دانشکده مهندسی کامپیوتر

گزارش پایانی درس: مبانی هوش محاسباتی

عنوان پژوهش: تمرین دوم بخش چهارم (پروژه فازی)

ارائه دهنده:

نرگس سادات موسوی جد

شادی شاهی محمدی

استاد درس:

دکتر کارشناس

بهار ۱۴۰۴

فهرست:

۱–پیاده‌سازی یک سیستم استنتاج فازی:

هدف این سیستم به‌طور کلی، استفاده از مدل فازی برای پیش‌بینی و ارزیابی عملکرد با استفاده از داده‌های آموزشی و آزمایشی است. در اینجا، به تفکیک به عملکرد هر بخش از کد می‌پردازیم:

۱–۱–کلاس‌های فازی :

FuzzySet : این کلاس پایه برای مجموعه‌های فازی است که متغیرهای is_continuous برای گسسته یا پیوسته بودن مجموعه و alpha برای برش آلفا را ذخیره می‌کند. یک متد membership_func برای محاسبه تابع عضویت دارد که توسط کلاس‌های فرزند پیاده‌سازی می‌شود.

```
class FuzzySet:
    def __init__(self, is_continuous, alpha):
        self.is_continuous = is_continuous
        self.alpha = alpha

    def cut(self, x):
        membership_value = self.membership_func(x)
        if membership_value > self.alpha:
            return self.alpha
        return membership_value

    @abstractmethod
    def membership_func(self, x):
        """Abstract method for area calculation"""
        pass
```

TriangleSet: این کلاس یک مجموعه فازی مثلثی است که یک تابع عضویت مثلثی را پیاده‌سازی می‌کند.

```
class TriangleSet(FuzzySet):
    def __init__(self, is_continuous, a, b, c):
        super().__init__(is_continuous, 1)
        self.a = a
        self.b = b
        self.c = c

    def membership_func(self, x):
        if not self.is_continuous:
            x = int(x)
        if self.a == self.b:
            return np.maximum(np.minimum(1, (self.c - x) / (self.c -
self.b)), 0)
        elif self.c == self.b:
            return np.maximum(np.minimum(1, (x - self.a) / (self.b -
self.a)), 0)
        return np.maximum(np.minimum((x - self.a) / (self.b - self.a),
(self.c - x) / (self.c - self.b)), 0)
```

CategoricalSet : این کلاس برای مجموعه‌های فازی دسته‌ای (categorical) است که مقادیر را به صورت دقیق (بله یا خیر) دسته‌بندی می‌کند. این دو کلاس نمونه‌های از کلاس مجموعه‌ی فازی هستند.

```
class CategoricalSet(FuzzySet):
    def __init__(self, category_num):
        super().__init__(False, 1)
        self.category_num = category_num

    def membership_func(self, x):
        if self.category_num == x:
            return 1
        else:
            return 0
```

LinguisticVariable : این کلاس برای متغیرهای زبانی استفاده می‌شود که دامنه‌ای از مقادیر عددی دارند و می‌توانند چندین مجموعه فازی داشته باشند:

```
class LinguisticVariable:
    def __init__(self, name, lower_limit, upper_limit):
        self.name = name
        self.lower_limit = lower_limit
        self.upper_limit = upper_limit
        self.fuzzy_sets = {}
```

Rule : این کلاس برای قوانین فازی تعریف شده که شامل پیش‌شرط‌ها(antecedents) ، نتیجه (result) و وزن (weight)است.

```
class Rule:
    def __init__(self, antecedents, result, weight):
        self.antecedents = antecedents
        self.result = result
        self.weight = weight

    def __eq__(self, other):
        if not isinstance(other, Rule):
            return False
        return (self.antecedents == other.antecedents) and (self.result ==
other.result)

    def __hash__(self):
        return hash((self.antecedents, self.result))
```

۱–۲–پیش پردازش داده‌ها:

preprocessing() : این تابع برای پیش‌پردازش داده‌ها، شامل بارگذاری، تقسیم داده‌ها به آموزش و آزمایش و حذف داده‌های گمشده است. همچنین اطلاعاتی از داده‌ها را نمایش می‌دهد.

```
def preprocessing():
    df = pd.read_csv("dataset.csv")

    print("head of dataset:")
    print(df.head(), "\n")
    print("information of dataset and its columns:")
    print(df.info(), "\n")
    print("some statistics:")
    print(df.describe(), "\n")

    print("check for missing data")
    print(df.isna().sum())
    df = df.dropna()

    df['Target'] = df['Target'].map({'Dropout': 0, 'Graduate': 1, 'Enrolled':
2})

    plot_dataset_columns(df)
    plot_correlation_matrix(df)

    return df
```

feature_selection() : این تابع برای انتخاب ویژگی‌ها از داده‌ها با استفاده از معیار اطلاعات متقابل (mutual information) به کار می‌رود. در نهایت ۱۰ ویژگی منتخب برای استفاده در سیستم استنتاج فازی بکار می‌روند.

```
def feature_selection(dataset):
    scores = mutual_info_classif(
        dataset.drop(columns='Target'),
        dataset['Target'],
        random_state=0
    )
    mi_series = pd.Series(scores,
index=dataset.drop(columns='Target').columns) \
        .sort_values(ascending=False)
    top_features = mi_series.index[:10].tolist()
    print("feature selection scores:")
    print(mi_series)

    selected_columns = top_features + ['Target']
    df_selected = dataset[selected_columns]
    return df_selected
```

