Machine Learning Project Documentation

Model Refinement

1. Overview

The model refinement phase is crucial for improving the performance of the machine learning model. By analyzing initial results and applying targeted techniques, this phase ensures that the model becomes more robust, accurate, and generalizable for real-world applications. This phase involves hyperparameter tuning, algorithm adjustments, feature selection, and optimized cross-validation strategies.

2. Model Evaluation

Initial model evaluation results indicated areas for improvement:

- **Metrics:** The baseline Random Forest model achieved an accuracy of 78%, but recall and precision metrics were slightly unbalanced.
- **Visualizations:** Confusion matrices showed a tendency to misclassify high-poverty regions, while ROC-AUC curves indicated moderate performance with an area under the curve of 0.85.
- **Insights:** The model struggled with imbalanced data, especially in minority classes, necessitating further refinement.

3. Refinement Techniques

The following techniques were applied to enhance the model's performance:

- **Hyperparameter Tuning:** Adjusted the number of estimators, maximum tree depth, and minimum samples per split for the Random Forest and Gradient Boosting models.
- **Algorithm Experimentation:** Conducted trials with additional algorithms such as XGBoost and CatBoost for comparison.
- **Ensemble Methods:** Combined predictions from multiple models using voting classifiers to leverage their collective strengths.

4. Hyperparameter Tuning

Extensive grid and random search techniques were employed:

- **Random Forest:** Tuned parameters such as n_estimators=200, max_depth=15, and min_samples_split=3, which increased accuracy by 3% and improved recall for minority classes.
- **Gradient Boosting:** Optimized learning_rate=0.05 and n_estimators=150 for better precision, resulting in an improved F1-score of 0.81.

Outcome: These adjustments significantly balanced precision and recall, reducing misclassification rates for high-poverty regions.

5. Cross-Validation

The cross-validation strategy was revised:

- Shifted from 5-fold to stratified 10-fold cross-validation to address class imbalance and ensure robust performance evaluation.
- Results showed reduced variance in validation metrics, indicating more reliable model performance across different subsets of data.

6. Feature Selection

Feature importance was evaluated using:

- **Permutation Importance:** Identified key features like GDP per capita, literacy rates, and healthcare expenditure.
- **Recursive Feature Elimination (RFE):** Reduced the feature set from 25 to 15 without sacrificing model accuracy.
- **Impact:** Streamlined feature set improved interpretability and reduced computational complexity.

Test Submission

1. Overview

The test submission phase involved finalizing the refined model and preparing it for evaluation on a held-out test dataset. Steps included data preparation, model application, and performance assessment to ensure readiness for real-world deployment.

2. Data Preparation for Testing

- **Normalization:** Ensured test data was scaled using the same Min-Max Scaler as the training data.
- **Encoding:** Applied consistent one-hot encoding for categorical variables to maintain feature alignment.
- **Imputation:** Addressed any remaining missing values using median imputation.

3. Model Application

The refined model was applied to the test dataset using the following code snippet:

```
# Applying the trained model to test data

predictions = refined_model.predict(X_test)
```

```
probs = refined_model.predict_proba(X_test)
```

4. Test Metrics

- **Accuracy:** 82%
- **Precision:** 0.84
- **Recall:** 0.80
- **F1-Score:** 0.82
- **ROC-AUC:** 0.87

Comparison with training metrics confirmed minimal overfitting, highlighting the generalizability of the model.

5. Model Deployment

No deployment actions have been taken yet.

6. Code Implementation

The key steps for refinement and test submission were implemented as follows:

```
# Hyperparameter tuning example
from sklearn.model_selection import GridSearchCV
param_grid = {
   'n_estimators': [100, 200],
   'max_depth': [10, 15],
   'min_samples_split': [2, 3]
}
grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=10)
grid_search.fit(X_train, y_train)


# Test evaluation
from sklearn.metrics import classification_report, roc_auc_score
print(classification_report(y_test, predictions))
print("ROC-AUC Score:", roc_auc_score(y_test, probs[:, 1]))
```

Conclusion

The refinement and testing phases significantly improved model performance, with the final model achieving an F1-score of 0.82 and an AUC of 0.87. These enhancements make the model a reliable tool for predicting poverty levels. Challenges like class imbalance were mitigated through improved cross-validation and hyperparameter tuning. Future work includes further deployment optimizations and monitoring to adapt to new data.

References

1. Scikit-learn documentation: https://scikit-learn.org/stable/
2. World Bank Development Indicators (WDI) Dataset: https://www.kaggle.com/datasets/kaggle/world-development-indicators