به نام خدا

گزارش تحلیلی تمرین اول هوش محاسباتی

موضوع : الگوریتم ژنتیک

استاد: دکتر حسین کارشناس

دستیاران آموزش :

رضا برزگر – علی شاه زمانی – آرمان خلیلی

اعضای گروه :

یونس ایوبی راد

پویا اسفندانی

وظیفه اول: پیش پردازش و اکتشاف داده ها

قسمت اول: بارگذاری دیتاست و بررسی مشکالت قالب بندی

```
df = pd.read_csv( filepath_or_buffer: 'data.csv', sep=';')
print(":اطلاعات کلی دیتاست")
print(df.info())
print("\nنمونه‌ای از داده‌ها:")
print(df.head())
print("\nبررسی مقادیر گمشده:")
print(df.isnull().sum())
```

.

پس از تحلیل خروجی های داده شده به این نتیجه رسیدیم که:

دیتاست ۴۴۲۴ نمونه و ۳۷ ویژگی دارد.

همه مقادیر دیتاست کامل هستند (هیچ داده گمشده‌ای وجود ندارد).

ویژگی‌ها شامل انواع داده‌ای مختلف مانند اعداد صحیح(int64) ، اعداد اعشاری (float64) و متنی (object) هستند.

قسمت دوم: پاکسازی و پیش پردازش داده ها

```
df_encoded = pd.get_dummies(df.drop( labels: 'Target', axis=1), drop_first=True)
target_mapping = {'Dropout': 0, 'Enrolled': 1, 'Graduate': 2}
df_encoded['Target'] = df['Target'].map(target_mapping)
```

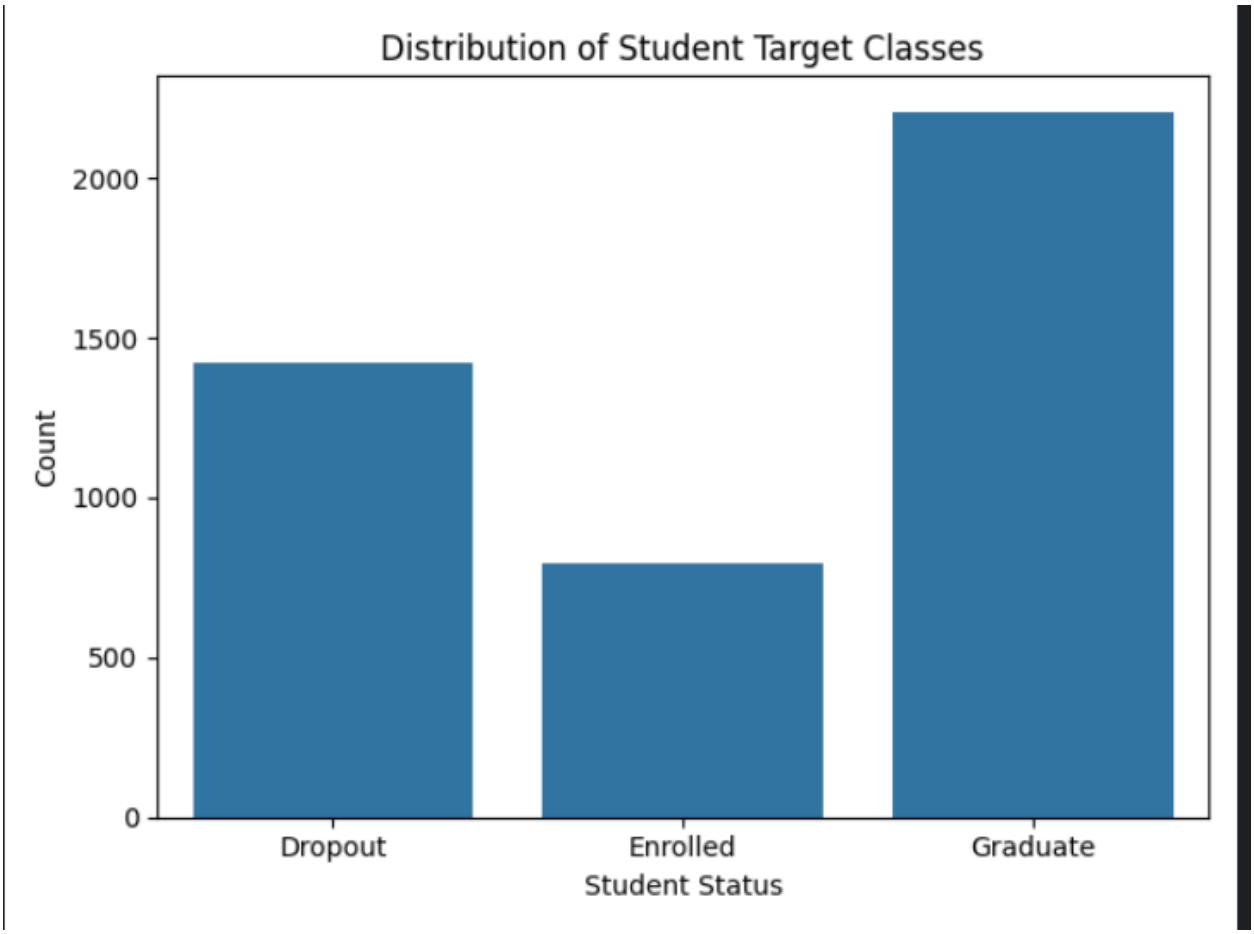
این کد ابتدا ستون Target از دیتاست حذف می‌شود تا تنها ویژگی‌ها باقی بمانند. سپس با استفاده از تابع pd.get_dummies تمامی ویژگی‌های دسته‌ای به مقادیر عددی تبدیل می‌شوند و با پارامتر drop_first=True تنها کدگذاری‌های لازم انجام می‌شود تا از هم‌خطی چندگانه جلوگیری گردد. در مرحله بعد، یک دیکشنری به نام target_mapping ساخته می‌شود که مقادیر متنی ستون Targetمانند"Dropout"، "Enrolled"و "Graduate" را به مقادیر عددی ۰، ۱ و ۲ تبدیل می‌کند. در نهایت با استفاده از تابع map مقادیر متنی ستون Target به مقادیر عددی در دیتافریم جدید df_encoded اختصاص داده می‌شود تا دیتاست آماده استفاده در مدل‌های یادگیری ماشین شود.

قسمت سوم: انجام تحلیل اکتشافی داده EDA و ترسیم الگوهای کلیدی.

