МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Институт Информационных Технологий
Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

Отчёт

по практическому занятию №1
по дисциплине «Моделирование программных систем»

Выполнили студенты группы *ИКБО-66-23* ***Хомченко В.А.***
Ян Х.
Смирнов А.Ю.

Работа выполнена «__» _____202__ г

«Зачтено» «__» _____202__ г.

Москва 2025

Содержание

Введение

Современные условия цифровизации и роста популярности дистанционного банковского обслуживания делают актуальным вопрос выбора оптимального финансового партнёра. Качество банковских услуг напрямую влияет на удобство клиентов, безопасность транзакций и надёжность операций. В связи с этим возникает необходимость в системном подходе к оценке и сравнению различных банков на основе объективных критериев.

Целью данной работы является оценка качества банковских услуг (Сбербанк, Т-Банк, Альфа-Банк) с использованием методов многокритериального принятия решений: **ELECTRE, TOPSIS и SAW**. Эти методы позволяют учитывать разнородные критерии, такие как Защищенность, эффективность, стоимость, удовлетворенность**,** и покрытие контекста, а также проводить выбор оптимального варианта.

Работа направлена на развитие навыков анализа финансовых учреждений и демонстрацию практического применения методов многокритериального принятия решений в банковском секторе. Полученные результаты могут быть использованы для оптимизации выбора банка в зависимости от потребностей клиентов и бизнес-задач.

Постановка задачи

Провести оценку качества банковских услуг и осуществить обоснованный выбор оптимального банка, предоставляющего наиболее выгодные условия для клиентов и бизнеса, в соответствии с указанной методикой.

В качестве программного обеспечения были выбраны (Сбербанк, Т-банк, Альфа-Банк), а в качестве методов принятия управленческих решений были выбраны методы ELECTRE, TOPSIS, SAW.

Ход работы

1. Выбор критериев оценки и построение таблицы оценки ПО

В качестве критериев оценки программного обеспечения выберем следующие:

- Защищённость;
- Эффективность;
- Удовлетворенность;
- Покрытие контекста;
- Стоимость

Далее построим таблицу с оцениваемыми банки в сроках и с критериями в столбцах в нормализованном виде, а также установим веса для каждого критерия:

Таблица 1. Оценка ПО

Веса	0,4	0,2	0,2	0,1	0,1
Характе-ристики / Плат-формы	Защищён ность	Эффективн ость	Удовлетвор енность	Покрытие контекста	**Стоимос ть**
СберБанк	0,95	0,9	0,9	0,7	0,7
Т-Банк	0,9	0,9	0,5	0,75	0,75
Альфа-Банк	0,98	0,9	0,75	0,75	0,8

2. Выбор и реализация методов

2.1. Метод ELECTRE

Методы ELECTRE, в общем и целом, предоставляют возможность оценить степень превосходства одного альтернативного решения над другими с помощью анализа их согласования. Процесс принятия решения начинается с раунда оценивания, когда эксперт оценивает все альтернативные решения по всем критериям. В результате формируется матрица решений $A = (xij)$, где xij обозначает оценку, данную по i-той альтернативе по j-тому критерию. Метод состоит из 9 последовательных шагов:

- расчёт нормализованной матрицы решения;
- расчёт взвешенной нормализованной матрицы решения;
- построение множеств согласия и несогласия;
- расчёт матрицы согласия;
- расчёт матрицы несогласия;
- построение матрицы индексов согласия превосходства;
- построение матрицы индексов несогласия превосходства;
- построение агрегированной матрицы превосходства;
- удаление наименее предпочтительных альтернатив.

На основе этого напишем код для подсчёта и получения ранжированного списка анализируемого ПО (Листинг 1).

Листинг 1. Метод ELECTRE

```
import numpy as np



alternatives = np.array([
    [0.95, 0.9, 0.9, 0.7, 0.7],  # Сбербанк
    [0.9, 0.9, 0.5, 0.75, 0.75],  # Т-Банк
    [0.98, 0.9, 0.75, 0.75, 0.8]  # Альфа-банк
])

weights = np.array([0.4, 0.2, 0.2, 0.1, 0.1])
names = ['Sber', 'T-bank', 'AlfaBank']
```

```
    norms = np.linalg.norm(alternatives, axis=0)
    normalized = alternatives / norms

    weighted = normalized * weights

    n = len(alternatives)
    c = np.zeros((n, n))
    d = np.zeros((n, n))

    all_diffs = weighted[:, np.newaxis] - weighted
    positive_diffs = all_diffs[all_diffs > 0]
    max_diff = np.max(positive_diffs) if len(positive_diffs) > 0 else 0

    for i in range(n):
        for j in range(n):
            if i == j:
                continue
            agreement = weighted[i] >= weighted[j]
            c[i, j] = np.sum(weights[agreement])

            mask = weighted[j] > weighted[i]
            if np.any(mask):
                disagreement = np.max(weighted[j][mask] -
weighted[i][mask])
                d[i, j] = disagreement / max_diff if max_diff != 0 else 0

    c_star = 0.7
    d_star = 0.5

    dominance = np.zeros((n, n), dtype=bool)
    for i in range(n):
        for j in range(n):
            if i == j:
                continue
            dominance[i, j] = (c[i, j] >= c_star) and (d[i, j] <= d_star)

    kernel = []
    for j in range(n):
        dominated = False
        for i in range(n):
            if dominance[i, j]:
                dominated = True
```

```
                break
        if not dominated:
            kernel.append(j)

    print("Матрица согласия:")
    print(c.round(2))
    print("\nМатрица несогласия:")
    print(d.round(2))
    print("\nЯдро (оптимальные альтернативы):")
    print([names[i] for i in kernel])

    scores = np.sum(dominance, axis=1)
    ranking = np.argsort(-scores)

    print("\nРанжированный список:")
    for i, idx in enumerate(ranking):
print(f"{i+1}. {names[idx]} ( имеет {scores[idx]} альтернатив)")
```

