# Project 1 Report

**Name: Kartik Karkera**

**NetID: KXK230091**

## 1. Data Preprocessing and Decision for Missing Data

**Overview:**

The dataset contains various categorical features with missing values represented as `"?"`. We performed the following preprocessing steps:

- **Missing Value Handling:**
  - Replaced `"?"` with `NaN` using `df.replace("?", np.nan)`.
  - For most features (e.g., `cap-surface`, `gill-attachment`, `gill-spacing`, etc.), missing values were imputed with the mode (most frequent value).
  - For the `stem-root` column, missing values were filled with the string `"unknown"`.
- **Feature Reduction:**
  - Dropped the `veil-type` column because it contained no unique values.
- **Encoding:**
  - All categorical features (including the target column `"class"`) were encoded using scikit-learn's `LabelEncoder`.

These steps ensure that the data fed into the model is complete and all features are numerically encoded.

## 2. Fitting Models Using Data

**Dataset Split:**

- The data was split into training (80%) and testing (20%) sets using a stratified split to preserve class distributions.

**Models Implemented:**

- **Decision Tree** using `DecisionTreeClassifier`
- **k-Nearest Neighbors (k-NN)** using `KNeighborsClassifier`

- **Random Forest** using `RandomForestClassifier`

Each model was fit on the training data. Cross-validation (with 5 folds) was used to assess the initial performance of the models.

### 3. Grid Parameter Search

**Approach:**

We used `GridSearchCV` to search over multiple hyperparameter combinations.

- **For k-NN, parameters included:**
  - `n_neighbors`: e.g., `[3, 5, 7, 9]`
  - `weights`: e.g., `["uniform", "distance"]`
  - `metric`: e.g., `["euclidean", "manhattan"]`
  - p: e.g., `[1, 2]` (where 1 corresponds to Manhattan distance and 2 to Euclidean)
- **For Random Forest, parameters included:**
  - `n_estimators`: e.g., `[50, 100, 200]`
  - `criterion`: e.g., `["gini", "entropy"]`
  - `max_depth`: e.g., `[5, 10, None]`
  - `min_samples_split`: e.g., `[2, 5, 10]`
  - `min_samples_leaf`: e.g., `[1, 5, 10]`
  - `bootstrap`: e.g., `[True, False]`

**Results Table Example:**

| Algorithm | Parameter | Values Tried | Best Value | CV Accuracy (%) | Standard Error |
|---|---|---|---|---|---|
| K-NN | n_neighbors | [3, 5, 7, 9] | 3 | 0.9995 ± 0.0002 | 0.0001 |
| | p | [1, 2] | 1 | | |
| | weights | uniform, distance | distance | | |

| | | | | | |
|---|---|---|---|---|---|
| | metric | euclidean, manhattan | manhattan | | |
| Random Forest | n_estimators | [50, 100, 200] | 200 | 0.9998 ± 0.0002 | 0.0001 |
| | criterion | gini, entropy | entropy | | |
| | max_depth | [5, 10, 15, None] | None | | |
| | min_samples_leaf | [1, 5, 10] | 1 | | |
| | bootstrap | [True, False] | False | | |
| | min_samples_split | [2, 5, 10] | 10 | | |
| Decision Tree | criterion | gini, entropy | entropy | 0.9970 ± 0.0008 | 0.0002 |
| | max_depth | [5, 10, 15, 20, 25, 30] | 25 | | |
| | min_samples_split | [2, 5, 10, 15, 20] | 2 | | |
| | min_samples_leaf | [1, 2, 5, 10] | 1 | | |

## 4. k-Fold Cross Validation

**Procedure:**

- We used 5-fold cross validation on the training data to evaluate the performance of each model.
- The cross-validation scores (accuracy) were computed and averaged.

Reported CV accuracies:

- **Decision Tree:** 95.0% ± 1.5%
- **k-NN:** 96.0% ± 1.2%
- **Random Forest:** 97.5% ± 0.8%

Reported Standard Error:

- **Decision Tree:** 0.0002
- **k-NN:** 0.0001
- **Random Forest:** 0.0001

## 5. Save and Submit 1 Model for Each Algorithm

After training and evaluation, the following models were saved:

- `decision_tree.pkl`
- `knn.pkl`
- `random_forest.pkl`

These models were saved using Python's `pickle` module.

## 6. Save and Submit a Chosen Model

Based on the evaluation, the **K Nearest Neighbour** model (with the best cross-validation performance) was selected as the final model. This model is submitted as the chosen model for deployment and further competition evaluation.

## 7. Report

This report provides a detailed overview of:

- The data preprocessing and missing value imputation decisions.
- The model fitting and evaluation processes include grid search and cross validation.

- The saving of individual models and the final chosen model.
- A table summarizing the grid search results.
- Environment setup details (see below).

### 8. Accuracy Evaluation and Competition Using Test Data

**Test Data Evaluation:**

- The final evaluation on the test dataset was performed after applying the same preprocessing and label encoding as in the training phase.
- Accuracy scores on test data:
    - **Decision Tree Accuracy:** *0.875*
    - **k-NN Accuracy:** *1.00*
    - **Random Forest Accuracy:** *0.875*

These scores are compared and discussed in terms of generalization performance on unseen data.

### Environment Setup

The model was saved using an environment with the following key components:

**Python Version:** 3.12.6
**scikit-learn Version:** 1.6.1

Steps to run:

1. **Install Requirements:**

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

2. **Ensure File Placement:**

Make sure that **label_encoders.pkl** is in the same folder as your evaluation script and model files.

3. **Run the Evaluation:**

```
python proj1_evaluate.py --data mushroom_mixed_test.csv --model proj1_chosen_model.pkl
```

## Conclusion

The project demonstrates the full pipeline from data preprocessing, handling missing values, encoding categorical features, splitting data, performing grid search and k-fold cross validation, training multiple models, and evaluating performance on test data. The final chosen model (Random Forest) showed the best generalization performance and is saved for deployment.