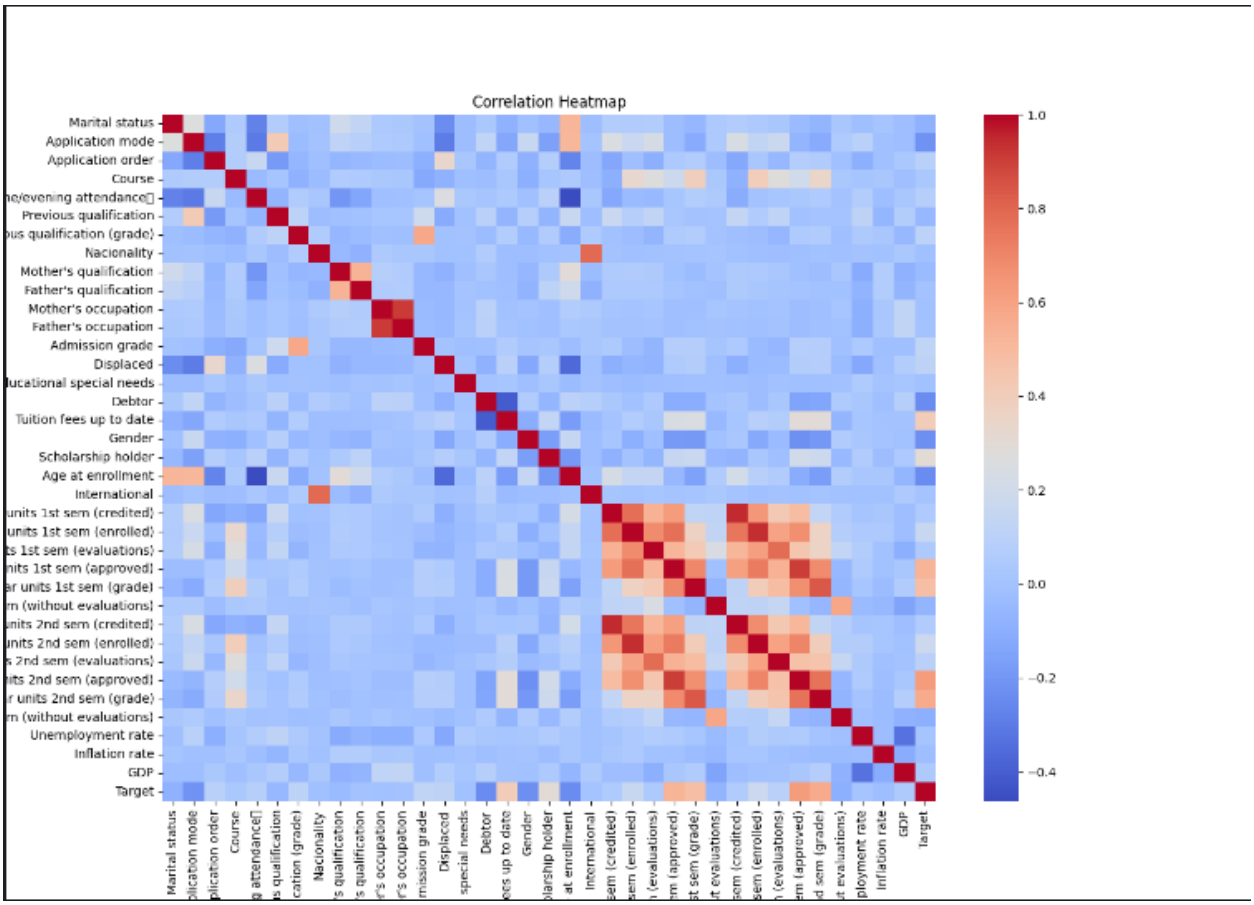
```
sns.countplot(x='Target', data=df_encoded)
plt.title('Distribution of Student Target Classes')
plt.xticks(ticks=[0, 1, 2], labels=['Dropout', 'Enrolled', 'Graduate'])
plt.xlabel('Student Status')
plt.ylabel('Count')
plt.tight_layout()
plt.show()

plt.figure(figsize=(14, 10))
sns.heatmap(df_encoded.corr(), cmap='coolwarm', cbar=True)
plt.title('Correlation Heatmap')
plt.show()
```

این کد شامل دو بخش اصلی است که برای تجزیه و تحلیل داده‌ها استفاده می‌شود. ابتدا با استفاده از sns.countplot یک نمودار شمارش از توزیع کلاس‌های مختلف ستون Target(که نمایانگر وضعیت تحصیلی دانشجویان است) ایجاد می‌شود. در اینجا محور x نشان‌دهنده کلاس‌های

مختلف مانند"Dropout"، "Enrolled"و "Graduate"است.



بخش دوم کد یک نقشه حرارتی از ماتریس همبستگی ویژگی‌ها با استفاده از sns.heatmap رسم می‌کند. این نقشه نشان‌دهنده ارتباط بین ویژگی‌های مختلف دیتاست است و از رنگ‌های coolwarm برای نمایش شدت همبستگی استفاده می‌شود



تحلیل هیت مپ همبستگی

همبستگی‌های قوی :

Curricular units 1st/2nd sem (approved و Curricular units 1st/2nd sem (grade) با Targetهمبستگی مثبت قوی (0.6–0.8) دارن: دانشجوهای با واحدهای پاس‌شده و نمرات بالا احتمال فارغ‌التحصیلی بیشتری دارن.

Debtor و Tuition fees up to date همبستگی منفی قوی (−0.6): بدهکارها معمولاً شهریه‌شون به‌روز نیست.

همبستگی‌های متوسط :

Admission grade و Scholarship holder با Targetهمبستگی مثبت (0.3): نمره پذیرش بالا و بورسیه احتمال فارغ‌التحصیلی رو زیاد می‌کنه.

Age at enrollment با Targetهمبستگی منفی (−0.3): دانشجوهای مسن‌تر احتمال انصراف بیشتری دارن.

همبستگی‌های ضعیف :

Unemployment rate، Inflation rate، GDP، Marital status، Course با Targetهمبستگی نزدیک به 0 دارن: تأثیر کمی دارن.

الگوها :

عملکرد تحصیلی (واحدهای پاس‌شده و نمرات) مهم‌ترین عامل برای پیش‌بینی Targetاست.

عوامل اقتصادی و اجتماعی (مثل تحصیلات والدین) تأثیر کمی دارن.

قسمت چهارم:تقسیم دیتاست

```
X = df_encoded.drop( labels: 'Target', axis=1)
y = df_encoded['Target']
X_train, X_test, y_train, y_test = train_test_split(
    *arrays: X, y, test_size=0.2, stratify=y, random_state=42
)
print("Train set shape:", X_train.shape)
print("Test set shape:", X_test.shape)
```

در این بخش ابتدا ویژگی‌ها (X) و برچسب هدف (y) از هم جدا می‌شوند، سپس داده‌ها به دو بخش آموزش و آزمون تقسیم می‌شوند به‌طوری که ۲۰٪ از داده‌ها برای آزمون استفاده می‌شود و توزیع کلاس‌ها در هر دو بخش با استفاده از stratify=y حفظ می‌شود.

```
Train set shape: (3539, 36)
Test set shape: (885, 36)
```

وظیفه دوم) فازی سازی ویژگی ها

تابع اول:

این تابع یک عضویت مثلثی برای فازی‌سازی مقادیر می‌سازد. با سه نقطه (شروع، قله، پایان) کار می‌کنه و درجه عضویت رو برای یه ورودی محاسبه می‌کنه. اگه شروع و پایان برابر باشن، خروجی باینری (۱ یا ۰) می‌ده، وگرنه با محاسبه شیب‌های چپ و راست، یه مقدار بین ۰ و ۱ تولید می‌کنه که در قله به ۱ می‌رسه و به‌سمت صفر کاهش پیدا می‌کنه.

```
def triangular_membership(x, a, b, c):  9 usages
    if a == c:
        return np.ones_like(x) if x == a else np.zeros_like(x)
    b_minus_a = b - a if b - a != 0 else 0.00000001
    c_minus_b = c - b if c - b != 0 else 0.00000001
    left = (x - a) / b_minus_a
    right = (c - x) / c_minus_b
    return np.maximum(np.minimum(left, right), 0)
```

تابع دوم:

این تابع مقادیر پیوسته رو به سه دسته فازی (کم، متوسط، زیاد) تبدیل می‌کنه. با استفاده از تابع عضویت مثلثی، محدوده‌ها رو از حداقل تا میانگین، از حداقل تا حداکثر، و از میانگین تا حداکثر تعریف می‌کنه. خروجی یه جدول با سه ستون برای هر دسته فازی هست که برای تحلیل‌های بعدی قابل استفاده است.

```
def fuzzify_continuous_feature(feature_values, feature_name):  2 usages
    low = triangular_membership(feature_values, feature_values.min(), feature_values.min(), feature_values.mean())
    medium = triangular_membership(feature_values, feature_values.min(), feature_values.mean(), feature_values.max())
    high = triangular_membership(feature_values, feature_values.mean(), feature_values.max(), feature_values.max())
    fuzzy_df = pd.DataFrame({
        f'{feature_name}_Low': low,
        f'{feature_name}_Medium': medium,
        f'{feature_name}_High': high
    })
    return fuzzy_df
```

تابع سوم:

این تابع مقادیر باینری رو به دو دسته فازی (خیر، آری) تبدیل می‌کنه. مقدار مکمل (۱ منهای مقدار) رو برای "خیر" و مقدار اصلی رو برای "آری" در نظر می‌گیره. خروجی یه جدول با دو ستون برای این دسته‌ها هست که برای ترکیب با داده‌های فازی‌شده مناسبه.

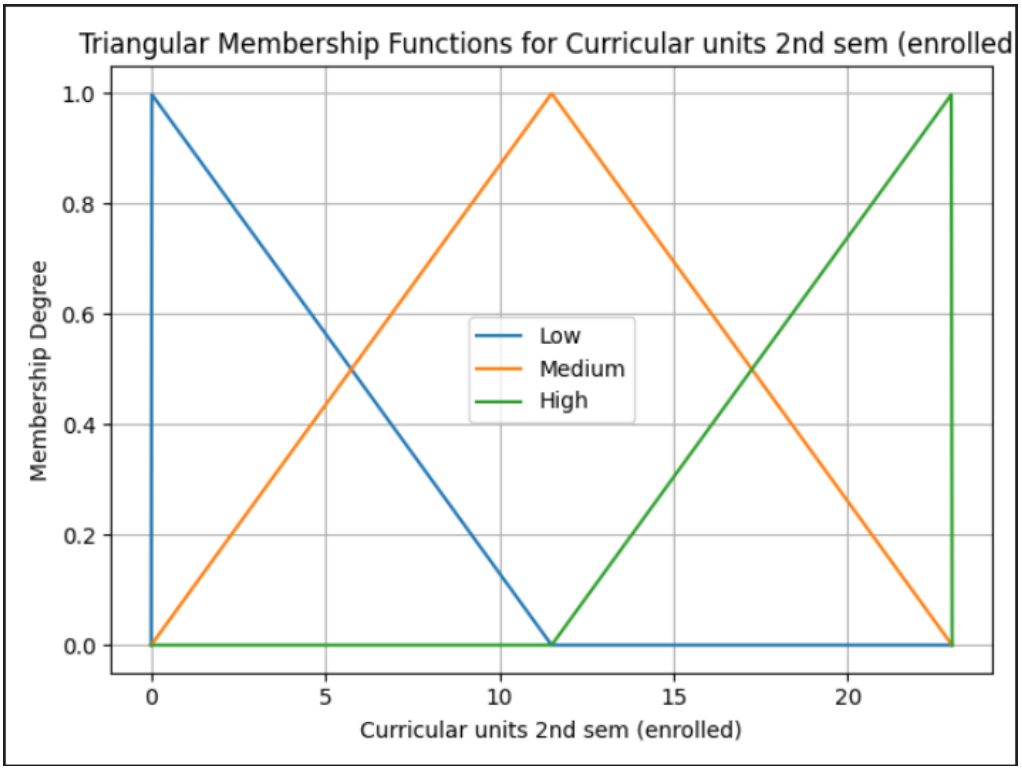
```
def fuzzify_binary_feature(feature_series, feature_name):  2 usages
    fuzzy_df = pd.DataFrame({
        f'{feature_name}_No': 1 - feature_series,
        f'{feature_name}_Yes': feature_series
    })
    return fuzzy_df
```

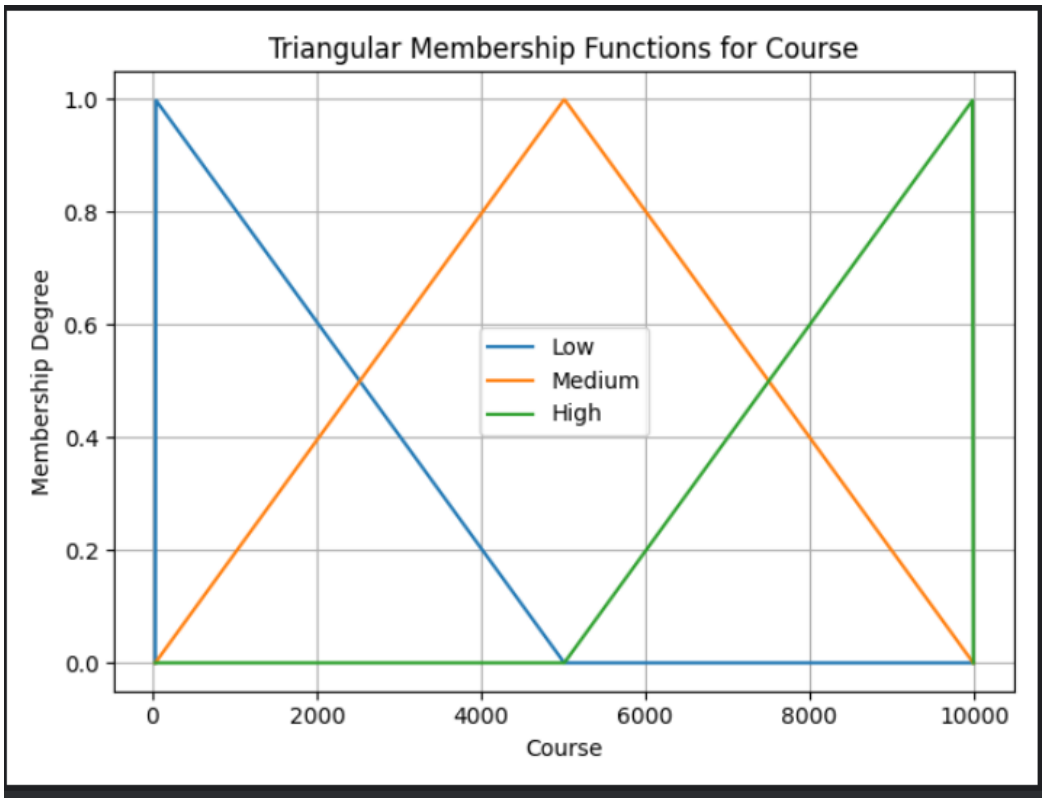
قسمت اصلی کد:

```
fuzzy_cont_train = [fuzzify_continuous_feature(X_train[feat], feat) for feat in cont_features]
fuzzy_cont_train_df = pd.concat(fuzzy_cont_train, axis=1)
fuzzy_bin_train = [fuzzify_binary_feature(X_train[feat], feat) for feat in binary_features]
fuzzy_bin_train_df = pd.concat(fuzzy_bin_train, axis=1)
X_train_fuzzy = pd.concat( objs: [fuzzy_cont_train_df, fuzzy_bin_train_df], axis=1)
fuzzy_cont_test = [fuzzify_continuous_feature(X_test[feat], feat) for feat in cont_features]
fuzzy_cont_test_df = pd.concat(fuzzy_cont_test, axis=1)
fuzzy_bin_test = [fuzzify_binary_feature(X_test[feat], feat) for feat in binary_features]
fuzzy_bin_test_df = pd.concat(fuzzy_bin_test, axis=1)
X_test_fuzzy = pd.concat( objs: [fuzzy_cont_test_df, fuzzy_bin_test_df], axis=1)
```

