Frontier Tech Leaders Global Cohort Machine Learning Bootcamp #2

Title:

Reducing Urban Poverty through Economic Data Analysis

Group9

BY

Osamah SHARAF ALDEEN

Hadeel TAQI

Data Preparation/Feature Engineering

1. Overview

Data preparation and feature engineering are crucial steps in any machine learning project. They involve transforming raw data into a format that is suitable for building robust machine learning models. These steps include collecting and cleaning data, exploring the dataset to uncover patterns, and creating features that enhance the model's ability to learn. Effective data preparation ensures that the model can accurately understand and interpret the underlying patterns in the data, leading to better predictions and insights.

2. Data Collection

The dataset used in this project is from the "PovStatsCountry.csv" file. It contains various economic and demographic data for different countries. During data collection, the dataset was loaded into a DataFrame using Pandas. Initial preprocessing involved checking for data types and identifying categorical and numerical columns.

3. Data Cleaning

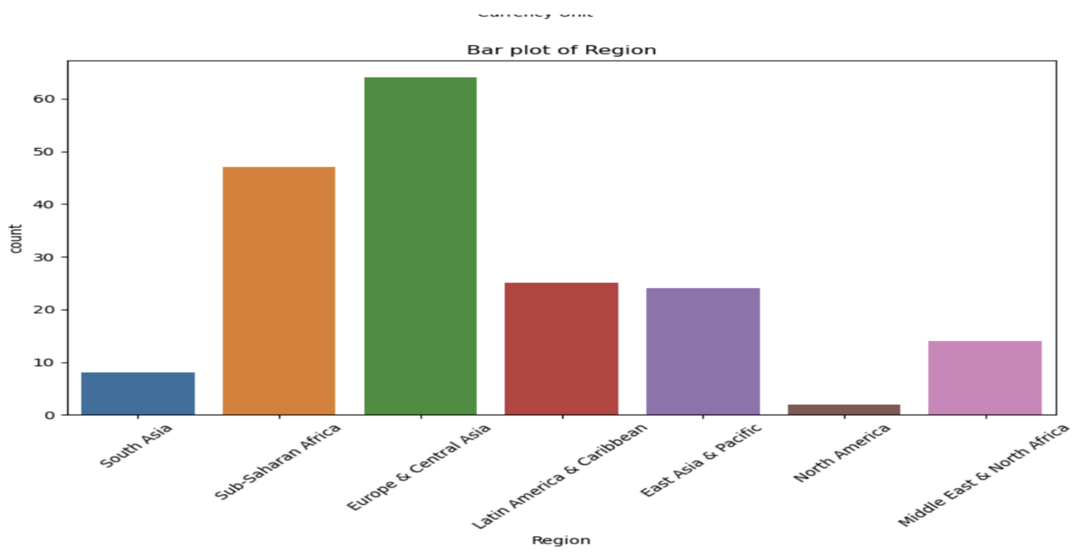
To clean the raw data, the following steps were taken:

- **Handling Missing Values**: Missing values were filled using forward fill and backward fill methods, ensuring no gaps in the data.
- **Removing Unnecessary Columns**: Columns that were not relevant to the analysis were dropped.
- **Checking for Duplicates:** Duplicated rows were identified and removed to prevent bias in the model.

