علی آرمان ۴۰۲۳۶۲۴۰۰۲

توضیحات ابتدایی:

- کامنت های نوشته شده در برنامه توسط اکستنش vscode comment code می باشد.
- در برخی توابع به دلیل عدم آشنایی بنده با نحوه کار کردن با توابع کتابخانه از وب سایت stackoverflow و openia کمک گرفته شده.
- تا حد امکان کپسول سازی انجام شده و اکثر موارد به صورت تابع پیاده سازی شده.
- کتابخانه های استفاده شده:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

۲-خواندن داده‌ها و تبدیل داده‌های رشته به عدد:

```
# Step 1: Read the dataset
Comment Code
def read_dataset(file_path):
    return pd.read_csv(file_path)
```

```
file_path = 'Telecust1.csv'
# file_path = 'Telecust1-Null.csv'
data = read_dataset(file_path)
```

داده‌های رشته با استفاده از کتابخانه LabelEncoder به داده‌های عددی تبدیل شده‌اند.

```
# Step 2: Encode categorical data using LabelEncoder
Comment Code
def encode_categorical_data(data):
    label_encoder = LabelEncoder()
    for column in data.select_dtypes(include=['object']).columns:
        data[column] = label_encoder.fit_transform(data[column])
    return data
```

۳- رفع مشکل مقادیر خالی:

دو روش برای رفع ارزش‌های خالی اعمال شده است: حذف ردیف های حاوی مقدار NaN و روش دگیر جایگذاری NaN با میانگین ستون.

```python
# Step 3: Handle missing values by dropping or replacing with the mean
Comment Code
def handle_missing_values(data):
    # First method: Delete rows with NaN values
    data_dropped = data.dropna()

    # Second method: Replace NaN values with the mean of the column
    data_filled = data.fillna(data.mean())

    # Report the difference in the number of data
    difference = len(data) - len(data_dropped)
    print(f"Difference in the number of data after handling missing values: {difference}")

    return data_filled
```

```
PROBLEMS 84  DEBUG CONSOLE  OUTPUT  TERMINAL  SEARCH TERMINAL OUTPUT  COMMENTS

4    41.875000
Name: employ, dtype: float64
Confusion Matrix:
[[25  9 15 11]
 [ 7  9 16  7]
 [10  8 26 11]
 [16  9  7 14]]
Accuracy: 0.37, Precision: 0.26, Recall: 0.23
(env_hw_1_ml) PS D:\sku\machine learning\hw_1_ml> python p5.py
Difference in the number of data after handling missing values: 439
```

با توجه به نتیجه کد ۴۳۹ داده دراپ شده اند که همان مقادیر null هستند.

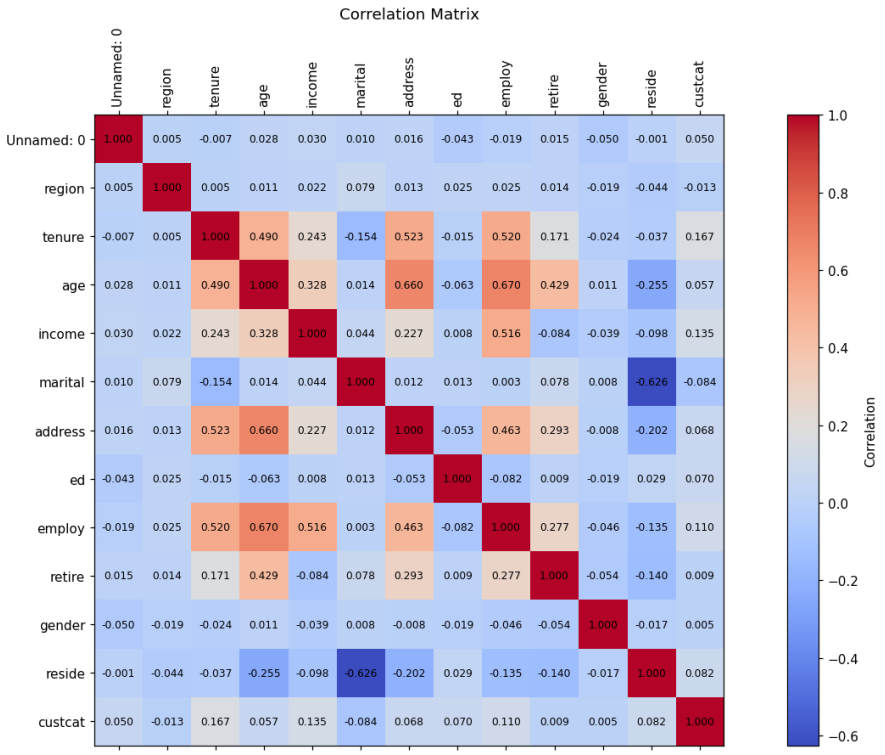
به دلیل اینکه در صورت سوال ذکر شده بود که مقادیر با روش دوم استفاده شود من نیز data_filled را برگشت داده ام.

۴- محاسبه ضرایب همبستگی و نمایش در نمودار:

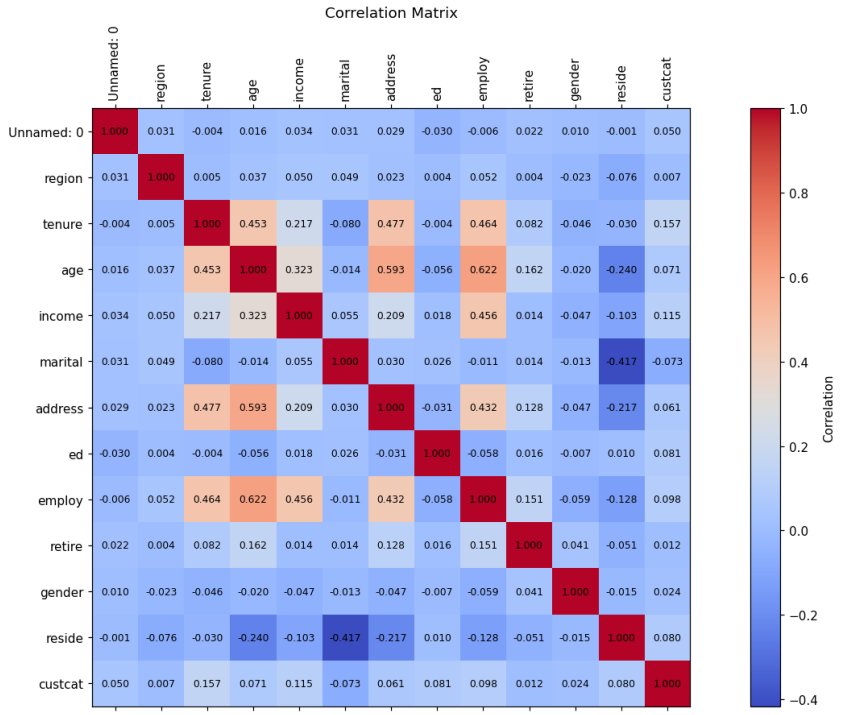
در تابع visualize_correlation ضریب همبستگی تمامی ستون ها با همدیگر حساب شده.

```python
# Step 4: Obtain and visualize correlation coefficients
Comment Code
def visualize_correlation(data):
    correlation_matrix = data.corr()
    plt.figure(figsize=(16, 14))

    # Use matshow instead of imshow to display the exact values
    plt.matshow(correlation_matrix, cmap='coolwarm', fignum=1)
    plt.colorbar(label='Correlation')

    # Display the values
    for (i, j), z in np.ndenumerate(correlation_matrix):
        plt.text(j, i, f'{z:.3f}', ha='center', va='center', fontsize=8)

    plt.xticks(range(len(correlation_matrix.columns)), correlation_matrix.columns, rotation='vertical')
    plt.yticks(range(len(correlation_matrix.columns)), correlation_matrix.columns)
    plt.title("Correlation Matrix")
    # plt.savefig('correlation_matrix.png')  # Save the correlation matrix plot
    plt.show()
```

تصویر زیر نتیجه نمودار برای فایل اول می باشد:



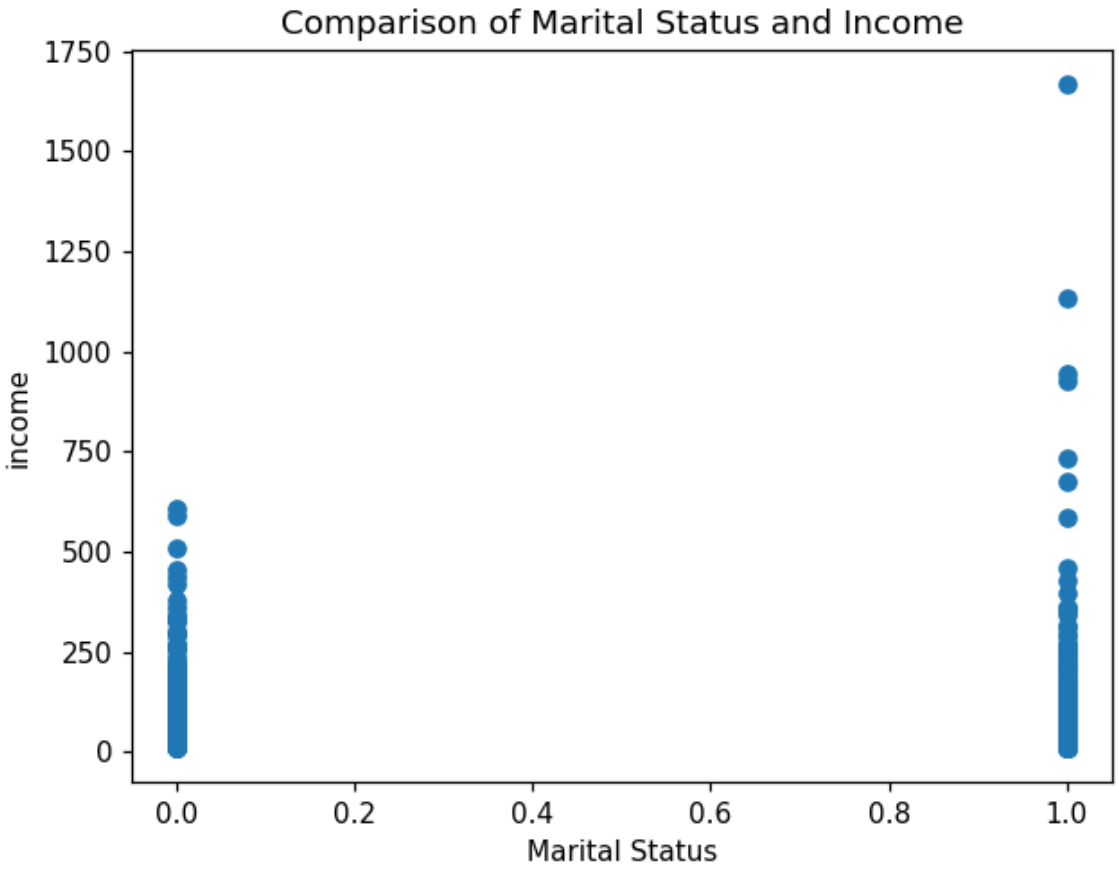
تصویر زیر نتیجه نمودار برای فایل دوم می باشد که با میانگین اصلاح شده:



همان طور که قابل مشاهده هست بیشترین همبستگی را هر ستون با خودش دارد و با توجه به اینکه در فایل دوم ما مقادیر میانگین قرار دادیم، بعضی از همبستگی ها متفاوت شده است.

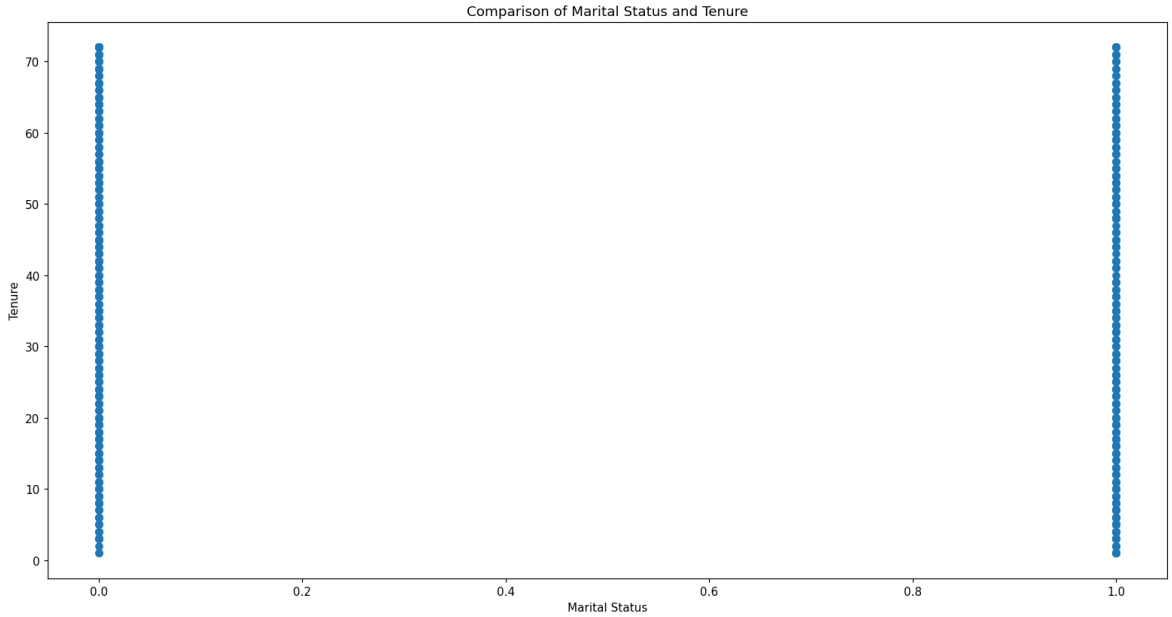
جهت مقایسه وضعیت تاهل، میزان درآمد و tenure برای مقایسه راحت تر در یک نمودار نیز به صورت جداگانه رسم شده تا بتوان راحت تر مقایسه نمود.

در آمد و تاهل:



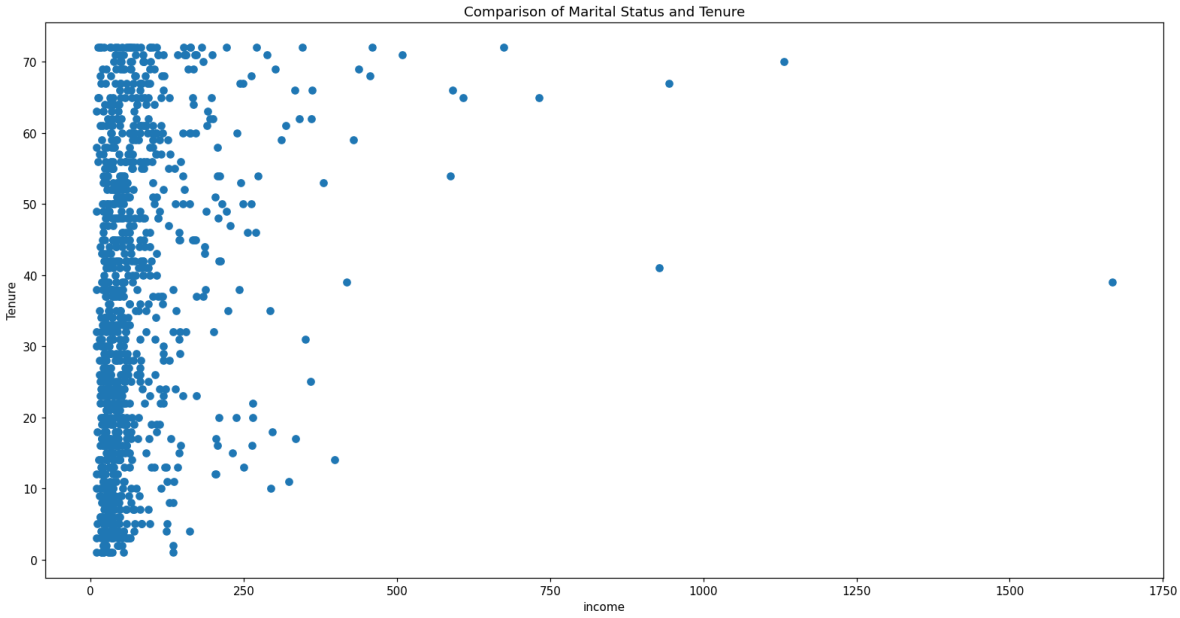
طبق نمودار افراد متاهل درآمد بیشتری داشته اند.

مقایسه tenure با وضعیت تاهل:



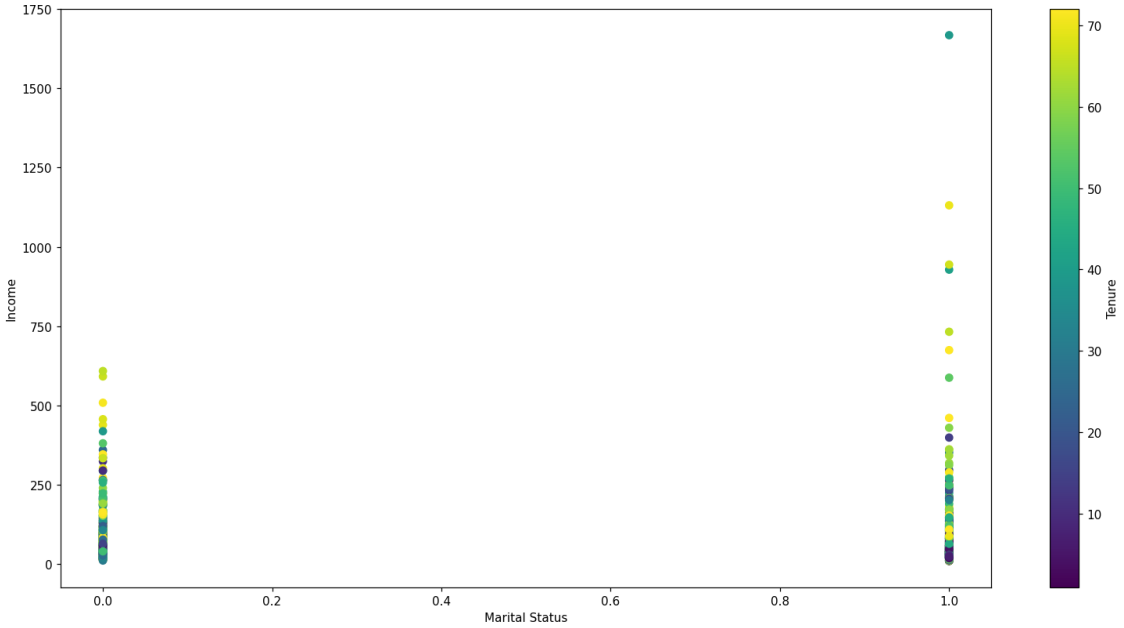
همان طور که پیداست تفاوت زیادی باهم دیگر ندارند.

مقایسه tenure و در آمد:

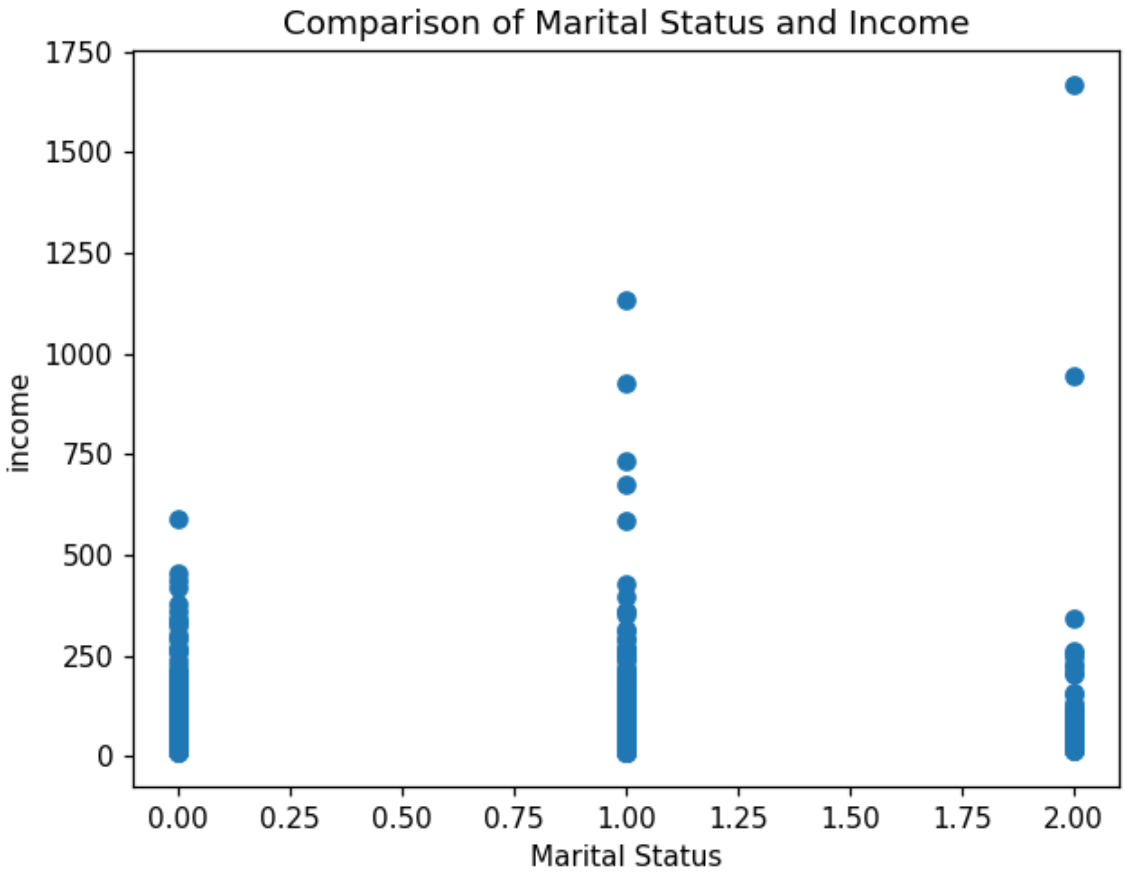


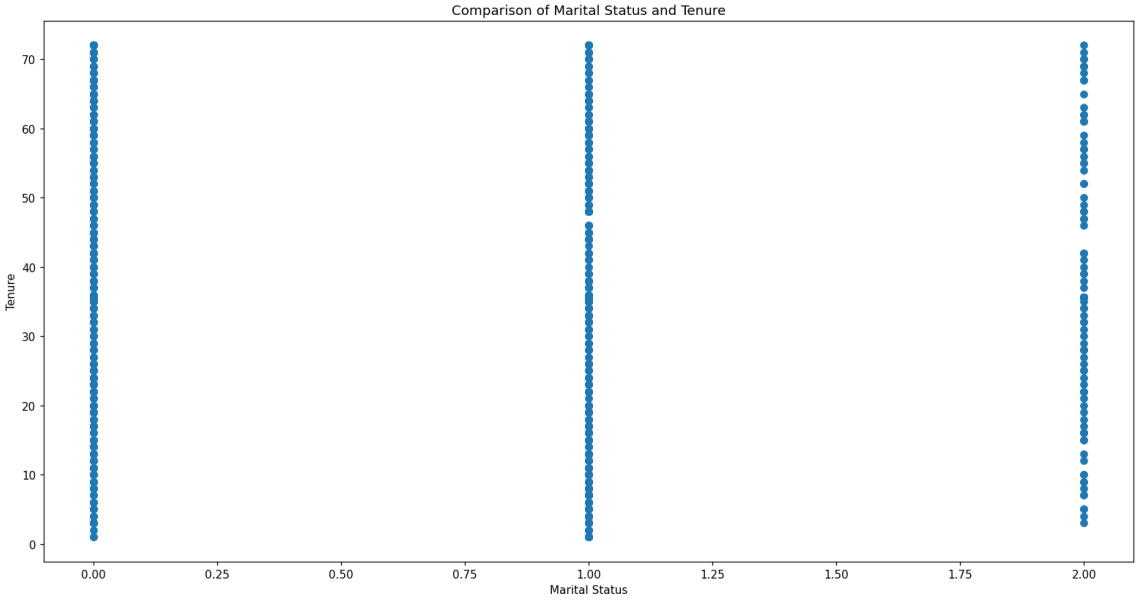
همان طور که پیداست مقدار tenure در درآمد های پایین بیشتر است.

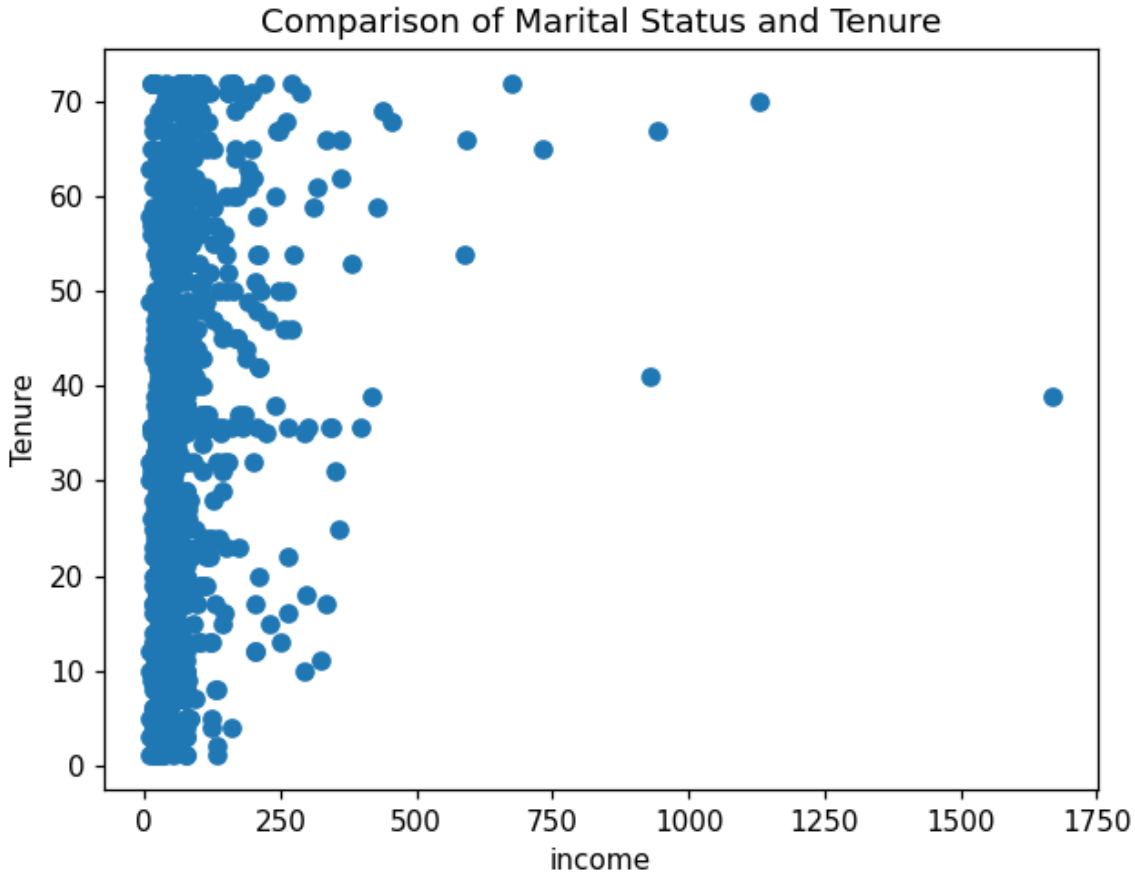
مقایسه همگانی:

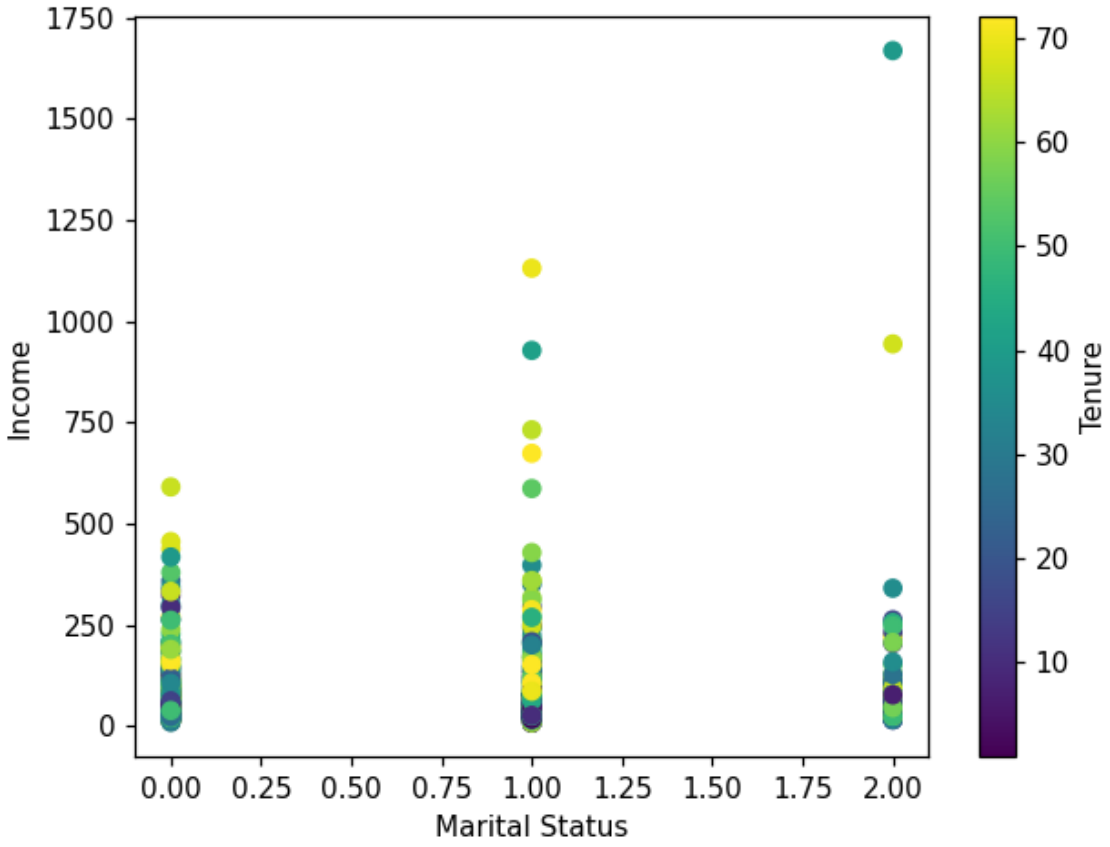


خروجی نمودار برای فایل دوم:




Comparison of Marital Status and Tenure
Tenure
Marital Status
0
10
20
30
40
50
60
70
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Comparison of Marital Status and Tenure
Tenure
income
0
10
20
30
40
50
60
70
0
250
500
750
1000
1250
1500
1750



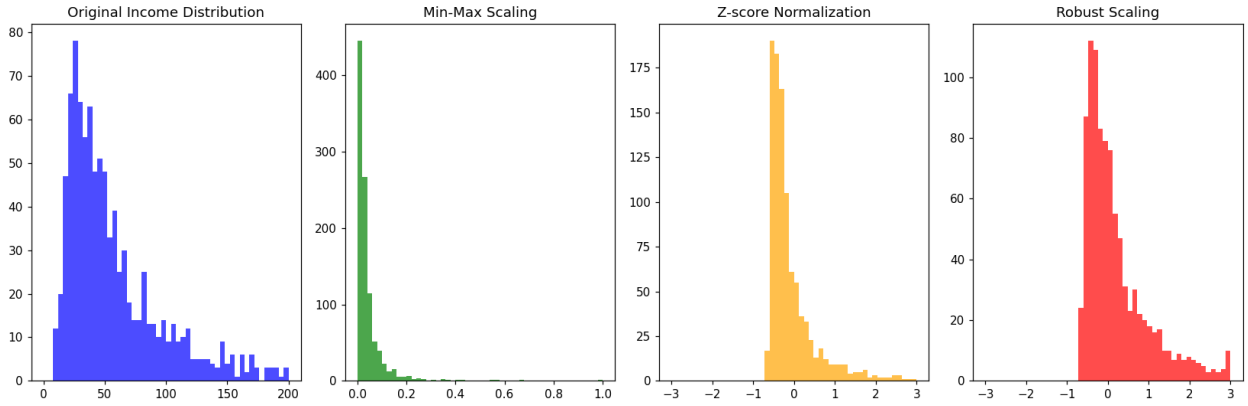
طبق نمودار های فایل دوم می توان نتیجه گرفت که اکثر مقادیر نال مربوط به افراد متاهل بوده و باعث ایجاد نویز بر روی داده ها شده

۵- نرمال‌سازی دو ستون income و tenure با سه روش مختلف:

روش‌هایMin-Max scaling ، Z-score normalization، Robust scaling برای نرمال‌سازی استفاده شده اند

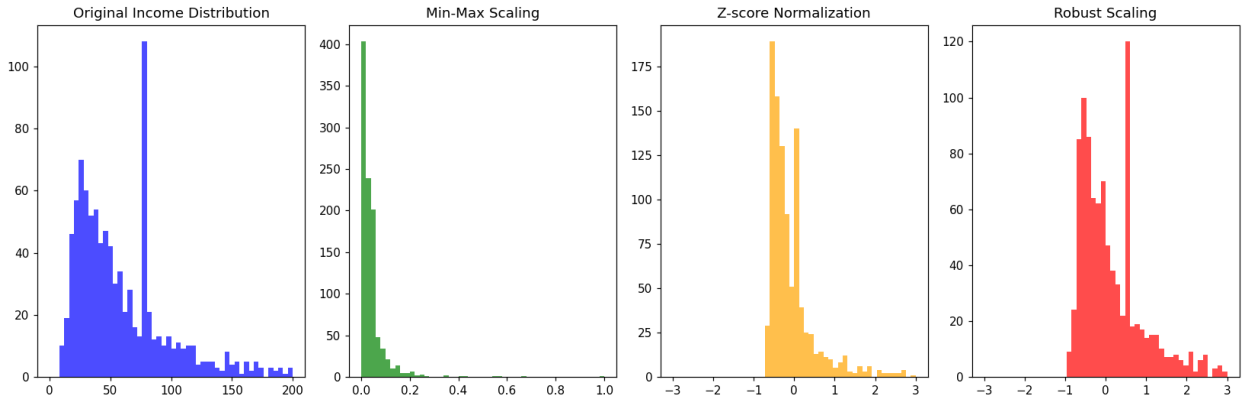
```python
def normalize_columns(data):
    # Normalize income using three different methods: Min-Max scaling, Z-score normalization, and Robust scaling
    # Set a custom range for the histograms
    income_range = (0, 200)  # Adjust this range based on your data distribution

    data['income_minmax'] = (data['income'] - data['income'].min()) / (data['income'].max() - data['income'].min())
    data['income_zscore'] = (data['income'] - data['income'].mean()) / data['income'].std()
    data['income_robust'] = (data['income'] - data['income'].median()) / (data['income'].quantile(0.75) - data['income'].quantile(0.25))

    # Plot the graphs for comparison
    plt.figure(figsize=(15, 5))

    # Adjust the bin count and range for better visualization
    plt.subplot(1, 4, 1)
    plt.hist(data['income'], bins=50, range=income_range, color='blue', alpha=0.7)
    plt.title('Original Income Distribution')

    plt.subplot(1, 4, 2)
    plt.hist(data['income_minmax'], bins=50, range=(0, 1), color='green', alpha=0.7)
    plt.title('Min-Max Scaling')

    plt.subplot(1, 4, 3)
    plt.hist(data['income_zscore'], bins=50, range=(-3, 3), color='orange', alpha=0.7)
    plt.title('Z-score Normalization')

    plt.subplot(1, 4, 4)
    plt.hist(data['income_robust'], bins=50, range=(-3, 3), color='red', alpha=0.7)
    plt.title('Robust Scaling')

    plt.tight_layout()
    plt.savefig('income_normalization.png')  # Save the normalization comparison plot
    plt.show()
```

خروجی برای نمودار درآمد در فایل اصلی:

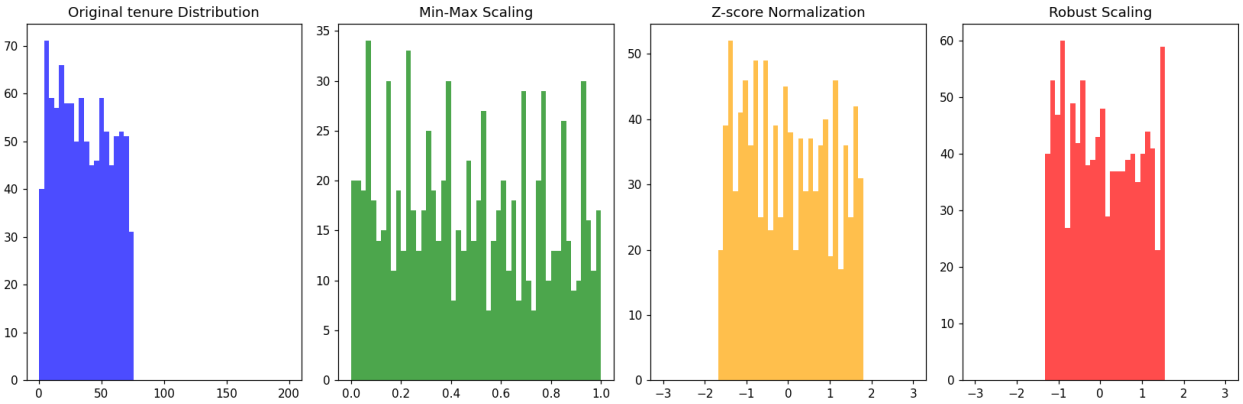


همان طور که پیداست داده های اصلی گستردگی زیادی دارند، ولی با نرمول سازی داده ها کم تر پخش هستند.

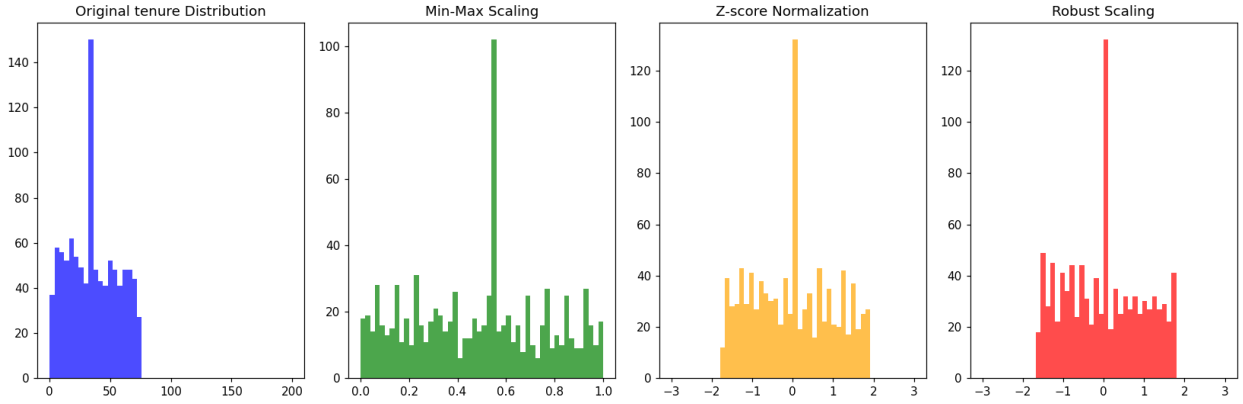
خروجی فایل دوم:



خروجی برای tenure:



خروجی نمودار در فایل دوم:



طبق نمودار های فایل دوم می توان نتیجه گرفت که اکثر مقادیر نال مربوط به افراد متاهل بوده و باعث ایجاد نویز بر روی داده ها شده.

۶-دسته بندی ستون employبا استفاده از روش میانگین دسته‌ها:

اجرا بر روی فایل اول با ۵ دسته بندی:

```
147     def smooth_employ_column(data, num_categories):
148         data['employ_smoothed'] = pd.cut(data['employ'], bins=num_categories, labels=False)
149         # Report the average of the categories
150         category_avg = data.groupby('employ_smoothed')['employ'].mean()
151         print(f"Average of categories after smoothing 'employ' column:\n{category_avg}")
152
```

PROBLEMS 117 DEBUG CONSOLE OUTPUT TERMINAL SEARCH TERMINAL OUTPUT COMMENTS

```
Difference in the number of data after handling missing values: 439
Average of categories after smoothing 'employ' column:
employ_smoothed
0     3.790927
1    12.623785
2    22.613139
3    32.431373
4    41.875000
```

۷- تحقیق و گزارش ماتریس ابهام و معیارها:

ماتریس درهم‌ریختگی: ماتریس درهم‌ریختگی یک ابزار ارزیابی است که در مسائل دسته‌بندی و پیش‌بینی بکار می‌رود. این ماتریس برخی از اطلاعات مهم را در مورد عملکرد مدل یا الگوریتم دسته‌بندی فراهم می‌کند. در یک ماتریس درهم‌ریختگی، داده‌ها به چهار دسته اصلی تقسیم می‌شوند:

۱. **True Positive (TP):** تعداد مثبت‌های صحیح که به درستی تشخیص داده شده‌اند.

۲. **True Negative (TN):** تعداد منفی‌های صحیح که به درستی تشخیص داده شده‌اند.

۳. **False Positive (FP):** تعداد منفی‌های اشتباه که به اشتباه تشخیص داده شده‌اند.

۴. **False Negative (FN):** تعداد مثبت‌های اشتباه که به اشتباه تشخیص داده شده‌اند.

معیارهای موجود در ماتریس درهم‌ریختگی:

۱. **صحت (Accuracy):** صحت نسبت تعداد پیش‌بینی‌های صحیح (TP و TN) به کل تعداد پیش‌بینی‌ها را نشان می‌دهد.

۲. **دقت (Precision):** دقت نسبت تعداد مثبت‌های صحیح (tp) به کل تعداد مثبت‌های تشخیص داده شده (tp+fp)را نشان می‌دهد.

۳. **حساسیت (Recall یا Sensitivity):** حساسیت نسبت تعداد مثبت‌های صحیح (tp) به کل تعداد مثبت‌ها (tp+fn) را نشان می‌دهد.

۴. **ویژگی F1 (F1-Score):** ویژگی F1 میانگین نمایش دقت و حساسیت است و برخی از مواقع به عنوان معیار ترکیبی استفاده می‌شود، که حاصل ضرب دقت و حساسیت تقسیم بر مجموع دقت و حساسیت است.

این معیارها به کمک ماتریس درهم‌ریختگی در ارزیابی کارایی مدل‌های دسته‌بندی به کار می‌روند و به تحلیل نقاط قوت و ضعف مدل کمک می‌کنند.

