Vitamin Deficiency Disease Prediction - Machine Learning Assignment

Deliverables:

1. GitHub Repository Link: https://github.com/verathagnus/disease-detector-ml/

a. Problem Statement

Vitamin deficiencies are a significant global health concern affecting millions of people worldwide. Early detection and diagnosis of vitamin deficiency-related diseases can help prevent serious health complications and improve patient outcomes. This project aims to develop a machine learning-based classification system that can predict vitamin deficiency diseases based on patient demographic information, lifestyle factors, dietary intake, laboratory test results, and clinical symptoms.

The classification problem involves predicting the disease diagnosis from multiple classes including:

- Healthy (no deficiency)
- Anemia (iron/vitamin B12/folate deficiency)
- Rickets/Osteomalacia (vitamin D deficiency)
- Scurvy (vitamin C deficiency)
- Beriberi (vitamin B1 deficiency)
- Pellagra (vitamin B3 deficiency)
- And other vitamin deficiency-related conditions

This is a multi-class classification problem where we need to accurately classify patients into one of several disease categories based on their clinical and lifestyle features.

b. Dataset Description

Dataset Source: Kaggle - Vitamin Deficiency Disease Prediction Dataset
Dataset Link: https://www.kaggle.com/datasets/nudratabbas/vitamin-deficiency-disease-prediction-dataset

Dataset Characteristics:

- **Total Instances:** 4,000 records
- **Total Features:** 34 attributes (12+ features as required)
- **Target Variable:** `disease_diagnosis` (multi-class classification)
- **Dataset Split:**
  - Training/Validation: 3,600 instances (90%)
  - Test: 400 instances (10%)
  - Stratified split to maintain class distribution

Feature Categories:

1. **Demographic Features:**
   - Age, Gender, BMI
2. **Lifestyle Features:**
   - Smoking Status, Alcohol Consumption, Exercise Level
   - Diet Type (Omnivore, Vegetarian, Pescatarian, Vegan)
   - Sun Exposure, Income Level, Latitude Region
3. **Nutritional Intake (Percentage of RDA):**
   - Vitamin A, C, D, E, B12, Folate
   - Calcium, Iron
4. **Laboratory Test Results:**
   - Hemoglobin (g/dL)
   - Serum Vitamin D (ng/mL)
   - Serum Vitamin B12 (pg/mL)
   - Serum Folate (ng/mL)
5. **Clinical Symptoms:**
   - Symptoms Count (0-9)
   - Symptoms List
   - Binary symptom flags: Night Blindness, Fatigue, Bleeding Gums, Bone Pain, Muscle Weakness, Numbness/Tingling, Memory Problems, Pale Skin
   - Multiple Deficiencies Flag
6. **Target Variable:**
   - Disease Diagnosis (multi-class)

Data Preprocessing:

- Removed duplicate records
- Handled missing values (mean imputation for numeric, 'Unknown' for categorical)
- Identified and converted fake numeric columns (≤10 unique values) to categorical
- Applied one-hot encoding for categorical features
- Used MinMaxScaler for feature scaling
- Stratified train-validation split (80-20) within training data

c. Models Used

Seven machine learning classification models were implemented and evaluated:

1. **Logistic Regression** - Linear classification model
2. **Decision Tree Classifier** - Non-linear tree-based model
3. **K-Nearest Neighbor (KNN) Classifier** - Instance-based learning
4. **Naive Bayes (Gaussian)** - Gaussian distribution assumption
5. **Naive Bayes (Multinomial)** - Multinomial distribution assumption

6. **Random Forest (Ensemble)** - Ensemble of decision trees
7. **XGBoost (Ensemble)** - Gradient boosting ensemble method

Note: Both Gaussian and Multinomial Naive Bayes variants were implemented. The Multinomial variant performed significantly better (79.44% vs 52.22% accuracy), indicating that count/frequency-based features are more suitable for this dataset.

Model Comparison Table

ML Model Name	Accuracy	AUC	Precision	Recall	F1	MCC
Logistic Regression	0.9292	0.9847	0.9275	0.9292	0.9272	0.8978
Decision Tree	0.9958	0.9995	0.9959	0.9958	0.9958	0.9940
KNN	0.7708	0.8405	0.7452	0.7708	0.7520	0.6640
Naive Bayes (Gaussian)	0.5222	0.7738	0.6190	0.5222	0.5092	0.3664
Naive Bayes (Multinomial)	0.7944	0.9368	0.7513	0.7944	0.7707	0.6987
Random Forest (Ensemble)	0.9486	0.9987	0.9504	0.9486	0.9432	0.9255
XGBoost (Ensemble)	0.9972	1.0000	0.9972	0.9972	0.9972	0.9960

