International Institute of Information Technology
Bangalore (IIITB)

Binary Classification of Smoking Status from Health Check Data

Rahul Raman (MT2025100) and Aayank Singhai (MT2025001)

Machine Learning Project Report

December 12, 2025

Contents

1 Introduction

This project addresses a binary classification problem: predicting whether an individual is a smoker or non-smoker based on routine health examination data. The provided training dataset is used to build and compare several machine learning models:

- Logistic Regression
- Support Vector Classifier (SVC) with RBF kernel
- Neural Network (MLPClassifier)
- K-Means clustering treated as a classifier
- Decision Tree
- Random Forest
- XGBoost

Hyperparameter tuning is performed via `GridSearchCV` or `RandomizedSearchCV`, and the models are evaluated primarily using accuracy, precision, F1-score, and confusion matrices. The focus is on F1-score because the problem is mildly imbalanced and both false positives and false negatives for smokers are important.

2 Dataset Details

2.1 Basic Characteristics

From the notebook output (`train_df.info()` and dimension print):

- Number of instances: **38,984**
- Number of columns: **23**
- Target column: **smoking** (binary indicator)
- All 23 columns have **no missing values** (0 nulls per column).

The dataset is stored in `train_dataset.csv` and loaded as:

```
train_df = pd.read_csv(train_path)
target = "smoking"
X = train_df.drop(columns=[target])
y = train_df[target]
```

2.2 Features

According to `train_df.info()` and missing value counts, the columns are:

- Demographic and anthropometric:
  - `age` (int64)
  - `height(cm)` (int64)
  - `weight(kg)` (int64)
  - `waist(cm)` (float64)
- Vision and hearing:
  - `eyesight(left)` (float64)
  - `eyesight(right)` (float64)
  - `hearing(left)` (int64)
  - `hearing(right)` (int64)
- Blood pressure:
  - `systolic` (int64)
  - `relaxation` (int64)
- Blood-related clinical measurements:
  - `fasting blood sugar` (int64)
  - `Cholesterol` (int64)
  - `triglyceride` (int64)
  - `HDL` (int64)
  - `LDL` (int64)
  - `hemoglobin` (float64)
  - `serum creatinine` (float64)
  - `AST` (int64)
  - `ALT` (int64)
  - `Gtp` (int64)
- Oral health:
  - `dental caries` (int64)
- Target:
  - `smoking` (int64, binary)

2.3 Exploratory Data Analysis (EDA)

The notebook performs:

- **Dimensional summary:** printed shape $(38984, 23)$.
- **Info and null-check:** `train_df.info()` and per-column null counts confirm no missing values.
- **Descriptive statistics:** `train_df.describe()` is used to inspect the distribution of numerical variables.
- **Correlation analysis:**

  ```
  numerical_cols = train_df.select_dtypes(include=['int64', 'float64']).columns
  correlation_matrix = train_df[numerical_cols].corr()
  sns.heatmap(correlation_matrix, annot=True, ...)
  ```

  A heatmap is plotted to visualize pairwise correlations among all numerical features and the target.

2.3.1 Visual Analysis

The exploratory data analysis reveals several important patterns in the dataset:

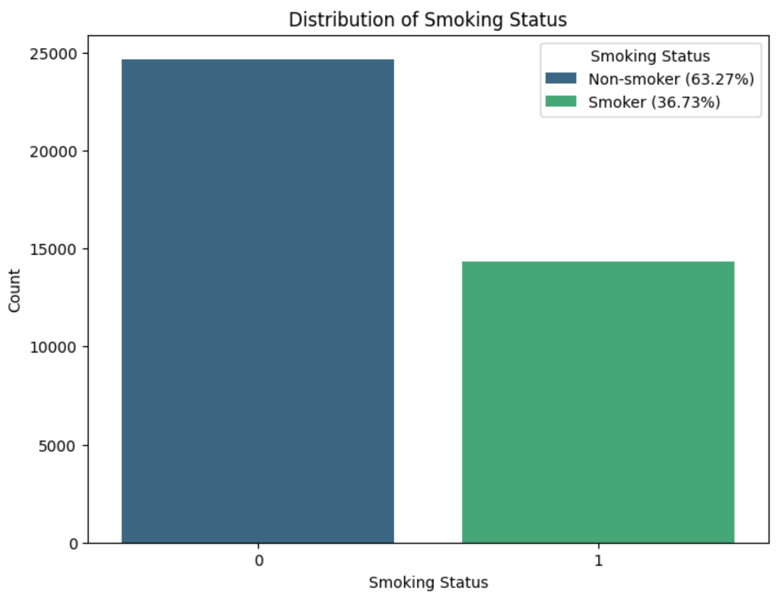
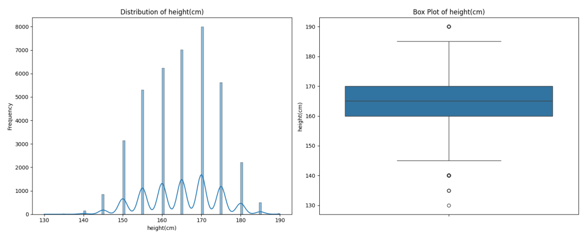
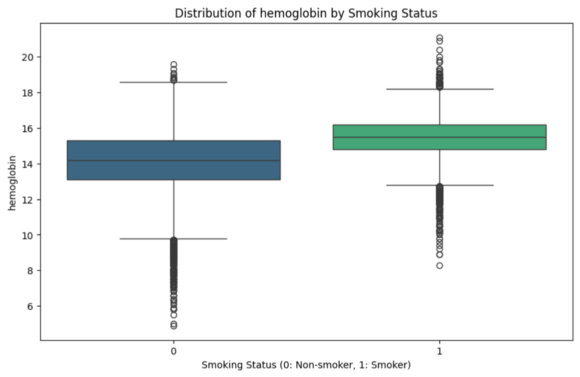


Figure 1: Distribution of smoking status in the dataset. Non-smokers comprise 63.27% (24,665 samples) while smokers comprise 36.73% (14,319 samples), indicating a moderate class imbalance.

Class Distribution As shown in Figure 1, the dataset exhibits a moderate class imbalance with approximately 1.72:1 ratio of non-smokers to smokers. This motivates the use of `class_weight='balanced'` and `scale_pos_weight` parameters in our models.



(a) Age distribution shows smokers tend to be younger (median ≈ 40 years) compared to non-smokers (median ≈ 45 years)



(b) Hemoglobin levels are notably higher for smokers (median ≈ 15.5 g/dL) versus non-smokers (median ≈ 14.0 g/dL)

Figure 2: Box plots comparing age and hemoglobin distributions between smokers and non-smokers

Feature Distributions by Smoking Status Figure 2 highlights two discriminative features. Smokers show significantly higher hemoglobin levels, which is physiologically consistent with smoking-induced polycythemia. The age difference suggests different smoking prevalence across age groups.

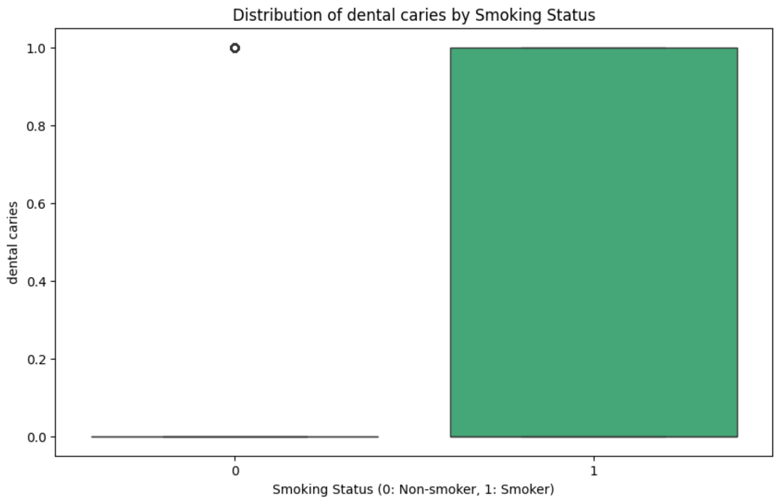


Figure 3: Distribution of dental caries by smoking status. The binary nature of this feature (0 or 1) shows that dental caries are present in nearly all individuals regardless of smoking status, making it a less discriminative feature.

