Оглавление

Глава 1

Введение

1.1 Наименование программы

Наименование программы - «Автоматизированная идентификация аномалий методами Process Mining».

1.2 Документы, на основании которых ведётся разработка

Приказ декана факультета компьютерных наук И.В. Аржанцева от 12.12.2018 № 2.3-02/1212-02 "Об утверждении тем, руководителей и консультантов курсовых работ студентовобразовательной программы Прикладная математика и информатика факультета компьютерных наук"

Глава 2

НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

2.1 Функциональное назначение

Во-первых, программа визуализирует бизнес-процесс, предварительно рассчитав оптимальный процент детализации.

Во-вторых, реализуемый инструмент оценивает эффективность работы предприятия при помощи общих Ключевых Показателей Эффективности (далее - КПЭ). Далее, мы анализируем процесс закупки, используя заранее выбранные КПЭ.

В-третьих, программа составляет отчет, описывающий аномалии в бизнес-процессе компании, которые могут свидетельствовать о неэффективном разделении обязанностей сотрудников, наличии повторяющихся событий в цепочке и слишком длительном выполнении цепочки бизнес-процесса.

2.2 Эксплуатационное назначение

Программа предназначена для использования во время анализа бизнеса-процесса. Она разделена на три модуля. Первый модуль отвечает за отрисовку графа бизнес-процесса. Второй модуль предназначен для подсчёта необходимых КПЭ. Третий модуль выявляет аномалии в бизнес-процессе компании. Бизнес-консультант, использующий программу, получает отчет, в котором содержится предварительный анализ бизнес-процесса.

Глава 3

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1 Постановка задачи на разработку программы

Цель данного проекта — разработка программы для выявления аномалий в данных из информационных систем, а также на основе анализа графа бизнес-процесса полученного путём Process mining. Также должны быть подобраны нужные КПЭ, которые впоследствии будут вычисляться, кроме того, надо будет визуализировать граф бизнес-процесса.

3.2 Общее описание структуры программы

Программа будет иметь 3 основных модуля:

1. Модуль создания графа бизнес-процессов и её визуализации. Этот модуль программы также включает в себя алгоритм, который должен будет определить необходимый процент точности визуализации.

2. Модуль расчёта КПЭ. После того, как нужные метрики будут выбраны, их надо будет вычислить с помощью подпрограмм. Всего есть два типа метрик

   (a) Общие метрики. Основные метрики качества, которые будут отражать эффективность деятельности предприятия (например, в качестве основной метрики может быть взято время выполнения той или иной цепочки событий в графе бизнес-процессов) Список общих метрик, которые будут вычисляться:

   1 Самые популярные события – будут вычисляться самые популярные события в информационных системах компании.

   2 Процент завершившихся цепочек.

   3 Сложные цепочки – считаем среднее количество вершин и циклов в цепочках. Если одновременно количество циклов и количество вершин превышают средние показатели для конкретно взятой цепочки более чем на два стандартных отклонения(так определяют 95 % стандартный интервал), то считаем данную цепочку «»сложной цепочкой»» Выводим количество сложных цепочек и процентное отношение количества сложных цепочек к количеству всех цепочек. (например: всего 100 сложных цепочек, и они составляют 5 % от общего количества цепочек).

   4 Общее количество кейсов.

   5 Общее количество уникальных пользователей – общее количество пользователей, которые являются сотрудниками компании и не являются автоматическими пользователями.

6 Уникальные цепочки – в этом КПЭ будет вычисляться общее количество уникальных цепочек.

7 Нестандартные цепочки по времени – Количество нестандартных цепочек по длительности. Нестандартными цепочками считаются такие цепочки, которые отклоняются от среднего времени прохождения по маршруту более чем на два стандартных отклонения.

8 Среднее количество кейсов, созданных за один день.

9 Среднее количество времени, проведенное в рамках одного класса событий

10 Среднее количество циклов на цепочку.

