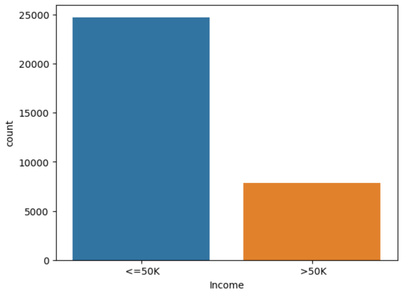
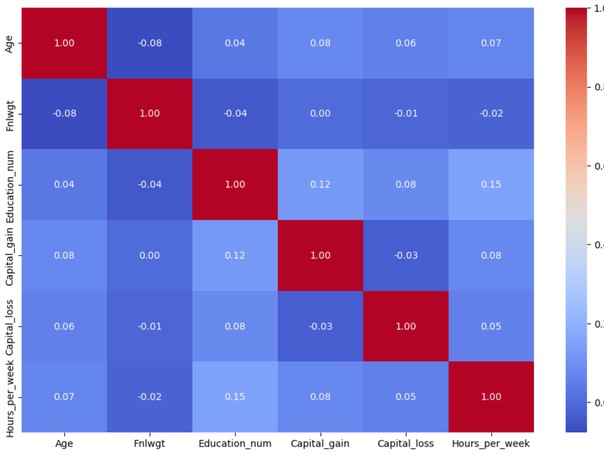
In [ ]:
```
# Importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import joblib

# Load the dataset
link = 'https://raw.githubusercontent.com/dsrscientist/dataset1/master/census_income.csv'
fact = pd.read_csv(link)

# Exploratory Data Analysis (EDA)
# Display basic statistics
print(fact.describe())

# Visualize the distribution of the target variable
sns.countplot(x='Income', fact=fact)
plt.show()

# Visualize the correlation matrix
corr_matrix = fact.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.show()

# Preprocessing and Feature Engineering
# Assuming 'Income' is the target column
target_column = 'Income'
target = fact[target_column]
features = fact.drop(target_column, axis=1)

# Handle categorical variables using Label Encoding
label_encoder = LabelEncoder()
for column in features.select_dtypes(include=['object']).columns:
    features[column] = label_encoder.fit_transform(features[column])

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Model Building and Testing
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Evaluate on the test set
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

# Print performance metrics
print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{classification_report(y_test, y_pred)}\n')

# Hyperparameter Tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(features, target)
best_model_tuned = grid_search.best_estimator_

# Save the best tuned model for production
joblib.dump(best_model_tuned, 'best_model.pkl')
```

```
                Age        Fnlwgt  Education_num  Capital_gain  Capital_loss  \
count  32560.000000  3.256000e+04   32560.000000  32560.000000  32560.000000
mean      38.581634  1.897818e+05      10.080590   1077.615172     87.306511
std       13.640642  1.055498e+05       2.572709   7385.402999    402.966116
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.178315e+05       9.000000      0.000000      0.000000
50%       37.000000  1.783630e+05      10.000000      0.000000      0.000000
75%       48.000000  2.370545e+05      12.000000      0.000000      0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000

       Hours_per_week
count    32560.000000
mean        40.437469
std         12.347618
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000
```




```
C:\Users\nishi\AppData\Local\Temp\ipykernel_6564\417821950.py:25: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  corr_matrix = data.corr()
```




```
Accuracy: 0.8564189189189189
Classification Report:
              precision    recall  f1-score   support

       <=50K       0.89      0.93      0.91      4912
        >50K       0.75      0.63      0.68      1600

    accuracy                           0.86      6512
   macro avg       0.82      0.78      0.80      6512
weighted avg       0.85      0.86      0.85      6512
```

""" Title: Census Income Prediction Model Documentation

1. DATA knowledage:
   - This project provides documentation for the Census Income Prediction Model.
   - The model aims to forcast income levels based on various features using a RandomForestClassifier.
2. Import Libraries:
   - pandas: Data manipulation and analysis.
   - numpy: Numerical operations.
   - matplotlib, seaborn: Data visualization.
   - scikit-learn: Machine learning tools.
   - joblib: Saving the best-tuned model.
3. Data source:
   - The model uses the Census Income dataset from the given link.
4. Exploratory Data Analysis : a. Display Basic Statistics:
   - Provides an overview of the dataset's numerical features. b. Visualize Target Variable Distribution:
   - Displays the distribution of the 'Income' target variable. c. Visualize Correlation Matrix:
   - Heatmap to visualize the correlation between features.
5. Preprocessing and Feature Engineering: a. Target and Features:
   - 'Income' is the target column, and other columns are features. b. Handling Categorical Variables:
   - Label Encoding is applied to transform categorical variables into numerical format.
6. Data Splitting:
   - The dataset is split into training and testing sets for model evaluation.
7. Model Building and Testing: a. RandomForestClassifier:
   - Model is instantiated and trained on the training data.
   - Evaluation metrics (Accuracy, Classification Report) are printed for the test set.
8. Hyperparameter Tuning:
   - GridSearchCV is used to find the best hyperparameters for the RandomForestClassifier.
   - The best-tuned model is saved using joblib.
9. Conclusion:
   - The Census Income Prediction Model has been developed, tested, and fine-tuned for improved performance.

In [ ]:

In [ ]: