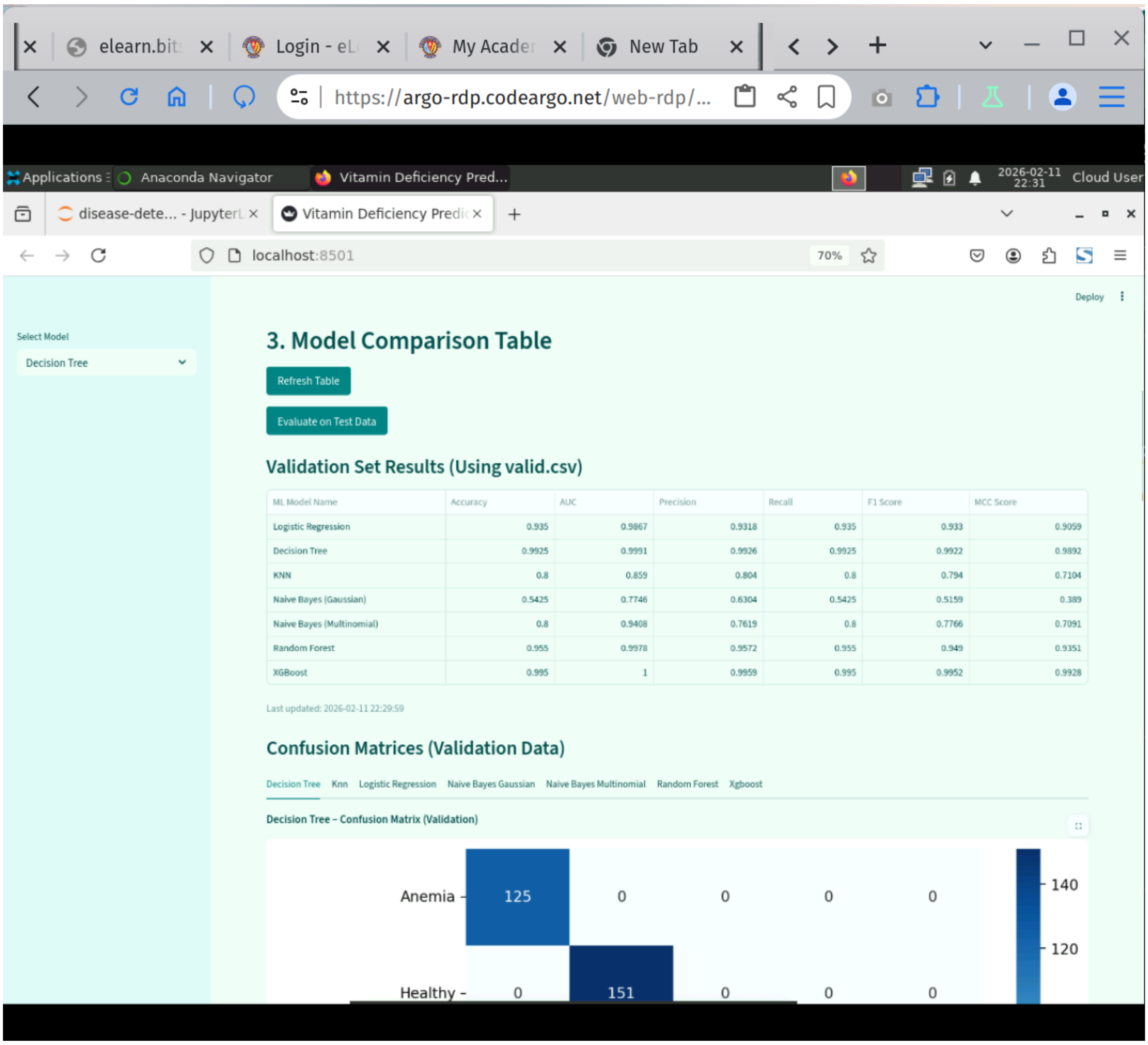
# Machine Learning Assignment 2 Submission Details

Deliverables:

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

## 2. Live Streamlit App Link: https://disease-detector-ml.streamlit.app/

## 3. Running application on BITS Lab Screenshot



### 3. Model Comparison Table

#### Validation Set Results (Using valid.csv)

| ML Model Name | Accuracy | AUC | Precision | Recall | F1 Score | MCC Score |
|---|---|---|---|---|---|---|
| Logistic Regression | 0.935 | 0.9867 | 0.9318 | 0.935 | 0.933 | 0.9059 |
| Decision Tree | 0.9925 | 0.9991 | 0.9926 | 0.9925 | 0.9922 | 0.9892 |
| KNN | 0.8 | 0.859 | 0.804 | 0.8 | 0.794 | 0.7104 |
| Naive Bayes (Gaussian) | 0.5425 | 0.7746 | 0.6304 | 0.5425 | 0.5159 | 0.389 |
| Naive Bayes (Multinomial) | 0.8 | 0.9408 | 0.7619 | 0.8 | 0.7766 | 0.7091 |
| Random Forest | 0.955 | 0.9978 | 0.9572 | 0.955 | 0.949 | 0.9351 |
| XGBoost | 0.995 | 1 | 0.9959 | 0.995 | 0.9952 | 0.9928 |

Last updated: 2026-02-11 22:29:59

#### Confusion Matrices (Validation Data)

Decision Tree – Confusion Matrix (Validation)

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Anemia | 125 | 0 | 0 | 0 | 0 |
| Healthy | 0 | 151 | 0 | 0 | 0 |

## 4. Content of the README.md file is given in the following section

## 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 5 classes:

- Healthy (no vitamin deficiency)
- Anemia (primarily iron, vitamin B12, or folate deficiency)
- Rickets and Osteomalacia (vitamin D deficiency affecting bone mineralization)
- Night Blindness (vitamin A deficiency affecting low-light vision)
- Scurvy (vitamin C deficiency affecting connective tissues)

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
- **Target Variable:** `disease_diagnosis` (5 classes: Healthy, Anemia, Rickets_Osteomalacia, Night_Blindness, Scurvy)
- **Dataset Split:**
    - Training: 3,200 instances (80%)
    - Validation: 400 instances (10%)
    - Test: 400 instances (10%) kept separate for user-directed evaluation from the Streamlit UI.
    - Stratified split was employed to ensure class distribution was maintained across all three subsets.

**Feature Categories:**

1. **Demographic Features:**
    - Age
    - Gender
    - BMI
2. **Lifestyle Features:**
    - Smoking Status
    - Alcohol Consumption
    - Exercise Level

- Sun Exposure
- Diet Type
- Income Level
- Latitude Region

3. **Nutritional Intake (Percentage of RDA):**
    - Vitamin A, C, D, E, B12, Folate (B9)
    - 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:**

- Performed a stratified 8:1:1 split of the full dataset into train (80%), validation (10%), and test (10%) subsets via the Streamlit app
- Handled missing values (mean imputation for numeric, 'Unknown' for categorical)
- Identified and converted pseudo numeric columns to categorical (Values are numeric but fixed discrete values only)
- Applied one-hot encoding for categorical features
- Used MinMaxScaler for feature scaling

## 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 (80% vs 54.25% 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 | 0.9375 | 0.9865 | 0.9341 | 0.9375 | 0.9351 | 0.9095 |
| Decision Tree | 0.9925 | 0.9991 | 0.9926 | 0.9925 | 0.9922 | 0.9892 |
| KNN | 0.8000 | 0.8590 | 0.8040 | 0.8000 | 0.7940 | 0.7104 |
| Naive Bayes (Gaussian) | 0.5425 | 0.7746 | 0.6304 | 0.5425 | 0.5159 | 0.3890 |
| Naive Bayes (Multinomial) | 0.8000 | 0.9408 | 0.7619 | 0.8000 | 0.7766 | 0.7091 |
| Random Forest (Ensemble) | 0.9550 | 0.9978 | 0.9572 | 0.9550 | 0.9490 | 0.9351 |
| XGBoost (Ensemble) | 0.9975 | 1.0000 | 0.9978 | 0.9975 | 0.9976 | 0.9964 |