(b) P2P (procure to pay) метрики или специальные метрики они специфичны для модели закупок. Список КПЭ процесса закупок, которые будут вычисляться:

1 Поставщики с высоким уровнем возвратов – Для каждого поставщика считаем след. Величину («взвешенное среднее»): число возвратов делённое на (максимальное число заказов по поставщикам – число заказов у фиксированного поставщика). Далее берем максимум по этим значениям и объявляем его за 100 процентный уровень возвратов. Далее считаем, что поставщик, у которого рассчитанное ранее значение больше 50 % от максимального, имеет высокий уровень возвратов, и наоборот. Данная метрика должна найти количество поставщиков, у которых уровень возвратов превышает 50 % от максимального.

2 Процент заказов у поставщиков с высоким уровнем возвратов – эта метрика, в которой должно быть вычислено общее количество заказов и должен быть найден процент заказов, которые были выполнены поставщиками с высоким уровнем возвратов.

3 Среднее время подтверждения заказа.

4 Количество заказов и услуг в процентом соотношении – эта метрика должна вычислить какой процент заказов — это услуги, а какой закупка товаров.

5 Процент товаров, которые поставляются исключительно внутренними поставщиками.

6 Общая сумма счетов – эта метрика считает общее количество денег, потраченных за определённый промежуток времени.

7 Среднее количество изменений заказа – данная метрика будет считать сколько раз в среднем менялись заказы компании.

8 Количество неправильно завершенных частичных поставок – эта метрика должна вывести количество таких случаев, когда не было полной поставки после частичной, а цепочка является завершенной.

9 Количество денег, потраченных на заказы и услуги в процентом соотношении.

10 Матожидание возврата – для каждого поставщика считаем вероятность возврата, то есть отношение количества возвратов к общему количеству заказов у данного поставщика, затем находим произведение этой вероятности на сумму средств по всем возвратам у этого поставщика и суммируем данное значение по всем поставщикам, выводим в отчёте полученное значение.

3. Модуль обнаружения аномалий. Будет включать в себя подпрограммы, которые в соответствии с выбранными критериями будут искать изъяны в бизнес-процессах предприятия. Будем выделять три критерия для выявления аномалий.

(a) Длительность выполнения того или иного процесса. Если время выполнения бизнес-процесса превышает средние показатели значит это будет определено как аномалия.

(b) Неэффективные процессы, в которых присутствуют циклы или присутствуют слишком часто повторяющиеся циклы также будут определены как аномалии.

(c) SOD (segregation of duties). Этот критерий подразумевает под собой разделение обязанностей внутри предприятия., соответственно если разделение обязанностей не соблюдается тогда эти случаи будут идентифицированы как аномалия. Также эта часть модуля должна выполнять функцию prediction, которая будет предсказывать вероятные повторения такого рода аномалий на предприятии.

3.3 Описание алгоритма визуализации

3.3.1 Используемые обозначения

Перейдем к описанию алгоритма. Для начала введем рассматриваемые в дальшейшем переменные. act - это некоторое событие в бизнес-процессе компании. C[i][j][l] - это трехмерный массив, где i - это номер кластера, j - номер номер цепочки, а l - номер ребра, которое состоит из двух act. J-ая цепочка кластера i хранится в виде $[(act_m, act_{m+1}), (act_{m+1}, act_{m+2}).....,(act_{n_{i,j}-1}, act_{n_{i,j}}), s[i][j]]$, где $n_{i,j}+1$ - это длина массива представляющего j-ую цепочку в i-ом кластере, а s[i][j]- это количество кейсов, совпадающих с j-ой цепочкой кластера i. Введем величину количества кепочек в кластере i и обозначим ее k[i]. Далее, рассмотрим функцию qor(C[i][j][l]), которая вычисляет количество всех кейсов, которое покрываются в соответствующем кластере i.

$$qor(C[i][j][l]) = \sum_{j=0}^{k[i]} C[i][j][n_{i,j}] \tag{3.1}$$

Теперь определим ещё две функции

$$qormin(C[0][j][l], C[1][j][l],....., C[p][j][l]) = \min_i(qor(C[i][j][l])), \tag{3.2}$$

- где p - это количество кластеров.

$$qormax(C[0][j][l], C[1][j][l],....., C[p][j][l]) = \max_i(qor(C[i][j][l])) \tag{3.3}$$

$T = \{T[0], T[1], T[2], ..., T[p]\}$, где T[i] - это количество уникальных вершин в i-ом кластере.