این کد فرآیند فازی‌سازی ویژگی‌های پیوسته و باینری رو برای داده‌های آموزشی و آزمونی انجام می‌ده. ابتدا لیست ویژگی‌های پیوسته و باینری تعریف می‌شه. سپس با استفاده از تابع فازی‌سازی پیوسته، هر ویژگی پیوسته به سه دسته فازی (کم، متوسط، زیاد) تبدیل و نتایج در یه جدول ترکیب می‌شن؛ همین‌طور، ویژگی‌های باینری با تابع فازی‌سازی باینری به دو دسته (خیر، آری) تبدیل و ترکیب می‌شن. در نهایت، داده‌های فازی‌شده پیوسته و باینری برای هر دو مجموعه آموزشی و آزمونی ترکیب می‌شن تا دو مجموعه داده فازی نهایی برای استفاده در مراحل بعدی (مثل استخراج قوانین) آماده بشن.

رسم نمودارهای فازی





وظیفه سوم: استخراج قوانین فازی

تابع دو ورودی، یعنی مجموعه ویژگی‌های فازی‌شده (X) و برچسب‌های کلاس(y) ، دریافت می‌کند. در ابتدا، یک دیکشنری به نام rules ایجاد می‌شود که هر شرط (به‌صورت تاپل از مقادیر فازی) را به‌عنوان کلید ذخیره می‌کند و اطلاعاتی مثل تعداد وقوع و لیست برچسب‌ها را به‌عنوان مقدار نگه می‌دارد. با پیمایش داده‌ها، اگر شرطی قبلاً وجود داشته باشد، شمارنده و لیست برچسب‌ها به‌روزرسانی می‌شوند، وگرنه یک ورودی جدید با مقدار اولیه ایجاد می‌شود. در مرحله بعد، برای هر شرط منحصربه‌فرد، با استفاده از سری پانداس، فراوانی برچسب‌ها محاسبه شده، برچسب غالب (با بیشترین تکرار) به‌عنوان خروجی انتخاب می‌شود، و وزن اطمینان (نسبت برچسب غالب به کل برچسب‌ها) محاسبه می‌شود. در نهایت، یک لیست از تاپل‌ها شامل شرط، برچسب نهایی و وزن اطمینان به‌عنوان قوانین فازی برگردانده می‌شود.

```python
def extract_rules(X, y):  1 usage
    rules = {}
    for i in range(len(X)):
        condition = tuple(X.iloc[i])
        label = y.iloc[i]
        if condition in rules:
            rules[condition]['count'] += 1
            rules[condition]['labels'].append(label)
        else:
            rules[condition] = {'count': 1, 'labels': [label]}
    final_rules = []
    for c, data in rules.items():
        label_counts = pd.Series(data['labels']).value_counts()
        final_label = label_counts.idxmax()
        confidence = label_counts.max() / sum(label_counts)
        final_rules.append((c, final_label, confidence))
    return final_rules
```

نمونه ای از قوانین

قانون 1:

اگر:

Admission grade_Low = 0.259648370117311

Admission grade_Medium = 0.740351629882689

Admission grade_High = 0.0

Unemployment rate_Low = 0.0

Unemployment rate_Medium = 0.49785011345773905

Unemployment rate_High = 0.502149886542261

Application mode_Low = 0.0

Application mode_Medium = 0.3651626597485297

Application mode_High = 0.6348373402514704

Course_Low = 0.0

Course_Medium = 0.6605186209864056

Course_High = 0.3394813790135944

Previous qualification_Low = 0.0

Previous qualification_Medium = 0.10446230250972231

Previous qualification_High = 0.8955376974902777

Curricular units 1st sem (enrolled)_Low = 0.0

Curricular units 1st sem (enrolled)_Medium = 0.7600836185048896

Curricular units 1st sem (enrolled)_High = 0.23991638149511033

Curricular units 2nd sem (grade)_Low = 0.0

Curricular units 2nd sem (grade)_Medium = 0.8093101549119738

Curricular units 2nd sem (grade)_High = 0.19068984508802625

Curricular units 2nd sem (enrolled)_Low = 0.0

Curricular units 2nd sem (enrolled)_Medium = 0.775050539083558

Curricular units 2nd sem (enrolled)_High = 0.22494946091644205

Gender_No = 0.0

Gender_Yes = 1.0

Debtor_No = 1.0

Debtor_Yes = 0.0

Scholarship holder_No = 1.0

Scholarship holder_Yes = 0.0

آنگاه: کلاس = Graduate

وزن اطمینان: 1.00

قانون 2:

اگر:

Admission grade_Low = 0.2157292056327445

Admission grade_Medium = 0.7842707943672554

Admission grade_High = 0.0

Unemployment rate_Low = 0.0

Unemployment rate_Medium = 0.757597998740038

Unemployment rate_High = 0.24240200125996197

Application mode_Low = 0.0

Application mode_Medium = 0.4694948482481096

Application mode_High = 0.5305051517518905

Course_Low = 0.0

Course_Medium = 0.7552543594545753

Course_High = 0.24474564054542466

Previous qualification_Low = 0.0

Previous qualification_Medium = 0.0

Previous qualification_High = 0.0

Curricular units 1st sem (enrolled)_Low = 0.20195733549812833

Curricular units 1st sem (enrolled)_Medium = 0.7980426645018717

Curricular units 1st sem (enrolled)_High = 0.0

Curricular units 2nd sem (grade)_Low = 0.0

Curricular units 2nd sem (grade)_Medium = 0.0

Curricular units 2nd sem (grade)_High = 0.0

Curricular units 2nd sem (enrolled)_Low = 0.19703226391977133

Curricular units 2nd sem (enrolled)_Medium = 0.8029677360802286

Curricular units 2nd sem (enrolled)_High = 0.0

Gender_No = 0.0

Gender_Yes = 1.0

Debtor_No = 0.0

Debtor_Yes = 1.0

Scholarship holder_No = 1.0

Scholarship holder_Yes = 0.0

آنگاه: کلاس = Dropout

وزن اطمینان: 1.00

وظیفه چهارم: انتخاب قوانین با استفاده از الگوریتم ژنتیک

توضیح توابع

```python
def evaluate_rule_set(rule_set, X_val, y_val):  1 usage
    correct = 0
    for i in range(len(X_val)):
        sample = tuple(X_val.iloc[i])
        true_label = y_val.iloc[i]
        scores = {0: 0, 1: 0, 2: 0}
        for condition, label, conf in rule_set:
            degree = 1.0
            for j, val in enumerate(sample):
                similarity = 1 - abs(condition[j] - val)
                degree *= max(0, similarity)
            scores[label] += degree * conf
        predicted_label = max(scores, key=scores.get) if max(scores.values()) > 0 else np.random.choice([0, 1, 2])
        if predicted_label == true_label:
            correct += 1
    acc = correct / len(X_val)
    penalty = 0.0001 * len(rule_set)
    return acc - penalty
```

```python
def evaluate_population_parallel(population, X_val, y_val, max_workers=4):  2 usages
    with ThreadPoolExecutor(max_workers=max_workers) as executor:
        futures = [executor.submit(evaluate_rule_set, *args: chromo, X_val, y_val) for chromo in population]
        return [f.result() for f in futures]

def tournament_selection(population, scores, k=3):  2 usages
    selected = random.sample(list(zip(population, scores)), k)
    selected.sort(key=lambda x: x[1], reverse=True)
    return selected[0][0]

def crossover(parent1, parent2):  1 usage
    if random.random() > CROSSOVER_RATE:
        return parent1[:], parent2[:]
    cut = random.randint( a: 1, min(len(parent1), len(parent2)) - 1)
    child1 = parent1[:cut] + parent2[cut:]
    child2 = parent2[:cut] + parent1[cut:]
    return child1, child2

def mutate(chromosome, all_rules):  2 usages
    if random.random() < MUTATION_RATE:
        idx = random.randint( a: 0, len(chromosome)-1)
        new_rule = random.choice(all_rules)
        chromosome[idx] = new_rule
    return chromosome
```

```python
def genetic_algorithm_optimized(initial_rules, X_val, y_val):  1 usage
    rule_len = min(15, len(initial_rules))
    population = [random.sample(initial_rules, rule_len) for _ in range(POP_SIZE)]

    for generation in range(NUM_GENERATIONS):
        scores = evaluate_population_parallel(population, X_val, y_val, MAX_WORKERS)
        print(f"نسل{generation+1} = بهترین دقت :{max(scores):.3f}")
        sorted_pop = [x for _, x in sorted(zip(scores, population), key=lambda x: x[0], reverse=True)]
        next_generation = sorted_pop[:ELITE_COUNT]

        while len(next_generation) < POP_SIZE:
            p1 = tournament_selection(population, scores)
            p2 = tournament_selection(population, scores)
            c1, c2 = crossover(p1, p2)
            next_generation.extend([
                mutate(c1, initial_rules),
                mutate(c2, initial_rules)
            ])

        population = next_generation[:POP_SIZE]

    final_scores = evaluate_population_parallel(population, X_val, y_val, MAX_WORKERS)
    best_idx = np.argmax(final_scores)
    best_rules = population[best_idx]
    print(f"\n :بهترین دقت نهایی {final_scores[best_idx]:.3f}")
    return best_rules
```

تابع evaluate_rule_set

عملکرد یک مجموعه قوانین فازی رو برای طبقه‌بندی ارزیابی می‌کنه. این تابع سه ورودی داره rule_set :مجموعه قوانین شامل شرط، برچسب و وزن اطمینان، X_valو y_val برای هر نمونه در داده‌ها، ابتدا شرط به‌صورت تاپل استخراج شده و برچسب واقعی ذخیره می‌شه. یک دیکشنری امتیاز برای سه کلاس (۰، ۱، ۲) تعریف می‌شه. سپس برای هر قانون، شباهت فازی با ضرب تفاوت مطلق مقادیر در حداکثر صفر محاسبه شده و با وزن اطمینان جمع می‌شه. کلاس با بیشترین امتیاز پیش‌بینی می‌شه، و اگه امتیازی وجود نداشته باشه، پیش‌بینی تصادفی انجام می‌شه. تعداد پیش‌بینی‌های درست شمرده شده و دقت با تقسیم بر تعداد کل نمونه‌ها محاسبه می‌شه. یه جریمه کوچک بر اساس تعداد قوانین اعمال می‌شه تا از پیچیدگی زیاد جلوگیری کنه، و در نهایت مقدار تعدیل‌شده دقت برگردانده می‌شه.

تابع evaluate_population_parallel

این تابع جمعیت رو موازی ارزیابی می‌کنه. با ThreadPoolExecutor، تابع ارزیابی رو برای هر کروموزوم اجرا کرده و امتیازات رو برمی‌گردونه.

تابع tournament_selection

روش انتخاب تورنمنتی رو اجرا می‌کنه k .کروموزوم تصادفی رو با امتیازشون انتخاب، مرتب می‌کنه و بهترین رو برمی‌گردونه.

تابع crossover

تقاطع رو برای دو والد اعمال می‌کنه. با احتمال مشخص، یه نقطه برش انتخاب می‌کنه و دو فرزند می‌سازه، وگرنه والدین رو برمی‌گردونه.

تابع mutate

جهش رو روی کروموزوم اعمال می‌کنه. با احتمال مشخص، یه قانون رو تصادفی جایگزین می‌کنه و کروموزوم رو برمی‌گردونه.

تابع genetic_algorithm_optimized

الگوریتم ژنتیک را برای بهینه‌سازی مجموعه قوانین فازی اجرا می‌کند. این تابع با دریافت قوانین اولیه، داده‌های فازی‌شده و برچسب‌ها، ابتدا یک جمعیت اولیه با نمونه‌گیری تصادفی از قوانین (تا حداکثر ۱۵ قانون) ایجاد می‌کند. در هر نسل از تعداد مشخص‌شده، امتیازات جمعیت با ارزیابی موازی محاسبه شده و بهترین دقت چاپ می‌شود؛ سپس جمعیت بر اساس امتیازات مرتب شده و تعداد محدودی از بهترین‌ها (نخبگان) به نسل بعدی منتقل می‌شوند. تا زمانی که جمعیت کامل نشده، با انتخاب تورنمنتی دو والد، اعمال تقاطع و جهش، کروموزوم‌های جدید تولید و به نسل اضافه می‌شوند. در پایان، پس از اتمام نسل‌ها، امتیازات نهایی محاسبه شده و کروموزوم با بهترین امتیاز به‌عنوان مجموعه قوانین بهینه انتخاب و با دقت نهایی چاپ و برگردانده می‌شود.

