FinalProject

October 18, 2024

1 IML CSCA 5622 Final Project

1.1 Analysis of the "Adult" dataset from UC Irvine Machine Learning Repository (https://archive.ics.uci.edu/dataset/2/adult)

Jupyter Notebook by Lucas Pozzi de Souza

1.2 Sources:

- Previous analysis by Chet Lemon, Chris Zelazo, Kesav Mulakaluri (https://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf)
- Helper script from the book "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition" by Aurélien Géron (https://books.apple.com/us/book/hands-on-machine-learning-with-scikit-learn-keras/id6443685464)
- Perplexity AI - code troubleshooting
- Khan Academy Migo - review, analysis and guidance

1.3 1. Four versions of the dataset and data hygiene

- X0: a full one,
- X1: one without the columns 'fnlwgt', 'relationship', 'capital-gain', 'capital-loss' as done in (https://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf),
- X2: one without any rows with missing values or question marks,
- X3: and one with the top 10 most important features

1.4 2. Fit Random Forest models with each dataset

1.5 3. Analysis

[2]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix, roc_curve, auc
from sklearn.model_selection import GridSearchCV, learning_curve
import seaborn as sns
import numpy as np
```

```python
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import label_binarize
from itertools import cycle
from sklearn.utils import column_or_1d
# %pip install ucimlrepo
from ucimlrepo import fetch_ucirepo

# Create 4 different datasets:
# X0: a full one,
# X1: one without the columns 'fnlwgt', 'relationship', 'capital-gain',␣
 ↪'capital-loss',
# X2: one without any rows with missing values or question marks,
# X3: and one with the top 10 most important features according to the␣
 ↪RandomForestClassifier.

# Fetch dataset as a dictionary
adult = fetch_ucirepo(id=2)

# data (as pandas dataframes)
X0 = adult.data.features
y0 = adult.data.targets

# Print information from one row of the dataset
print(f'{X0.iloc[0]} \n')

# metadata
print(f'Metadata: \n{adult.metadata}\n')

# variable information
print(f'Variables: \n{adult.variables}')

# Report on the number of rows and columns in the dataset
def report_dataset_shape(dataset, dataset_name):
    rows = dataset.shape[0]
    columns = dataset.shape[1]
    print(f'{dataset_name} Number of rows: {rows}')
    print(f'{dataset_name} Number of columns: {columns}')

report_dataset_shape(X0, 'X0')


# Create X1
# Remove columns with either too much bad data or not enough useful data
# Columns to remove: 'fnlwgt', 'relationship', 'capital-gain', 'capital-loss'
# https://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf
X1 = X0.drop(columns=['fnlwgt', 'relationship', 'capital-gain', 'capital-loss'])
```

```python
# Ensure y is filtered to match the columns of X
y1 = y0.loc[X1.index]
report_dataset_shape(X1, 'X1')

# Create X2
# Remove any rows with missing values
X2 = X0.dropna()
y2 = y0.loc[X2.index] # Ensure y is filtered to match the rows of X

# Remove any rows where a value is a question-mark "?"
X2 = X2.replace('?', None).dropna()
y2 = y2.loc[X2.index] # Ensure y is filtered to match the rows of X

# Report on the number of rows and columns in the dataset after removing␣
↪missing values
report_dataset_shape(X2, 'X2')

# We will create X3 after we fit the first 3 models to get the top 10 most␣
↪important features

# Split each dataset into features and target
X0_train, X0_test, y0_train, y0_test = train_test_split(X0, y0, test_size=0.2,␣
↪random_state=42)
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2,␣
↪random_state=42)
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.2,␣
↪random_state=42)


def create_pipeline(X):
    categorical_columns = X.select_dtypes(include=['object']).columns
    numeric_columns = X.select_dtypes(exclude=['object']).columns

    preprocessor = ColumnTransformer(
        transformers=[
            ('num', 'passthrough', numeric_columns),
            ('cat', OneHotEncoder(drop='first', sparse_output=False,␣
↪handle_unknown='ignore'), categorical_columns)
        ])

    return Pipeline([
        ('preprocessor', preprocessor),
        ('classifier', RandomForestClassifier(random_state=42))
    ]), categorical_columns

# Create pipelines and store categorical columns
```

```python
pipeline0, categorical_columns0 = create_pipeline(X0)
pipeline1, categorical_columns1 = create_pipeline(X1)
pipeline2, categorical_columns2 = create_pipeline(X2)

# Fit the pipelines
pipeline0.fit(X0_train, y0_train.values.ravel())
pipeline1.fit(X1_train, y1_train.values.ravel())
pipeline2.fit(X2_train, y2_train.values.ravel())


# Print model summaries
# Helper script from the book "Hands-On Machine Learning with Scikit-Learn,␣
↪Keras, and TensorFlow, 2nd Edition" by Aurélien Géron
def print_model_summary(pipeline, X_test, y_test, dataset_name):
    print(f"\nModel Summary for {dataset_name}:")