Теперь определим функции

$$f_1(C[i][j][l]) = \frac{qor(C[i][j][l]) - qormin(C[0][j][l],....., C[p][j][l])}{qormax(C[0][j][l], C[1][j][l],....., C[p][j][l]) - qormin(C[0][j][l], C[1][j][l],....., C[p][j][l])} \tag{3.4}$$

$$g_1(T[i]) = \frac{T[i] - \min(T[0], T[1],..., T[p])}{\max(T[0], T[1],..., T[p]) - \min(T[0], T[1],..., T[p])} \tag{3.5}$$

$$h_1(C[i][j][l], T[i]) = f_1(C[i][j][l]) + g_1(T[i]) \tag{3.6}$$

$VisClust = [qor(C[0][j][l]), ..., qor(C[r][j][l])]$ - это массив, каждый элемент которого это сумма всех кейсов цепочек внутри соответствующего визуализированного кластера.

$$qoavr(VisClust) = \sum_i^r VisClust[i] \tag{3.7}$$

Quantity of all visualized repetitions(qoavr) - количество кейсов, которые уже были визуализированы $AllClust = [qor(C[0][j][l]), ..., qor(C[p][j][l])]$ - это массив, который содержит сумму кейсов каждого кластера

$$TotalClusterSum(AllClust) = \sum_{i=0}^{p} (AllClust[i]) \tag{3.8}$$

Пусть $VisVert = [t[0], ...,t[r]]$ (массив всех визуализированных уникальных вершин)

$$qoavv(VisVert) = \sum_{i=0}^{r} VisVert[i] \tag{3.9}$$

Quantity of all visualized vertexes(qoavv) - количество уникальных вершин, которые уже были визуализированы Пусть $AllVert = [t[0], ...,t[p]]$ (массив всех уникальных вершин)

$$TotalVertQuant(AllVert) = \sum_{i=0}^{p} AllVert[i] \tag{3.10}$$

Теперь определим функции:

$$f_2(qoavr(VisClust)) = \frac{qoavr(VisClust) - qormin}{TotalClusterSum(AllClust) - qormin} \tag{3.11}$$

$$g_2(qoavv(VisVert)) = \frac{qoavv(VisVert) - \min(T[0], T[1],..., T[p])}{TotalVertQuant - \min(T[0], T[1],..., T[p])} \tag{3.12}$$

$$h_2(qoavr, qoavv) = f_2(qoavr(VisClust)) + g_2(qoavv(VisVert)) \tag{3.13}$$

3.3.2 Описание алгоритма на псевдокоде

Псевдокод алгоритма:
i = 0
$while(h_1(C[i][j][l], T[i]) > h_2(qoavr(VisClust), qoavv(VisVert))) :$
    VisualizeCluster(C[i][j][l])
    VisClust.append(qor(C[i][j][l]))
    VisVert.append(T[i])
    i += 1

3.4 Описание общих KPI метрик

1. Самые популярные категории событий.
   Данная метрика вычисляет популярные категории событий относительно количества выполненных событий и пользователей. Вход: dataframe - eventlog.
   Выход: Кортеж (act_am_dict, users_am_dict)
   act_am_dict - словарь, в котором ключ - название категории событий('Activity Category')
   Значение - отношение количества событий, входящих в эту категорию, и количества всех событий.
   users_am_dict - словарь, в котором ключ - название категории событий('Activity Category')
   Значение - количество уникальных пользователей, которые участвовали в этой категории событий.

```
def O_1(df):
def O_1(df):
act_am_dict = df.groupby('Activity Category')['Activity'].
agg(['count']).div(df.shape[0]).to_dict()['count']
users_am_dict = df.groupby('Activity Category').
agg({"User": lambda x: x.nunique()}).to_dict()['User']
```

```
sorted_act_am_dict = sorted(act_am_dict.items(),
key=lambda kv: kv[1], reverse=True)
sorted_users_am_dict = sorted(users_am_dict.items(),
key=lambda kv: kv[1], reverse=True)

return sorted_act_am_dict[:5], sorted_users_am_dict[:5]
```