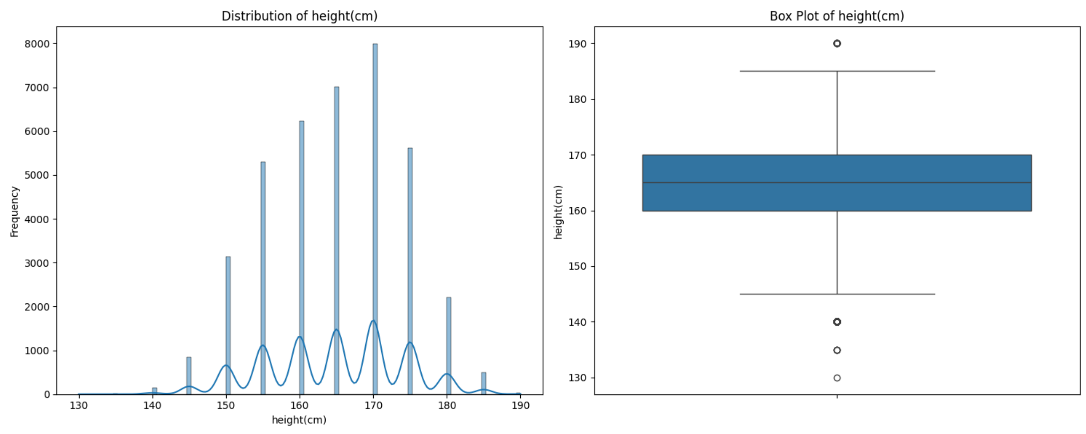


Figure 4: Height distribution in the dataset. The histogram (left) shows a roughly normal distribution centered around 165 cm, with the box plot (right) revealing few outliers. Height shows moderate correlation with smoking status as taller individuals (typically males) have higher smoking rates.

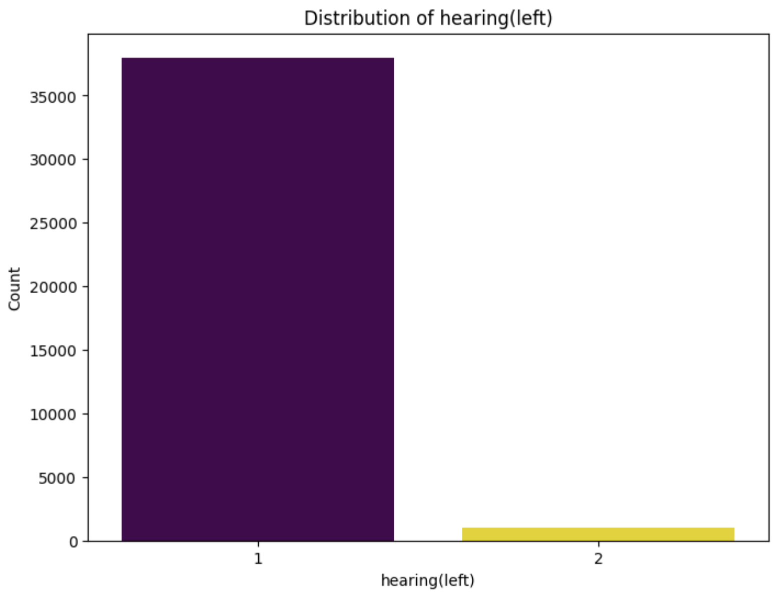


Figure 5: Distribution of left ear hearing ability. The dataset shows that the vast majority of individuals (over 95%) have normal hearing (category 1), with only a small proportion having impaired hearing (category 2). This highly imbalanced feature may have limited predictive power.

