Technical Report: Wine Quality Prediction Application

1 Summary

The objective of this project is to develop an application that predicts wine quality based on its chemical properties. The application allows users to input data, obtain a quality prediction, save this data, and visualize a wine inventory and wines created by users. The implementation is done in Python, utilizing libraries such as `pandas`, `scikit-learn`, `imblearn`, `joblib`, and `tkinter`.

2 Code Structure

The code is organized into several modules, each responsible for a specific part of the workflow:

- **Data Loading and Preprocessing**: `data_preprocessing.py`
- **Model Training and Saving**: `model.py`
- **Model Evaluation**: `evaluation.py`
- **User Interface**: `app.py`
- **Utilities**: `utils.py`

3 Design Decisions

3.1 Data Loading and Preprocessing

3.1.1 Function `load_data(file_path)`

- **Decision**: Use `pandas` to load data from a CSV file.
- **Motivation**: `pandas` is efficient and provides robust functionality for handling tabular data.

3.1.2 Function `preprocess_data(data)`

- **Decision**: Remove unnecessary columns, separate features (`X`) and labels (`y`), and apply SMOTE to handle class imbalance.
- **Motivation**: Ensure the dataset is clean and balanced to improve model performance.

3.1.3 Function `split_data(X, y)`

- **Decision**: Split the data into training and testing sets using `train_test_split`.
- **Motivation**: Allow model evaluation on unseen data, ensuring a fair assessment.

3.2 Model Training and Saving

3.2.1 Function `train_model(X_train, y_train)`

- **Decision**: Use `RandomForestClassifier` with `GridSearchCV` to find the best hyperparameters.
- **Motivation**: `RandomForest` is a robust algorithm for classification, and `GridSearchCV` helps optimize model performance.

3.2.2 Function `save_model(model, file_path)`

- **Decision**: Save the trained model using `joblib`.
- **Motivation**: Allow reuse of the model without retraining, saving time and computational resources.

3.2.3 Function `load_model(file_path)`

- **Decision**: Load a previously saved model using `joblib`.
- **Motivation**: Facilitate prediction without the need to retrain the model each time.

3.3 Model Evaluation

3.3.1 Function `evaluate_model(model, X_test, y_test)`

- **Decision**: Evaluate the model using metrics such as `accuracy_score` and `classification_report`.
- **Motivation**: Obtain a detailed evaluation of model performance in terms of accuracy and other classification metrics.

3.3.2 Function `print_evaluation(accuracy, report)`

- **Decision**: Print the evaluation results clearly and concisely.
- **Motivation**: Facilitate interpretation of results by the user.

3.4 User Interface

3.4.1 Graphical Interface with `tkinter`

- **Decision**: Use `tkinter` to create a simple and accessible user interface.
- **Motivation**: `tkinter` is part of Python's standard library and is sufficient for creating basic GUI applications.

3.4.2 Function `predict_quality()`

- **Decision**: Collect user input, make predictions, and display the result in a popup window.
- **Motivation**: Provide an interactive way for users to use the model and obtain predictions.

3.4.3 Function `save_user_data(features, quality)`

- **Decision**: Save user input and the prediction to a CSV file.
- **Motivation**: Allow data persistence for future analysis or audits.

3.4.4 Function `open_prediction_window()`

- **Decision**: Create a window with entry fields for wine features.
- **Motivation**: Provide a graphical interface for users to input necessary data for prediction.

3.4.5 Functions `show_inventory()` and `show_user_wines()`

- **Decision**: Display inventory data and user-created wines in tree views (`Treeview`).
- **Motivation**: Facilitate the visualization and management of large datasets in the graphical interface.

4 Additional Considerations

4.1 Robustness and Error Handling

Basic error handling is implemented to ensure user inputs are valid and to provide appropriate feedback in case of errors.

4.2 Data Persistence

User data and prediction results are saved in CSV files, allowing for persistence and future analysis.

4.3 Modularity

The code is structured in a modular way to facilitate maintenance and scalability of the project.

5 Conclusion

The project is designed to be a comprehensive system for predicting wine quality, from data loading and preprocessing to user interface for data input and result visualization. The decisions made at each step are based on simplicity, efficiency, and usability, ensuring a clear and effective workflow.