2. Процент завершившихся цепочек
   Считается процент начатых и незаконченных цепочек от общего числа цепочек.
   Вход: dataframe - eventlog.
   Выход: отношение кол-ва завершенных цепочек (т.е. таких цепочек, которые закончились Категорией События (’Activity Category’) == ’Платеж (выравнивание)’) и кол-ва всех цепочек.

```
def O_2(df):
started_cases = set(df.iloc[np.where(df['Activity Category']
== 'Заказ на поставку создан')]['CaseID'])
finished_cases = set(df.iloc[np.where(df['Activity Category']
== 'Платеж(выравнивание)')]['CaseID'])

return len(started_cases & finished_cases) / len(df['CaseID'].unique())
```

3. Сложные цепочки. Считает количество сложных цепочек. Сложная цепочка - такая цепочка, которая отклоняется от среднего больше, чем на одно стандартное отклонение, по количеству кейсов и количеству циклов.
   Вход: dataframe-eventlog.
   Выход: количество сложных цепочек.

```
def convert_to_adj_list(activ):
    decode = list(set(activ))
    dec_dict = {}

    for i in range(len(decode)):
        dec_dict[decode[i]] = i

    res = [[] for i in range(len(decode))]

    for i in range(len(activ) - 1):
        res[dec_dict[activ[i]]].append(dec_dict[activ[i + 1]])

    return res

def O_3(df):
    num_of_act_in_case = df.groupby('CaseID').
    agg({'Activity': 'count'})['Activity'].to_dict()

    num_of_loop_in_case = {}

    cases = df['CaseID'].unique()
    begin = time.time()
    for case in cases:
        chain = df.iloc[np.where(df['CaseID'] == case)]
```

```
            chain = chain.sort_values(by='Event end')
            activ = chain['Activity'].tolist()

            l_am = 0

            if len(activ) != len(set(activ)):

                conv_activ = convert_to_adj_list(activ)
                visited = [0 for i in range(len(conv_activ))]

                def dfs(vert, l_am):
                    visited[vert] = 1
                    for v in conv_activ[vert]:
                        if not visited[v]:
                            l_am = dfs(v, l_am)
                        else:
                            l_am += 1
                    return l_am

                for i in range(len(conv_activ)):
                    if not visited[i]:
                        l_am += dfs(i, l_am)

            num_of_loop_in_case[case] = l_am

        loop_in_case = np.array(list(num_of_loop_in_case.values()))
        act_in_case = np.array(list(num_of_act_in_case.values()))

        av_loop = np.mean(loop_in_case)
        std_loop = np.std(loop_in_case)
        ind_loop = np.where(loop_in_case > av_loop + 2 * std_loop)

        av_act = np.mean(act_in_case)
        std_act = np.std(act_in_case)
        ind_act = np.where(act_in_case > av_act + 2 * std_act)

        return len(set(ind_act[0]) & set(ind_loop[0]))
```

4. Общее количество цепочек. Вычисляет общее количество цепочек.
   Вход: dataframe - eventlog.
   Выход: общее количество цепочек.

```
def O_4(df):
     return len(df['CaseID'].unique())
```

5. Общее количество уникальных пользователей.
   Вход: dataframe - eventlog.
   Выход: количество уникальных пользователей.

```
def O_5(df):
    return len(df['User'].dropna().unique())
```

6. Уникальные цепочки. Считает количество уникальных цепочек.
   Вход: dataframe - eventlog.
   Выход: количество уникальных цепочек.

```
def O_6(df):
    cases = df['CaseID'].unique()

    chain = df.iloc[np.where(df['CaseID'] == cases[0])]
    activ = chain['Activity'].tolist()

    eq_chains = []
    eq_chains.append(activ)

    num_cases = [[] for i in range(len(cases))]
    num_cases[0].append(cases[0])

    d = {}
    d[0] = 1
    k = 1

    for case in cases[1:]:
        chain = df.iloc[np.where(df['CaseID'] == case)]
        activ = chain['Activity'].tolist()
        i = 0

        for key in d.keys():
            if eq_chains[key] == activ:
                d[key] += 1
                num_cases[key].append(case)
                break
            if i == k:
                break
            i += 1
        if i == k:
            d[k] = 1
            eq_chains.append(activ)
            num_cases[k].append(case)
            k += 1
    return d, eq_chains, num_cases
```

7. Нестандартные цепочки по времени. В рамках одного маршрута вычисляет нестандартные цепочки по времени. Нестандартная цепочка - цепочка, которая отклоняется по времени от среднего более чем на два стандартных отклонения.
   Вход: dataframe-eventlog.
   Выход: кортеж(tuple).
   Первое значение, возвращаемое функцией - количество нестандартных по времени цепочек. Второе значение, возвращаемое функцией - процентное соотношение этих цепочек ко всем цепочкам.

```
def O_7(df):
    cases = df['CaseID'].unique()
```

```
    all_time = df.groupby(
        ['CaseID'])['Event end'].apply(
        lambda a: (a.max() - a.min()).total_seconds()
    )
    av = np.mean(all_time)
    std = np.std(all_time)
    ind = np.where(all_time > av + 2 * std)

    return len(ind[0]), len(ind[0]) / len(cases)
```

8. Среднее количество цепочек, созданных за день. Вычисляет среднее количество цепочек, созданных за день. (т. е. таких, у которых Событие('Activity') == 'Заказ на поставку создан')
   Вход: dataframe - eventlog.
   Выход: среднее количество цепочек, созданных за день.

```
def O_8(df):
    df_n = pd.DataFrame(df, columns=['Event end', 'CaseID', 'Activity'])
    df_n['Event end'] = df_n['Event end'].astype(str).apply(lambda x: x[:10])
    am = sum(Counter(df_n.iloc[np.where(df_n['Activity'] ==
    'Заказ на поставку создан')]['Event end']).values())
    days = len(df_n['Event end'].unique())

    return am / days
```

9. Среднее количество времени, проведенное в рамках одного класса.

```
def O_9(df):
    df.sort_values(by='Event end', inplace=True)
    all_time = {cat : 0 for cat in df['Activity Category'].unique()}
    cases = df['CaseID'].unique()

    for case in cases:
        chain = df.iloc[np.where(df['CaseID'] == case)]
        cat = chain['Activity Category'].tolist()
        time = chain['Event end'].tolist()

        for i in range(len(cat) - 2):
            all_time[cat[i]] += (time[i + 1] - time[i]).total_seconds()
    return all_time
```

10. Среднее количество циклов на цепочку. Данная функция вычисляет среднее количество циклов на одну цепочку.
    Вход: dataframe - eventlog.
    Выход: среднее количество циклов на цепочку.

```
def O_10(df):
    cases = df['CaseID'].unique()

    loops = []
```

```
    for case in cases:
        chain = df.iloc[np.where(df['CaseID'] == case)]
        chain = chain.sort_values(by='Event end')
        activ = chain['Activity'].tolist()

        events = set()
        loop_am = 0

        for act in activ:
            if act in events:
                loop_am += 1
                events.add(act)
            else:
                events.add(act)

        loops.append(loop_am)

    return np.array(loops).mean()
```