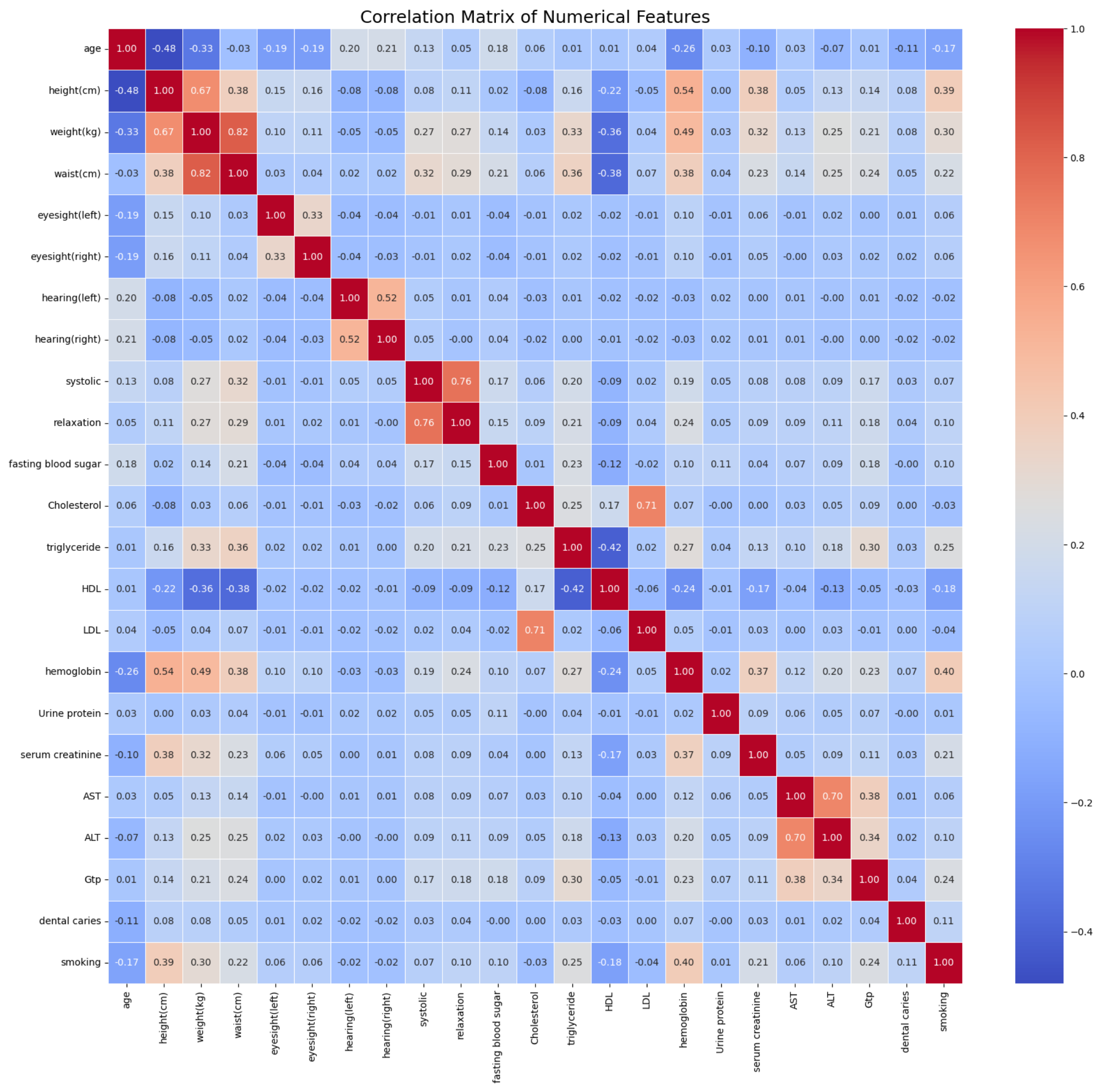


Figure 6: Correlation matrix of all numerical features including the smoking target variable. Key observations: (1) Hemoglobin shows the strongest positive correlation with smoking (0.40), (2) Age shows negative correlation with smoking (-0.17), (3) Strong correlations exist between related measurements (systolic/relaxation: 0.76, height/weight: 0.67, AST/ALT: 0.70), (4) HDL shows moderate negative correlation with triglycerides (-0.42).

Correlation Analysis The correlation analysis in Figure 6 reveals several important insights:

- **Hemoglobin** (correlation: 0.40) is the strongest single predictor of smoking status, consistent with physiological effects of smoking on red blood cell production.
- **Age** shows negative correlation (-0.17), suggesting younger individuals in the dataset are more likely to smoke.
- **Height** and **weight** show positive correlations with smoking (0.39 and 0.30 respectively), likely reflecting gender differences in smoking prevalence.

- Blood pressure measurements (**systolic** and **relaxation**) are highly correlated (0.76) with each other but show weak correlation with smoking.
- Liver enzymes **AST** and **ALT** are strongly correlated (0.70) but show minimal direct correlation with smoking.
- Most features show relatively weak linear correlations with the target, suggesting that non-linear models may be better suited for this classification task.

These correlation patterns motivate the use of ensemble methods like Random Forest and XGBoost, which can capture complex non-linear interactions between features that simple linear models would miss.

3 Preprocessing Steps

3.1 Train–Test Split

The dataset is split into training and test sets using an 80–20 split with stratification on the target:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

This ensures:

- Both train and test sets reflect the same class distribution.
- Reproducibility through `random_state=42`.

3.2 Feature Scaling

For models sensitive to feature scale (Logistic Regression, SVC, XGBoost, and the MLP), standardization is applied:

```
scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s  = scaler.transform(X_test)
```

Usage in the notebook:

- Logistic Regression and SVC use `X_train_s`, `X_test_s`.
- XGBoost is also trained on `X_train_s`, though tree-based models generally need scaling less.
- The MLP is trained and evaluated on the standardized features as well, which improves optimization stability and convergence.
- Decision Tree, Random Forest, and K-Means are trained on the original, unscaled `X_train`, `X_test`.