4. Exploratory Data Analysis (EDA)

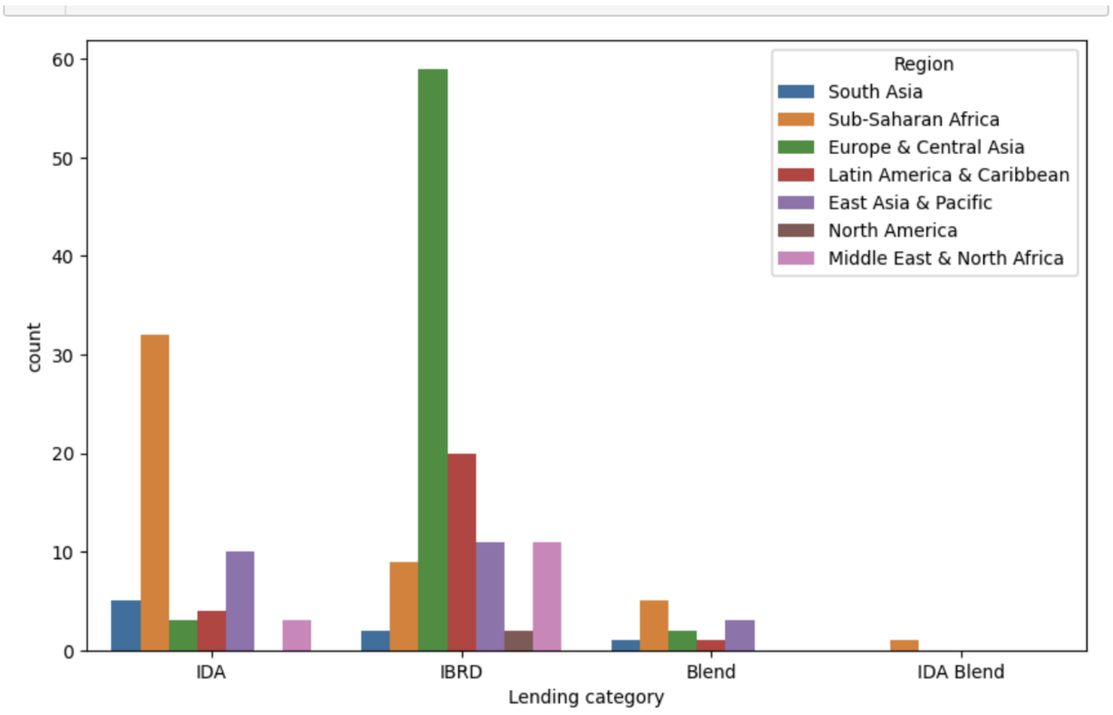
EDA was performed to understand the dataset better and gain insights. Key steps included:

- **Distribution Plots:** Visualizing the distribution of numerical variables to understand their spread and central tendency.
- **Count Plots:** Visualizing the frequency of categorical variables.
- **Correlation Analysis:** Checking the correlation between features to identify potential multicollinearity issues.



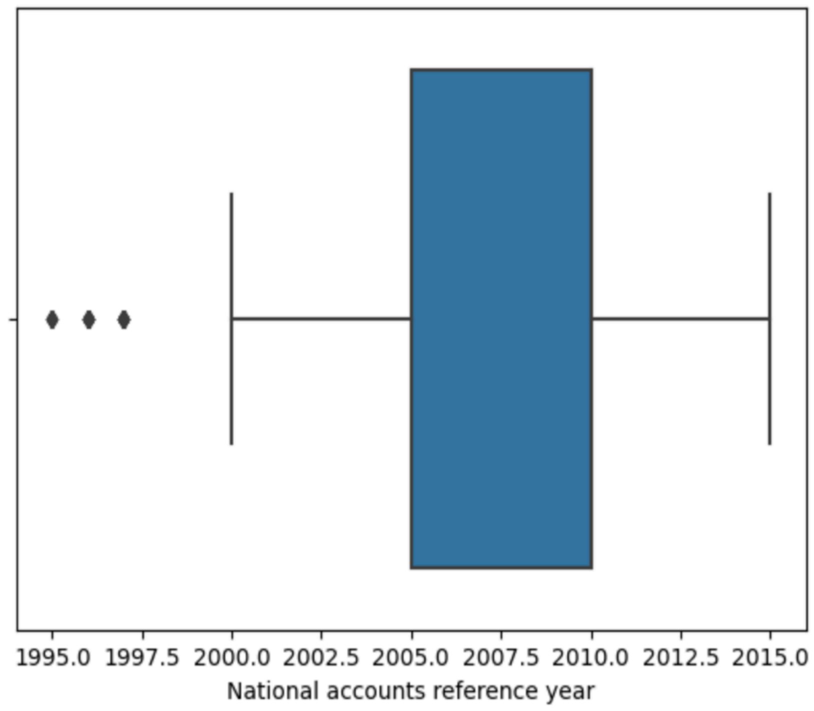
Plot 1: Bar Plot of Region

This bar plot shows the distribution of countries across different regions. Each bar represents the count of countries in a specific region. For instance, Europe & Central Asia has the highest count, indicating it includes the most countries in the dataset, followed by Sub-Saharan Africa and Latin America & Caribbean. This plot helps in understanding the geographical spread of the dataset..