3.5 Описание КПЭ процеса закупок

1. Поставщики с высоким уровнем возвратов.

```
def P2P_1(df):
    k = 0
    cases = df['CaseID'].unique()
    ret_cases = []

    d_n = {}
    d_ret = {}
    for case in cases:
        c = df.iloc[np.where(df['CaseID'] == case)]
        act = c['Activity Category'].tolist()
        sup = c['Supplier'].tolist()[0]
        if sup in d_n:
            d_n[sup] += 1
        else:
            d_n[sup] = 1
        if 'Поступление материала-Возврат' in act:
            if sup in d_ret:
                d_ret[sup] += 1
            else:
                d_ret[sup] = 1
            ret_cases.append(c['Supplier'].tolist()[0])

    for key in d_ret.keys():
        d_ret[key] /= (1212 - d_n[key])

    weighted = sorted(d_ret.items(), key=lambda x: x[1])
    check = True
    danger_suppliers = []
    quantity_of_danger_sup = 0
```

```
    for i in range(len(weighted)):
        if(check):
            main_weight = weighted[i][1]/2
            check = False
        if(weighted[i][1] > main_weight):
            quantity_of_danger_sup += 1
            danger_suppliers.append(weighted[i][0])
        else:
            break
    print("Количество поставщиков с высоким уровнем возвратов:",
    quantity_of_danger_sup)
```

2. Процент заказов у поставщиков с высоким уровнем возвратов.

```
def P2P_2(df):
    cases = df['CaseID'].unique()
        n = 0
        quantity_of_danger_orders = 0
        for i in range(len(df["CaseID"])):
                if(cases[n] == df["CaseID"][i]):
                        if(df['Supplier'][i] in danger_suppliers):
                                quantity_of_danger_orders += 1
                        i = 0
                        n += 1
                if(n == len(cases)):
                        break
        print("Количество заказов у поставщиков с высоким уровнем возвратов:",
        quantity_of_danger_orders)
        print("Процент заказов у поставщиков с высоким уровнем возвратов:",
        round(quantity_of_danger_orders/len(cases) * 100), "%")
```

3. Среднее время подтверждения заказа.

```
def P2P_3(df):
    cases = df['CaseID'].unique()
        cases = list(cases)
        case_time_list = []
        for i in range(len(cases)):
                case_time_list.append([cases[i], False])
        approved_times = []
        approved = ['Заказ на поставку согласован 1',
        'Заказ на поставку согласован 2']
        df_time_sort = df.sort_values(by = 'Event end')
        df_time_sort = df_time_sort.reset_index(drop = True)
        for j in range(len(df_time_sort) - 1, -1, -1):
                case_index = cases.index(df_time_sort['CaseID'][j])
                if(df_time_sort["Activity"][j] in approved and not
                case_time_list[case_index][1]):
                        try:
                                approved_time =
                                time.mktime(time.strptime(
                                str(df_time_sort['Event end'][j]),
```

```
                                "%Y-%m-%d %H:%M:%S.%f"))
                        except ValueError:
                                approved_time =
                                time.mktime(time.strptime(
                                str(df_time_sort['Event end'][j]),
                                "%Y-%m-%d %H:%M:%S"))
                        case_time_list[case_index].append(approved_time)
                        case_time_list[case_index][1] = True
                if(df_time_sort["Activity"][j]
                == 'Заказ на поставку создан'):
                        try:
                                starting_time =
                                time.mktime(time.strptime(
                                str(df_time_sort['Event end'][j]),
                                "%Y-%m-%d %H:%M:%S.%f"))
                        except ValueError:
                                starting_time =
                                time.mktime(time.strptime(
                                str(df_time_sort['Event end'][j]),
                                "%Y-%m-%d %H:%M:%S"))
                        case_time_list[case_index].append(starting_time)

        cases_without_confirm = 0
        for n in range(len(case_time_list)):
                if(len(case_time_list[n]) == 4):
                        approved_times.append(case_time_list[n][2]
                        - case_time_list[n][3])
                if(len(case_time_list[n]) == 3):
                        cases_without_confirm += 1

        if(len(approved_times) == 0):
                print("Нету согласованных заказов")
        else:
                print("среднее время подтверждения заказа:",
                sum(approved_times)/len(approved_times), "сек.")
                print("Процент кейсов без согласования:",

          round(cases_without_confirm/(cases_without_confirm +
          len(approved_times)) * 100), "%")
                print("Процент согласованных кейсов:",

          round(len(approved_times)/(cases_without_confirm +
          len(approved_times)) * 100), "%")
```