۱–۳–کلاس سیستم استنتاج فازی:

FIS : این کلاس نماینده سیستم استنتاج فازی است که شامل قوانین فازی (rules) و متغیرهای زبانی (linguistic_variables)می‌باشد.

```
class FIS:
    def __init__(self, train_set, linguistic_variables_list):
        self.train_set = train_set
        self.rules = []
        self.linguistic_variables = linguistic_variables_list

    def find_matching_rule(self, target_rule):
        for rule in self.rules:
            if rule == target_rule:
                return rule
        return None
```

wang_mendel() : این متد برای یادگیری قوانین فازی به روش Wang-Mendel استفاده می‌شود. برای هر ردیف از داده‌های آموزشی، بهترین مجموعه فازی برای پیش‌شرط‌ها انتخاب می‌شود و قانونی تولید می‌گردد و در صورتی که قانون تکراری نبود، به قوانین موجود افزوده می‌شود.

```
def wang_mendel(self, max_rules=50):
    count = 0
    for _, row in self.train_set.iterrows():
        if count > max_rules:
            break
        antecedents = []
        cf = 1
        row_enum = enumerate(row)
        for i, x in row_enum:
            membership_degrees = [fs.membership_func(x) for fs in
self.linguistic_variables[i].fuzzy_sets.values()]
            set_names = list(self.linguistic_variables[i].fuzzy_sets.keys())
            max_membership_degree = max(membership_degrees)
            best_set = set_names[np.argmax(membership_degrees)]
            if i < len(row) - 1:
                cf *= max_membership_degree
                antecedents.append(best_set)
            else:
                rule = Rule(copy.deepcopy(antecedents), best_set, cf)
                same_rule = self.find_matching_rule(rule)
                if same_rule is None:
                    self.rules.append(rule)
                else:
                    if same_rule.weight < rule.weight:
                        same_rule.weight = rule.weight
        count += 1
```

mamdani() : این متد برای محاسبه نتیجه سیستم فازی به روش Mamdani است. در اینجا، برای هر قانون، عضویت پیش‌شرط‌ها محاسبه شده و سپس نتیجه خروجی محاسبه می‌شود.

```
def mamdani(self, inputs):
    weight_prod_z = 0
    sum_weight = 0
    for rule in self.rules:
        antecedents_memberships = []
        i = 0
        for antecedent in rule.antecedents:
            membership =
self.linguistic_variables[i].fuzzy_sets[antecedent].membership_func(inputs.il
oc[i])
            antecedents_memberships.append(membership)
            i += 1
        alpha = min(antecedents_memberships)
        output_set = self.linguistic_variables[i].fuzzy_sets[rule.result]
        if isinstance(output_set, CategoricalSet):
            weight_prod_z += output_set.category_num * alpha
            sum_weight += alpha
    if sum_weight == 0:
        return 0
    return int(weight_prod_z / sum_weight)
```

۱–۴–کلاس کروموزوم (Chromosome) و الگوریتم ژنتیک:(Genetic Algorithm)

Chromosome : این کلاس برای نمایندگی یک کروموزوم از الگوریتم ژنتیک استفاده می‌شود. کروموزوم شامل لیستی از قوانین فازی است. در واقع در اینجا کروموزوم‌ها با روش پتزبورگ پیاده‌سازی شده‌اند و طول متفاوتی دارند.

```
class Chromosome:
    def __init__(self, num_rules, chromosome_list, num_element_in_rule=11):
        self.num_rules = num_rules
        self.chromosome_list = chromosome_list
        self.num_element_in_rule = num_element_in_rule

    def modify_gen(self, rule_idx, gen_index):
        rule = self.chromosome_list[rule_idx]
        target_index = next(i for i, var in
enumerate(fis.linguistic_variables) if var.name == "Target")

        if gen_index == self.num_element_in_rule - 1:
            fuzzy_set_names =
list(fis.linguistic_variables[target_index].fuzzy_sets.keys())
            rule.result = random.choice(fuzzy_set_names)
        else:
            fuzzy_set_names =
list(fis.linguistic_variables[gen_index].fuzzy_sets.keys())
            rule.antecedents[gen_index] = random.choice(fuzzy_set_names)
```