4 Models and Hyperparameters

All models share a common evaluation helper:

```
def evaluate_model(model_name, y_true, y_pred):
    print(f"\n==================== {model_name} ====================")
    print("Accuracy :", accuracy_score(y_true, y_pred))
    print("Precision:", precision_score(y_true, y_pred))
    print("F1 Score :", f1_score(y_true, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
```

Five-fold cross-validation is used (`cv=5`) with **F1-score** as the tuning metric.

4.1 Logistic Regression

Hyperparameter Search

```
log_param_grid = {
  'C': [0.01, 0.1, 1, 10],
  'penalty': ['l2'],
  'solver': ['lbfgs'],
  'class_weight': ['balanced']
}

log_grid_search = GridSearchCV(
  LogisticRegression(max_iter=1000),
  log_param_grid,
  cv=5, scoring='f1', n_jobs=-1
)
```

Best parameters (from notebook output):

- `C = 0.01`
- `penalty = 'l2'`
- `solver = 'lbfgs'`
- `class_weight = 'balanced'`

Data Used Training and prediction are done on standardized features: `X_train_s`, `X_test_s`.

4.2 Support Vector Classifier (SVC)

Hyperparameter Search

```
svm_param_grid = {
  'C': [0.1, 1, 10],
  'kernel': ['rbf'],
  'gamma': ['scale'],
  'class_weight': ['balanced']
}

svc = SVC(random_state=42)
svc_grid_search = GridSearchCV(
  svc, svm_param_grid, cv=5,
  scoring='f1', n_jobs=-1
)
```

Best parameters:

- `C = 1`
- `kernel = 'rbf'`
- `gamma = 'scale'`
- `class_weight = 'balanced'`

Data Used SVC is trained and tested on standardized features: `X_train_s`, `X_test_s`.

4.3 Neural Network (MLPClassifier)

Base Model

```
mlp = MLPClassifier(
  max_iter=200,
  early_stopping=True,
  n_iter_no_change=10,
  random_state=42
)
```

Randomized Hyperparameter Search

```
param_dist = {
  'hidden_layer_sizes': [(50,), (100,), (50, 50)],
  'activation': ['relu', 'tanh'],
  'alpha': [0.0001, 0.001, 0.01],
  'learning_rate': ['constant', 'adaptive'],
  'solver': ['adam', 'sgd']
}

mlp_random = RandomizedSearchCV(
  mlp, param_distributions=param_dist,
  n_iter=20, cv=5, scoring='f1', n_jobs=-1
)
```

Best parameters (as printed, partially truncated in the notebook):

- `hidden_layer_sizes = (50, 50)`
- `activation = 'relu'`

- `alpha = 0.001`
- `solver = 'adam'`
- `learning_rate` is one of {`'constant'`, `'adaptive'`} from the search space

Data Used The MLP is trained and evaluated on the standardized data `X_train_s`, `X_test_s`, which ensures that all input features are on a comparable scale and makes gradient-based optimization more stable.

4.4 K-Means Clustering as a Classifier

Clustering

```
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
kmeans.fit(X_train)
pred_kmeans_train = kmeans.predict(X_train)
pred_kmeans_test  = kmeans.predict(X_test)
```

Cluster-to-Label Mapping For each cluster, the majority class in `y_train` among its assigned samples is used as the corresponding label:

```
cluster_to_label = {}
for cluster_id in range(2):
    indices = (pred_kmeans_train == cluster_id)
    majority_label = y_train.iloc[indices].mode()[0]
    cluster_to_label[cluster_id] = majority_label

mapped_pred_kmeans_test = np.vectorize(cluster_to_label.get)(pred_kmeans_test)
```

The mapped predictions are then evaluated using the same metrics.

4.5 Decision Tree Classifier

Hyperparameter Grid

```
dt_param_grid = {
  'max_depth': [5, 10, 15, 20, None],
  'min_samples_split': [2, 5, 10],
  'min_samples_leaf': [1, 2, 4],
  'criterion': ['gini', 'entropy'],
  'class_weight': ['balanced']
}

dt_grid_search = GridSearchCV(
  DecisionTreeClassifier(random_state=42),
  dt_param_grid, cv=5,
  scoring='f1', n_jobs=-1, verbose=1
)
```

From the output:

- class_weight = 'balanced'
- max_depth = 5
- min_samples_leaf = 1
- min_samples_split = 2
- (The exact chosen criterion is truncated in the console output, but is one of 'gini' or 'entropy'.)