4. Какой процент заказов — это услуги, а какой закупка товаров.

```
def P2P_4(df):
        cases = df['CaseID'].unique()
        n = 0
        services = 0
        goods = 0
        services_total_amount = 0
```

```
        goods_total_amount = 0
        Purchase_type = df['Purchase type'].unique()
        for i in range(len(df["CaseID"])):
                if(cases[n] == df["CaseID"][i]):
                        if(df['Purchase type'][i] == Purchase_type[0]):
                                services += 1
                                if(not np.isnan(df["Amount"][i])):
                                        services_total_amount += df["Amount"][i]
                        else:
                                if(not np.isnan(df["Amount"][i])):
                                        goods_total_amount += df["Amount"][i]
                                goods += 1
                        i = 0
                        n += 1
                if(n == len(cases)):
                        break
        serv_per = services/(services + goods) * 100
        goods_per = goods/(services + goods) * 100
        print("процент заказов, которые являются услугами:", round(serv_per), "%",
        "\nпроцент заказов, которые являются закупкой товаров:", round(goods_per),
```

5. Процент товаров, которые поставляются исключительно внутренними поставщиками.

```
def P2P_5(df):
        Material = df['Material'].unique()
        n = 0
        total_amount_domestic_mat = 0
        for i in range(len(df["CaseID"])):
                if(Material[n] == df["Material"][i]):
                        if(df["Supplier_type"][i] == 'Внутренние'):
                                total_amount_domestic_mat += 1
                                i = 0
                                n += 1
                        else:
                                i = 0
                                n += 1
                if(n == len(Material)):
                        break

print("общее количество внутренних товаров:", total_amount_domestic_mat)
print("процент товаров поставляющихся внутренними поставщиками:",
      round(total_amount_domestic_mat/len(Material) * 100), "%")
```

6. Общее количество потраченных денег.

```
def P2P_6(df):
        cases = df['CaseID'].unique()
        n = 0
        total_amount = 0
        for i in range(len(df["CaseID"])):
                if(cases[n] == df["CaseID"][i]):
```

```
                        if(not np.isnan(df["Amount"][i])):
                                total_amount += df["Amount"][i]
                        i = 0
                        n += 1
                if(n == len(cases)):
                        break
        print("общее количество потраченных денег:", total_amount)
```

7. Среднее количество изменений заказа.

```
def P2P_7(df):
        changes = ['Заказ на поставку бессрочно заблокирован',
        'Заказ на поставку изменен: лимит на недопоставку',
             'Заказ на поставку изменен: группа закупки',
                        'Заказ на поставку изменен: лимит на сверх-поставку',
             'Заказ на поставку изменен: материал',
                        'Заказ на поставку изменен: налоговые условия',
             'Заказ на поставку изменен: запланированный срок доставки в днях',
              'Заказ на поставку изменен: эффективная стоимость',
                         'Заказ на поставку изменен: уменьшена стоимость',
             'Заказ на поставку изменен: уменьшена цена',
                        'Заказ на поставку изменен: уменьшено количество',
             'Заказ на поставку изменен: увеличена стоимость',
                        'Заказ на поставку изменен: увеличено количество',
             'Заказ на поставку изменен: увеличена цена',
                        'Заказ на поставку изменен: срока предоставления скидки',
             'Заказ на поставку изменен: условия оплаты',
                        'Заказ на поставку изменен: завод',
              'Заказ на поставку изменен: статус наличия счета',
                         'Заказ на поставку изменен: валюта',
             'Заказ на поставку изменен: поставщик',
                        'Заказ на поставку изменен: тип документа']

        quantity_of_changes = 0
        for i in range(len(df["CaseID"])):
                if(df["Activity"][i] in changes):
                        quantity_of_changes += 1
        print("всего изменений заказа:", quantity_of_changes)
        print("среднее количество изменений заказа:",
        quantity_of_changes/len(cases))
```

