# Assignment 4: Building Classifiers

## 1 Instructions

Through this assignment, you will advance your understanding of two foundational classification algorithms: ***k*-Nearest Neighbors (KNNs)** and **Logistic Regression (LR)**. You will find a suitable dataset, pre-process it, and then use the `scikit-learn` library to build, tune, and evaluate these models. You will then prune the feature set to half the number of features and see if the performance improves or degrades. Finally, you will leverage your insights to build a third, superior classification model of your choice.

*The only supported platform for completing this assignment is Google Colab. However, you are welcome to use a local version of Python and Jupyter notebooks if you are comfortable doing that.*

## 2 Submission Format

In Gradescope, you will simply submit the `.ipynb` Jupyter notebook file. Note that you can download Google Colab notebooks as ipynb files. **Make sure to save the notebook with all output cells rendered** so your report can be generated correctly. It is critical that we can fully reproduce your results by running all cells sequentially after installing any dependencies you list.

## 3 Notebook Structure and Content

Your submission must be a single Jupyter notebook organized into the following YAML header cell and eight content cells. Please see the template notebook file for a template of the cells including the YAML header.

### 3.1 Cell 0: Quarto YAML Header (Markdown)

Please use the following YAML header as the first cell in your notebook (this instructs Quarto how to render the notebook for the gallery). Note that you can use the "Markdown" cell in Google Colab for this but it will render a little weird in Colab but that's okay.

```yaml
---
format:
  html:
    code-fold: true
jupyter: python3
---
```

### 3.2 Cell 1: Dataset Justification (Markdown)

In this cell, provide a detailed explanation of your dataset selection and preprocessing strategy.

- **Dataset Selection**: Identify an interesting dataset from Kaggle's catalog—this link filters to classification datasets that are CSVs and less than 100MB. You can choose any dataset you are interested in. You are encouraged to browse the datasets and find one that you find particularly interesting. The dataset must be less than 100MB in size. Do not use the Iris dataset.
- **Preprocessing Plan**: Provide a brief justification for each preprocessing step you will take. For example, explain why you are using one-hot encoding for a specific categorical feature or why feature scaling is important or helpful.

### 3.3 Cell 2: Data Loading and Preprocessing (Code)

This cell should contain all the code for downloading, loading, and preprocessing your data.

- Install and import all necessary dependencies at the top of this cell.
- Ideally, this cell would automatically download the dataset but that is not a requirement as long as you clearly state where you got the dataset.
- Perform the preprocessing steps you justified in the previous cell.
- Split your data into training and testing sets.
- **Output**: The cell's output must display the names of the features (or generic names like `feature_01`, `feature_02`, etc. if no feature names are given), the shape of your final input data `X`, and the shape of your target labels `y`.

### 3.4 Cell 3: Model Training Plan (Markdown)

Explain your methodology for training and tuning the KNN and Logistic Regression models using **only the training dataset** for all training and hyperparameter tuning (the test set should NOT be used for hyperparameter tuning).

- **Hyperparameter Tuning**: Clearly state that you will use `scikit-learn`'s `sklearn.model_selection.GridSearchCV` to find the optimal hyperparameters.
- **Models**: You must use `scikit-learn`'s implementations for KNN (`KNeighborsClassifier`) and Logistic Regression (`LogisticRegression`).
- **Hyperparameter Ranges**: Justify your choice for the range of hyperparameters you will search over.
  - For **KNN**, explain your range of values for `k` (the number of neighbors). For instance, you might search over odd numbers to avoid ties in binary classification.
  - For **Logistic Regression**, explain your range of values for the regularization parameter `C`. A logarithmic scale (e.g., `[0.01, 0.1, 1, 10, 100]`) is often a good starting point.
- **Validation**: Explain how you will call `GridSearchCV` to perform cross-validation to prevent overfitting and provide a robust estimate of performance on unseen data.

### 3.5 Cell 4: KNN and Logistic Regression Training (Code)

In this cell, implement the training and evaluation process.

