SMOKE DETECTION USING NEURAL NETWORK,SVM,LOGISTIC REGRESSION

Project Report

Author: Rohit Bansal, Prasanna Kumar M
Roll Number: MT2025105, MT2025090
Institution: International Institute of Information Technology, Bangalore

Submission Date: 12-12-2025

Introduction

This report details the implementation and evaluation of various machine learning models—specifically Logistic Regression, Support Vector Machines (SVM), and Neural Networks—to predict smoking status. The primary objective of this project, titled "SMOKE DETECTION USING NEURAL NETWORKS, SVM, LOGISTIC REGRESSION", is to develop a robust classifier using a comprehensive health-related dataset.

The scope of Project 1 encompasses the essential stages of a typical machine learning workflow: Exploratory Data Analysis (EDA), Feature Enhancement (Engineering), Feature Normalization (Z-Score standardization), and Model Building and evaluation using Logistic Regression and SVM. The overall goal is to compare the performance of these classical machine learning techniques in identifying the target variable, `smoking`. The notebook also transitions to Project 2B, which focuses on a large-scale classification task using Neural Networks on the `covtype.csv` dataset[cite: 380, 382].

1.Project 1 → SMOKE DETECTION

Exploratory Data Analysis (EDA)

The data exploration began by examining the dataset structure. The training dataset used, `train_dataset.csv`[cite: 15], contains multiple health-related features.

Missing Values

A check for missing values using `data.isna().sum()` confirmed that there were no null values across any of the 29 columns[cite: 23, 24].

Data Types

The dataset is composed of numerical data types, including `int64` and `float64`, with one feature, `blood_pressure_category`, identified as a `category` type before initial data cleaning [cite: 25, 27-84].

DATA VISUALISATION

FEATURE ENHANCEMENT and NORMALIZATION

Feature Enhancement (Feature Engineering)

New features were engineered to capture derived relationships between existing variables [cite: 275-280]:

- BMI was calculated from weight(kg) and height(cm)[cite: 276].
- waist_height_ratio was calculated as waist(cm) divided by {height(cm)[cite: 276].
- avg_eyesight and avg_hearing were calculated as the average of the left and right measurements[cite: 278, 279].
- cholesterol_ratio was calculated as HDL divided by LDL[cite: 280].

The target variable, smoking, was then separated from the features (`train_data`), and the categorical column blood_pressure_category was dropped[cite: 217, 218, 222]. The data was split into training and testing sets with a test size of 20% and random_state = 0[cite: 219].

Feature Normalization (Z-Score)

Non-binary features were standardized using the Z-Score method, $\left(Z = \frac{X-\mu}{\sigma}\right)$, to center the data around zero and scale it by the standard deviation [cite: 281-284, 325-327].

Listing 1: Z-Score Normalization Implementation

```
# Identify binary columns
binary = []
for i in data.columns:
    if data[i].nunique() <= 2:
        binary.append(i)

# Normalization for non-binary features in X_train
for i in X_train.columns:
   if i not in binary:
       X_train[i] = (X_train[i] - X_train[i].mean()) / X_train[i].std()

# Normalization for non-binary features in X_test
for i in X_test.columns:
    if i not in binary:
        X_test[i] = (X_test[i] - X_test[i].mean()) / X_test[i].std()
```

2.LOGISTIC REGRESSION

A Logistic Regression model was trained to establish a baseline performance.

Listing 2: Logistic Regression Model Training

```
from sklearn.linear_model import LogisticRegression
# Model Definition and Training
model = LogisticRegression(random_state=42,
fit_intercept=True)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Accuracy_Score

The Logistic Regression model achieved the following accuracy:

```
accuracy_score(y_test, y_pred)
% Output: 0.7293830960625882
```

Feature_Importance

The feature importance was calculated using the absolute values of the fitted model's coefficients (abs(model.coef_[0]))[cite: 295]. Features like waist(cm), waist_height_ratio, and weight(kg) exhibited the highest absolute coefficient values[cite: 308, 309, 332].

Listing 3: Feature Importance Visualization

```
imp = abs(model.coef_[0])
plt.figure(figsize=(15,6))
plt.xticks(rotation=90)
plt.bar(X_train.columns, imp)
```

SVM

A Support Vector Classification (SVC) model was initialized and trained using its default settings, which utilize a Radial Basis Function (RBF) kernel[cite: 353, 354, 370].

Listing 4: SVM Model Training

```
from sklearn.svm import SVC
```

```
# Model Definition and Training
model = SVC()
model.fit(X_train, y_train)
y_predict_svm = model.predict(X_test)
```

Accuracy_Score

The SVC model demonstrated a slight improvement in accuracy over Logistic Regression:

```
accuracy_score(y_test, y_predict_svm)
% Output: 0.7496473002436834
```

The classification report provides a comprehensive view of the SVM performance[cite: 377, 378]:

Table 1: SVM Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.81	0.80	0.81	4975
1	0.66	0.66	0.66	2822
Accuracy			**0.75**	**7797**
Weighted Avg	0.75	0.75	0.75	7797

Featue Importance

Feature importance was not calculated for the SVC model in the provided analysis[cite: 379].

3.PROJECT 2B: NEURAL NETWORKS

The analysis briefly shifts to Project 2B, focused on a different classification task using Neural Networks[cite: 380]. This section initializes the process for the `covtype.csv` dataset[cite: 382].

- **Dataset:** `covtype.csv`[cite: 382].
- **Data Shape:** The dataset contains 581,012 rows and 55 columns[cite: 384].
- **Data Types:** All features are of type `int64`[cite: 386].

0.1 Data Preparation and Feature Engineering

Feature Engineering

Five new composite features were generated from the raw health data to provide the model with better proxies for metabolic health and overall physiological risk.

```
data['BMI'] = data['weight(kg)'] / ((data['height(cm)'] / 100) ** 2)
data['waist_height_ratio'] = data['waist(cm)'] / data['height(cm)']
data['avg_eyesight'] = (data['eyesight(left)'] + data['eyesight(right)']) /
data['avg_hearing'] = (data['hearing(left)'] + data['hearing(right)']) / 2
data['cholesterol_ratio'] = data['HDL'] / data['LDL']
```

Data Splitting and Scaling

The dataset was split into training and testing sets (80/20) with stratification on the target variable (y) to preserve the class balance. Feature scaling using StandardScaler was applied, which is critical for stabilizing the training of Neural Networks.

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Initialize StandardScaler
```

```
scaler = StandardScaler()

# Fit scaler on training data and transform both training and test data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

input_dim = X_train_scaled.shape[1]
# Input dimension (number of features): 22
```

Neural Network Model Architecture

The model is a Sequential Deep Neural Network designed for binary classification. It consists of three hidden layers with decreasing numbers of neurons, using ReLU activation and Dropout layers to prevent overfitting.

```
def build_nn_model(input_dim):
    model = Sequential([
        # First Hidden Layer
        Dense(128, activation='relu', input_shape=(input_dim,)),
        Dropout(0.3),

        # Second Hidden Layer
        Dense(64, activation='relu'),
        Dropout(0.3),

        # Third Hidden Layer
        Dense(32, activation='relu'),
        Dropout(0.2),

        # Output Layer: Sigmoid for binary probability output
        Dense(1, activation='sigmoid')
    ])

    model.compile(
        optimizer=Adam(learning_rate=0.001),
        loss='binary_crossentropy',
        metrics=['accuracy', 'AUC']
```

```
    )
    return model
```

0.1.1 Model Summary

The final model structure has a total of 13,313 trainable parameters.

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 128)	2,944
dropout_3 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 64)	8,256
dropout_4 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 32)	2,080
dropout_5 (Dropout)	(None, 32)	0
dense_7 (Dense)	(None, 1)	33

```
Total params: 13,313 (52.00 KB)
Trainable params: 13,313 (52.00 KB)
Non-trainable params: 0 (0.00 B)
```

4.Final Model Performance

The model was evaluated on the held-out test set ($\sim 20\%$ of the data).

Table 2: Model Evaluation on Test Set

Metric	**Value**
Test Loss	0.4738
Test Accuracy	0.7557
Test AUC	0.8366

Accuracy Score and ROC AUC Score

The Neural Network achieved a Test Accuracy of **0.7557** and a ROC AUC Score of **0.8366**, demonstrating strong predictive performance for the binary classification task.

Classification Report

The detailed classification report provides insights into the model's performance on both classes (Non-Smoker=0, Smoker=1).

```
--- Classification Report ---
              precision    recall  f1-score   support

           0       0.81      0.80      0.81      4933
           1       0.66      0.68      0.67      2864

    accuracy                           0.76      7797
   macro avg       0.74      0.74      0.74      7797
weighted avg       0.76      0.76      0.76      7797
```

Confusion Matrix

The confusion matrix shows the raw counts of correct and incorrect predictions on the test set.

```
--- Confusion Matrix ---
[[3956  977]
 [ 928 1936]]
```

Interpretation:

- True Negatives (Correctly predicted Non-Smoker): 3,956
- False Positives (Incorrectly predicted Smoker): 977
- False Negatives (Incorrectly predicted Non-Smoker): 928
- True Positives (Correctly predicted Smoker): 1,936

Conclusion

The Deep Neural Network, incorporating scaled features and enhanced with composite features, successfully modeled the non-linear relationship between health metrics and smoker status. The final ROC AUC of **0.8366** indicates a high degree of separability between the two classes.