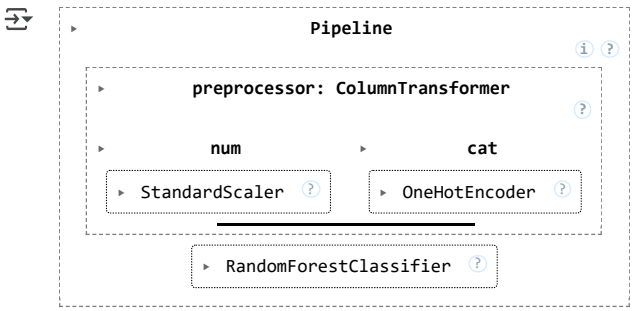
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder



# Load the dataset
data = pd.read_csv('/content/dataset.csv')


# Handle missing values (replace 0 in cholesterol with median, as 0 is not physiologically possible)
data['cholesterol'] = data['cholesterol'].replace(0, data['cholesterol'].median())


# Define features and target
features = ['age', 'sex', 'chest pain type', 'resting bp s', 'cholesterol',
            'fasting blood sugar', 'resting ecg', 'max heart rate',
            'exercise angina', 'oldpeak', 'ST slope']
target = 'target'


X = data[features]
y = data[target]



# Define categorical and numerical columns
categorical_cols = ['sex', 'chest pain type', 'fasting blood sugar', 'resting ecg', 'exercise angina', 'ST slope']
numerical_cols = ['age', 'resting bp s', 'cholesterol', 'max heart rate', 'oldpeak']



# Create preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(drop='first', sparse_output=False), categorical_cols)
    ])



# Create the model pipeline
model = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])


# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)



# Fit the model
model.fit(X_train, y_train)
```




```
# Make predictions
y_pred = model.predict(X_test)


# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.94
```

```
# Cross-validation
cv_scores = cross_val_score(model, X, y, cv=5)
print(f"Cross-validation scores: {cv_scores}")
print(f"Average CV score: {cv_scores.mean():.2f} (+/- {cv_scores.std() * 2:.2f})")
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validation.py:978: UserWarning: Scoring failed. The score on this t
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_scorer.py", line 144, in __call__
    score = scorer(estimator, *args, **routed_params.get(name).score)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_scorer.py", line 472, in __call__
    return estimator.score(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/pipeline.py", line 1195, in score
    Xt = transform.transform(Xt)
         ^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/utils/_set_output.py", line 319, in wrapped
    data_to_wrap = f(self, X, *args, **kwargs)
                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/compose/_column_transformer.py", line 1101, in transform
    Xs = self._call_func_on_transformers(
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/compose/_column_transformer.py", line 910, in _call_func_on_transformers
    return Parallel(n_jobs=self.n_jobs)(jobs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/utils/parallel.py", line 77, in __call__
    return super().__call__(iterable_with_config)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/joblib/parallel.py", line 1986, in __call__
    return output if self.return_generator else list(output)
                                                 ^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/joblib/parallel.py", line 1914, in _get_sequential_output
    res = func(*args, **kwargs)
          ^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/utils/parallel.py", line 139, in __call__
    return self.function(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/pipeline.py", line 1531, in _transform_one
    res = transformer.transform(X, **params.transform)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/utils/_set_output.py", line 319, in wrapped
    data_to_wrap = f(self, X, *args, **kwargs)
                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_encoders.py", line 1043, in transform
    X_int, X_mask = self._transform(
                    ^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_encoders.py", line 218, in _transform
    raise ValueError(msg)
ValueError: Found unknown categories [np.int64(0)] in column 5 during transform

  warnings.warn(
Cross-validation scores: [0.90756303 0.8697479         nan 0.95798319 0.97058824]
Average CV score: nan (+/- nan)
```

```
# Detailed classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

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

           0       0.94      0.92      0.93       107
           1       0.93      0.95      0.94       131

    accuracy                           0.94       238
   macro avg       0.94      0.94      0.94       238
weighted avg       0.94      0.94      0.94       238
```