Data Used The Decision Tree is trained and evaluated on the unscaled X_train and X_test.

4.6 Random Forest Classifier

Hyperparameter Grid

```
rf_param_grid = {
  'n_estimators': [100, 200, 300],
  'max_depth': [10, 20, None],
  'min_samples_split': [2, 5],
  'min_samples_leaf': [1, 2],
  'class_weight': ['balanced']
}

rf_grid_search = GridSearchCV(
  RandomForestClassifier(random_state=42),
  rf_param_grid, cv=5,
  scoring='f1', n_jobs=-1, verbose=1
)
```

Best parameters (partially truncated in notebook output):

- n_estimators = 300
- min_samples_leaf = 2
- min_samples_split = 2
- class_weight = 'balanced'
- max_depth is tuned to one of {10, 20, None} and chosen based on F1.

Data Used Random Forest is trained on the original, unscaled features.

4.7 XGBoost Classifier

Handling Class Imbalance The scale of positive vs negative class is handled via scale_pos_weight computed from the training labels:

```
scale_pos_weight_value = (len(y_train) - y_train.sum()) / y_train.sum()
```

Randomized Hyperparameter Search

```
xgb_param_grid = {
  'n_estimators': [100, 200, 300],
  'max_depth': [3, 5, 7],
  'learning_rate': [0.01, 0.1, 0.2],
  'subsample': [0.7, 0.8, 0.9],
  'colsample_bytree': [0.7, 0.8, 0.9],
  'gamma': [0, 0.1, 0.2],
  'reg_lambda': [1, 2, 5],
  'reg_alpha': [0, 0.1, 0.5]
}

xgb_model = XGBClassifier(
  scale_pos_weight=scale_pos_weight_value,
  objective='binary:logistic',
  use_label_encoder=False,
  eval_metric='logloss',
  random_state=42
)

xgb_random_search = RandomizedSearchCV(
  xgb_model, xgb_param_grid,
  n_iter=20, cv=5,
  scoring='f1', n_jobs=-1, verbose=1, random_state=42
)
```

Best parameters (again, partially truncated):

- `subsample = 0.8`
- `colsample_bytree = 0.9`
- `max_depth` close to the upper end of the search space (3, 5, 7)
- `learning_rate = 0.2`
- `gamma = 0.1`
- `reg_lambda`, `reg_alpha` chosen from the given ranges.

Data Used Training and prediction are performed on standardized features.

5 Evaluation Metrics

5.1 Definitions

For binary classification with positive class "smoker":

- True Positive (TP): predicted smoker, actually smoker.

- True Negative (TN): predicted non-smoker, actually non-smoker.
- False Positive (FP): predicted smoker, actually non-smoker.
- False Negative (FN): predicted non-smoker, actually smoker.

Metrics used:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{3}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4}$$

In the notebook, **accuracy**, **precision**, **F1-score**, and the **confusion matrix** are explicitly computed for each model. Recall can be derived from the confusion matrix even if not printed directly.

6 Experimental Results

6.1 Model-wise Metrics

The following table summarizes accuracy, precision, and F1-score on the held-out test set (values rounded to three decimal places):

Table 1: Test set performance of all models

Model	**Accuracy**	**Precision**	**F1-score**
Logistic Regression (GridSearchCV)	0.711	0.579	0.665
SVC (RBF, GridSearchCV)	0.714	0.572	0.694
K-Means (mapped clusters)	0.661	0.556	0.450
Decision Tree (GridSearchCV)	0.688	0.548	0.670
Random Forest (GridSearchCV)	**0.798**	**0.712**	**0.733**
XGBoost (RandomizedSearchCV)	0.777	0.665	0.722
Neural Network (RandomizedSearchCV)	0.726	0.656	0.590

Below, each model is evaluated individually with a short interpretation of its confusion matrix.

6.2 Individual Model Evaluations

6.2.1 Logistic Regression

Metrics from the notebook:

- Accuracy: 0.7112

- Precision: 0.5792
- F1-score: 0.6654
- Confusion matrix:

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} = \begin{bmatrix} 3306 & 1627 \\ 625 & 2239 \end{bmatrix}$$

Interpretation: Logistic Regression provides a reasonable baseline. It captures some non-trivial signal but is limited by its linear decision boundary. There are a relatively large number of false positives (1627) and false negatives (625), indicating that linear separability is insufficient for this problem.