Observations on Model Performance

ML Model Name	Observation about model performance
Logistic Regression	Strong performance (92.92% accuracy, 98.47% AUC). Balanced precision (92.75%) and recall (92.92%) indicate excellent generalization. MCC of 0.8978 shows very strong predictive correlation. Suitable for interpretability and baseline comparison.
Decision Tree	Excellent performance (99.58% accuracy, 99.95% AUC). Near-perfect precision and recall with minimal overfitting on validation. MCC of 0.994 demonstrates exceptional predictive power. May require careful validation on new data.
KNN	Moderate performance (77.08% accuracy, 84.05% AUC). Lower precision (74.52%) suggests false positives. Performance limited by curse of dimensionality with many one-hot encoded features. MCC of 0.664 indicates reasonable but not outstanding performance.
Naive Bayes (Gaussian)	Poor performance (52.22% accuracy, 77.38% AUC). Gaussian distribution assumption unsuitable for this dataset. Demonstrates importance of choosing appropriate distribution assumptions.
Naive Bayes (Multinomial)	Good performance (79.44% accuracy, 93.68% AUC). Significantly outperforms Gaussian variant, indicating count/frequency-based features are more suitable. Balanced precision and recall with solid predictive capability (MCC: 0.6987).

ML Model Name	Observation about model performance
Random Forest (Ensemble)	Excellent performance (94.86% accuracy, 99.87% AUC). Ensemble approach reduces overfitting while maintaining high accuracy. Robust precision and recall. MCC of 0.9255 shows very strong predictive power. Strong candidate for deployment.
XGBoost (Ensemble)	Outstanding performance (99.72% accuracy, 100% AUC). Near-perfect precision and recall. MCC of 0.996 indicates exceptional predictive accuracy. Best-performing model due to ability to capture complex non-linear relationships. Requires careful validation on independent test data.

Overall Observations:

- **Best Models:** XGBoost and Decision Tree achieve exceptional performance, with XGBoost showing near-perfect metrics.
- **Ensemble Methods:** Random Forest and XGBoost significantly outperform individual models, demonstrating ensemble learning effectiveness.
- **Linear Models:** Logistic Regression balances performance and interpretability, suitable for baseline comparisons.
- **Naive Bayes:** Multinomial variant (79.44%) significantly outperforms Gaussian (52.22%), highlighting the importance of appropriate distribution assumptions.
- **Feature Engineering:** Preprocessing steps (fake numeric handling, one-hot encoding, scaling) improved performance across all algorithms.

Repository Structure

```
project-folder/
├── app.py                  # Main Streamlit application
├── train_models.py         # Model training script
├── requirements.txt        # Python dependencies
├── README.md               # This file
├── download_dataset.sh     # Dataset download script
├── model/                  # Saved model files
│   ├── *.pkl               # Trained model files
│   ├── scaler.pkl          # Feature scaler
│   ├── label_encoder.pkl   # Target encoder
│   └── feature_columns.pkl # Feature column names
├── dataset/                # Dataset files
│   ├── data.csv            # Original dataset
│   ├── train_valid.csv     # Training/validation split
│   ├── test.csv            # Test split
│   └── default_values.json # Default values for form inputs
├── model_comparison_table.csv       # Training metrics
├── model_comparison_test.csv        # Test metrics
└── .streamlit/            # Streamlit configuration
    └── config.toml         # Theme configuration
```

Installation and Usage

Prerequisites

- Python 3.8 or higher
- pip package manager

Installation

1. Clone the repository:

```
git clone https://github.com/verathagnus/disease-detector-ml/
cd disease-detector-ml
```

2. Install dependencies:

```
pip install -r requirements.txt
```

3. Download the dataset:

```
chmod +x download_dataset.sh
./download_dataset.sh
```

Training Models (Can be done from the streamlit app)

Run the training script to train all models:

```
python train_models.py
```

This will:

- Load and preprocess training data
- Train all 7 classification models
- Evaluate models on validation set
- Save models and generate `model_comparison_table.csv`

Running the Streamlit App

```
streamlit run app.py
```

The app will open in your browser with the following features:

- Dataset management (download, delete)
- Dataset preparation (remove duplicates, train-valid and test split)

- Model training and retraining
- Model comparison tables
- Test data evaluation
- CSV file upload for predictions
- Single instance prediction form
- Confusion matrix visualization
- Evaluation metrics display

Deployment

The application is deployed on Streamlit Community Cloud:

- **Live App Link:** https://disease-detector-ml.streamlit.app/

Evaluation Metrics Explanation

- **Accuracy:** Overall percentage of correct predictions
- **AUC (Area Under ROC Curve):** Measures the model's ability to distinguish between classes (multi-class: One-vs-Rest)
- **Precision:** Ratio of true positives to all predicted positives (weighted average for multi-class)
- **Recall:** Ratio of true positives to all actual positives (weighted average for multi-class)
- **F1 Score:** Harmonic mean of precision and recall
- **MCC (Matthews Correlation Coefficient):** Balanced measure for multi-class classification, ranges from -1 to +1

Technologies Used

- **Python 3.10**
- **Streamlit** - Web application framework
- **Scikit-learn** - Machine learning library
- **XGBoost** - Gradient boosting framework
- **Pandas** - Data manipulation
- **NumPy** - Numerical computing
- **Matplotlib & Seaborn** - Data visualization
- **Joblib** - Model serialization