8. Количество неправильно завершенных частичных поставок.

```
def P2P_8(df):
        activ = ['Поступление материала-Возврат-Частичная поставка',
        'Поступление материала-Получение-Завершающая поставка']
        quantity_of_wrong_cases = 0
        for i in range(len(df["CaseID"])):
                bool = True
                curr_case = df["CaseID"][i]
                if(df["Activity"][i] == activ[0]):
                        j = i
```

```
                        while(curr_case == df["CaseID"][j]):
                                if(df["Activity"][i] == activ[1]):
                                        bool = False
                                        break
                                j += 1
                        if(bool):
                                quantity_of_wrong_cases += 1
        print("количество неправильных кейсов:", quantity_of_wrong_cases)
```

9. Количество денег, потраченных на заказы и услуги в процентом соотношении.

```
def P2P_9(services_total_amount, total_amount):
        print("количество денег, которые были потрачены на услуги в процентах:",
     int(round(services_total_amount/total_amount * 100)), "%",
    "\nколичество денег, которые были потрачены на товары в процентах:",
    int(round(goods_total_amount/total_amount * 100)), "%")
```

10. Оборотные капитал: издержки возвратов.

```
def P2P_10(df):
        suppliers = df['Supplier'].unique()
        suppliers = list(suppliers)
        suppliers_list = []
        cases_summed = []
        for i in range(len(suppliers)):
                suppliers_list.append([suppliers[i], 0, 0, 0])
        for j in range(len(df["CaseID"])):
                supplier_index = suppliers.index(df['Supplier'][j])
                if(df['CaseID'][j] not in suppliers_list[supplier_index]):
                        suppliers_list[supplier_index].append(df['CaseID'][j])
                        suppliers_list[supplier_index][3] += 1
                if(df['Activity Category'][j] == 'Поступление материала-Возврат'):
                        if(df['CaseID'][j] not in cases_summed):
                                if(not np.isnan(df["Amount"][j])):
                                        suppliers_list[supplier_index][1]
                                        += df["Amount"][j]
                                cases_summed.append(df['CaseID'][j])
                                suppliers_list[supplier_index][2] += 1
        expec = 0
        for n in range(len(suppliers_list)):
                expec += suppliers_list[n][2]/suppliers_list[n][3] *
                suppliers_list[n][1]
        print("Матожидание возврата:", expec)
```

Глава 4

Технико-экономические показатели

4.1 Ориентировочная экономическая эффективность.

В настоящее время в связи с автоматизацией бизнес-процессов, ведения электронного учета в информационных системах происходит накопление огромного объема данных. Разрабатываемый инструмент предназначается для:

1. Автоматизированного анализа бизнес-процесса.
2. Автоматизированной идентификации аномалий в бизнес-процессах.
3. Вычислении КПЭ для предоставления актуальной информации о работе компании
4. Визуализации граф бизнес-процессов

Инструмент при работе в этих направлениях основывается на работе с методами машинного обучения и методов Process Mining. Одной из ключевых возможностей инструмента является обнаружения часто встречающихся отклонений от стандартного процесса и выявление потенциальных причин его появления, что позволит оптимизировать сам бизнес-процесс и уйти от лишних издержек. Также немаловажным приложением инструмента будет обнаружение нетипичных цепочек событий, что может потенциально снизить риск мошенничества.

4.2 Предполагаемая потребность.

Предполагаемая потребность обуславливается тем фактом, что на данный момент не существует инструмента, позволяющего помимо обнаружения аномалий в бизнес-процессе выявлять причины их появления, который существенно поможет бизнес-аналитикам в исследовании цепочек событий.

Глава 5

ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

5.1 Список источников

1. ГОСТ 19.101-77 Виды программ и программных документов. \\Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

2. ГОСТ 19.102-77 Стадии разработки. \\Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

3. ГОСТ 19.103-77 Обозначения программ и программных документов. \\Единая система программной документации. – М.: ИПК Издательство стандартов, 2001

4. ГОСТ 19.104-78 Основные надписи. \\Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

5. ГОСТ 19.105-78 Общие требования к программным документам. \\Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. \\Единая система программной документации. – М.: ИПК Издательство стандартов, 2001

7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. \\Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.