A group of graphs showing different colored squares

Description automatically generated

A group of graphs with different colored squares

Description automatically generated

A group of graphs showing different colored squares

Description automatically generated

A group of graphs showing different colored squares

Description automatically generated

Conclusion from numerical features vs recommended meal plans:

 **All 4 features seem to have similar distributions across diets** (medians and IQRs are quite close).

 There’s **no strong visual separation** between meal plans based on these individual features.

* E.g., people with high cholesterol get all kinds of meal plans, not just "Low-Fat Diet."
* These features **may not be useful on their own**, but:
  + **Could still help in combination** with others (interactions).
  + Might gain value in **nonlinear models** (like trees or neural nets) that can detect subtler patterns.

Predictive Modelling

How I’m evaluating my models: Because the target I’m trying to predict is categorical, I will be using accuracy score, classification reports, and analyzing confusion matrices to compare my models.

1/ Creating a Baseline Model

I chose to use uniform as the strategy

1. “**uniform**”: This strategy assigns class labels randomly and uniformly, without considering the class distribution in the training data. It is useful when there is no specific pattern or information to guide the predictions.

2/ Linear Regression Model

3/KNN

\* Curse of dimensionality: KNN is more appropriate to use when you have a small number of inputs. If the number of variables grows, the KNN algorithm will have a hard time predicting the output of a new data point.

\* Class imbalance can be an issue: If we have an imbalanced class data, the algorithm might wrongly pick the majority class.

4/ Decision Tree

\* Decision trees are effective in capturing *non-linear relationships* which can be difficult to achieve with other algorithms like Support Vector Machine and Linear Regression.

\* As the tree grows in size, it becomes prone to overfitting and requires pruning

Since all of my models performed so poorly with a big gap between train accuracy and test accuracy, which indicates that my models are overfitted, I’m going to try to fix the overfitting issue by:

**Use fewer or better features**

* Remove irrelevant or highly correlated features
* Try feature selection or dimensionality reduction (e.g., PCA)

**Cross-validation**

* Helps you tune hyperparameters and get a more robust performance estimate

Ok, so in narrowing down the features, I’m going to choose overlaps in 4-5 features that I see in the top features list across my lgr, knn, and decision tree models. This will hopefully help reduces dimensionality, which improves performance and interpretability; speeds up training and testing, especially helpful with a large dataset; Reduces overfitting risk, especially in models like Decision Trees and KNN.

A screenshot of a table

Description automatically generated

Looking at these lists, 5 features seem to consistently perform decently across the three models:

1. Chronic disease
2. Fat intake
3. BMI
4. Food Aversions
5. Blood Pressure Diastolic

# Testing Neural Networks with fewer features

* 🌳 **Decision trees** can uncover complex, nonlinear interactions between features.
* 🤖 **Neural networks** also excel at capturing nonlinear relationships — *but* they can be more sensitive to irrelevant or noisy input features.
* ✅ So, reducing dimensionality using decision tree–based feature selection can help a neural network model **converge better** and **generalize more effectively**.

**A few suggestions as you try this:**

* **Normalize/scale** your input features before feeding them into the neural network (especially if using gradient descent optimizers).
* **Try a shallow neural net** first (e.g., 1 or 2 hidden layers) to avoid overfitting, especially with fewer features.
* Use **early stopping or dropout** if you still suspect overfitting

Protein\_Intake, Caloric\_Intake, BMI, Daily\_Steps, Carbohydrate\_Intake, Fat\_Intake, Cholesterol\_Level, Age, Blood\_Sugar\_Level, Blood\_Pressure\_Diastolic

With 3 layers:

Classification report:

A screenshot of a graph

Description automatically generated

A graph with a line

Description automatically generated

With 4 layers:

A screenshot of a graph

Description automatically generated

A graph with a line drawn on it

Description automatically generated

Comparing my losses intraining the data for multiclass neural network, the one for 3 layers declined more smoothly than my 4 layer neural network model -the 4 layer neural network model loss also showed decline in loss overtime, but it had 3 wonky spikes near the tail. What does that mean

Analyze: **Spikes in loss**, especially near the end of training, often suggest:

* Learning rate may be too high → causing the model to "jump" around the optimal point.
* **Overfitting** due to excess capacity → more layers means more parameters, which can memorize noise.
* **Unstable gradients** or poor batch composition late in training (depending on your batch size or data shuffling).

Adjustment: Reduced lr from 0.01 to 0.001to smooth out the spikes

A graph with a line graph

Description automatically generated A screenshot of a graph

Description automatically generated

### Interpretation: **Overall Performance**

* **Accuracy: 74%** — This is well above random guessing for 4 classes (random chance = 25%), indicating your model is learning meaningful patterns.
* **Macro avg F1-score: 0.74** — This treats all classes equally, so a good macro score means your model is not heavily favoring one class

 **Class 0 and 3 perform best**:

* Precision and recall are both **above 0.83**, meaning the model is confidently and correctly identifying these classes.

 **Class 2 needs improvement**:

* **Recall is 0.56**, meaning it misses quite a few actual class 2s.
* This could be due to:
  + Lower representation (but class support seems fairly balanced).
  + Less distinguishable feature patterns.