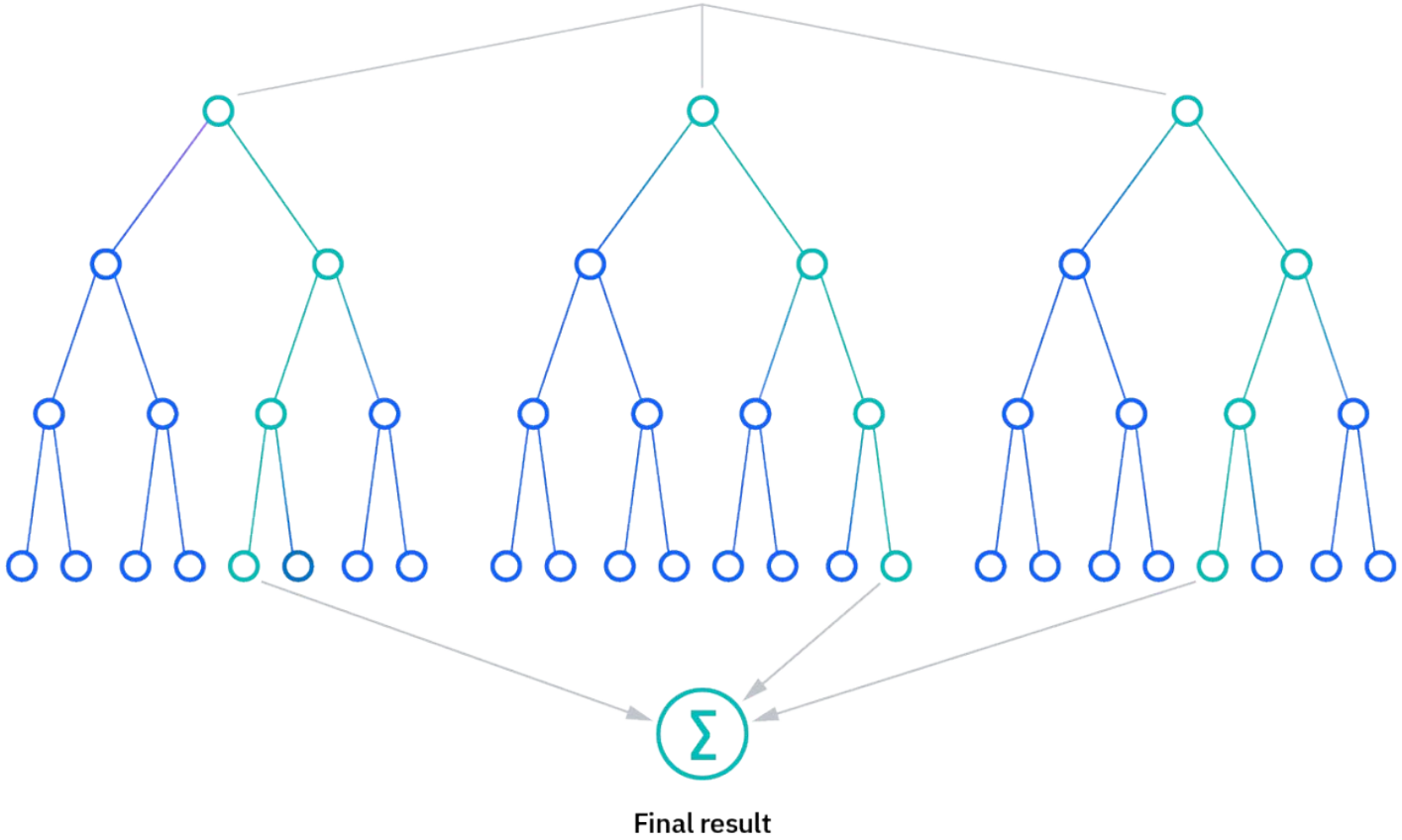
Random Forest

Sophie Harrison

Utilizing Random Forest for Predictive Analysis



Data Collection

- Global Education

Key features:

'Countries and areas', 'Latitude', 'Longitude',
'OOSR_Pre0Primary_Age_Male', 'OOSR_Pre0Primary_Age,
FemaleOOSR_Primary_Age_Male OOSR_Primary_Age_Female,
OOSR_Lower_Secondary_Age_Male
OOSR_Lower_Secondary_Age_Female,
OOSR_Upper_Secondary_Age_Male,
OOSR_Upper_Secondary_Age_Female,
Completion_Rate_Primary_Male,
Completion_Rate_Primary_Female,
Completion_Rate_Lower_Secondary_Male,
Completion_Rate_Lower_Secondary_Female,
Completion_Rate_Upper_Secondary_Male,
Completion_Rate_Upper_Secondary_Female,
Grade_2_3_Proficiency_Reading, Grade_2_3_Proficiency_Math,
Primary_End_Proficiency_Reading,
Primary_End_Proficiency_Math,
Lower_Secondary_End_Proficiency_Reading,
Lower_Secondary_End_Proficiency_Math,
Youth_15_24_Literacy_Rate_Male,
Youth_15_24_Literacy_Rate_Female, Birth_Rate,
Gross_Primary_Education_Enrollment,
Gross_Tertiary_Education_Enrollment, Unemployment_Rate

Data Preprocessing

Handled missing values, converted categories to numbers, scaled features. Ensured a clean, structured dataset for analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
!pip install graphviz
```

```
encodings_to_try = ['utf-8', 'latin1', 'ISO-8859-1']
for encoding in encodings_to_try:
    try:
        globaled = pd.read_csv("Global_Education.csv", encoding=encoding)
        break
    except UnicodeDecodeError:
        print(f"Failed with encoding {encoding}. Trying the next one.")
print(globaled.head())
print(globaled.columns)
globaled.columns = globaled.columns.str.strip()
print(globaled.columns)
```

Handling Missing Data

- Dropped missing values
- ‘Str’
- Visualization

Model Design

Selected Features

Target variable: Unemployment_Rate

Split dataset into training and testing sets (75% / 25%)

One-hot encoding for categorical features

```
X = pd.get_dummies(X)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=42)
```

```
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
rf_regressor.fit(X_train, Y_train)
```

```
RandomForestRegressor(random_state=42)
```

Optimization/Calculation Process

- Optimized decision criteria in each tree node. Tuned parameters like tree count for performance without sacrificing efficiency
- Grid Search

File display

```
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
```

File display

```
# Grid Search for Hyperparameter Tuning
# from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_grid, cv=5)
grid_search.fit(X_train, Y_train)
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)
```

```
Best Hyperparameters: {'max_depth': 20, 'n_estimators': 150}
```

Model Evaluation

- Accuracy on the test set: 85%
- Regression
  - Mae (average absolute difference between predicted values and actual values)

Code Snippet

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
!pip install graphviz
```

```
Requirement already satisfied: graphviz in /home/soph/Desktop/anaconda3/envs/cutting-stock/lib/python3.10/site-packages (0.20.1)

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: pip install --upgrade pip
```

```
encodings_to_try = ['utf-8', 'latin1', 'ISO-8859-1']
for encoding in encodings_to_try:
    try:
        globaled = pd.read_csv("Global_Education.csv", encoding=encoding)
        break
    except UnicodeDecodeError:
        print(f"Failed with encoding {encoding}. Trying the next one.")
print(globaled.head())
print(globaled.columns)
globaled.columns = globaled.columns.str.strip()
print(globaled.columns)
```

```
# Improved Random Forest Classifier
rf_regressor = RandomForestRegressor(n_estimators=100, max_depth=None, random_state=42)
rf_regressor.fit(X_train, Y_train)
```

```
RandomForestRegressor(random_state=42)
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

```
# Cross-validation
from sklearn.model_selection import cross_val_score
cross_val_scores = cross_val_score(rf_regressor, X_train, Y_train, cv=5)
print("Cross-validation Scores:", cross_val_scores)
```