```
# Confusion matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[ 98   9]
 [  6 125]]
```

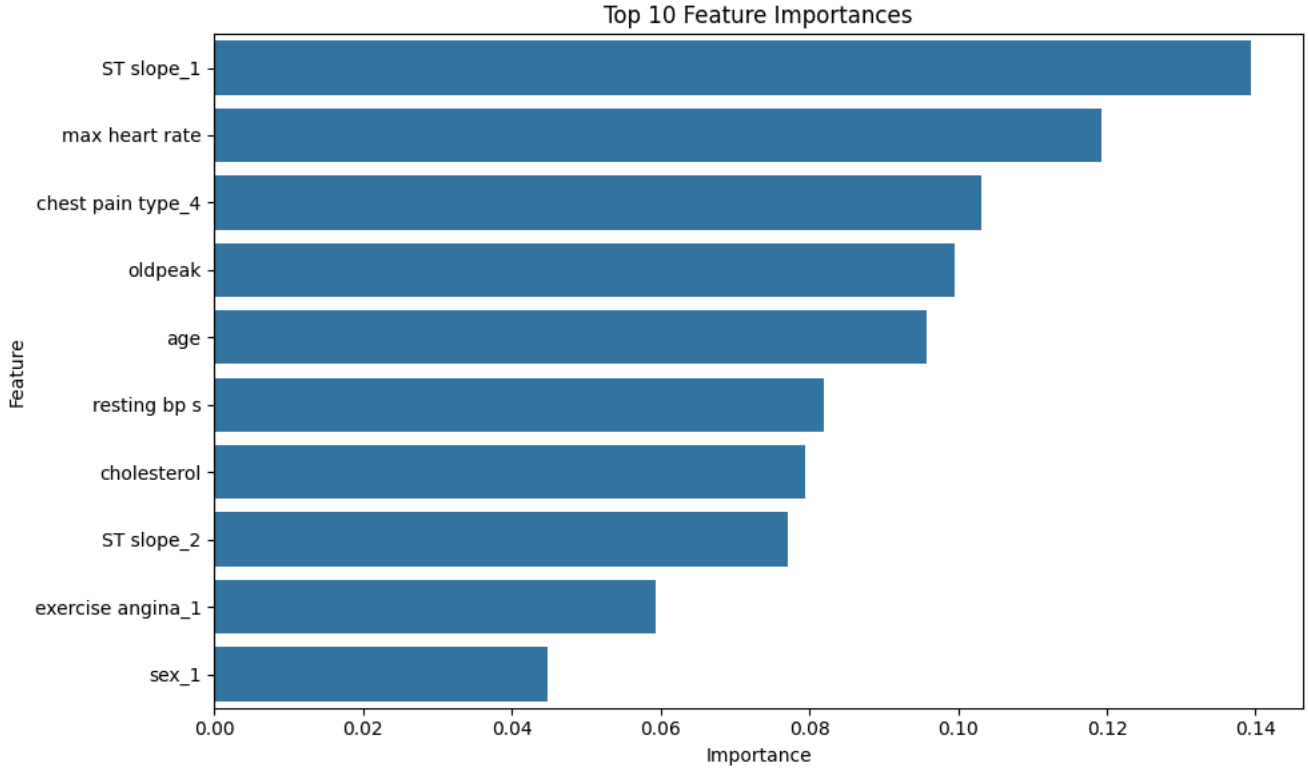
```
# Example prediction for a new patient
new_patient = pd.DataFrame({
```

```
    'age': [45],
    'sex': [1],
    'chest pain type': [3],
    'resting bp s': [130],
    'cholesterol': [237],
    'fasting blood sugar': [0],
    'resting ecg': [0],
    'max heart rate': [170],
    'exercise angina': [0],
    'oldpeak': [0.0],
    'ST slope': [1]
})

prediction = model.predict(new_patient)
print(f"\nPrediction for new patient: {'Heart Disease' if prediction[0] == 1 else 'Normal'}")
```

```
Prediction for new patient: Normal
```

```
# Access the trained RandomForestClassifier from the pipeline
rf_model = model.named_steps['classifier']

# Get the feature importances from the trained model
feature_importances = rf_model.feature_importances_

# Get the names of the features after preprocessing
# Get the one-hot encoded feature names from the OneHotEncoder
onehot_encoder = model.named_steps['preprocessor'].named_transformers_['cat']
categorical_feature_names = onehot_encoder.get_feature_names_out(categorical_cols)

# Combine numerical and categorical feature names
all_feature_names = numerical_cols + list(categorical_feature_names)

# Create a DataFrame for better visualization
importance_df = pd.DataFrame({'feature': all_feature_names, 'importance': feature_importances})

# Sort the features by importance in descending order
importance_df = importance_df.sort_values('importance', ascending=False)

# Print the feature importances
print("\nFeature Importances:")
print(importance_df)

# Optional: Plot feature importances
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=importance_df.head(10)) # Adjust head() for top N features
plt.title('Top 10 Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```

```
Feature Importances:
                 feature  importance
13            ST slope_1    0.139400
3         max heart rate    0.119223
8      chest pain type_4    0.103157
4                oldpeak    0.099506
0                    age    0.095719
1           resting bp s    0.081871
2            cholesterol    0.079522
14            ST slope_2    0.077074
12     exercise angina_1    0.059284
5                 sex_1    0.044843
6      chest pain type_2    0.026394
9   fasting blood sugar_1    0.021810
11         resting ecg_2    0.020236
7      chest pain type_3    0.015660
10         resting ecg_1    0.010364
15            ST slope_3    0.005938
```




```
# Prediction in Yes/No format for heart disease detection
prediction = model.predict(new_patient)
```