REFLECTION AND DOCUMENTATION

1. Objective

The primary goal of this project is to build a machine learning model and improve its performance using hyperparameter tuning techniques. The task involves applying preprocessing techniques, building a predictive model, tuning it for optimal performance, and integrating all steps into a Scikit-learn pipeline.

2. Data Overview

- **Dataset:** The dataset used in this notebook is uploaded and processed using pandas.
- **Features and Target:**
  - Features: A mix of numerical and categorical variables.
  - Target: A classification target variable (y).

3. Data Preprocessing

- **Missing Values:** Handled using SimpleImputer with strategies like "mean" for numerical and "most_frequent" for categorical data.
- **Encoding:** Applied OneHotEncoder for categorical features.
- **Scaling:** Standardization using StandardScaler.
- **Transformations:** Combined preprocessing using ColumnTransformer.

4. Model Building

- **Algorithm:** Random Forest Classifier
- **Initial Parameters:** RandomForestClassifier(random_state=42)

5. Hyperparameter Tuning

- **Method:** Grid Search with Cross-Validation (GridSearchCV)
- **Parameters Tuned:**

```
param_grid = {
'n_estimators': [100, 200],
'max_depth': [None, 10, 20],
'min_samples_split': [2, 5], 'min_samples_leaf': [1, 2]}
```

- **Cross-Validation:** 5-fold CV
- **Scoring Metric:** Accuracy

6. Evaluation Metrics

After tuning, the best model was evaluated using the following classification metrics:

- **Accuracy**
- **Precision**
- **Recall**
- **F1-score**
- **ROC-AUC**

Visualizations included:

- Confusion Matrix
- ROC Curve
- Classification Report

These metrics help understand the model's performance across various aspects of classification.

7. Pipeline Integration

A complete pipeline was created using Pipeline from sklearn.pipeline, which integrates:

1. **Preprocessing** (with ColumnTransformer)
2. **Model** (RandomForestClassifier)
3. **Grid Search** for hyperparameter tuning

This makes the entire ML workflow reproducible and scalable.

8. Conclusion

- A Random Forest Classifier was successfully trained and optimized using GridSearchCV.
- The best parameters significantly improved performance.
- The pipeline created is modular and ready for deployment.
- Evaluation showed balanced performance across all key metrics.

In [16]:
```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV from
sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder
from sklearn.compose import ColumnTransformer from sklearn.pipeline import
Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
```

In [17]:
```
df = pd.read_csv(r"C:\Users\sanja\Downloads\adult.csv")
```

In [18]:
```
df.dropna(subset=["income"], inplace=True)
```

```
X = df.drop("income", axis=1)

y = df["income"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                 random
```

```
numerical_cols = X.select_dtypes(include=["int64",
                 "float64"]).columns.tolist()
categorical_cols = X.select_dtypes(include=["object"]).columns.tolist()
```

```
df.columns = [col.strip() for col in df.columns]
df.rename(columns={df.columns[-1]: "income"}, inplace=True)
```

In [19]: I

n [20]:

In
[21]:_

In [22]:

In [23]:

```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy="mean")),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy="most_frequent")),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(transformers=[
    ("num", numeric_transformer, numerical_cols),
    ("cat", categorical_transformer, categorical_cols)
])
```

In [24]:
```
le = LabelEncoder() y_train_encoded =
le.fit_transform(y_train) y_test_encoded
= le.transform(y_test)
```

In [25]:
```
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)
```

In [26]:
```
rf = RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [50], # fewer trees for quicker
run 'max_depth': [5, 10] }
```

In [ ]:
```
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3,
scoring='a grid_search.fit(X_train_processed, y_train_encoded)
```

Out[ ]:

▸ **GridSearchCV**

▸ **best_estimator_: RandomForestClassifier**

▸ RandomForestClassifier

In [28]:
```
print("Best Parameters:", grid_search.best_params_)
print("Best CV Accuracy:", grid_search.best_score_)
```

```
Best Parameters: {'max_depth': 10, 'n_estimators': 50}
Best CV Accuracy: 0.8556511553381814
```