    # Get the RandomForestClassifier from the pipeline
    rf_classifier = pipeline.named_steps['classifier']

    # Print basic model information
    print(f"Number of trees: {rf_classifier.n_estimators}")
    print(f"Max depth: {rf_classifier.max_depth}")
    print(f"Min samples split: {rf_classifier.min_samples_split}")
    print(f"Min samples leaf: {rf_classifier.min_samples_leaf}")

    # Make predictions
    y_pred = pipeline.predict(X_test)

    # Print accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.4f}")

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

    # Print confusion matrix
    print("\nConfusion Matrix:")
    print(confusion_matrix(y_test, y_pred))

    print("\n" + "="*50 + "\n")

print_model_summary(pipeline0, X0_test, y0_test, "Dataset 0 (Full Dataset)")
print_model_summary(pipeline1, X1_test, y1_test, "Dataset 1 (Removed Columns)")
print_model_summary(pipeline2, X2_test, y2_test, "Dataset 2 (Removed Missing␣
↪Values)")
```

```
# Continue with the rest of your code...

# Make predictions
y0_pred = pipeline0.predict(X0_test)
y1_pred = pipeline1.predict(X1_test)
y2_pred = pipeline2.predict(X2_test)

# Calculate accuracy
accuracy0 = accuracy_score(y0_test, y0_pred)
print(f'Model accuracy X0: {accuracy0}')
accuracy1 = accuracy_score(y1_test, y1_pred)
print(f'Model accuracy X1: {accuracy1}')
accuracy2 = accuracy_score(y2_test, y2_pred)
print(f'Model accuracy X2: {accuracy2}')

# Function to get feature names and importances
def get_feature_info(model, X):
    feature_names = (model.named_steps['preprocessor']
                     .named_transformers_['cat']
                     .get_feature_names_out(X.select_dtypes(include=['object']).
 ↪columns).tolist() +
                     X.select_dtypes(exclude=['object']).columns.tolist())
    importances = model.named_steps['classifier'].feature_importances_
    return importances, feature_names



# Get feature importances and names
importances0, feature_names0 = get_feature_info(pipeline0, X0)
importances1, feature_names1 = get_feature_info(pipeline1, X1)
importances2, feature_names2 = get_feature_info(pipeline2, X2)

# Sort feature importances in descending order (do this only once)
indices0 = np.argsort(importances0)[::-1]
indices1 = np.argsort(importances1)[::-1]
indices2 = np.argsort(importances2)[::-1]

# Function to print feature ranking
def print_feature_ranking(importances, feature_names, indices, X, top_n=None,␣
 ↪dataset_name=None):
    print(f"{dataset_name} Feature ranking:")
    for f in range(min(top_n or len(indices), X.shape[1])):
        print("%d. %s (%f)" % (f + 1, feature_names[indices[f]],␣
 ↪importances[indices[f]]))

# Print the feature rankings (you can specify top_n if you want to limit the␣
 ↪number of features shown)
```

```python
print_feature_ranking(importances0, feature_names0, indices0, X0, top_n=10,␣
  ↪dataset_name='X0')
print_feature_ranking(importances1, feature_names1, indices1, X1, top_n=10,␣
  ↪dataset_name='X1')
print_feature_ranking(importances2, feature_names2, indices2, X2, top_n=10,␣
  ↪dataset_name='X2')

# Get the number of features for plotting
n_features0 = min(len(importances0), X0.shape[1])
n_features1 = min(len(importances1), X1.shape[1])
n_features2 = min(len(importances2), X2.shape[1])

# Function to plot feature importances
def plot_feature_importances(importances, feature_names, indices, n_features,␣
  ↪title):
    plt.figure(figsize=(10, n_features // 3))
    plt.title(title)
    plt.barh(range(n_features), importances[indices[:n_features]])
    plt.yticks(range(n_features), [feature_names[i] for i in indices[:
  ↪n_features]])
    plt.xlabel("Relative Importance")
    plt.tight_layout()
    plt.show()


# Plot the feature importances
plot_feature_importances(importances0, feature_names0, indices0, n_features0,␣
  ↪"Feature Importances (Dataset 0)")
plot_feature_importances(importances1, feature_names1, indices1, n_features1,␣
  ↪"Feature Importances (Dataset 1)")
plot_feature_importances(importances2, feature_names2, indices2, n_features2,␣
  ↪"Feature Importances (Dataset 2)")


# Create X3 using the top 10 most important features from X0
print("Original X0 columns:", X0.columns.tolist())

# Print top 10 feature names and their importances
print("\nTop 10 features and their importances:")
for i in range(10):
    print(f"{feature_names0[indices0[i]]}: {importances0[indices0[i]]}")

# Create a mapping from encoded feature names to original column names
feature_mapping = {}
for original_col in X0.columns:
    for encoded_feature in feature_names0:
```

```
        if original_col in encoded_feature:
            feature_mapping[encoded_feature] = original_col

# Select top 10 original features
top_features = []
for feature in [feature_names0[i] for i in indices0]:
    original_feature = feature_mapping.get(feature)
    if original_feature and original_feature not in top_features:
        top_features.append(original_feature)
    if len(top_features) == 10:
        break

print("\nSelected top 10 original features:", top_features)

# Create X3 using the top 10 most important original features
X3 = X0[top_features]
y3 = y0

# Verify that X3 has the correct features
print("\nFeatures in X3:", X3.columns.tolist())
print("Shape of X3:", X3.shape)



# Split the data
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.2,␣
↪random_state=42)

# Create pipeline
pipeline3, categorical_columns3 = create_pipeline(X3)

# Define the parameter grid
param_grid = {
    'classifier__n_estimators': [10, 40, 98],
    'classifier__max_depth': [None, 5, 10, 15],
    'classifier__min_samples_split': [2, 5, 10],
    'classifier__min_samples_leaf': [1, 5, 10]
}

# Create GridSearchCV object
# Verify that X3 has the correct features
print("Features in X3:", X3.columns)

# Check for any empty columns
if X3.shape[1] == 0:
    raise ValueError("X3 has no features. Please check the feature selection␣
↪process.")
```

```
grid_search = GridSearchCV(pipeline3, param_grid, cv=10, scoring="accuracy",␣
↪n_jobs=-1, verbose=0)

# Fit the grid search
grid_search.fit(X3_train, y3_train.values.ravel())

# Get the best model
best_model = grid_search.best_estimator_

# Make predictions
y3_pred = best_model.predict(X3_test)

# Print model summary
print_model_summary(best_model, X3_test, y3_test, "Dataset 3 (Top 10 Features␣
↪with Hyperparameter Tuning)")

# Print best parameters
print("Best parameters:", grid_search.best_params_)

# Calculate accuracy
accuracy3 = accuracy_score(y3_test, y3_pred)
print(f'Model accuracy X3: {accuracy3}')

# Get feature importances
importances3, feature_names3 = get_feature_info(best_model, X3)

# Identify categorical columns
categorical_columns = X3.select_dtypes(include=['object']).columns

# Perform one-hot encoding
X3_encoded = pd.get_dummies(X3, columns=categorical_columns)

# Create correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(X3_encoded.corr(), annot=False, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of All Features (Encoded)')
plt.tight_layout()
plt.show()

# Feature importance comparison
def plot_feature_importance_comparison(importances_list, feature_names_list,␣
↪model_names):
    plt.figure(figsize=(12, 8))
    x = np.arange(len(importances_list[0]))
    width = 0.2
```

```python
    for i, (importances, feature_names) in enumerate(zip(importances_list, feature_names_list)):
        plt.bar(x + i*width, importances, width, label=model_names[i])

    plt.xlabel('Features')
    plt.ylabel('Importance')
    plt.title('Feature Importance Comparison Across Models')
    plt.xticks(x + width, feature_names_list[0], rotation=90)
    plt.legend()
    plt.tight_layout()
    plt.show()

min_features = min(len(importances0), len(importances1), len(importances2), len(importances3))
importances_list = [
    importances0[:min_features],
    importances1[:min_features],
    importances2[:min_features],
    importances3[:min_features]
]

feature_names_list = [
    feature_names0[:min_features],
    feature_names1[:min_features],
    feature_names2[:min_features],
    feature_names3[:min_features]
]
model_names = ['Model 0', 'Model 1', 'Model 2', 'Model 3']

plot_feature_importance_comparison(importances_list, feature_names_list, model_names)

# ROC curve comparison
def plot_roc_curves(models, X_test_list, y_test_list, model_names):
    plt.figure(figsize=(10, 8))

    for model, X_test, y_test, name in zip(models, X_test_list, y_test_list, model_names):
        # Ensure y_test is flattened
        y_test = y_test.values.ravel()

        # Binarize the output
        n_classes = len(np.unique(y_test))
        y_test_bin = label_binarize(y_test, classes=np.unique(y_test))

        # Compute ROC curve and ROC area for each class
        y_score = model.predict_proba(X_test)
```

```
        fpr = dict()
        tpr = dict()
        roc_auc = dict()
        for i in range(n_classes):
            fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
            roc_auc[i] = auc(fpr[i], tpr[i])

        # Compute micro-average ROC curve and ROC area
        fpr["micro"], tpr["micro"], _ = roc_curve(y_test_bin.ravel(), y_score.
↪ravel())
        roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
        print(f'{name} micro-average AUC: {roc_auc["micro"]:.2f}')

        # Plot ROC curves
        plt.plot(fpr["micro"], tpr["micro"],
                 label=f'{name} (micro-average AUC = {roc_auc["micro"]:.2f})',
                 linewidth=2)

    plt.plot([0, 1], [0, 1], 'k--', linewidth=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('Receiver Operating Characteristic (ROC) Curve Comparison')
    plt.legend(loc="lower right")
    plt.show()


# Now call the function with your data
models = [pipeline0, pipeline1, pipeline2, best_model]
X_test_list = [X0_test, X1_test, X2_test, X3_test]
y_test_list = [y0_test, y1_test, y2_test, y3_test]
model_names = ['Model 0', 'Model 1', 'Model 2', 'Model 3']

plot_roc_curves(models, X_test_list, y_test_list, model_names)


# Learning curve comparison
def plot_learning_curve(estimator, X, y, title):
    # Ensure y is 1-dimensional
    y = column_or_1d(y, warn=False)

    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=5, n_jobs=-1,
        train_sizes=np.linspace(0.1, 1.0, 5))

    train_scores_mean = np.mean(train_scores, axis=1)
```

```
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.figure(figsize=(10, 6))
    plt.title(title)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1, color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training␣
↪score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",␣
↪label="Cross-validation score")

    plt.legend(loc="best")
    plt.show()

plot_learning_curve(best_model, X3, y3.values.ravel(), "Learning Curve for␣
↪Model 3")
```

```
age                           39
workclass              State-gov
fnlwgt                     77516
education              Bachelors
education-num                 13
marital-status     Never-married
occupation          Adm-clerical
relationship       Not-in-family
race                       White
sex                         Male
capital-gain                2174
capital-loss                   0
hours-per-week                40
native-country     United-States
Name: 0, dtype: object
```

Metadata:

{'uci_id': 2, 'name': 'Adult', 'repository_url':
'https://archive.ics.uci.edu/dataset/2/adult', 'data_url':
'https://archive.ics.uci.edu/static/public/2/data.csv', 'abstract': 'Predict
whether annual income of an individual exceeds $50K/yr based on census data.
Also known as "Census Income" dataset. ', 'area': 'Social Science', 'tasks':

['Classification'], 'characteristics': ['Multivariate'], 'num_instances': 48842, 'num_features': 14, 'feature_types': ['Categorical', 'Integer'], 'demographics': ['Age', 'Income', 'Education Level', 'Other', 'Race', 'Sex'], 'target_col': ['income'], 'index_col': None, 'has_missing_values': 'yes', 'missing_values_symbol': 'NaN', 'year_of_dataset_creation': 1996, 'last_updated': 'Tue Sep 24 2024', 'dataset_doi': '10.24432/C5XW20', 'creators': ['Barry Becker', 'Ronny Kohavi'], 'intro_paper': None, 'additional_info': {'summary': "Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))\n\nPrediction task is to determine whether a person's income is over $50,000 a year.\n", 'purpose': None, 'funded_by': None, 'instances_represent': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': 'Listing of attributes:\r\n\r\n>50K, <=50K.\r\n\r\nage: continuous.\r\nworkclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.\r\nfnlwgt: continuous.\r\neducation: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.\r\neducation-num: continuous.\r\nmarital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.\r\noccupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.\r\nrelationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.\r\nrace: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.\r\nsex: Female, Male.\r\ncapital-gain: continuous.\r\ncapital-loss: continuous.\r\nhours-per-week: continuous.\r\nnative-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.', 'citation': None}}

```
Variables:
              name     role         type      demographic  \
0              age  Feature      Integer              Age
1        workclass  Feature  Categorical           Income
2           fnlwgt  Feature      Integer             None
3        education  Feature  Categorical  Education Level
4    education-num  Feature      Integer  Education Level
5   marital-status  Feature  Categorical            Other
6       occupation  Feature  Categorical            Other
7     relationship  Feature  Categorical            Other
8             race  Feature  Categorical             Race
9              sex  Feature       Binary              Sex
10    capital-gain  Feature      Integer             None
11    capital-loss  Feature      Integer             None
```

```
12  hours-per-week  Feature      Integer            None
13  native-country  Feature  Categorical           Other
14          income   Target       Binary          Income

                                       description units missing_values
0                                              N/A  None             no
1   Private, Self-emp-not-inc, Self-emp-inc, Feder…  None            yes
2                                             None  None             no
3    Bachelors, Some-college, 11th, HS-grad, Prof-…  None             no
4                                             None  None             no
5   Married-civ-spouse, Divorced, Never-married, S…  None             no
6   Tech-support, Craft-repair, Other-service, Sal…  None            yes
7   Wife, Own-child, Husband, Not-in-family, Other…  None             no
8   White, Asian-Pac-Islander, Amer-Indian-Eskimo,…  None             no
9                                    Female, Male.  None             no
10                                            None  None             no
11                                            None  None             no
12                                            None  None             no
13  United-States, Cambodia, England, Puerto-Rico,…  None            yes
14                                    >50K, <=50K.  None             no
X0 Number of rows: 48842
X0 Number of columns: 14
X1 Number of rows: 48842
X1 Number of columns: 10
X2 Number of rows: 45222
X2 Number of columns: 14

Model Summary for Dataset 0 (Full Dataset):
Number of trees: 100
Max depth: None
Min samples split: 2
Min samples leaf: 1
```

```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [7] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(
```

```
Accuracy: 0.5488

Classification Report:
              precision    recall  f1-score   support

       <=50K       0.61      0.81      0.69      4936
      <=50K.       0.40      0.19      0.26      2478
        >50K       0.48      0.51      0.50      1562
       >50K.       0.24      0.10      0.15       793
```

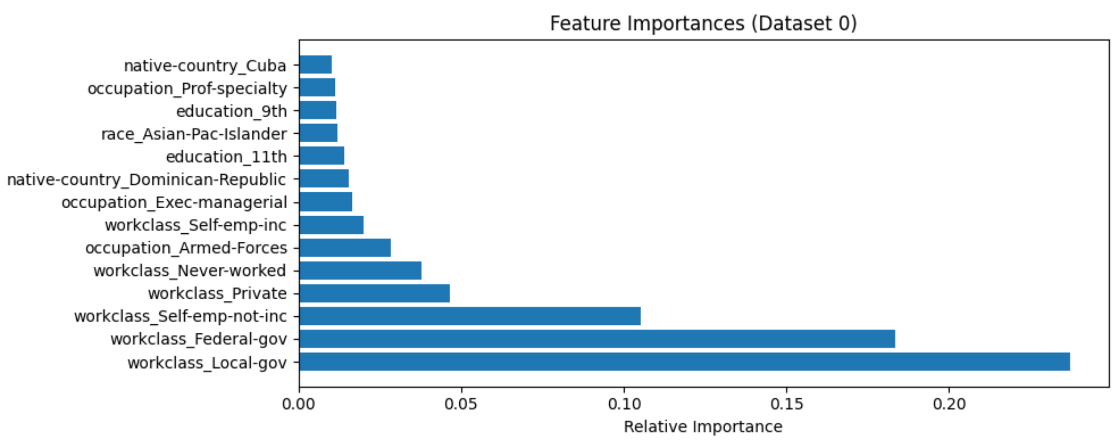
```
    accuracy                           0.55      9769
   macro avg       0.43      0.41      0.40      9769
weighted avg       0.51      0.55      0.51      9769


Confusion Matrix:
[[3997  557  301   81]
 [1800  481  163   34]
 [ 514  104  800  144]
 [ 259   58  393   83]]

==================================================


Model Summary for Dataset 1 (Removed Columns):
Number of trees: 100
Max depth: None
Min samples split: 2
Min samples leaf: 1
```

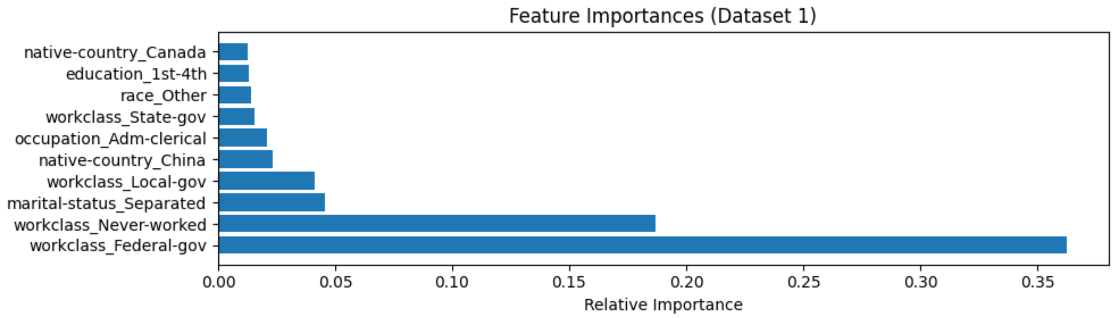
```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [6] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(
```

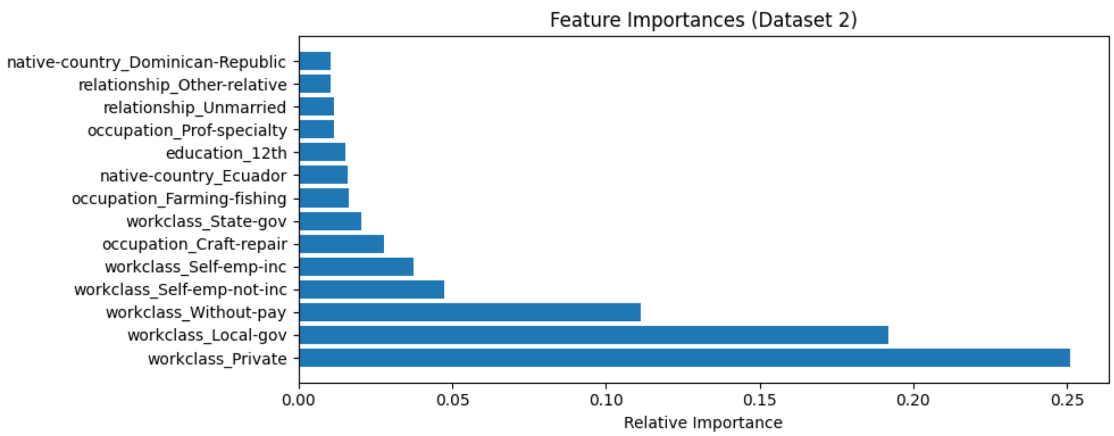
```
Accuracy: 0.5108

Classification Report:
              precision    recall  f1-score   support

       <=50K       0.60      0.73      0.66      4936
      <=50K.       0.36      0.25      0.29      2478
        >50K       0.43      0.43      0.43      1562
       >50K.       0.24      0.16      0.19       793

    accuracy                           0.51      9769
   macro avg       0.41      0.39      0.39      9769
weighted avg       0.48      0.51      0.49      9769


Confusion Matrix:
[[3586  829  399  122]
 [1601  608  200   69]
 [ 538  152  671  201]
 [ 264  104  300  125]]

==================================================
```

```
Model Summary for Dataset 2 (Removed Missing Values):
Number of trees: 100
Max depth: None
Min samples split: 2
Min samples leaf: 1
Accuracy: 0.5261

Classification Report:
              precision    recall  f1-score   support

       <=50K       0.60      0.79      0.68      4513
      <=50K.       0.28      0.13      0.18      2232
        >50K       0.50      0.53      0.52      1538
       >50K.       0.24      0.12      0.16       762

    accuracy                           0.53      9045
   macro avg       0.40      0.39      0.38      9045
weighted avg       0.47      0.53      0.48      9045


Confusion Matrix:
[[3557  596  279   81]
 [1762  293  139   38]
 [ 436  109  818  175]
 [ 221   64  386   91]]

==================================================

/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [7] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [6] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(

Model accuracy X0: 0.5487767427577029
Model accuracy X1: 0.5107994677039616
Model accuracy X2: 0.526147042564953
X0 Feature ranking:
1. workclass_Local-gov (0.237267)
2. workclass_Federal-gov (0.183593)
3. workclass_Self-emp-not-inc (0.105246)
4. workclass_Private (0.046697)
```

```
5. workclass_Never-worked (0.037884)
6. occupation_Armed-Forces (0.028504)
7. workclass_Self-emp-inc (0.019956)
8. occupation_Exec-managerial (0.016587)
9. native-country_Dominican-Republic (0.015495)
10. education_11th (0.014164)
X1 Feature ranking:
1. workclass_Federal-gov (0.362461)
2. workclass_Never-worked (0.186766)
3. marital-status_Separated (0.045636)
4. workclass_Local-gov (0.041314)
5. native-country_China (0.023524)
6. occupation_Adm-clerical (0.020861)
7. workclass_State-gov (0.015605)
8. race_Other (0.014008)
9. education_1st-4th (0.013095)
10. native-country_Canada (0.012547)
X2 Feature ranking:
1. workclass_Private (0.251032)
2. workclass_Local-gov (0.191962)
3. workclass_Without-pay (0.111421)
4. workclass_Self-emp-not-inc (0.047457)
5. workclass_Self-emp-inc (0.037292)
6. occupation_Craft-repair (0.027675)
7. workclass_State-gov (0.020610)
8. occupation_Farming-fishing (0.016373)
9. native-country_Ecuador (0.016003)
10. education_12th (0.015156)
```



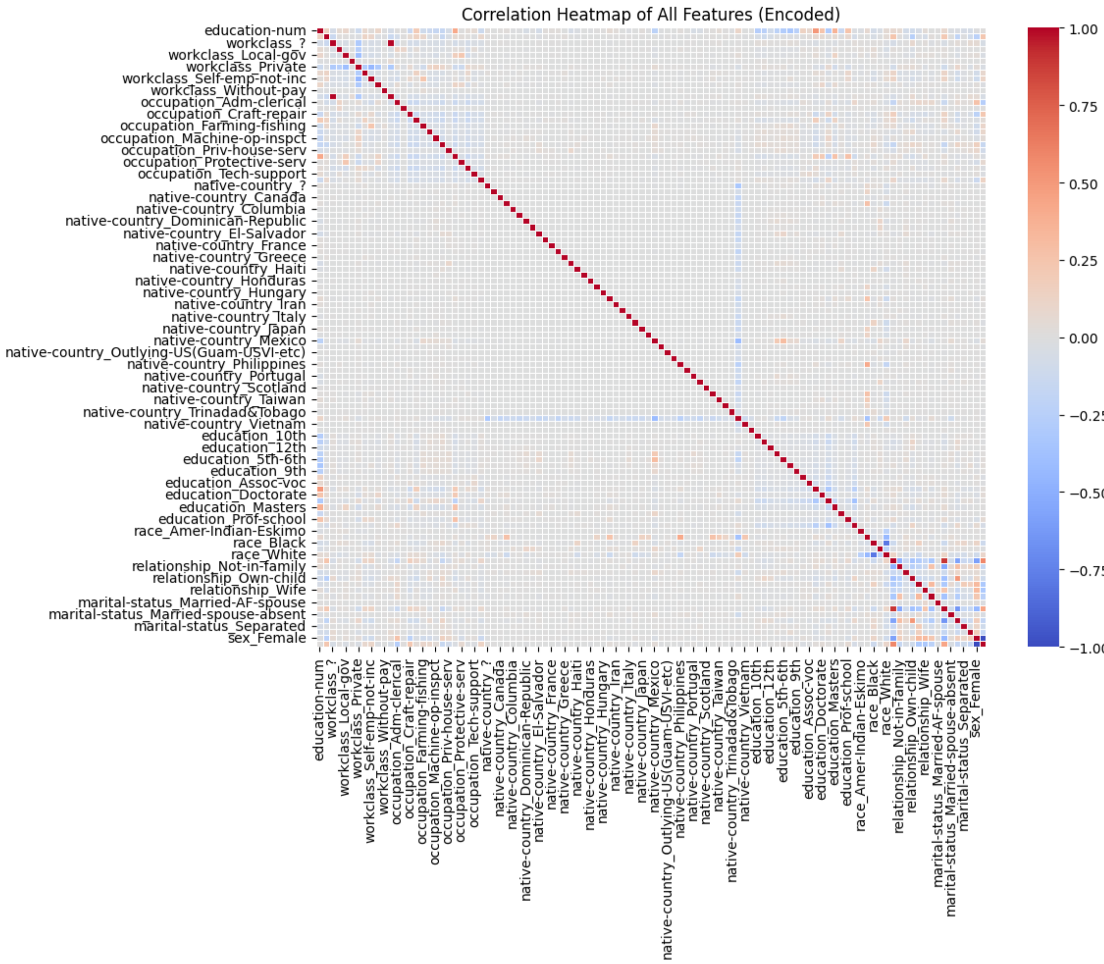


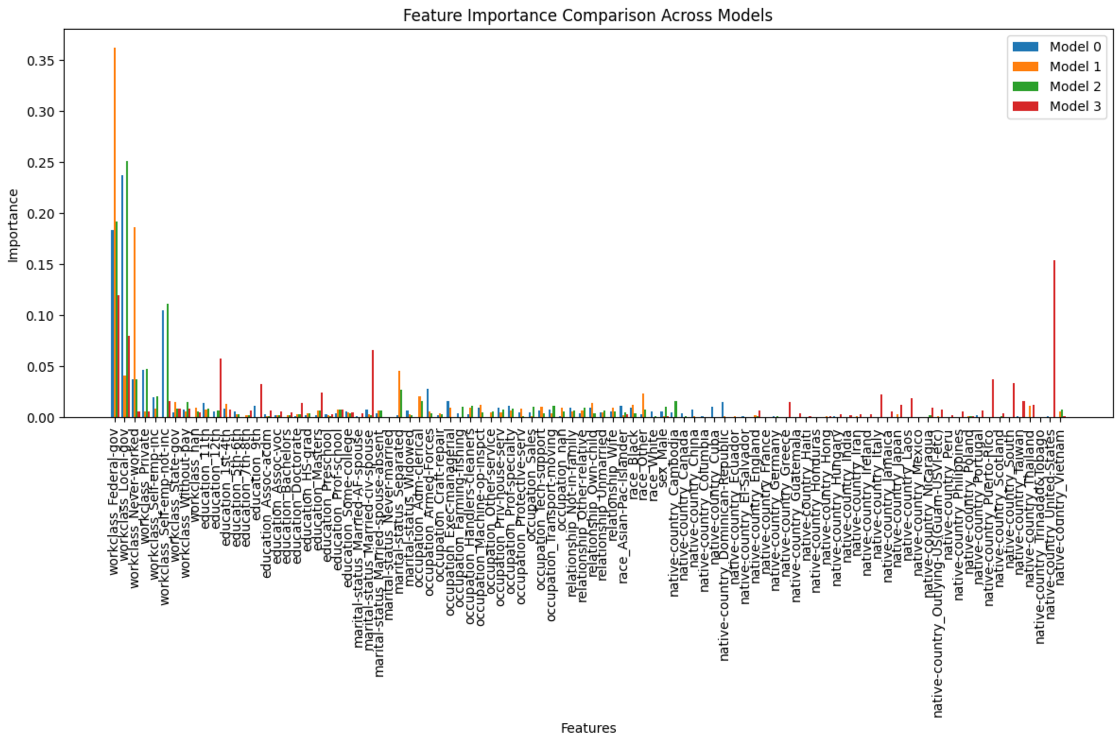



```
Original X0 columns: ['age', 'workclass', 'fnlwgt', 'education', 'education-
num', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'capital-
gain', 'capital-loss', 'hours-per-week', 'native-country']

Top 10 features and their importances:
workclass_Local-gov: 0.23726718555717502
workclass_Federal-gov: 0.1835932315561079
workclass_Self-emp-not-inc: 0.1052460715073518
workclass_Private: 0.046697369201038424
workclass_Never-worked: 0.0378836078832112
occupation_Armed-Forces: 0.02850437435785696
workclass_Self-emp-inc: 0.019955910830872923
occupation_Exec-managerial: 0.0165867357094631
native-country_Dominican-Republic: 0.015495148238840845
education_11th: 0.014164117101791648

Selected top 10 original features: ['workclass', 'occupation', 'native-country',
'education', 'race', 'relationship', 'education-num', 'marital-status', 'sex',
'hours-per-week']
```

```
Features in X3: ['workclass', 'occupation', 'native-country', 'education',
'race', 'relationship', 'education-num', 'marital-status', 'sex', 'hours-per-
week']
Shape of X3: (48842, 10)
Features in X3: Index(['workclass', 'occupation', 'native-country', 'education',
'race',
       'relationship', 'education-num', 'marital-status', 'sex',
       'hours-per-week'],
      dtype='object')

/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [2] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [2] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(


Model Summary for Dataset 3 (Top 10 Features with Hyperparameter Tuning):
Number of trees: 98
Max depth: None
Min samples split: 2
Min samples leaf: 5
Accuracy: 0.5766

Classification Report:
              precision    recall  f1-score   support

       <=50K       0.59      0.94      0.72      4936
      <=50K.       0.91      0.09      0.16      2478
        >50K       0.48      0.50      0.49      1562
       >50K.       0.92      0.01      0.03       793

    accuracy                           0.58      9769
   macro avg       0.72      0.38      0.35      9769
weighted avg       0.68      0.58      0.48      9769


Confusion Matrix:
[[4624    0  312    0]
 [2115  212  150    1]
 [ 776    0  786    0]
 [ 373   21  388   11]]

==================================================
```

```
Best parameters: {'classifier__max_depth': None, 'classifier__min_samples_leaf':
5, 'classifier__min_samples_split': 2, 'classifier__n_estimators': 98}
Model accuracy X3: 0.5766199201555943
```





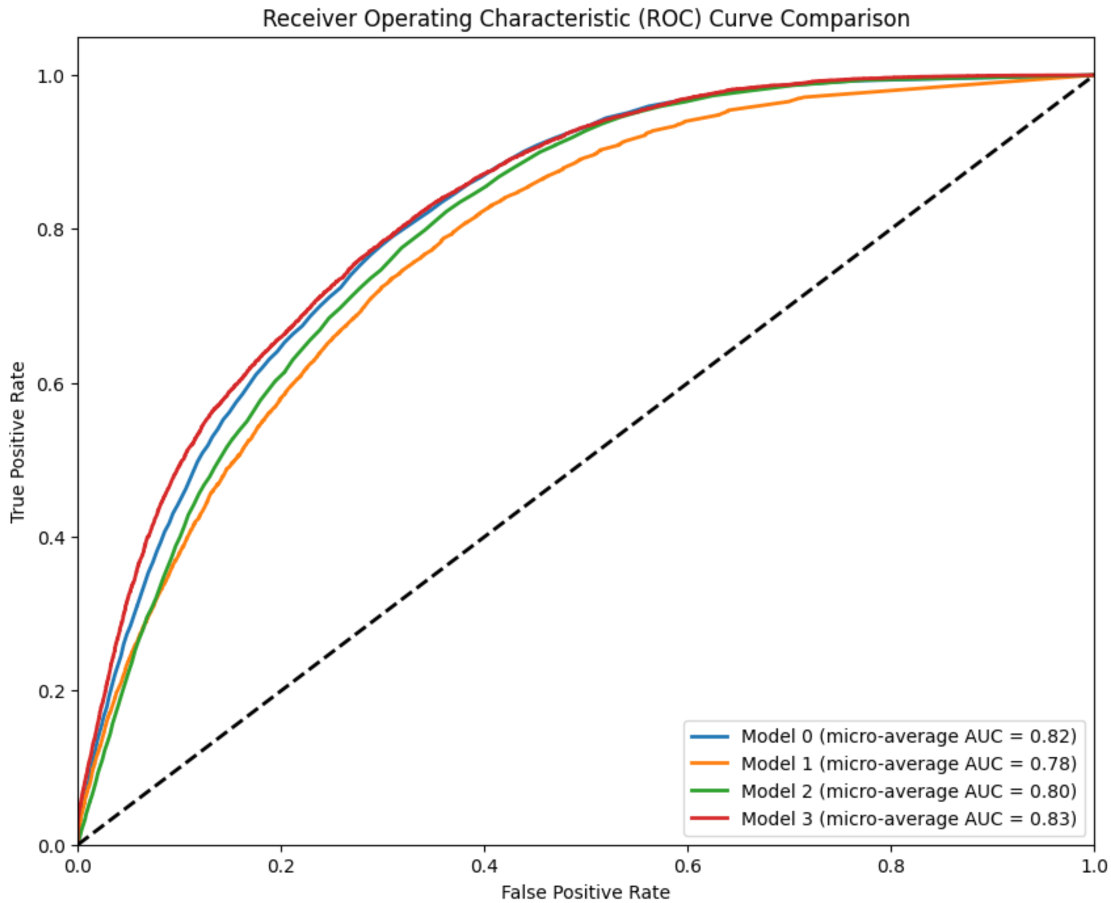

```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [7] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(
```

```
Model 0 micro-average AUC: 0.82
```

```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [6] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(
```

```
Model 1 micro-average AUC: 0.78
Model 2 micro-average AUC: 0.80
Model 3 micro-average AUC: 0.83
```

```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [2] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(
```




```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
```

```
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
```

```
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [2] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [0, 1, 2] during transform. These unknown categories will
be encoded as all zeros
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-
packages/sklearn/preprocessing/_encoders.py:242: UserWarning: Found unknown
categories in columns [2] during transform. These unknown categories will be
encoded as all zeros
  warnings.warn(
```