۸- ترین کردن :

```
# Step 9: Split the dataset into training and testing sets
Comment Code
def split_train_test_data(data):
    X = data.drop('custcat', axis=1)
    y = data['custcat']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    return X_train, X_test, y_train, y_test
```

۹- مرحله ۷

قطعه کد نوشته شده برای نمایش ماتریس درهم ریختگی:

```
153     # Step 8: confusion matrix and criteria
        Comment Code
154 +   def report_confusion_matrix_criteria(y_test, y_pred):
155         cm = confusion_matrix(y_test, y_pred)
156         accuracy = np.sum(np.diag(cm)) / np.sum(cm)
157         precision = cm[1, 1] / np.sum(cm[:, 1])
158         recall = cm[1, 1] / np.sum(cm[1, :])
159         print(f"Confusion Matrix:\n{cm}")
160         print(f"Accuracy: {accuracy:.2f}, Precision: {precision:.2f}, Recall: {recall:.2f}")
161
```

```
PROBLEMS 117   DEBUG CONSOLE   OUTPUT   TERMINAL   SEARCH TERMINAL OUTPUT   COMMENTS

4    41.875000
Name: employ, dtype: float64
Confusion Matrix:
[[21  9 16 14]
 [10  9 14  6]
 [12  8 26  9]
 [11  9 10 16]]
Accuracy: 0.36, Precision: 0.26, Recall: 0.23
```

۱۰- مراحل ۸ و ۹:

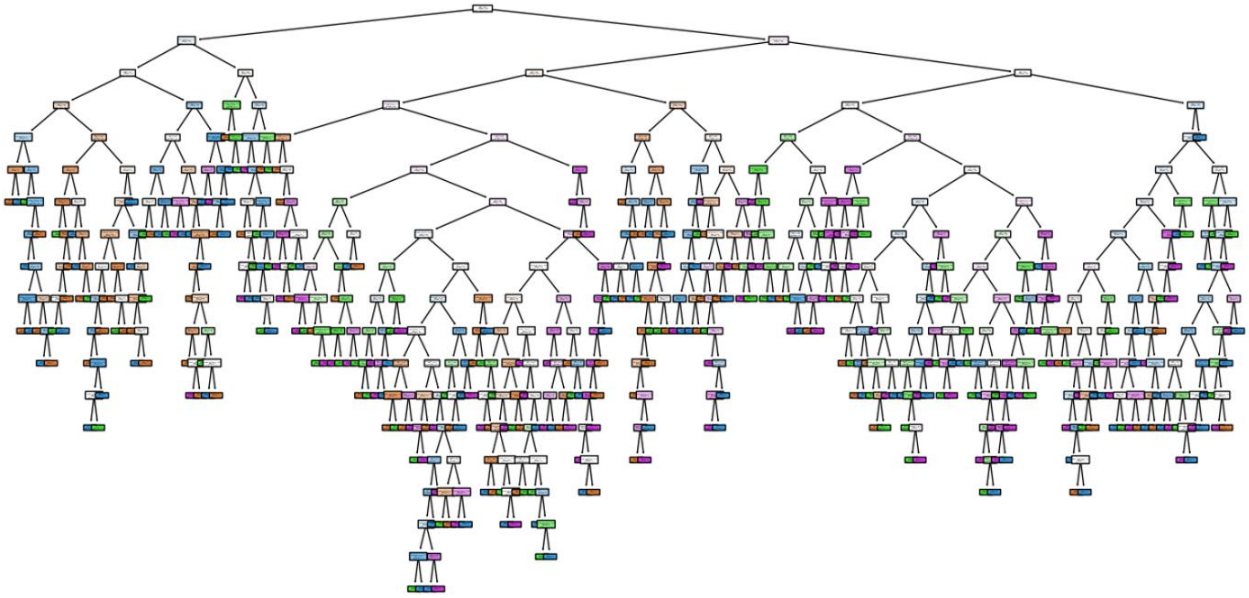
خروجی برای فایل شماره دو:

```
Confusion Matrix:
[[23 11 15 11]
 [ 9  9 14  7]
 [12  9 25  9]
 [13  8 11 14]]
Accuracy: 0.35, Precision: 0.24, Recall: 0.23
```

خروجی برای فایل شماره یک:

```
Confusion Matrix:
[[21  9 16 14]
 [10  9 14  6]
 [12  8 26  9]
 [11  9 10 16]]
Accuracy: 0.36, Precision: 0.26, Recall: 0.23
```

خروجی درخت تصمیم (انجام شده به صورت اضافی):



قسمت اضافی:

```python
from sklearn.decomposition import PCA
Comment Code
def reduce_features(data, num_components=9):
    # Extract features (X) and target variable (y)
    X = data.drop('custcat', axis=1)
    y = data['custcat']

    # Apply PCA
    pca = PCA(n_components=num_components)
    X_reduced = pca.fit_transform(X)
    reduced_data = pd.DataFrame(data=X_reduced, columns=[f'PC{i}' for i in range(1, num_components + 1)])
    reduced_data['custcat'] = y

    return reduced_data
```

کاهش ویژگی با این قسمت انجام شده که تعداد ویژگی بر روی ۹ قرار داده شده و جواب بهتری ارائه شده.

با تغییر criterion به entropy میزان Precision و Recall کاهش یافت.

```
Results for the default decision tree:
Confusion Matrix:
[[22 11 15 12]
 [ 6 10 10 13]
 [13  9 18 15]
 [14  9 12 11]]
Accuracy: 0.30, Precision: 0.26, Recall: 0.26

Results for the decision tree with entropy criterion:
Confusion Matrix:
[[18 14 15 13]
 [ 8  9 13  9]
 [16 13 17  9]
 [ 9 10 11 16]]
Accuracy: 0.30, Precision: 0.20, Recall: 0.23

Results for the decision tree on reduced features:
Confusion Matrix:
[[15 13 17 15]
 [ 7 13  9 10]
 [13  9 26  7]
 [ 8 10 13 15]]
Accuracy: 0.34, Precision: 0.29, Recall: 0.33
```

کد نوشته شده برای این قسمت:

```
Comment Code
def train_decision_tree_with_parameters(X_train, y_train, criterion='gini'):
    model = DecisionTreeClassifier(criterion=criterion)
    model.fit(X_train, y_train)
    return model

reduced_data = reduce_features(data_filled)
X_train_reduced, X_test_reduced, y_train_reduced, y_test_reduced = split_train_test_data(reduced_data)

# Train decision tree with default parameters
model_default = train_decision_tree_with_parameters(X_train, y_train)
y_pred_default = model_default.predict(X_test)
print("Results for the default decision tree:")
report_confusion_matrix_criteria(y_test, y_pred_default)

# Train decision tree with different criterion
model_entropy = train_decision_tree_with_parameters(X_train, y_train, criterion='entropy')
y_pred_entropy = model_entropy.predict(X_test)
print("\nResults for the decision tree with entropy criterion:")
report_confusion_matrix_criteria(y_test, y_pred_entropy)

# Train decision tree on reduced features
model_reduced = train_decision_tree_with_parameters(X_train_reduced, y_train_reduced)
y_pred_reduced = model_reduced.predict(X_test_reduced)
print("\nResults for the decision tree on reduced features:")
report_confusion_matrix_criteria(y_test_reduced, y_pred_reduced)
```