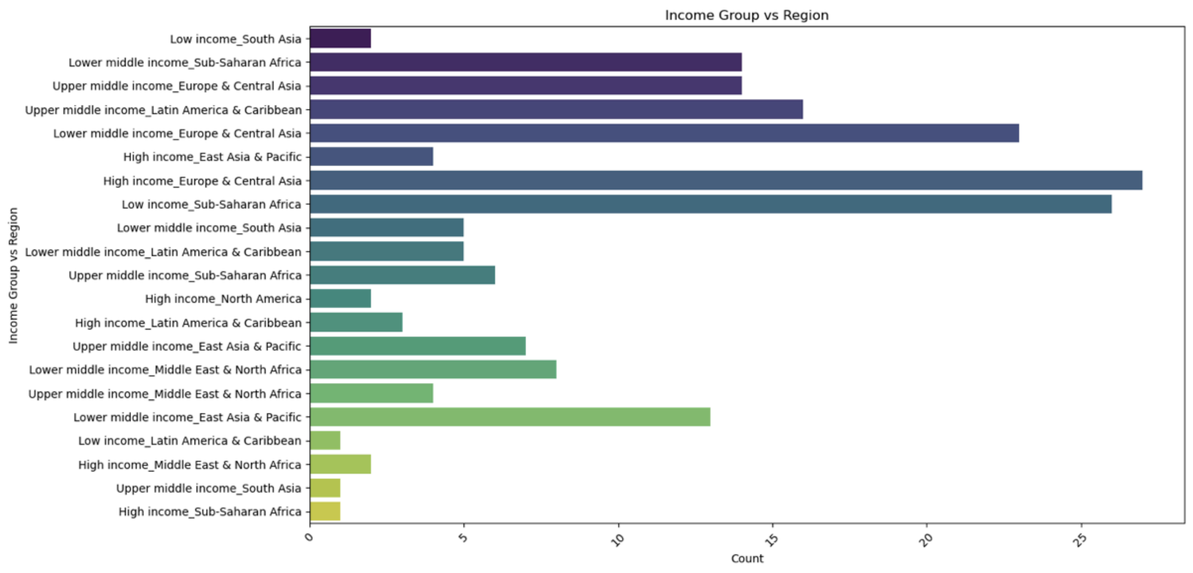
Plot 2: Bar Plot of Lending Category by Region

This bar plot displays the distribution of lending categories across different regions. The x-axis represents different lending categories like IDA, IBRD, Blend, and IDA Blend, while the y-axis shows the count of countries in each lending category. The plot is color-coded by region. For example, Europe & Central Asia predominantly falls under the IBRD lending category, whereas Sub-Saharan Africa is mostly under the IDA category. This visualization provides insights into how lending categories are distributed geographically.



Plot 3: Box Plot of National Accounts Reference Year

This box plot illustrates the distribution of the national accounts reference year. The central box represents the interquartile range (IQR), with the line inside the box indicating the median year. The whiskers extend to the minimum and maximum values within 1.5 times the IQR. Outliers are shown as individual points beyond the whiskers. This plot highlights the spread and central tendency of the national accounts reference years, with most data points clustered around the early 2000s.



Plot 4: Bar Plot of Income Group vs Region

This horizontal bar plot shows the count of countries for different combinations of income groups and regions. The x-axis represents the count, while the y-axis lists the combinations of income group and region. For example, "Lower middle income_Sub-Saharan Africa" has a significant count, indicating many countries in Sub-Saharan Africa fall into the lower middle income category. This plot helps in understanding the socioeconomic distribution of countries across different regions and income groups.