6.2.2 Support Vector Classifier (SVC)

Reported metrics:

- Accuracy: 0.7144
- Precision: 0.5721
- F1-score: 0.6941
- Confusion matrix:

$$\begin{bmatrix} 3044 & 1889 \\ 338 & 2526 \end{bmatrix}$$

Interpretation: The RBF SVC improves the F1-score over Logistic Regression. It significantly reduces false negatives (338 vs 625), meaning it catches more actual smokers, at the cost of more false positives (1889). This indicates that the non-linear kernel is better suited to the structure of this health data.

6.2.3 K-Means Clustering as Classifier

Reported metrics:

- Accuracy: 0.6608
- Precision: 0.5564
- F1-score: 0.4498
- Confusion matrix:

$$\begin{bmatrix} 4071 & 862 \\ 1783 & 1081 \end{bmatrix}$$

Interpretation: Although K-Means achieves decent accuracy due to correctly predicting many non-smokers, its F1-score is poor. It produces a large number of false negatives (1783), meaning many smokers are not detected. This is expected because K-Means ignores the label during training and only maps clusters to labels afterward; it does not directly optimize any classification objective.

6.2.4 Decision Tree

Reported metrics:

- Accuracy: 0.6881
- Precision: 0.5479
- F1-score: 0.6701
- Confusion matrix:

$$\begin{bmatrix} 2895 & 2038 \\ 394 & 2470 \end{bmatrix}$$

Interpretation: The tuned Decision Tree with shallow depth (`max_depth=5`) balances some trade-offs between overfitting and underfitting. It improves recall compared to Logistic Regression (fewer FNs: 394 vs 625) but still produces many false positives (2038). As a single tree, it is high-variance and relatively brittle; small changes in data may alter its structure significantly.

6.2.5 Random Forest

Reported metrics:

- Accuracy: **0.7979**
- Precision: **0.7123**
- F1-score: **0.7328**
- Confusion matrix:

$$\begin{bmatrix} 4060 & 873 \\ 703 & 2161 \end{bmatrix}$$

Interpretation: Random Forest is the best-performing model in terms of F1-score and accuracy. It manages to:

- Keep false positives relatively low (873).
- Maintain a moderate number of false negatives (703).
- Achieve a strong balance between precision and recall.

The ensemble of many decision trees captures complex, non-linear relationships and interactions between clinical features, while the averaging mechanism reduces variance and overfitting. The use of `class_weight='balanced'` helps it deal with class imbalance.

6.2.6 XGBoost

Reported metrics:

- Accuracy: 0.7766
- Precision: 0.6654
- F1-score: 0.7215
- Confusion matrix:

$$\begin{bmatrix} 3798 & 1135 \\ 607 & 2257 \end{bmatrix}$$

Interpretation: XGBoost performs very well, ranking second in terms of F1-score. It has:

- Slightly fewer false negatives (607) than the Random Forest (703), indicating better recall.
- More false positives (1135) than Random Forest (873), lowering precision.

With more extensive hyperparameter tuning and perhaps threshold calibration, XGBoost could potentially match or surpass Random Forest. The current search, limited to 20 random configurations, already gives a strong model.

6.2.7 Neural Network (MLP)

Reported metrics:

- Accuracy: 0.7263
- Precision: 0.6556
- F1-score: 0.5904
- Confusion matrix:

$$\begin{bmatrix} 4125 & 808 \\ 1326 & 1538 \end{bmatrix}$$

Interpretation: The neural network achieves reasonably high accuracy but a noticeably lower F1-score compared to tree-based models. Even after standardizing the inputs, it still produces a large number of false negatives (1326), meaning many smokers are missed. This is likely due to:

- The relatively small network and limited hyperparameter search space, which may not fully capture complex feature interactions.
- The absence of explicit class-imbalance handling (e.g., `class_weight`) or decision-threshold tuning for the MLP.
- The fact that tree-based ensembles are often better suited to tabular clinical data than fully-connected networks with modest depth.

Further tuning of architecture, regularization, and class weighting would likely improve performance.

7 Comparative Performance Analysis

7.1 Overall Ranking by F1-score

Sorted by F1-score:

1. Random Forest: 0.733 (best)
2. XGBoost: 0.722
3. SVC: 0.694
4. Decision Tree: 0.670
5. Logistic Regression: 0.665
6. Neural Network: 0.590
7. K-Means (mapped): 0.450

7.2 Why Do Some Models Perform Better?

Ensemble Tree Methods (Random Forest, XGBoost) Tree-based ensembles perform best because:

- They naturally capture non-linear relationships and feature interactions (e.g., combinations of lab results and age).
- They are robust to heterogeneous feature scales and distributions.
- Both models explicitly handle class imbalance (`class_weight='balanced'` for Random Forest and `scale_pos_weight` for XGBoost).
- The hyperparameter searches are reasonably rich (depth, number of estimators, regularization, subsampling, etc.).