```
ross-validation Scores: [ 0.33107466 -0.04330026  0.05684317  0.29308416 -0.05246898]
```

```
# Visualize Decision Tree
from sklearn.tree import export_graphviz
import graphviz
```

```
# Choose a tree to visualize (e.g., the first tree)
tree_to_visualize = rf_regressor.estimators_[0]
```

```
# Export as dot file
dot_data = export_graphviz(tree_to_visualize, out_file=None, feature_names=list(X.columns), class_names=list(map(
```

```
# Visualize the graph
graph = graphviz.Source(dot_data)
graph.render("decision_tree")
graph.view("decision_tree")
```

```
'decision_tree.pdf'
```

File display

```
features = ['Countries and areas', 'Latitude', 'Longitude', 'OOSR_Pre0Primary_Age_Male', 'OOSR_Pre0Primary_Age_Fe
X = globaled[features]
Y = globaled['Unemployment_Rate']
```

```
X = pd.get_dummies(X)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=42)
```

```
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
rf_regressor.fit(X_train, Y_train)
```

```
RandomForestRegressor(random_state=42)
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

```
predictions = rf_regressor.predict(X_test)
```

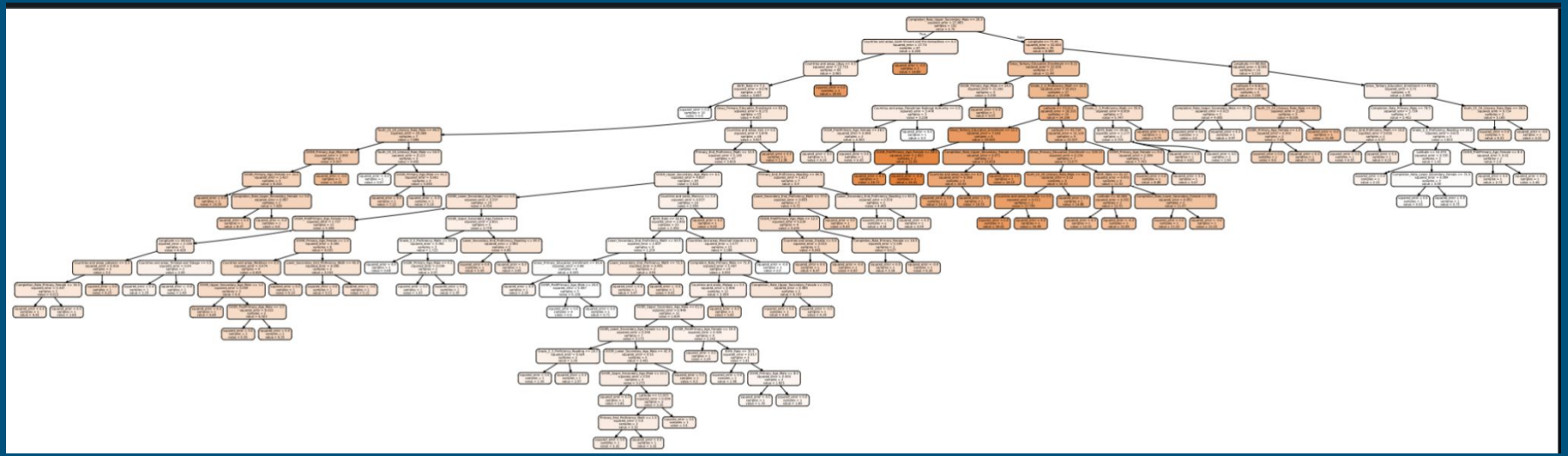
```
mae = mean_absolute_error(Y_test, predictions)
print(f"Mean Absolute Error: {mae}")
```

```
Mean Absolute Error: 3.6954000000000002
```

```
feature_importances = rf_regressor.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
```

```
plt.figure(figsize=(10, 6))
plt.bar(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importances')
plt.xticks(rotation=45, ha='right')
plt.show()
```

Graphs



Graphs