### Observations on Model Performance

| ML Model Name | Observation about model performance |
|---|---|
| Logistic Regression | Remarkable performance (93.75% accuracy, 98.65% AUC) for a linear classifier. Balanced precision (93.41%) and recall (93.75%) with F1 (93.51%) indicate no major class imbalance. MCC of 0.9095 shows strong predictive correlation. It serves as a highly interpretable baseline |
| Decision Tree | Near pefect metrics (99.25% accuracy, 99.91% AUC). Near-perfect precision(99.26%) and recall (99.25%) with F1 (99.22%) shows minimal overfitting on validation set. MCC of 0.9892 demonstrates exceptional predictive power. However, a single tree is likely sensitive to training data pertubations and the reported validation metrics may need futher confirmation in future to be well-suited for deployment. For now, this is the second best model. |

| ML Model Name | Observation about model performance |
|---|---|
| KNN | Moderate performance (80.00% accuracy, 85.90% AUC). Precision (80.40%) and recall (80.00%) along with F1 (79.40%) are also moderate. This is attributable to the curse of dimensionality due to many one-hot encoded features causing a high-dimensional sparse feature space in which Euclidean distance loses discriminatory power. MCC of 0.7104 again indicates moderate predictive performance. |
| Naive Bayes (Gaussian) | Poor performance (54.25% accuracy, 77.46% AUC, 63.04% Precision, 54.25% Recall, 51.59% F1). Gaussian distribution assumption is unsuitable for this dataset characteristics. It demonstrates importance of choosing appropriate distribution assumptions. |
| Naive Bayes (Multinomial) | Good performance (80.00% accuracy, 94.08% AUC). Significantly improvement over Gaussian variant, indicating count/frequency-based features are more suitable yielding balanced precision (76.19%) and recall (80.00%) with balanced F1 (77.66%) along with solid predictive capability (MCC: 0.7091). |
| Random Forest (Ensemble) | Excellent performance (95.50% accuracy, 99.78% AUC). Ensemble approach mitigates overfitting while maintaining high accuracy by retaining non-linear relationships. Robust precision (95.72%) and recall (95.5%) with F1 (94.9%) indicating high generalizability. MCC of 0.9351 shows very strong predictive power. This model offers an excellent bias-variance trade-off and is well-suited for deployment. |
| XGBoost (Ensemble) | Outstanding performance (99.75% accuracy, 100% AUC). Near-perfect precision (99.78%) and recall (99.75%) with F1 (99.76%). MCC of 0.9964 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 (80.00%) significantly outperforms Gaussian (54.25%), highlighting the importance of appropriate distribution assumptions.
- **Feature Engineering:** Preprocessing steps (fake numeric handling, one-hot encoding, scaling) were needed to incorporate all features appropriately.

## Repository Structure

```
project-folder/
├── app.py                      # Main Streamlit application
├── train_models.py             # Model training script
├── requirements.txt            # Python dependencies
├── README.md                   # GitHub README 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 folder
    └── config.toml             # Theme configuration
```

## Installation and Usage

### Prerequisites

- Python 3 with pip or conda package manager. (Preferably use miniconda environment or venv)

### 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. Terminal Based Usage

Download the dataset (Can be done from the streamlit app as well)

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

Run the dataset splitting script to partition the downloaded dataset:

```
python prepare_dataset.py
```

Training Models

Run the training script to train all models:

```
python train_models.py
```

This will:

- Download the dataset as data.csv in dataset folder
- Split the dataset into train.csv, valid.csv, test.csv
- 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 the browser with the following features:

- Dataset Download
- Dataset Preparation
- Model Training and Retraining
- Model comparison tables and confusion matrices view
- Test data evaluation and confusion matrix display
- CSV file upload for generating predictions (and evaluation if target column is present in uploaded csv file.)
- Single instance prediction form
- Confusion matrix visualization
- Evaluation metrics display