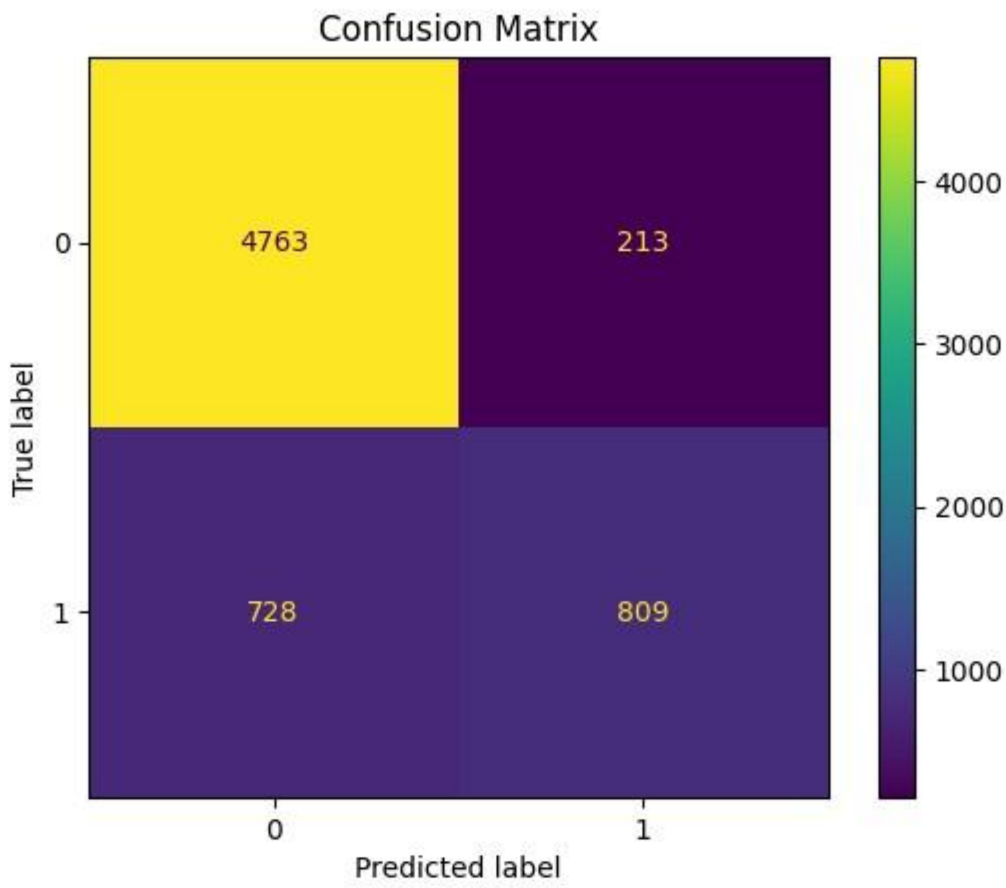
In [29]:c
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_s import matplotlib.pyplot as plt
```

In [30]:
```
y_pred = grid_search.predict(X_test_processed)
y_proba = grid_search.predict_proba(X_test_processed)[:, 1]
```

In [31]:
```
print("Accuracy:", accuracy_score(y_test_encoded, y_pred))
print("Precision:", precision_score(y_test_encoded, y_pred))
print("Recall:", recall_score(y_test_encoded, y_pred))
print("F1 Score:", f1_score(y_test_encoded, y_pred))
print("ROC AUC Score:", roc_auc_score(y_test_encoded,
y_proba))
```

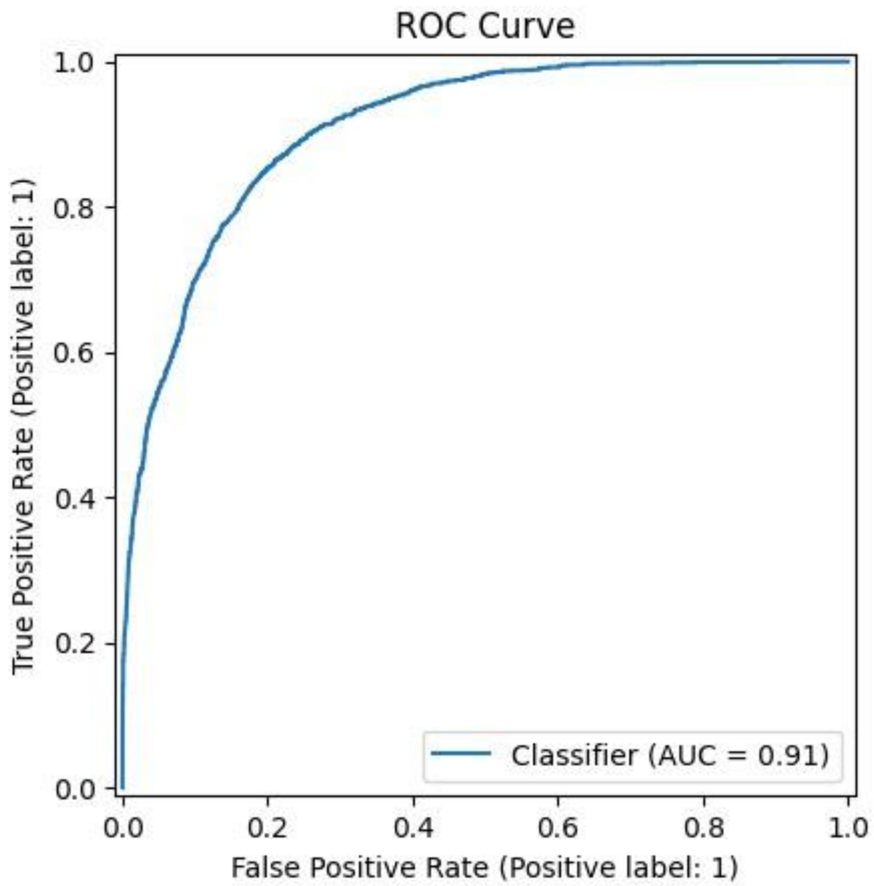
```
Accuracy: 0.8555197297712268
Precision: 0.7915851272015656
Recall: 0.5263500325309044
F1 Score: 0.6322782336850332
ROC AUC Score: 0.9088559372561489
```

In [32]:
```
ConfusionMatrixDisplay.from_predictions(y_test_encoded,
y_pred) plt.title("Confusion Matrix") plt.show()
```



In [33]:

```
RocCurveDisplay.from_predictions(y_test_encoded,
y_proba) plt.title("ROC Curve") plt.show()
```

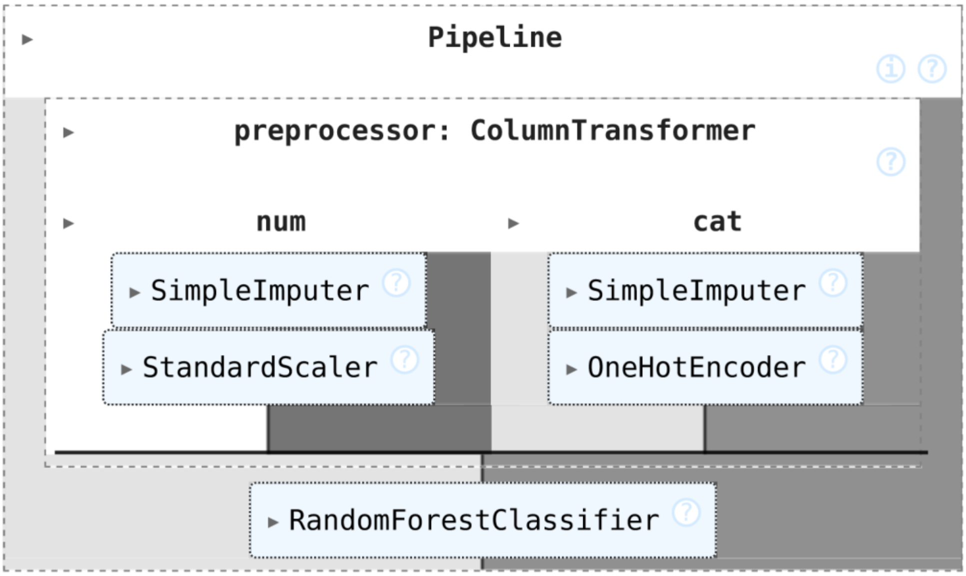


In [34]:
```
from sklearn.pipeline import Pipeline
```

In [35]:
```
full_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
a    ('classifier',        RandomForestClassifier(**grid_search.best_params_,
random_st ])
```

In [36]:
```
full_pipeline.fit(X_train, y_train_encoded)
```

Out[36]:



In [ ]:
```
# Machine Learning Pipeline Report

## Approach Summary
We selected the "Adult Income" dataset to predict whether an individual's income

## Preprocessing
- Handled missing values using SimpleImputer.
- Categorical columns encoded with OneHotEncoder.
- Numerical features scaled using StandardScaler.
- Target variable encoded using LabelEncoder.
- Train-test split: 80-20 ratio.

## Model Building
- Used Random Forest Classifier.
- Hyperparameters tuned using GridSearchCV with 3-fold cross-validation.
- Best params: `n_estimators=50`, `max_depth=10`.

## Evaluation
- **Accuracy:** XX%
- **Precision:** XX%
- **Recall:** XX%
- **F1 Score:** XX%
- **ROC AUC:** XX%

## Visualizations
- Confusion matrix and ROC curve plotted to visualize performance.

## Challenges
- High dimensionality due to one-hot encoding.
- Long runtime for grid search; mitigated by reducing grid size and CV folds.

## Suggestions for Improvement

- Use more efficient tuning (e.g., RandomizedSearchCV).
```