- Use `GridSearchCV` to perform a grid search with cross-validation on the training data for both KNN and Logistic Regression to find the best hyperparameters. `GridSearchCV` will automatically retrain the best model on the full training dataset—hint: see `refit=True` hyperparameter. Importantly, you should only pass the **training dataset** to the `GridSearchCV` fit method.
- Evaluate the final, tuned models on your held-out test set.
- **Output**: The cell's output must clearly report the following for **both** KNN and Logistic Regression:

  1. The best hyperparameter chosen (e.g., `k=5` for KNN, `C=10` for LR).
  2. The mean cross-validation accuracy for that best hyperparameter.
  3. The final accuracy on the held-out test set.

### 3.6 Cell 5: Feature Pruning Plan (Markdown)

Explain your strategy for reducing the number of features in your dataset to **at most half** of the original number.

- **Methodology**: Describe the method you will use. This could be a statistical approach (e.g., selecting features with high mutual information or using EDA to find important features), a brute-force approach, or logical reasoning based on your understanding of the features. Justify your choice of method.
- **Hypothesis**: After explaining your method, discuss whether you think the performance will improve or degrade based on your understanding. Briefly justify why or why not.

### 3.7 Cell 6: Training with Pruned Features (Code)

Implement your feature pruning strategy and re-run the complete training and evaluation pipeline from Cell 4.

- Apply your chosen pruning method to create a new, smaller dataset.
- Use `GridSearchCV` on this new dataset to find the best hyperparameters for KNN and Logistic Regression.
- Evaluate the final models on the (correspondingly pruned) held-out test set.
- **Output**: The cell's output must report the same metrics as Cell 4, but clearly labeled as being for the **pruned feature set**:

1. The best hyperparameter for each model.
2. The mean cross-validation accuracy for each model.
3. The final test accuracy for each model.

### 3.8 Cell 7: "Choose Your Own Adventure" Plan (Markdown)

In this section, you will design a third model with the goal of outperforming both KNN and Logistic Regression on your dataset. You are free to use any model and any library (e.g., scikit-learn, PyTorch, etc.).

- **Model Justification**: Propose your third model. Explain why you chose this approach and why you hypothesize it will generalize more effectively than KNN and LR.
- **Training Plan**: Briefly describe how you will train and select hyperparameters for this new model.

### 3.9 Cell 8: "Choose Your Own Adventure" Implementation (Code)

Implement, train, and evaluate your third model.

- Include all code for training and tuning your chosen classifier.
- Evaluate its performance on the held-out test set.
- **Output**: The final output of this cell must be the **final test accuracy** of your new classifier.

---

## 4 Rubric

Each of the four major parts of the assignment will be weighted equally (25% each).

| Criterion | Excellent (5) | Good (4) | Satisfactory (3) | Okay (2) | Poor (1) |
|---|---|---|---|---|---|
| **Dataset & Pre-Processing** | A carefully justified dataset is chosen containing structure amenable to KNNs / LRs, and is processed perfectly. | A well justified and well-processed dataset is presented but justifications for pre-processing steps are vague. | The dataset is well-justified but is not adequately pre-processed. | A valid dataset is chosen but isn't justified, with poorly pre-processed features. | The selected dataset is too large or is the Iris dataset. |
| **KNN & LR Implementations** | Bug-free implementations with rigorous hyperparameter search and validation. | Bug-free implementations with a rigorous hyperparameter search but inadequate validation. | Bug free implementations but with no hyperparameter search or validation. | Significant bugs are included in the implementations that make results unreliable. | No implementation, or `scikit-learn` is not used for the implementation. |
| **Feature Pruning** | Excellently justified feature pruning with rigorous methodology and critically analyzed feature subsets. | The feature subset is well reasoned but the follow-up analysis is missing. | The feature subset is sound but unsupported by the methodology. | The chosen feature subset is suboptimal with vague or unsound justifications. | Dataset pruning retains more than half of the original feature set. |

| Criterion | Excellent (5) | Good (4) | Satisfactory (3) | Okay (2) | Poor (1) |
|---|---|---|---|---|---|
| **Choose Your Own Adventure** | A perfectly reasoned alternative approach is implemented that outperforms the two prior approaches with rigorous tuning and analysis. | A strong effort is made to apply insights from the previous sections, but the performance is on-par with the best of the prior two approaches. | The alternative approach is reasonable and implemented correctly but justification is missing or results are significantly below KNN and logistic regression. | An approach is implemented but is incorrect or lacks justification. | No meaningful approach is proposed. |