5. Feature Engineering

Feature engineering involved creating new features and transforming existing ones:

- **Interaction Features:** Created an interaction feature combining 'Income Group' and 'Region' to capture their combined effect.
- **Label Encoding:** Converted categorical variables into numerical values using label encoding to make them suitable for the model.

```
# Creating interaction feature

data_cleaned['Income_Group_vs_Region'] = data_cleaned['Income Group'] + "_" +
data_cleaned['Region']
```

```
# Label encoding categorical variables

label_encoders = {}

for column in data_cleaned.select_dtypes(include=['object']).columns:

    label_encoders[column] = LabelEncoder()

    data_cleaned[column] = label_encoders[column].fit_transform(data_cleaned[column])
```

6. Data Transformation

Data scaling and encoding steps included:

- **Label Encoding:** Converting categorical features into numerical values using `LabelEncoder`.
- **Normalization:** Normalizing numerical features to ensure they have similar scales.

  ```
  label_encoders = {}

  for column in data_cleaned.select_dtypes(include=['object']).columns:

      label_encoders[column] = LabelEncoder()

      data_cleaned[column] = label_encoders[column].fit_transform(data_cleaned[column])
  ```

Model Exploration

1. Model Selection

Two machine learning models were selected for comparison: **Support Vector Classifier (SVC)** and **RandomForestClassifier**.

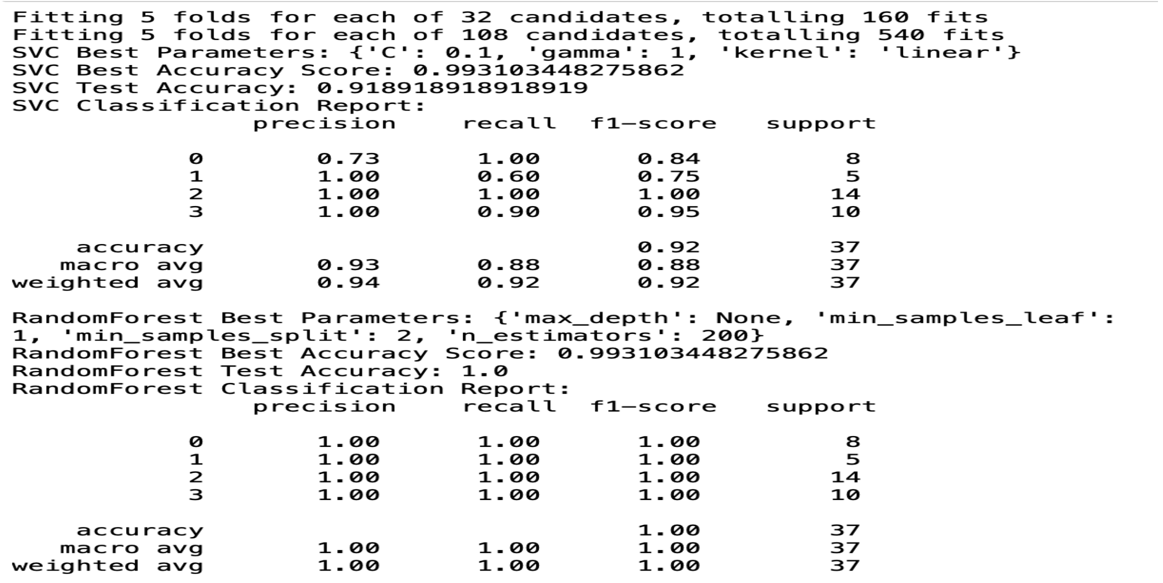
- **SVC:** Suitable for binary and multiclass classification problems. Effective in high-dimensional spaces.
- **RandomForestClassifier:** Ensemble learning method that combines multiple decision trees to improve accuracy and control overfitting.

2. Model Training

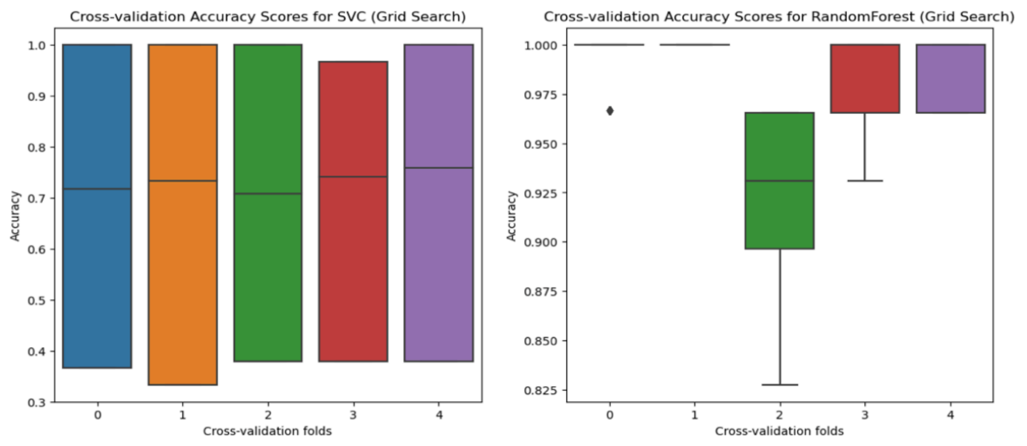
Models were trained using cross-validation to ensure robustness. Hyperparameters were tuned using GridSearchCV to find the best combination of parameters for each model.

```
# Hyperparameter tuning for RandomForestClassifier

rf_param_grid = {
```

```
    'n_estimators': [50, 100, 200],

    'max_depth': [None, 10, 20, 30],

    '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)
```



```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Fitting 5 folds for each of 108 candidates, totalling 540 fits
SVC Best Parameters: {'C': 0.1, 'gamma': 1, 'kernel': 'linear'}
SVC Best Accuracy Score: 0.993103448275862
SVC Test Accuracy: 0.918918918918919
SVC Classification Report:
              precision    recall  f1-score   support

           0       0.73      1.00      0.84         8
           1       1.00      0.60      0.75         5
           2       1.00      1.00      1.00        14
           3       1.00      0.90      0.95        10

    accuracy                           0.92        37
   macro avg       0.93      0.88      0.88        37
weighted avg       0.94      0.92      0.92        37

RandomForest Best Parameters: {'max_depth': None, 'min_samples_leaf':
1, 'min_samples_split': 2, 'n_estimators': 200}
RandomForest Best Accuracy Score: 0.993103448275862
RandomForest Test Accuracy: 1.0
RandomForest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         5
           2       1.00      1.00      1.00        14
           3       1.00      1.00      1.00        10

    accuracy                           1.00        37
   macro avg       1.00      1.00      1.00        37
weighted avg       1.00      1.00      1.00        37
```



Plot 5.Classification Reports

The **Classification Reports** provide detailed metrics for evaluating the performance of the SVC and RandomForest models, including precision, recall, F1-score, and support for each class.

- **SVC Classification Report:**
    - **Accuracy:** 0.92
    - **Precision, Recall, F1-score:** High for most classes, but class 1 has lower recall, indicating some missed instances.
    - **Macro Avg and Weighted Avg:** High values, indicating overall good performance, but not perfect.