Передадим данные в нашу программу и получим результат, как показано ниже:

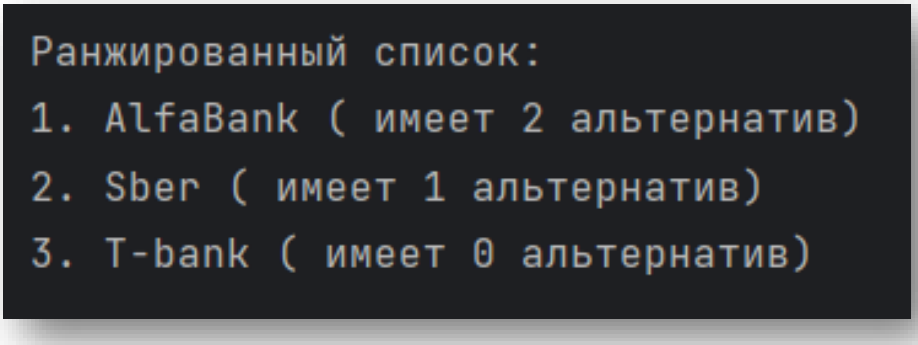

Рисунок 1 – Результат метода ELECTRE

В итоге получилось, что AlfaBank на 1-ом месте.

2.2. Метод TOPSIS

TOPSIS – технология, разработанная Хвонгом и Юном в 1981 г. Данный метод используется для решения многокритериальных задач. Суть метода состоит в поиске альтернатив, значения оценок которых наиболее близки к идеально-позитивному решению и наиболее отдалены от идеально-негативного решения. Идеально-позитивное решение представляет собой вектор максимальных значений матрицы взвешенных оценок альтернатив. Идеальнонегативное решение, напротив, является вектором минимальных значений.

Метод состоит из 6 последовательных шагов:

- расчёт нормализованной матрицы решения;
- расчёт взвешенной нормализованной матрицы решения;
- определение «идеального» и «идеально-негативного» ожидаемого состояния;
- расчёт метрики разделения;
- расчёт относительной близости к «идеальному» состоянию;
- ранжирование критериев.

На основе этого напишем код для подсчёта и получения ранжированного списка анализируемого ПО (Листинг 2).

Листинг 2. Метод TOPSIS

```
import numpy as np



alternatives = ['Sber', 'T-bank', 'AlfaBank']
criteria_weights = np.array([0.4, 0.2, 0.2, 0.1, 0.1])
decision_matrix = np.array([
    [0.95, 0.9, 0.9, 0.7, 0.7],  # Сбербанк
    [0.9, 0.9, 0.5, 0.75, 0.75],  # Т-Банк
    [0.98, 0.9, 0.75, 0.75, 0.8]  # Альфа-банк
])



squared_sum = np.sum(decision_matrix**2, axis=0)
normalized_matrix = decision_matrix / np.sqrt(squared_sum)
```

```
      weighted_matrix = normalized_matrix * criteria_weights

      ideal_best = np.max(weighted_matrix, axis=0)
      ideal_worst = np.min(weighted_matrix, axis=0)

      distance_best = np.sqrt(np.sum((weighted_matrix - ideal_best)**2,
axis=1))
      distance_worst = np.sqrt(np.sum((weighted_matrix - ideal_worst)**2,
axis=1))

      closeness = distance_worst / (distance_best + distance_worst)

      ranked_indices = np.argsort(-closeness)
      ranked_alternatives = [alternatives[i] for i in ranked_indices]

      print("Относительная близость к идеальному решению:")
      for alt, score in zip(alternatives, closeness):
          print(f"{alt}: {score:.4f}")

      print("\nРанжированный список:")
      for i, alt in enumerate(ranked_alternatives):
    print(f"{i+1}. {alt}")
```

Передадим данные в нашу программу и получим результат, как показано ниже:

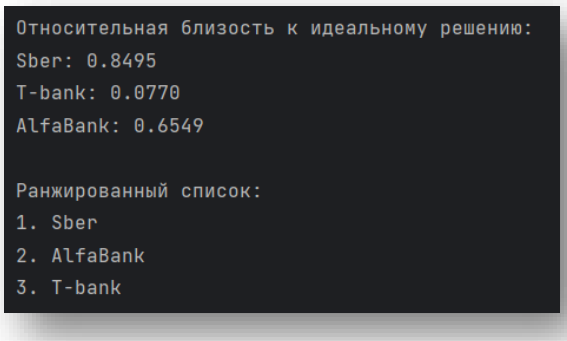

Рисунок 2 – Результат метода TOPSIS

В итоге получилось, что Sber на 1-ом месте, AlfaBank на 2-ом, а T-bank на 3-ем.

2.3. Метод SAW

Метод SAW, или метод простого аддитивного взвешивания, является одним из самых известных и широко используемых методов многоатрибутивного принятия решений. В целом процесс нахождения наилучшего поставщика может быть разделен на следующие этапы:

- анализ по критериям;
- определение весов критериев;
- нормирование критериев;
- определение рейтинга путем умножения значений критериев на веса.

На основе полученного рейтинга принимается решение о выборе. Однако результат оценки может иметь субъективный и неоднозначный характер, т.к. определение критериев оценки и присвоение удельной значимости факторам носят неформализованный характер и зависят от конкретных ситуаций. В зависимости от выбираемых критериев оценки наиболее предпочтительный выбор может меняться. Чаще всего для определения рейтинга используется экспертная оценка показателей с использованием балльной шкалы. Критерии могут оцениваться по пятибалльной, либо десятибалльной шкале. Критерии выбора, веса и оценки определяет менеджер по логистике, после чего происходит подсчет суммарного рейтинга.

На основе этого напишем код для подсчёта и получения ранжированного списка анализируемого ПО (Листинг 3).

Листинг 3. Метод SAW

```python
import numpy as np
data = np.array([
    [0.95, 0.9, 0.9, 0.7, 0.7],  # Сбербанк
    [0.9, 0.9, 0.5, 0.75, 0.75],  # Т-Банк
    [0.98, 0.9, 0.75, 0.75, 0.8]   # Альфа-банк
])
weights = np.array([0.4, 0.2, 0.2, 0.1, 0.1])

norm_data = data / data.max(axis=0)

scores = np.dot(norm_data, weights)

banks = ["Сбербанк", "Т-Банк", "Альфа-банк"]

for i, score in enumerate(scores):

    print(f"{banks[i]}: {score:.3f}")

best_bank = banks[np.argmax(scores)]

print(f"Лучший банк по методу SAW: {best_bank}")
```

Передадим данные в нашу программу и получим результат, как показано ниже:

```
Сбербанк: 0.969
Т-Банк: 0.872
Альфа-банк: 0.967
Лучший банк по методу SAW: Сбербанк
```

Рисунок 3 – Результат метода SAW

В итоге получилось, что Сбербанк на 1-ом месте, Альфа-Банк на 2-ом, , а Т-Банк на 3-ом.

3. Выбор оптимального ПО

На основе результатов выбранных методов получается, что во всех из трёх результатов 1-ое место получило ПО “СберБанк”. Из чего можно сделать вывод, что данное ПО будет самым оптимальным выбором из просмотренных.

Вывод

В ходе выполнения практической работы я научился:

- Выбирать и обосновывать критерии оценки ПО на основе профессиональных требований (сопровождаемость, защищённость, удобство использования и др.).
- Строить нормализованные матрицы оценок с учётом весовых коэффициентов для объективного сравнения альтернатив.
- Реализовывать алгоритмы методов ELECTRE, TOPSIS и SAW на языке программирования Python.
- Сравнивать результаты разных методов (ELECTRE, TOPSIS, SAW) и делать обоснованные выводы о выборе оптимального ПО.

Список источников

1. А. В. Демидовский. Сравнительный анализ методов многокритериального принятия решений: ELECTRE, TOPSIS и ML-LDM [Электронный ресурс]: Электронная статья: Национальный Исследовательский Университет Высшая Школа Экономики, 2020. – режим доступа: URL: https://scm.etu.ru/assets/files/2020/scm20/papers/4/234.pdf, свободный (дата обращения 27.02.2023)

2. Катаржина Халицкая. Выбор технологий с помощью метода TOPSIS [Электронный ресурс]: Электронная статья: Белостокский технический университет, 2020. – режим доступа: URL: https://foresight-journal.hse.ru/data/2020/03/20/1567702093/6-Халицкая-85-96.pdf, свободный (дата обращения 27.02.2023)

3. Леонас Устинович, Зенонас Турскис, Галина Шевченко. ПРИМЕНЕНИЕ МЕТОДА SAW ДЛЯ МНОГОКРИТЕРИАЛЬНОГО СРАВНИТЕЛЬНОГО АНАЛИЗА ВАРИАНТОВ РИСКА ИНВЕСТИЦИЙ В СТРОИТЕЛЬСТВЕ [Электронный ресурс]: Электронная статья: Вильнюсский технический университет им. Гедиминаса, 2006. – режим доступа: URL: https://www.tsi.lv/sites/default/files/editor/science/Research_journals/Tr_Tel/2006/V3/art06-2.pdf, свободный (дата обращения 27.02.2023)