Fall 2025 CPSC436/536 Project1 - Exploring Labeled Data Using kNN

Objective: Gain understanding of kNN and ML models in general and enhance proficiency by applying these techniques to a real-world dataset.

Dataset: You'll work on a dataset (attached) extracted from National Health and Nutrition Examination Survey: https://www.cdc.gov/nchs/nhanes/index.htm.

- The dataset contained health records of n NHANES participants.
  - The attribute list includes:
    - age, gender, race, blood pressure readings (systolic and diastolic), lab work (levels of total cholesterol (TCHOL), LDL, HDL, triglyceride), and certain medical conditions such as diabetes. We also know whether he/she is a current smoker (smoker).
    - In addition to the above attributes, medical professionals consider some interaction terms are important, such as age* Systolic, age* TCHOL, age*HDL, age* smoker. You might want to consider them.
  - Target variable: MI. (whether the participant had a heart attack (myocardial infarction)).

Goal: Predict the probability of a participant who experienced a heart attack (MI).

Output: Utilize your <u>most optimal model</u> to forecast the likelihood of individuals in the testing dataset who had experienced a heart attack (MI).

Things to Consider When You Tune Your kNN Models:

1. How many features/attributes does the dataset have?
   - More features can increase computational cost and worsen performance due to the curse of dimensionality.
   - Consider dimensionality reduction (feature selection) if needed.
     - drop obviously irrelevant columns (IDs, constant columns, etc.)
     - use domain knowledge (e.g., medical intuition)
     - remove highly correlated features
     - keep the ones that are most correlated with the target
     - apply univariate feature selection (e.g., ANOVA F-test) to rank features based on individual predictive power
     - use variance thresholding - drop features with very low variance
     - explore recursive feature elimination to select features based on model performance
     - ...
2. Have you considered how to handle missing values properly?
   - Drop rows (if very few are affected)
   - Impute using mean/median/mode
3. What is the class distribution?
   - Check the number of instances in each class (e.g., NHANES MI = 1 vs. MI = 2).
   - If the dataset is imbalanced, consider:
     - Resampling techniques (e.g., SMOTE, undersampling)
     - Evaluation metrics like F1-score, precision-recall, confusion matrix
4. What's the best value of k?
   - Use cross-validation to select k.
   - Too small → overfitting; too large → underfitting.
   - Try odd values to avoid classification ties.
5. What distance metric should you use?
   - Common choices: Euclidean, Manhattan, Minkowski
   - Remember: kNN is sensitive to scale, so normalize features using StandardScaler or MinMaxScaler.
6. Are categorical variables encoded properly?
   - Categorical features like gender and race must be numerically encoded.
   - Use one-hot encoding or ordinal encoding, depending on the variable type.
7. Should you add interaction terms or derived features?
   - Expert-informed interactions like age × TCHOL or age × smoker may improve predictive power.
   - Try manual feature engineering.

8. Have you validated your model properly?
   - Use stratified train/test splits to preserve class distribution.
   - Apply k-fold cross-validation to estimate generalization performance.
9. Are you using appropriate performance metrics? Consider:
   - Accuracy
   - F1-score, precision, and recall
   - Confusion matrix
11. Is your model efficient and scalable?
   - kNN can be slow for large datasets.
   - Have you tried different neighbor-finding algorithms?

What to submit: Your Jupyter notebook and your predictions for MI = 1 for the participants in the testing dataset.

Evaluation: Your project will be evaluated based on how well your model predicts the probability of a heart attack (MI). Specifically, we will use two metrics:

- **Accuracy**: Measures how often the model's predicted class (based on a 0.5 threshold) matches the true label.
- **Kullback-Leibler (KL) Divergence**: Measures how close your predicted **probability distribution** is to the true label distribution. Lower values indicate better calibrated probabilistic predictions.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \, \log\left(\frac{P(x)}{Q(x)}\right),$$

where $P(x)$ is the true distribution, $Q(x)$ is the predicted distribution. This is equivalent to log loss (cross-entropy), where $\hat{p} = Q(x)$ is the predicted probability:

$$D_{KL}(P \parallel Q) = y \cdot log\left(\frac{1}{\hat{p}}\right) + (1-y) \cdot log\left(\frac{1}{1-\hat{p}}\right)$$

You can try this metric yourself:

```
from sklearn.metrics import log_loss
kl_div = log_loss(y_true, y_pred_prob)
```

Attributes keys:

Age	Continues
BMI	Continues
CurrentSmoker	1 yes; 2 no
Diabetes	1 yes; 2 no
Diastolic	Continues
Edu	1- Less than 9th grade; 2- 9-11th grade (Includes 12th grade with no diploma); 3- High school graduate/GED or equivalent; 4- Some college or AA degree; 5- College graduate or above
HDL	Continues
Income	Ratio of family income to poverty
isActive	1 yes; 2 no
Insurance	Categorical
kidneys_eGFR	Continues
LDL	Continues
Pulse	Continues
Race*	1 Mexican American, 2 Other Hispanic, 3 Non-Hispanic White, 4 Non-Hispanic Black, 5. None, 6. Non-Hispanic Asian, 7. Other Race - Including Multi-Racial
Sex	1 male; 2 female
Systolic	Continues
TCHOL	Continues
Trig	Continues

*Sometime people consider only three race groups: white, black, and others. Blank for missing values.