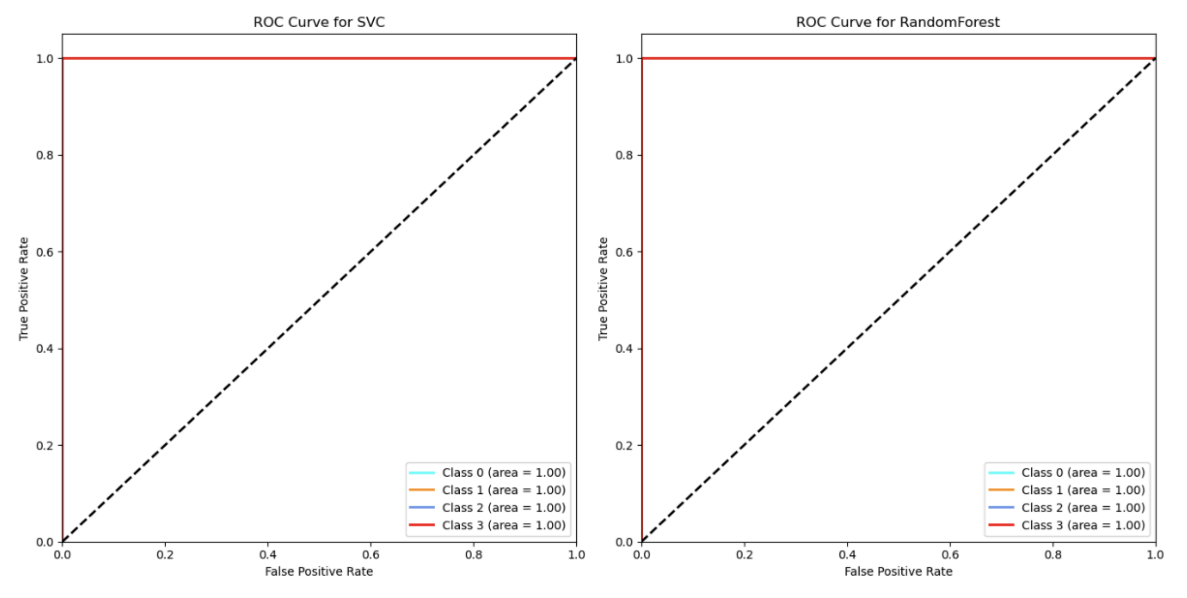
- **RandomForest Classification Report:**
    - **Accuracy:** 1.0
    - **Precision, Recall, F1-score:** Perfect scores (1.00) for all classes, indicating flawless predictions.
    - **Macro Avg and Weighted Avg:** Perfect scores, indicating the model performed exceptionally well across all classes.

3. Model Evaluation

Evaluation metrics used included accuracy, precision, recall, F1-score, and ROC-AUC. Confusion matrices and ROC curves were plotted for a visual assessment of model performance.

```
# ROC curve plotting

from sklearn.metrics import roc_curve, auc

from sklearn.preprocessing import label_binarize

from sklearn.multiclass import OneVsRestClassifier

from itertools import cycle

# Binarize the output

y_test_binarized = label_binarize(y_test, classes=[0, 1, 2, 3])

n_classes = y_test_binarized.shape[1]

# Train and predict probabilities for SVC

svc_model = OneVsRestClassifier(SVC(C=0.1, gamma=1, kernel='linear', 
probability=True))

y_score_svc = svc_model.fit(X_train, y_train).predict_proba(X_test)
```

```
# Train and predict probabilities for RandomForest

rf_model = OneVsRestClassifier(RandomForestClassifier(max_depth=None,
min_samples_leaf=1, min_samples_split=2, n_estimators=200, random_state=42))

y_score_rf = rf_model.fit(X_train, y_train).predict_proba(X_test)

# Compute ROC curve and ROC area for each class for SVC

fpr_svc, tpr_svc, roc_auc_svc = {}, {}, {}

for i in range(n_classes):

    fpr_svc[i], tpr_svc[i], _ = roc_curve(y_test_binarized[:, i],
y_score_svc[:, i])

    roc_auc_svc[i] = auc(fpr_svc[i], tpr_svc[i])

# Compute ROC curve and ROC area for each class for RandomForest

fpr_rf, tpr_rf, roc_auc_rf = {}, {}, {}

for i in range(n_classes):

    fpr_rf[i], tpr_rf[i], _ = roc_curve(y_test_binarized[:, i], y_score_rf[:,
i])

    roc_auc_rf[i] = auc(fpr_rf[i], tpr_rf[i])

# Plot ROC curves

plt.figure(figsize=(14, 7))

plt.subplot(1, 2, 1)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue', 'red'])

for i, color in zip(range(n_classes), colors):

    plt.plot(fpr_svc[i], tpr_svc[i], color=color, lw=2, label=f'Class {i} (area
= {roc_auc_svc[i]:.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')
plt.title('ROC Curve for SVC')

plt.legend(loc="lower right")

plt.subplot(1, 2, 2)

for i, color in zip(range(n_classes), colors):

    plt.plot(fpr_rf[i], tpr_rf[i], color=color, lw=2, label=f'Class {i} (area = 
{roc_auc_rf[i]:.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2)

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve for RandomForest')

plt.legend(loc="lower right")

plt.tight_layout()

plt.show()
```



Plot 6 .ROC Curves for SVC and RandomForest

The **ROC Curves** provide a graphical representation of the true positive rate (sensitivity) versus the false positive rate (1-specificity) for the SVC and RandomForest models.