چند نمونه از قوانین(به دلیل طولانی بودن قوانین فقط دوتای اول آن به نمایش گذاشته می‌شوند)

چند قانون بهینه‌شده نهایی:

قانون 1:

Admission grade_Low = 0.4196396121682309

Admission grade_Medium = 0.580360387831769

Admission grade_High = 0.0

Unemployment rate_Low = 0.5477541052272871

Unemployment rate_Medium = 0.452245894772713

Unemployment rate_High = 0.0

Application mode_Low = 0.0

Application mode_Medium = 0.4694948482481096

Application mode_High = 0.5305051517518905

Course_Low = 0.09491811975285559

Course_Medium = 0.9050818802471444

Course_High = 0.0

Previous qualification_Low = 0.0

Previous qualification_Medium = 0.0

Previous qualification_High = 0.0

Curricular units 1st sem (enrolled)_Low = 0.04234880259775401

Curricular units 1st sem (enrolled)_Medium = 0.957651197402246

Curricular units 1st sem (enrolled)_High = 0.0

Curricular units 2nd sem (grade)_Low = 0.0

Curricular units 2nd sem (grade)_Medium = 0.0

Curricular units 2nd sem (grade)_High = 0.0

Curricular units 2nd sem (enrolled)_Low = 0.0364387167037256

Curricular units 2nd sem (enrolled)_Medium = 0.9635612832962744

Curricular units 2nd sem (enrolled)_High = 0.0

Gender_No = 1.0

Gender_Yes = 0.0

Debtor_No = 1.0

Debtor_Yes = 0.0

Scholarship holder_No = 1.0

Scholarship holder_Yes = 0.0

کلاس: Dropout

وزن اطمینان: 1.00

قانون 2:

Admission grade_Low = 0.5608083551543369

Admission grade_Medium = 0.4391916448456631

Admission grade_High = 0.0

Unemployment rate_Low = 0.12063298238639177

Unemployment rate_Medium = 0.8793670176136082

Unemployment rate_High = 0.0

Application mode_Low = 0.0

Application mode_Medium = 0.0

Application mode_High = 0.0

Course_Low = 0.0

Course_Medium = 0.4306967369247346

Course_High = 0.5693032630752654

Previous qualification_Low = 0.0

Previous qualification_Medium = 0.0

Previous qualification_High = 0.0

Curricular units 1st sem (enrolled)_Low = 0.0

Curricular units 1st sem (enrolled)_Medium = 0.9627725834395269

Curricular units 1st sem (enrolled)_High = 0.03722741656047308

Curricular units 2nd sem (grade)_Low = 0.0

Curricular units 2nd sem (grade)_Medium = 0.705478564625931

Curricular units 2nd sem (grade)_High = 0.2945214353740691

Curricular units 2nd sem (enrolled)_Low = 0.0

Curricular units 2nd sem (enrolled)_Medium = 0.8942890835579516

Curricular units 2nd sem (enrolled)_High = 0.10571091644204851

Gender_No = 1.0

Gender_Yes = 0.0

Debtor_No = 1.0

Debtor_Yes = 0.0

Scholarship holder_No = 0.0

Scholarship holder_Yes = 1.0

کلاس: Graduate

وزن اطمینان: 1.00

وظیفه :۵ استنتاج فازی برای طبقه بندی

```
def fuzzy_inference(rule_set, X_val):  3 usages
    predictions = []
    for i in range(len(X_val)):
        sample = tuple(X_val.iloc[i])
        scores = {0: 0, 1: 0, 2: 0}
        for condition, label, conf in rule_set:
            degree = 1.0
            for j, val in enumerate(sample):
                similarity = 1 - abs(condition[j] - val)
                degree *= max(0, similarity)
            scores[label] += degree * conf
        predicted_label = max(scores, key=scores.get) if max(scores.values()) > 0 else np.random.choice([0, 1, 2])
        predictions.append(predicted_label)
    return predictions
y_pred = fuzzy_inference(best_rules, X_test_fuzzy)

errors = sum(1 for i in range(len(y_test)) if y_test.iloc[i] != y_pred[i])
error_rate = (errors / len(y_test)) * 100
print(f"\n :خطای کلی {errors} مورد خطا از {len(y_test)}) نمونه {error_rate:.2f}(خطا %")
```

این کد فرآیند استنتاج فازی و محاسبه خطای پیش‌بینی را انجام می‌دهد. تابع fuzzy_inference با دریافت مجموعه قوانین و داده‌های فازی‌شده، برای هر نمونه پیش‌بینی کلاس را انجام می‌دهد؛ ابتدا یک نمونه به تاپل تبدیل شده و امتیاز هر کلاس (۰، ۱، ۲) محاسبه می‌شود، به این صورت که شباهت فازی بین نمونه و هر قانون (با ضرب تفاوت مقادیر در حداکثر صفر) با وزن اطمینان جمع‌آوری شده و کلاس با بیشترین امتیاز انتخاب می‌شود، یا در صورت نبود امتیاز، تصادفی پیش‌بینی می‌شود. پیش‌بینی‌ها در یک لیست ذخیره می‌شوند. سپس با فراخوانی تابع روی داده‌های آزمونی فازی‌شده، پیش‌بینی‌ها (y_pred) تولید می‌شوند. در نهایت، خطاها با مقایسه پیش‌بینی‌ها و برچسب‌های واقعی (y_test) شمرده شده، نرخ خطا به درصد محاسبه می‌شود و نتیجه به‌صورت تعداد خطاها و درصد خطا چاپ می‌شود.

نتیجه پیش‌بینی

```
تعداد قوانین نهایی انتخابشده: 15

خطای کلی: 312 مورد خطا از 885 نمونه (%35.25 خطا)
```

وظیفه ششم: ارزیابی مدل

```python
def evaluate_model(y_true, y_pred, title="Model Evaluation"):  1 usage
    pred_counts = pd.Series(y_pred).value_counts()
    print(f"\n توزیع پیش‌بینی‌ها در {title}:")
    for cls, count in pred_counts.items():
        print(f" کلاس{['Dropout', 'Enrolled', 'Graduate'][cls]}: {count}نمونه ")
    accuracy = accuracy_score(y_true, y_pred)
    precision, recall, f1, _ = precision_recall_fscore_support(
        y_true, y_pred, average='weighted', zero_division=0
    )
    print(f"\n {title}")
    print(f"Accuracy: {accuracy:.3f}")
    print(f"Precision (weighted): {precision:.3f}")
    print(f"Recall (weighted): {recall:.3f}")
    print(f"F1-Score (weighted): {f1:.3f}")
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Dropout', 'Enrolled', 'Graduate'],
                yticklabels=['Dropout', 'Enrolled', 'Graduate'])
    plt.title(f'Confusion Matrix - {title}')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.tight_layout()
    plt.show()

    return accuracy, precision, recall, f1
```