Genetic : این کلاس پیاده‌سازی الگوریتم ژنتیک است که از جمعیت کروموزوم‌ها (که هر کروموزوم مجموعه‌ای از قوانین است) استفاده می‌کند. این الگوریتم از انتخاب والدین(parent selection) ، ترکیب (crossover)و جهش (mutation) برای تولید نسل‌های جدید استفاده می‌کند.

```
class Genetic:
    def __init__(self, fis_model, train_set, population_size=20,
mutation_rate=0.4,
                 crossover_rate=0.9, generations=35, patience=10):
        self.fis_model = fis_model
        self.patience = patience
        self.train_set, self.valid_set = train_test_split(train_set,
test_size=0.2, random_state=42,

stratify=train_set["Target"])
        self.population_size = population_size
        self.mutation_rate = mutation_rate
        self.crossover_rate = crossover_rate
        self.generations = generations
        self.population = self.init_population()
        self.feature_selection_history = []
        self.accuracy_history = []
```

init_population() : این متد برای ایجاد جمعیت اولیه کروموزوم‌ها از مجموعه قوانین موجود استفاده می‌شود. برای حفظ تنوع ژن‌ها بیشتر قوانین به‌صورت تصادفی ساخته شده و بکار می‌روند ولی برای استفاده از تجربیات قبلی، از قوانین ساخته شده با روش wang mendel نیز استفاده می‌شود.

```
def init_population(self):
    rules = copy.deepcopy(self.fis_model.rules)
    for i in range(200):
        rule = self.make_random_rule()
        if rule not in rules:
            rules.append(rule)
    chromosome_rules_size = int(len(rules) / self.population_size)
    rule_idx = 0
    population = []
    for i in range(self.population_size):
        chromosome_list = []
        for _ in range(chromosome_rules_size):
            chromosome_list.append(
                Rule(rules[rule_idx].antecedents, rules[rule_idx].result,
                     rules[rule_idx].weight))
            rule_idx += 1
        population.append(Chromosome(chromosome_rules_size,
copy.deepcopy(chromosome_list)))
    while rule_idx < len(self.fis_model.rules):
        population[self.population_size - 1].chromosome_list.append(
            Rule(rules[rule_idx].antecedents, rules[rule_idx].result,
                 rules[rule_idx].weight))
        rule_idx += 1
    return population
```

fitness() : تابع برازندگی برای ارزیابی کروموزوم‌ها با استفاده از دقت و F1 score و همچنین تعداد قوانین بکار می‌رود. این تابع به هرکدام از این معیارها وزنی داده و در نهایت مقدار نهایی برازندگی هر کروموزوم را که درواقع یک پایگاه قوانین است محاسبه می‌کند.

```
def fitness(self, chromosome):
    temp_fis = FIS(self.train_set, self.fis_model.linguistic_variables)
    temp_fis.rules = copy.deepcopy(chromosome.chromosome_list)
    y_true = []
    y_pred = []
    for _, row in self.valid_set.iterrows():
        predicted = temp_fis.mamdani(row)
        y_pred.append(predicted)
        y_true.append(row['Target'])

    acc = accuracy_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred, average='weighted')
    penalty = chromosome.num_rules / len(self.fis_model.rules)
    fitness_value = 0.5 * acc + 0.3 * f1 + 0.2 * (1 - penalty)

    return fitness_value
```

crossover() : متد کراس‌اور برای ترکیب دو کروموزوم و ایجاد دو فرزند استفاده می‌شود. در اینجا از روش کراس اور تک نقطه‌ای استفاده شده است.

```
def crossover(self, parent1, parent2):
    rules1 = parent1.chromosome_list
    rules2 = parent2.chromosome_list

    min_len = min(len(rules1), len(rules2))
    # too short for crossover
    if min_len < 2:
        return copy.deepcopy(parent1), copy.deepcopy(parent2)

    crossover_point = random.randint(1, min_len - 1)

    child1_rules = copy.deepcopy(rules1[:crossover_point] +
rules2[crossover_point:])
    child2_rules = copy.deepcopy(rules2[:crossover_point] +
rules1[crossover_point:])

    child1 = Chromosome(len(child1_rules), child1_rules)
    child2 = Chromosome(len(child2_rules), child2_rules)

    return child1, child2
```

mutation() : تابع mutationیک بخش از الگوریتم ژنتیک است که برای ایجاد تغییرات تصادفی در یک والد به‌منظور افزایش تنوع ژنتیکی استفاده می‌شود. ابتدا با احتمال mutation_rate مشخص می‌شود که آیا جهش انجام شود یا نه. سپس بسته به مقدار عدد تصادفی rand_op، یکی از سه نوع جهش اعمال می‌شود: اگر rand_op

0.5 > باشد، جهش در ژن‌های قوانین موجود رخ می‌دهد به‌طوری که با احتمال ۵۰٪ هر ژن از هر قانون ممکن است تغییر یابد (با استفاده از modify_gen). اگر مقدار rand_op بین ۰.۵ و ۰.۷۵ باشد و تعداد قوانین بیشتر از ۵ باشد، یکی از قوانین به‌صورت تصادفی حذف می‌شود. در غیر این صورت، یک قانون جدید تولید شده و اگر تکراری نباشد، به مجموعه قوانین افزوده می‌شود. این طراحی باعث ایجاد تعادل بین اکتشاف (افزایش تنوع) و بهره‌برداری (بهبود جواب موجود) در فضای جست‌وجو می‌شود

```
def mutation(self, parent):
    if random.random() < self.mutation_rate:
        rand_op = random.random()
        if rand_op < 0.5:
            for rule_idx in range(parent.num_rules):
                for gen_idx in range(parent.num_element_in_rule):
                    if random.random() < 0.5:
                        parent.modify_gen(rule_idx, gen_idx)
        elif 0.5 <= rand_op < 0.75 and parent.num_rules > 5:
            remove_idx = random.randint(0, parent.num_rules - 1)
            del parent.chromosome_list[remove_idx]
            parent.num_rules -= 1
        else:
            rule = self.make_random_rule()
            if rule not in parent.chromosome_list:
                parent.chromosome_list.append(rule)
                parent.num_rules += 1
```

