Project Report: Mushroom Classification Analysis

Project Overview

Project Title: Mushroom Classification Using Machine Learning

GitHub Repository Link: https://github.com/broncodo/cmse492_project.git

Brief Project Description:
This project aims to build a machine learning model to classify whether a mushroom is edible or poisonous based on various attributes. The data is preprocessed and analyzed using Python and machine learning libraries to develop a reliable classification model.

Project Setup

Repository Structure:

- **mushroom-classification-analysis.ipynb**: Contains the code for data preprocessing, exploratory data analysis, model training, and testing.
- **data/**: https://www.kaggle.com/datasets/uciml/mushroom-classification
- **models/**: Directory for storing trained machine learning models.
- **submission.csv**: The final output file for predictions.

Dependencies:

- Python 3.x
- Pandas
- Scikit-learn
- Matplotlib
- Jupyter Notebook

Setup Instructions:

1. Clone the repository from GitHub.
2. Install the dependencies using pip install -r requirements.txt.
3. Run the Jupyter Notebook mushroom-classification-analysis.ipynb to execute the project.

(Screenshot of the project structure can be included here if needed.)

Completed Tasks

1. **Data Preprocessing**:
    - The dataset was preprocessed using the KNN Imputer to handle missing values.
    - Object-type columns were encoded using categorical encoding before applying the imputer.
2. **Exploratory Data Analysis**:
    - A histogram of the class distribution (edible vs. poisonous mushrooms) was created to understand the target variable's distribution.
    - Feature analysis was conducted to evaluate which mushroom characteristics were most correlated with the target variable.
3. **Feature Imputation**:
    - The KNN imputation method was applied to handle missing values in the dataset, ensuring that the model receives complete data for training.

Each task was necessary to ensure the dataset was clean and ready for machine learning model training.

Initial Analysis and Findings

- The dataset consists of several categorical features representing different mushroom attributes.
- Key insight: The class distribution is relatively balanced, which is beneficial for model training.
- Initial histograms and visualizations show clear differences in attribute values between edible and poisonous mushrooms, suggesting that the model can differentiate them based on these features.

(You can include and discuss relevant visualizations and plots here, such as the histogram of the class distribution and feature importance.)

Proposed Approach

Machine Learning Model:

- **Proposed Algorithm**: Random Forest Classifier

- **Justification**: The Random Forest is an ensemble method that combines multiple decision trees, which is highly suitable for classification tasks involving categorical features. It is robust to overfitting and provides interpretable results through feature importance.

Model Pipeline:

1. Data preprocessing (including handling missing values and categorical encoding).
2. Train-test split (using an 80-20 ratio).
3. Training the Random Forest classifier.
4. Model evaluation using cross-validation.

(Diagrams or flowcharts outlining the approach can be added here.)

Preliminary Results

- **Accuracy**: The initial run of the Random Forest classifier showed promising results with an accuracy of [Insert Value Here].
- **Feature Importance**: Attributes such as [List important features here] played a significant role in determining the mushroom's edibility.

Include visualizations like confusion matrices, accuracy curves, and feature importance plots to present preliminary results.

Challenges and Solutions

1. **Missing Data**: Several features in the dataset had missing values. The solution was to use the KNN Imputer for imputation, which predicts missing values based on the nearest neighbors.
    - **Solution Code Snippet**:

python

Copy code

```
from sklearn.impute import KNNImputer

knn_imputer = KNNImputer(n_neighbors=5)

df_imputed = pd.DataFrame(knn_imputer.fit_transform(df_encoded),
columns=df_encoded.columns)
```

2. **Categorical Encoding**: Dealing with categorical variables required converting them into numerical values before training the model.
    - **Solution Code Snippet**:

python

Copy code

```
for col in df.select_dtypes(include='object').columns:
  df_encoded[col] = df_encoded[col].astype('category').cat.codes
```

Next Steps

1. **Hyperparameter Tuning**: Tune the Random Forest classifier using grid search to improve performance.
2. **Model Testing**: Test the model on unseen data to evaluate its generalization capability.
3. **Additional Models**: Experiment with other machine learning algorithms, such as Gradient Boosting or SVM, to compare performance.

Timeline and Milestones:

- **Week 1**: Complete hyperparameter tuning.
- **Week 2**: Finalize model testing and comparison of different algorithms.
- **Week 3**: Prepare the final report with detailed results and conclusions.

Conclusion

- **Current Project Status**: The project is currently in the model development and evaluation stage. Initial results are promising, but further refinement is needed.
- **Lessons Learned**: Handling missing data and categorical encoding are critical preprocessing steps in classification problems involving real-world datasets.
- **Outlook for Completion**: The project is on track for successful completion, with the next steps focusing on improving model performance and evaluating different algorithms.