تابع evaluate_modelبرای بررسی کارایی یک مدل طبقه‌بندی طراحی شده و برچسب‌های واقعی، پیش‌بینی‌شده و یک عنوان دلخواه) به‌صورت پیش‌فرض "Model Evaluation" را می‌گیرد. ابتدا تعداد پیش‌بینی‌ها برای هر دسته) مانندDropout ، Enrolled، Graduateرا شمرده و به‌صورت متنی نشان می‌دهد تا توزیع پیش‌بینی‌ها مشخص شود. سپس معیارهای ارزیابی شامل دقت کلی، دقت جزئی، بازخوانی و امتیاز F1 (با میانگین وزنی برای تعادل دسته‌ها و جلوگیری از خطای تقسیم بر صفر) را با ابزارهای موجود محاسبه و چاپ می‌کند. بعد، جدول درهم‌ریختگی را می‌سازد که تعداد پیش‌بینی‌های درست و نادرست را برای هر دسته نشان می‌دهد و آن را با یک نمودار گرافیکی ترسیم می‌کند؛ محور افقی برای پیش‌بینی‌شده، محور عمودی برای واقعی، و برچسب‌ها با نام دسته‌ها تنظیم می‌شوند. در نهایت، این معیارها) دقت، دقت جزئی، بازخوانی و امتیاز (F1 را به‌صورت یک مجموعه برمی‌گرداند تا برای تحلیل‌های بعدی استفاده شود. این روش هم اعداد و هم نمودار را برای درک بهتر کارایی مدل ارائه می‌دهد.

```python
def analyze_class_distribution(y, dataset_name="Dataset"):  2 usages
    class_counts = pd.Series(y).value_counts()
    total = len(y)
    print(f"\n توزیع کلاس‌ها در {dataset_name}:")
    for cls, count in class_counts.items():
        percentage = (count / total) * 100
        print(f" کلاس{['Dropout', 'Enrolled', 'Graduate'][cls]}: {count}) نمونه {percentage:.2f}%)")
```

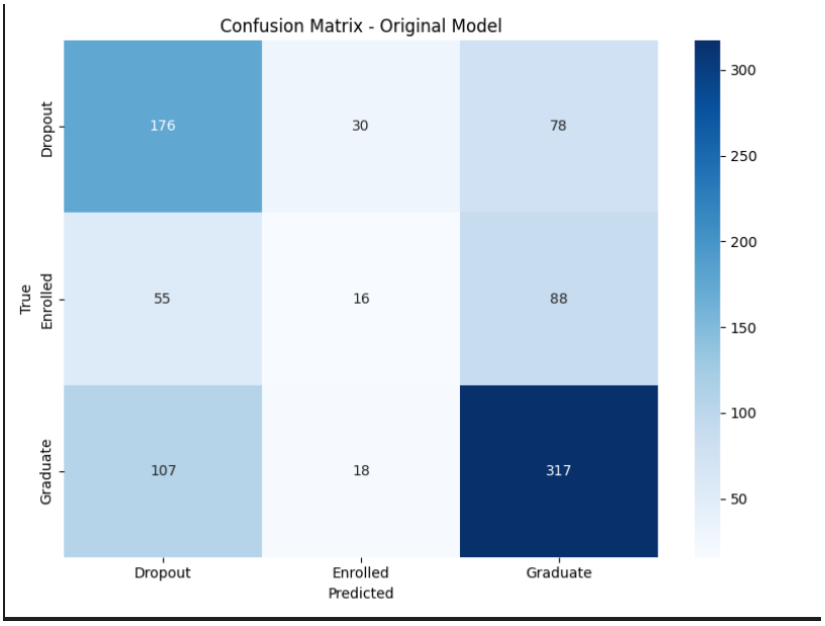
تابع analyze_class_distribution برای بررسی توزیع کلاس‌ها در یک مجموعه داده طراحی شده و برچسب‌های کلاس‌ها yو یک نام دلخواه برای مجموعه داده به صورت پیش فرض Dataset را دریافت می‌کند. ابتدا با استفاده از pd.Series(y).value_counts تعداد نمونه‌ها برای هر کلاس ۰، ۱، ۲ که به ترتیب بهDropout ، Enrolledو Graduate نگاشت شده‌اند محاسبه می‌شود. سپس با تقسیم تعداد هر کلاس بر کل نمونه‌ها و ضرب در ۱۰۰، درصد هر کلاس به دست آمده و به صورت متنی چاپ می‌شود. این خروجی شامل تعداد نمونه‌ها و درصد آن‌ها برای هر کلاس است، که به تحلیلگر کمک می‌کند تا تعادل یا عدم تعادل کلاس‌ها را در مجموعه داده موردنظر شناسایی کند.

نتیجه نهایی

```
 :Training Set توزیع کلاس‌ها در
(%49.93) نمونه Graduate: 1767 کلاس
(%32.13) نمونه Dropout: 1137 کلاس
(%17.94) نمونه Enrolled: 635 کلاس

 :Test Set توزیع کلاس‌ها در
(%49.94) نمونه Graduate: 442 کلاس
(%32.09) نمونه Dropout: 284 کلاس
(%17.97) نمونه Enrolled: 159 کلاس
```

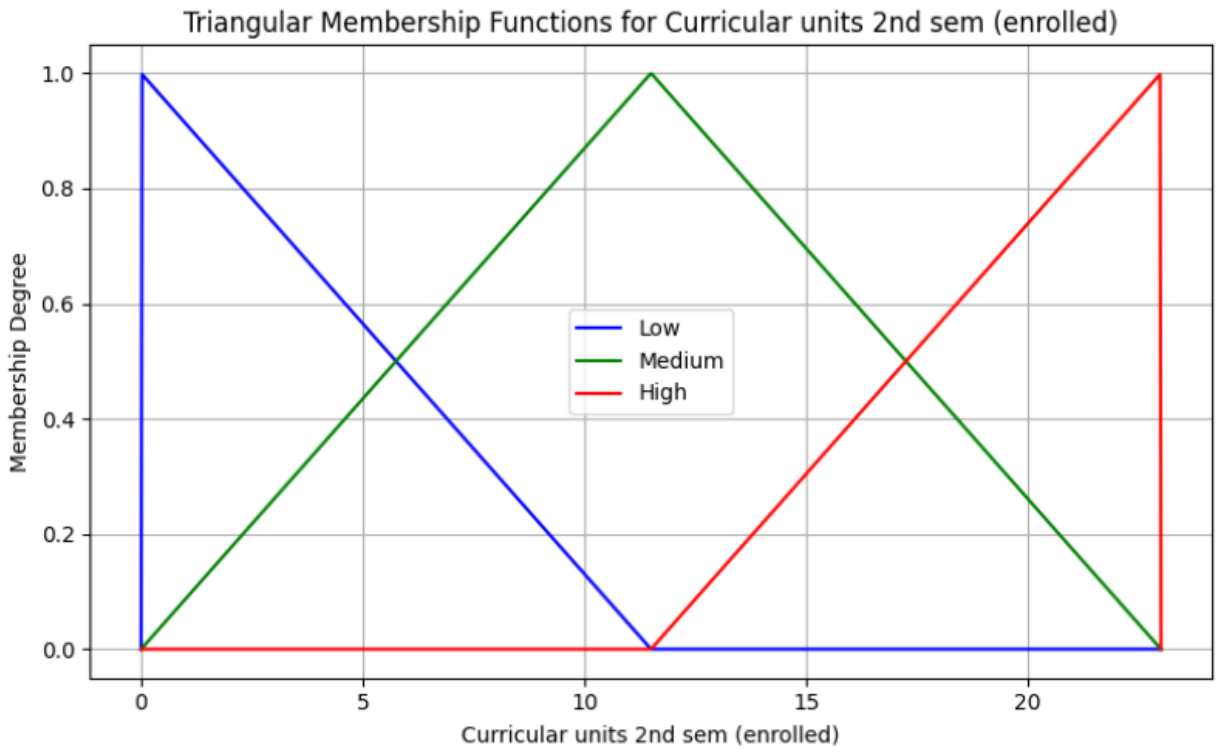
عدم توازن کلاس ها وجود نداشت.