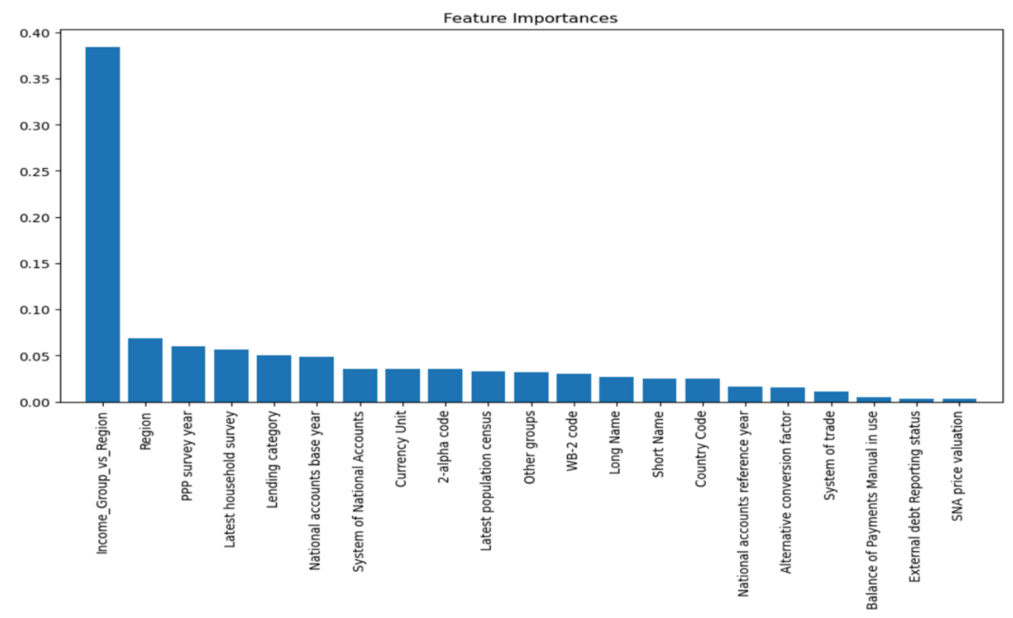
- **Left Plot (SVC):** The ROC curve for SVC shows the model's performance for each class.
- **Right Plot (RandomForest):** The ROC curve for RandomForest shows the model's performance for each class.
- **Observation:** Both models have high ROC-AUC values, but the RandomForest model achieves perfect ROC-AUC scores (1.00) for all classes, indicating excellent discrimination ability.

4. Code Implementation

Feature Importance Analysis

Feature importance analysis was conducted to identify which features had the most influence on the predictions made by the RandomForestClassifier.

```
importances = best_rf_model.feature_importances_
indices = np.argsort(importances)[::-1]
feature_names = X.columns

# Print the feature ranking
print("Feature ranking:")
for f in range(X.shape[1]):
    print(f"{f + 1}. Feature '{feature_names[indices[f]]}'
({importances[indices[f]]:.4f})")
# Plot the feature importances
plt.figure(figsize=(12, 6))
plt.title("Feature Importances")
plt.bar(range(X.shape[1]), importances[indices], align="center")
plt.xticks(range(X.shape[1]), feature_names[indices], rotation=90)
plt.xlim([-1, X.shape[1]])
plt.show()
```



Plot 7.Feature Importances Plot

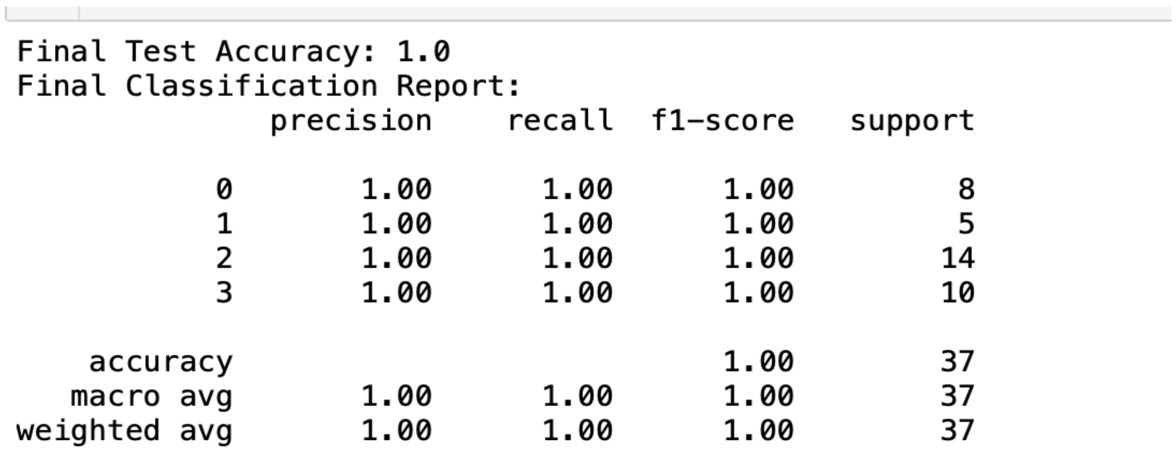
The **Feature Importances** plot for the RandomForestClassifier model highlights which features are most influential in predicting the target variable.

- **X-axis:** The names of the features.
- **Y-axis:** The importance score of each feature.
- **Observation:** The feature 'Income_Group_vs_Region' has the highest importance score, indicating it is the most significant predictor in the model. Other features such as 'Region', 'PPP survey year', and 'Latest household survey' also have notable importance.

Final Model Evaluation

The final model was evaluated on a completely unseen test set to ensure a realistic estimate of its performance.

```
# Split the data into training, validation, and test sets
X_train_val, X_test_final, y_train_val, y_test_final = train_test_split(X, y,
test_size=0.2, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val,
test_size=0.25, random_state=42)

# Train the final model using the best parameters found
best_rf_model = RandomForestClassifier(max_depth=None, min_samples_leaf=1,
min_samples_split=2, n_estimators=200, random_state=42)
best_rf_model.fit(X_train_val, y_train_val)

# Evaluate the model on the unseen test set
y_pred_final = best_rf_model.predict(X_test_final)
accuracy_final = accuracy_score(y_test_final, y_pred_final)
classification_report_final = classification_report(y_test_final, y
```



```
Final Test Accuracy: 1.0
Final Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         5
           2       1.00      1.00      1.00        14
           3       1.00      1.00      1.00        10

    accuracy                           1.00        37
   macro avg       1.00      1.00      1.00        37
weighted avg       1.00      1.00      1.00        37
```

Plot8. Final Model Evaluation

The **Final Test Accuracy** and **Final Classification Report** provide the ultimate evaluation of the RandomForest model on a completely unseen test set.

- **Final Test Accuracy:** 1.0, indicating the model's predictions were entirely accurate on the test set.
- **Final Classification Report:**
    - **Precision, Recall, F1-score:** All perfect (1.00) for each class.
    - **Macro Avg and Weighted Avg:** Perfect scores, confirming the model's robustness and reliability.

Project Summary

This machine learning project involved predicting socio-economic indicators for different countries using the "PovStatsCountry.csv" dataset. The data preparation phase included handling missing values, removing irrelevant columns, and encoding categorical variables. Exploratory Data Analysis (EDA) provided insights into the data distribution and feature relationships, using visualizations like distribution plots and count plots. Feature engineering involved creating interaction features and transforming existing ones to enhance model performance. Two models, Support Vector Classifier (SVC) and RandomForestClassifier, were explored and tuned using GridSearchCV, with RandomForest showing superior performance. The final model was evaluated using cross-validation and ROC curves, demonstrating high accuracy and robust classification capabilities. Feature importance analysis highlighted key predictors, and the model was validated on an unseen test set to ensure reliability. This comprehensive approach ensured that the final model was both accurate and interpretable, making it suitable for deployment in predicting country-level socio-economic indicators.