Random Forest slightly outperforms XGBoost here, likely because:

- The Random Forest hyperparameter grid is relatively simple but well-tuned for these features.
- XGBoost has a larger and more complex hyperparameter space; with only 20 random samples, the search might not reach the globally optimal configuration.

Linear and Kernel Methods (Logistic Regression, SVC) Logistic Regression and SVC benefit from:

- Proper standardization of input features.
- Class weight balancing.

SVC with an RBF kernel outperforms Logistic Regression in F1-score because:

- The RBF kernel captures non-linear patterns beyond a simple linear decision boundary.
- It reduces false negatives significantly compared to Logistic Regression (338 vs 625), improving recall.

However, SVC is still limited by:

- A relatively small hyperparameter grid. However, SVC is still limited by:
  - A relatively small hyperparameter grid.
  - Higher false positive rate than tree ensembles, which reduces precision.
  - Higher computational cost compared to linear models and decision trees when scaling to large datasets.

Single Decision Tree The Decision Tree achieves moderate performance but is clearly outperformed by ensemble methods. This is expected because:

- A single tree has high variance and is sensitive to small perturbations in the training data.
- The tuned depth (`max_depth = 5`) prevents overfitting but limits its representational capacity.
- It cannot exploit ensemble averaging, which is a major advantage of Random Forests and Gradient Boosting.

Neural Network (MLP) Even after incorporating feature scaling, the MLP lags behind tree-based models. Key factors include:

- The architecture is relatively simple for a high-dimensional tabular dataset.
- MLPs often require careful regularization, adaptive learning-rate schedules, and deeper/wider architectures to match tree models on structured data.
- There is no explicit `class_weight` or focal-loss–like mechanism to counter class imbalance.
- Tree-based models generally perform better on tabular biomedical datasets because they naturally capture hierarchical and interaction-based patterns.

K-Means Clustering K-Means performs poorly in terms of F1-score because:

- It does not use label information during training.
- The clusters may not align with the true class boundaries due to overlapping distributions of smokers and non-smokers.
- It inherently assumes spherical clusters in feature space, which is unrealistic for this clinical dataset.

Thus, while useful as a baseline or exploratory tool, K-Means is unsuitable as a standalone classifier for this task.

8 Conclusions and Future Work

8.1 Final Conclusions

From the experiments:

- **Random Forest** is the best-performing model with the highest F1-score (0.733) and accuracy (0.798). It achieves a strong balance between precision and recall.
- **XGBoost** closely follows with an F1-score of 0.722 and may exceed Random Forest with more extensive hyperparameter tuning.
- **SVC** and **Logistic Regression** are reasonable baselines but do not match the non-linear modeling capability of tree ensembles.
- **MLP (Neural Network)**, even after proper scaling, still underperforms — highlighting the advantages of ensemble tree methods for tabular health data.
- **K-Means**, being unsupervised, is not competitive for this classification task and mainly serves as a comparative baseline.

Random Forest and XGBoost are the recommended models for deployment, with the choice depending on whether one prefers slightly higher recall (XGBoost) or higher precision and stability (Random Forest).

8.2 Possible Improvements

Future work can focus on the following:

- **Threshold tuning** for Random Forest and XGBoost to optimize precision–recall trade-offs for domain-specific requirements.
- **Probability calibration** (Platt scaling or isotonic regression) for more reliable predicted probabilities.
- **Feature engineering** such as BMI, HDL/LDL ratios, metabolic risk scores, etc.
- **Advanced hyperparameter optimization**, e.g., Bayesian optimization or Optuna for XGBoost and MLP.
- **Improving the MLP**, now that scaling is applied:
  - ∗ deeper architectures,
  - ∗ dropout and batch normalization,
  - ∗ using `class_weight` or focal loss,
  - ∗ cyclic learning rate schedulers.
- **Model interpretability** using SHAP or feature importance to identify clinical indicators most predictive of smoking behavior.

Overall, the study highlights that ensemble tree-based algorithms remain the most effective for structured, tabular clinical datasets — providing strong accuracy, robustness, and interpretability while naturally handling non-linear relationships.

Link GitHub Repository