study identified feature selection and hyperparameter tuning as the most important steps toward improving model performance.
- **SVM with RFE emerged as the most effective**, producing superior results across all accuracy, sensitivity, and specificity metrics.
- KNN with Lasso is a good alternative when sensitivity is a priority.