ماتریس اشفتگی



وظیفه :۷ تفسیر و بصری سازی

نمودار عضویت



نمونه ای از فعال سازی و توضیحات آن

نمونه 1 (کلاس واقعی: Graduate, کلاس پیش‌بینی‌شده: Graduate)

قوانین فعال‌شده:

قانون 14: درجه فعال‌سازی = 0.050, کلاس = Graduate, وزن اطمینان = 1.00

مقدمه:

Admission grade_Low = 0.049

Admission grade_Medium = 0.951

Admission grade_High = 0.000

Unemployment rate_Low = 0.196

Unemployment rate_Medium = 0.804

Unemployment rate_High = 0.000

Application mode_Low = 0.000

Application mode_Medium = 0.000

Application mode_High = 0.000

Course_Low = 0.000

Course_Medium = 0.431

Course_High = 0.569

Previous qualification_Low = 0.000

Previous qualification_Medium = 0.000

Previous qualification_High = 0.000

Curricular units 1st sem (enrolled)_Low = 0.000

Curricular units 1st sem (enrolled)_Medium = 0.912

Curricular units 1st sem (enrolled)_High = 0.088

Curricular units 2nd sem (grade)_Low = 0.000

Curricular units 2nd sem (grade)_Medium = 0.743

Curricular units 2nd sem (grade)_High = 0.257

Curricular units 2nd sem (enrolled)_Low = 0.000

Curricular units 2nd sem (enrolled)_Medium = 0.894

Curricular units 2nd sem (enrolled)_High = 0.106

Gender_No = 1.000

Gender_Yes = 0.000

Debtor_No = 1.000

Debtor_Yes = 0.000

Scholarship holder_No = 0.000

Scholarship holder_Yes = 1.000

قانون 3: درجه فعال‌سازی = 0.000, کلاس = Graduate, وزن اطمینان = 1.00

Admission grade_Low = 0.937

Admission grade_Medium = 0.063

Admission grade_High = 0.000

Unemployment rate_Low = 0.000

Unemployment rate_Medium = 0.152

Unemployment rate_High = 0.848

Application mode_Low = 0.000

Application mode_Medium = 0.469

Application mode_High = 0.531

Course_Low = 0.095

Course_Medium = 0.905

Course_High = 0.000

Previous qualification_Low = 0.000

Previous qualification_Medium = 0.000

Previous qualification_High = 0.000

Curricular units 1st sem (enrolled)_Low = 0.042

Curricular units 1st sem (enrolled)_Medium = 0.958

Curricular units 1st sem (enrolled)_High = 0.000

Curricular units 2nd sem (grade)_Low = 0.000

Curricular units 2nd sem (grade)_Medium = 0.865

Curricular units 2nd sem (grade)_High = 0.135

Curricular units 2nd sem (enrolled)_Low = 0.036

Curricular units 2nd sem (enrolled)_Medium = 0.964

Curricular units 2nd sem (enrolled)_High = 0.000

Gender_No = 1.000

Gender_Yes = 0.000

Debtor_No = 1.000

Debtor_Yes = 0.000

Scholarship holder_No = 0.000

Scholarship holder_Yes = 1.000

قوانین کلیدی

قانون 1 (کلاس: Graduate, وزن اطمینان: 1.00):

مقدمه:

Admission grade_Low = 0.197

Admission grade_Medium = 0.803

Admission grade_High = 0.000

Unemployment rate_Low = 0.673

Unemployment rate_Medium = 0.327

Unemployment rate_High = 0.000

Application mode_Low = 0.000

Application mode_Medium = 0.000

Application mode_High = 0.000

Course_Low = 0.000

Course_Medium = 0.282

Course_High = 0.718

Previous qualification_Low = 0.000

Previous qualification_Medium = 0.000

Previous qualification_High = 0.000

Curricular units 1st sem (enrolled)_Low = 0.042

Curricular units 1st sem (enrolled)_Medium = 0.958

Curricular units 1st sem (enrolled)_High = 0.000

Curricular units 2nd sem (grade)_Low = 0.000

Curricular units 2nd sem (grade)_Medium = 0.546

Curricular units 2nd sem (grade)_High = 0.454

Curricular units 2nd sem (enrolled)_Low = 0.036

Curricular units 2nd sem (enrolled)_Medium = 0.964

Curricular units 2nd sem (enrolled)_High = 0.000

Gender_No = 0.000

Gender_Yes = 1.000

Debtor_No = 1.000

Debtor_Yes = 0.000

Scholarship holder_No = 1.000

Scholarship holder_Yes = 0.000

تفسیر:

این قانون نشان‌دهنده ویژگی‌هایی است که احتمال فارغ‌التحصیلی را افزایش می‌دهند.

قانون 2 (کلاس: Dropout, وزن اطمینان: 1.00):

مقدمه:

Admission grade_Low = 0.000

Admission grade_Medium = 0.792

Admission grade_High = 0.208

Unemployment rate_Low = 0.000

Unemployment rate_Medium = 0.758

Unemployment rate_High = 0.242

Application mode_Low = 0.660

Application mode_Medium = 0.340

Application mode_High = 0.000

Course_Low = 0.000

Course_Medium = 0.755

Course_High = 0.245

Previous qualification_Low = 0.461

Previous qualification_Medium = 0.539

Previous qualification_High = 0.000

Curricular units 1st sem (enrolled)_Low = 0.202

Curricular units 1st sem (enrolled)_Medium = 0.798

Curricular units 1st sem (enrolled)_High = 0.000

Curricular units 2nd sem (grade)_Low = 0.000

Curricular units 2nd sem (grade)_Medium = 0.000

Curricular units 2nd sem (grade)_High = 0.000

Curricular units 2nd sem (enrolled)_Low = 0.197

Curricular units 2nd sem (enrolled)_Medium = 0.803

Curricular units 2nd sem (enrolled)_High = 0.000

Gender_No = 0.000

Gender_Yes = 1.000

Debtor_No = 1.000

Debtor_Yes = 0.000

Scholarship holder_No = 1.000

Scholarship holder_Yes = 0.000

تفسیر:

این قانون نشان‌دهنده ویژگی‌هایی است که احتمال ترک تحصیل را افزایش می‌دهند.