parent_selection(): متد انتخاب والدین به روش تورنومنت ۵ تایی می‌باشد.

```
def parent_selection(self, fitness_scores, k=5):
    selected = random.sample(list(enumerate(fitness_scores)), k)
    winner = max(selected, key=lambda x: x[1])
    return self.population[winner[0]]
```

run(): این متد برای اجرای الگوریتم ژنتیک در چندین نسل (تکرار) استفاده می‌شود. در هر نسل، والدین انتخاب شده و با استفاده از ترکیب و جهش فرزندان جدید تولید می‌شوند. سپس نسل جدید جایگزین نسل قبلی می‌شود.

```
def run(self):
    print("Genetic Algorithm Running...")
    fitness_scores = [self.fitness(ch) for ch in self.population]
    best_fitness = max(fitness_scores)
    for generation in range(self.generations):
        new_population = []
        for _ in range(self.population_size // 2):
            parent1 = self.parent_selection(fitness_scores)
            parent2 = self.parent_selection(fitness_scores)
            if random.random() < self.crossover_rate:
                child1, child2 = self.crossover(parent1, parent2)
            else:
                child1, child2 = copy.deepcopy(parent1),
copy.deepcopy(parent2)
            self.mutation(child1)
            self.mutation(child2)
            new_population.append(child1)
```

```
            new_population.append(child2)
        print(max(fitness_scores))

        self.accuracy_history.append(best_fitness)
        self.population = new_population

        fitness_scores = [self.fitness(ch) for ch in self.population]
        best_fitness = max(fitness_scores)

    print("Genetic Algorithm Completed.")
    return self.population[np.argmax(fitness_scores)]
```

۱–۵–تست مدل:

در نهایت به کمک تابع test_model مدل استنتاجی فازی با قوانین جدید تست می‌شود. این تابع برای ارزیابی مدل روی داده‌های آزمایشی است که دقت، دقت مثبت، بازیابی، F1score و ماتریس آشفتگی را محاسبه می‌کند.

```
def test_model(test_set, fis_model):
    y_true = []
    y_pred = []

    for _, row in test_set.iterrows():
        y_true.append(row["Target"])
        y_pred.append(fis_model.mamdani(row))

    acc = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted',
zero_division=0)
    recall = recall_score(y_true, y_pred, average='weighted',
zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)
    cm = confusion_matrix(y_true, y_pred)

    print("Evaluation on Test Set")
    print(f"num_rules: {len(fis_model.rules)}")
    print(f"Accuracy:  {acc:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall:    {recall:.4f}")
    print(f"F1 Score:  {f1:.4f}")
    print("\nConfusion Matrix:")
    print(cm)
```

۲–نتایج:

۲–۱–تحلیل اکتشافی داده‌ها:

نمایش هدرهای دیتاست:

```
head of dataset:
   Marital status  Application mode  Application order  Course  \
0               1                 8                  5       2
1               1                 6                  1      11
2               1                 1                  5       5
3               1                 8                  2      15
4               2                12                  1       3

   Daytime/evening attendance  Previous qualification  Nacionality  \
0                           1                       1            1
1                           1                       1            1
2                           1                       1            1
3                           1                       1            1
4                           0                       1            1

   Mother's qualification  Father's qualification  Mother's occupation
...  \
0                      13                      10                    6
...
1                       1                       3                    4
...
2                      22                      27                   10
...
3                      23                      27                    6
...
4                      22                      28                   10
...

4                                    6

   Curricular units 2nd sem (approved)  Curricular units 2nd sem (grade)
\
0                                    0                         0.000000
1                                    6                        13.666667
2                                    0                         0.000000
3                                    5                        12.400000
4                                    6                        13.000000

   Curricular units 2nd sem (without evaluations)  Unemployment rate  \
0                                               0               10.8
1                                               0               13.9
2                                               0               10.8
3                                               0                9.4
 4                                              0               13.9
Inflation rate   GDP    Target
```

```
0              1.4  1.74   Dropout
1             -0.3  0.79  Graduate
2              1.4  1.74   Dropout
3             -0.8 -3.12  Graduate
4             -0.3  0.79  Graduate

[5 rows x 35 columns]
```

نمایش اطلاعات هر ستون دیتاست:

```
information of dataset and its columns:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4424 entries, 0 to 4423
Data columns (total 35 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Marital status                                  4424 non-null   int64
 1   Application mode                                4424 non-null   int64
 2   Application order                               4424 non-null   int64
 3   Course                                          4424 non-null   int64
 4   Daytime/evening attendance                      4424 non-null   int64
 5   Previous qualification                          4424 non-null   int64
 6   Nacionality                                     4424 non-null   int64
 7   Mother's qualification                          4424 non-null   int64
 8   Father's qualification                          4424 non-null   int64
 9   Mother's occupation                             4424 non-null   int64
 10  Father's occupation                             4424 non-null   int64
 11  Displaced                                       4424 non-null   int64
 12  Educational special needs                       4424 non-null   int64
 13  Debtor                                          4424 non-null   int64
 14  Tuition fees up to date                         4424 non-null   int64
 15  Gender                                          4424 non-null   int64
 16  Scholarship holder                              4424 non-null   int64
 17  Age at enrollment                               4424 non-null   int64
 18  International                                   4424 non-null   int64
 19  Curricular units 1st sem (credited)             4424 non-null   int64
 20  Curricular units 1st sem (enrolled)             4424 non-null   int64
 21  Curricular units 1st sem (evaluations)          4424 non-null   int64
 22  Curricular units 1st sem (approved)             4424 non-null   int64
 23  Curricular units 1st sem (grade)                4424 non-null float64
 24  Curricular units 1st sem (without evaluations)  4424 non-null   int64
 25  Curricular units 2nd sem (credited)             4424 non-null   int64
 26  Curricular units 2nd sem (enrolled)             4424 non-null   int64
 27  Curricular units 2nd sem (evaluations)          4424 non-null   int64
 28  Curricular units 2nd sem (approved)             4424 non-null   int64
 29  Curricular units 2nd sem (grade)                4424 non-null float64
 30  Curricular units 2nd sem (without evaluations)  4424 non-null   int64
 31  Unemployment rate                               4424 non-null float64
 32  Inflation rate                                  4424 non-null float64
 33  GDP                                             4424 non-null float64
 34  Target                                          4424 non-null  object
```

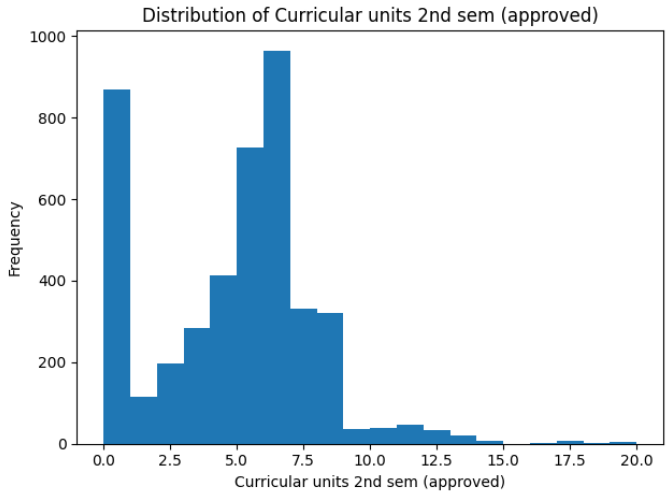
```
dtypes: float64(5), int64(29), object(1)
memory usage: 1.2+ MB
```

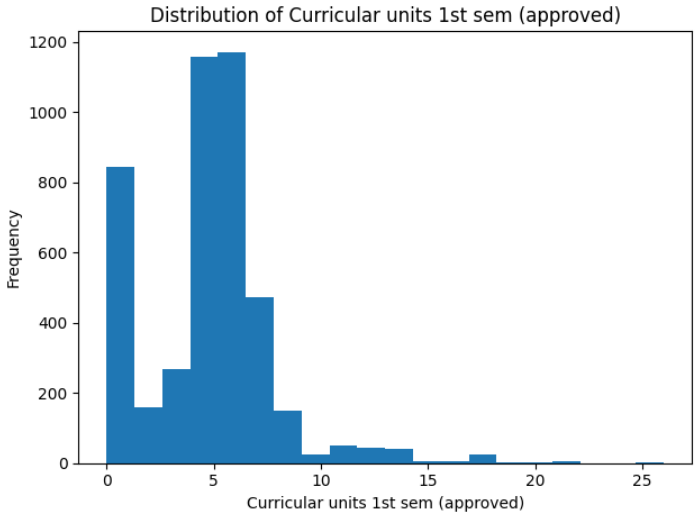
None

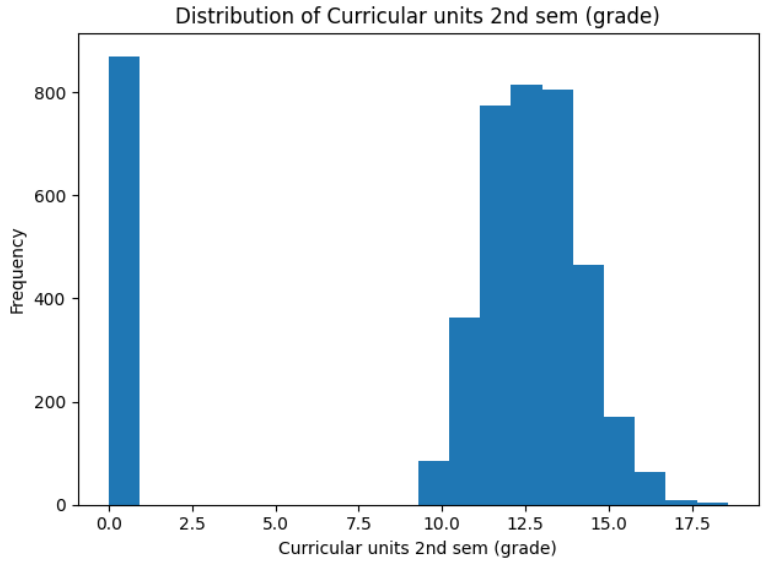
نمایش اطلاعاتی آماری دیتاست: (برای بقیه داده‌ها نیز به همین ترتیب)

