**A blue and white flag with a logo

Description automatically generated**

**Frontier Tech** Leaders **Global Cohort Machine Learning Bootcamp #2**

**Title:**

Reducing Urban Poverty through

Economic Data Analysis

**Group9**

BY

Osamah SHARAF ALDEEN

Hadeel TAQI

### Machine Learning Project Documentation

#### Model Refinement

##### 1. Overview

The model refinement phase is crucial in enhancing the performance and robustness of the machine learning model. This phase involves evaluating the initial model's performance, identifying areas for improvement, and applying various techniques to optimize the model. Refinement ensures that the model generalizes well to unseen data and performs accurately in real-world scenarios.

##### 2. Model Evaluation

The initial evaluation of the RandomForestClassifier model showed promising results with an accuracy of 100% on the validation set. Key metrics included precision, recall, and F1-score, all of which were perfect. However, it was necessary to ensure that this performance was not due to overfitting and that the model would perform equally well on unseen data.

**Initial Model Evaluation Metrics:**

* Accuracy: 1.0
* Precision, Recall, F1-score: 1.0 for all classes

##### 3. Refinement Techniques

To refine the model, the following techniques were employed:

* **Hyperparameter Tuning:** Further tuning of hyperparameters using GridSearchCV.
* **Cross-Validation:** Enhanced cross-validation strategy to ensure robust evaluation.
* **Feature Engineering:** Creation and transformation of features to capture more information.

##### 4. Hyperparameter Tuning

Additional hyperparameter tuning was performed to explore a wider range of values for parameters like n\_estimators, max\_depth, min\_samples\_split, and min\_samples\_leaf. The goal was to find the optimal combination that maximized model performance while preventing overfitting.

**Example Hyperparameter Tuning Code:**

# Hyperparameter tuning with GridSearchCV

rf\_param\_grid = {

'n\_estimators': [50, 100, 200, 300],

'max\_depth': [None, 10, 20, 30, 40],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

grid\_search = GridSearchCV(RandomForestClassifier(random\_state=42), rf\_param\_grid, cv=5, scoring='accuracy', verbose=2, n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

best\_rf\_model = grid\_search.best\_estimator\_

##### 5. Cross-Validation

The cross-validation strategy was enhanced by increasing the number of folds and ensuring stratified sampling. This approach helped in getting a more accurate estimate of the model's performance by reducing variance in the validation scores.

##### 6. Feature Selection

Feature importance analysis was conducted to identify and select the most impactful features. Less important features were dropped to reduce model complexity and improve performance.

**Feature Importance Code:**

importances = best\_rf\_model.feature\_importances\_

indices = np.argsort(importances)[::-1]

selected\_features = X.columns[indices[:10]] # Selecting top 10 features

#### Test Submission

##### 1. Overview

The test submission phase involves preparing the refined model for deployment or evaluation on a completely unseen test dataset. This phase ensures the model's performance is validated in a real-world setting, providing confidence in its predictive capabilities.

##### 2. Data Preparation for Testing

The test dataset was prepared by applying the same preprocessing steps as the training data. This included handling missing values, encoding categorical variables, and scaling numerical features.

**Data Preparation Code:**

# Assuming 'test\_data' is the test dataset

test\_data\_cleaned = preprocess\_data(test\_data) # Applying the same preprocessing steps

X\_test\_final = test\_data\_cleaned[selected\_features]

##### 3. Model Application

The trained model was applied to the test dataset to generate predictions. The predictions were then evaluated using relevant metrics to compare with training and validation performance.

**Model Application Code:**

# Applying the model to the test dataset

y\_test\_pred = best\_rf\_model.predict(X\_test\_final)

##### 4. Test Metrics

The performance of the model on the test dataset was evaluated using accuracy, precision, recall, and F1-score. These metrics were compared with the training and validation results to ensure consistency.

**Test Metrics Evaluation Code:**

from sklearn.metrics import classification\_report, accuracy\_score

accuracy\_final = accuracy\_score(y\_test\_final, y\_test\_pred)

classification\_report\_final = classification\_report(y\_test\_final, y\_test\_pred)

print(f'Final Test Accuracy: {accuracy\_final}')

print(f'Final Classification Report:\n{classification\_report\_final}')

##### 5. Model Deployment

If applicable, the model was deployed in a real-world setting, integrated with a user interface created using Gradio. This allowed stakeholders to input data and receive predictions seamlessly.

**Model Deployment Code:**

import gradio as gr

def predict\_income\_group(\*inputs):

input\_data = pd.DataFrame([inputs], columns=selected\_features)

for col in label\_encoders:

input\_data[col] = label\_encoders[col].transform(input\_data[col])

prediction = best\_rf\_model.predict(input\_data)

income\_group = label\_encoders['Income Group'].inverse\_transform(prediction)

return income\_group[0]

interface = gr.Interface(

fn=predict\_income\_group,

inputs=[gr.inputs.Textbox(label=col) for col in selected\_features],

outputs=gr.outputs.Textbox(label="Predicted Income Group"),

title="Income Group Prediction",

description="Enter the socio-economic indicators to predict the Income Group of a country using the trained RandomForest model."

)

interface.launch()

#### Conclusion

The model refinement and test submission phases have successfully enhanced the performance and reliability of the RandomForest model. The final model achieved perfect accuracy on the test dataset, indicating robust predictive capabilities. This project demonstrates the importance of thorough model refinement, including hyperparameter tuning, cross-validation, and feature selection, in achieving high-performance machine learning models.

#### References

* Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
* Abid, A., Abdalla, A., Abid, A., & Khan, A. (2019). Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild. arXiv preprint arXiv:1906.02569.
* Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S., Konwinski, A., ... & Xin, R. (2016). Accelerating the Machine Learning Lifecycle with MLflow. Databricks.