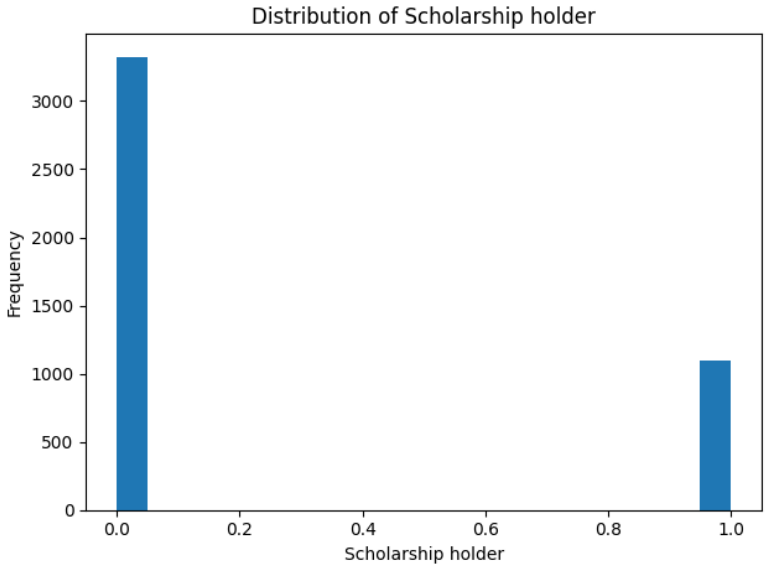
```
some statistics:
       Marital status  Application mode  Application order       Course  \
count     4424.000000       4424.000000        4424.000000  4424.000000
mean         1.178571          6.886980           1.727848     9.899186
std          0.605747          5.298964           1.313793     4.331792
min          1.000000          1.000000           0.000000     1.000000
25%          1.000000          1.000000           1.000000     6.000000
50%          1.000000          8.000000           1.000000    10.000000
75%          1.000000         12.000000           2.000000    13.000000
max          6.000000         18.000000           9.000000    17.000000
```

نمودار برخی از ستون‌ها که در فرایند انتخاب ویژگی انتخاب شده‌اند:

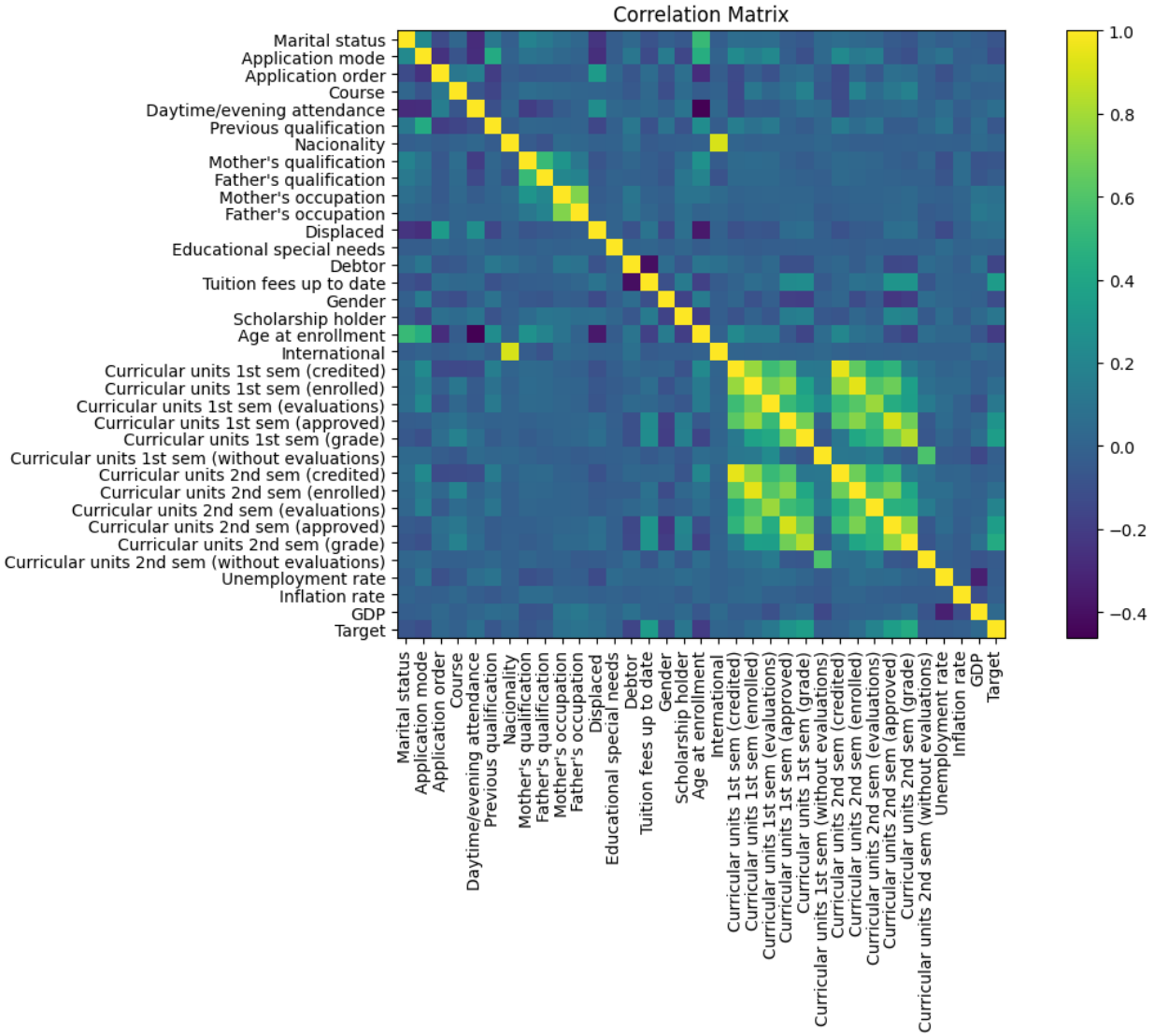





Distribution of Curricular units 2nd sem (grade)
Frequency
0
200
400
600
800
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Curricular units 2nd sem (grade)


Distribution of Scholarship holder
Frequency
0
500
1000
1500
2000
2500
3000
0.0
0.2
0.4
0.6
0.8
1.0
Scholarship holder

نمایش ماتریس همبستگی:



۲–۲–نتایج مربوط به انتخاب ویژگی:

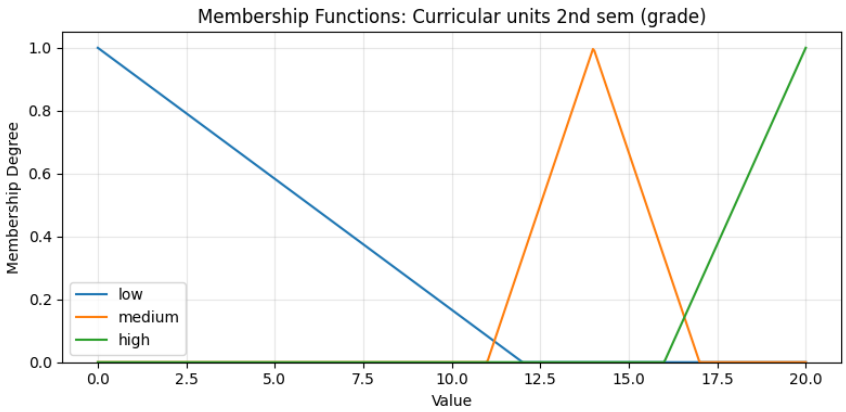
ده ویژگی اول در مدل استفاده شده است.

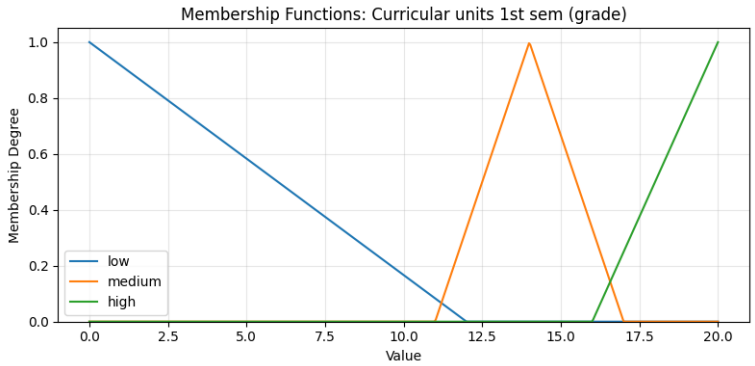
```
feature selection scores:
Curricular units 2nd sem (approved)              0.309306
Curricular units 1st sem (approved)              0.244813
Curricular units 2nd sem (grade)                 0.238680
Curricular units 1st sem (grade)                 0.194403
Tuition fees up to date                          0.101759
Curricular units 2nd sem (evaluations)           0.087349
Curricular units 1st sem (evaluations)           0.083303
Course                                           0.076519
Curricular units 1st sem (enrolled)              0.060009
```

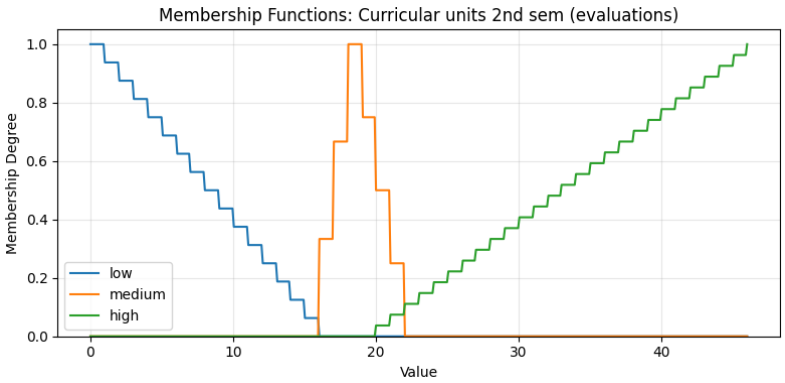
```
Scholarship holder                                  0.054155
Application mode                                    0.048580
Curricular units 2nd sem (enrolled)                 0.046073
Age at enrollment                                   0.044143
Debtor                                              0.035538
Mother's qualification                              0.033370
Mother's occupation                                 0.028415
Gender                                              0.025723
Inflation rate                                      0.018512
Educational special needs                           0.015218
Father's qualification                              0.015068
Previous qualification                              0.014537
Father's occupation                                 0.010913
Unemployment rate                                   0.009486
Application order                                   0.008450
Curricular units 1st sem (without evaluations)      0.008039
Curricular units 2nd sem (credited)                 0.008035
Daytime/evening attendance                          0.007567
Marital status                                      0.007201
Curricular units 2nd sem (without evaluations)      0.006746
GDP                                                 0.002705
Curricular units 1st sem (credited)                 0.002604
International                                       0.001808
Displaced                                           0.001186
Nacionality                                         0.000000
dtype: float64
Train size: 3539
Test size: 885
```

۲-۳-توابع عضویت:

دو تابع عضویت پیوسته و یک تابع گسسته در زیر به نمایش در آمده است:

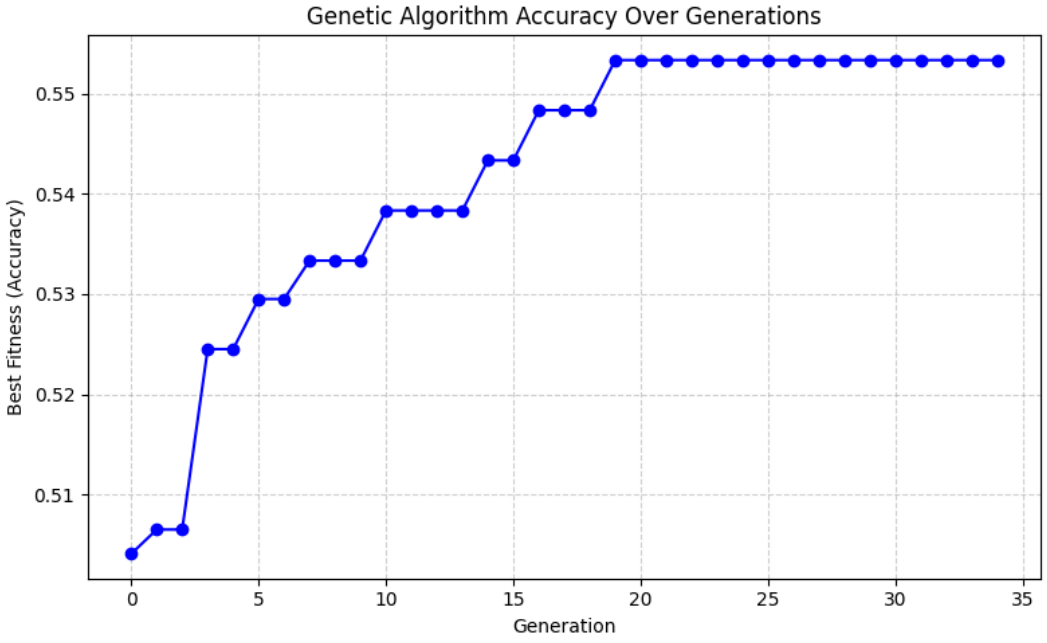






۲-۴-نتایج الگوریتم ژنتیک:

نمودار دقت الگوریتم ژنتیک در طول نسل‌های مختلف در زیر به نمایش در آمده است همانطور که مشاهده می‌شود الگوریتم با شیب خوبی به سمت مقادیر بهینه حرکت می‌کند:



۲–۵–نتایج تست سامانه استنتاج فازی:

آمارهای سامانه استنتاج اولیه به صورت زیر است:

```
Evaluation on Test Set
num_rules: 40
Accuracy:  0.5514
Precision: 0.5803
Recall:    0.5514
F1 Score:  0.5222

Confusion Matrix:
[[235  32  17]
 [195 243   4]
 [106  43  10]]
```

در حالی که سامانه‌ی جدید با تغییر نه چندان زیاد دقت توانسته است، تعداد قوانین را از ۴۰ به ۶ قانون کاهش دهد:Evaluation on Test Set

```
num_rules: 6
Accuracy:  0.4712
Precision: 0.5750
Recall:    0.4712
F1 Score:  0.4257

Confusion Matrix:
[[265  12   7]
 [272 145  25]
[135  17   7]]
```

کد برنامه: https://colab.research.google.com/drive/1R-v6x_Usxu6_18e3SMW71e0OR-oEqbZf?usp=sharing