МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г. Шухова)**

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
И УПРАВЛЯЮЩИХ СИСТЕМ

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

Экспериментальная работа
Получение объективных данных с помощью когнитивных графов

Выполнил:
ст. группы ПВ-233

_____ Мороз Р. А.

Принял:

_____ Поляков В.М.

Белгород 2025 г.

Цель работы: Разработка когнитивной модели для выявления объективных причинно-следственных связей прокрастинации у студентов с использованием нейросетевого автоэнкодера.

В данном исследовании рассматривается проблема студенческой прокрастинации. С целью выявления объективных причинно-следственных связей, лежащих в основе этого явления, была поставлена задача разработки когнитивной модели. Работа включает в себя сбор и анализ данных студенческих опросов, кластеризацию текстовых данных для выявления ключевых тем прокрастинации, статистическую проверку гипотез и построение когнитивного графа с использованием нейросетевого автоэнкодера для моделирования взаимосвязей между факторами.

Задачи работы:

1. Провести сбор данных по теме прокрастинации среди студентов, выполнить их очистку и предварительную обработку, включая кластеризацию текстовых данных.

2. Выполнить исследовательский анализ данных (EDA) для выявления основных статистик и визуализации паттернов.

3. Сформулировать и статистически протестировать гипотезы о связях между различными факторами и прокрастинацией.

4. Разработать, обучить и оценить когнитивную модель на основе нейросетевого автоэнкодера для построения карты влияния факторов.

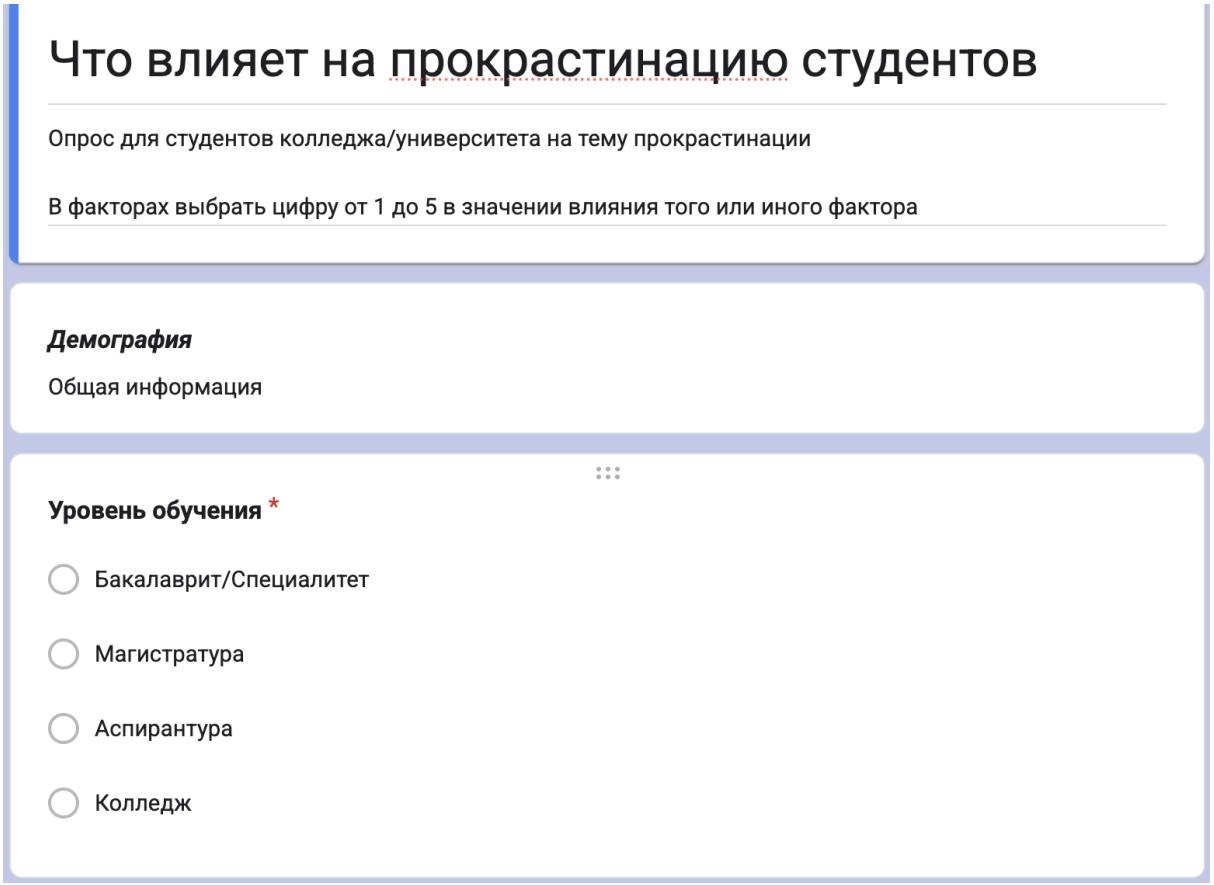
Ход работы

Этап 1. Сбор и первичный анализ данных

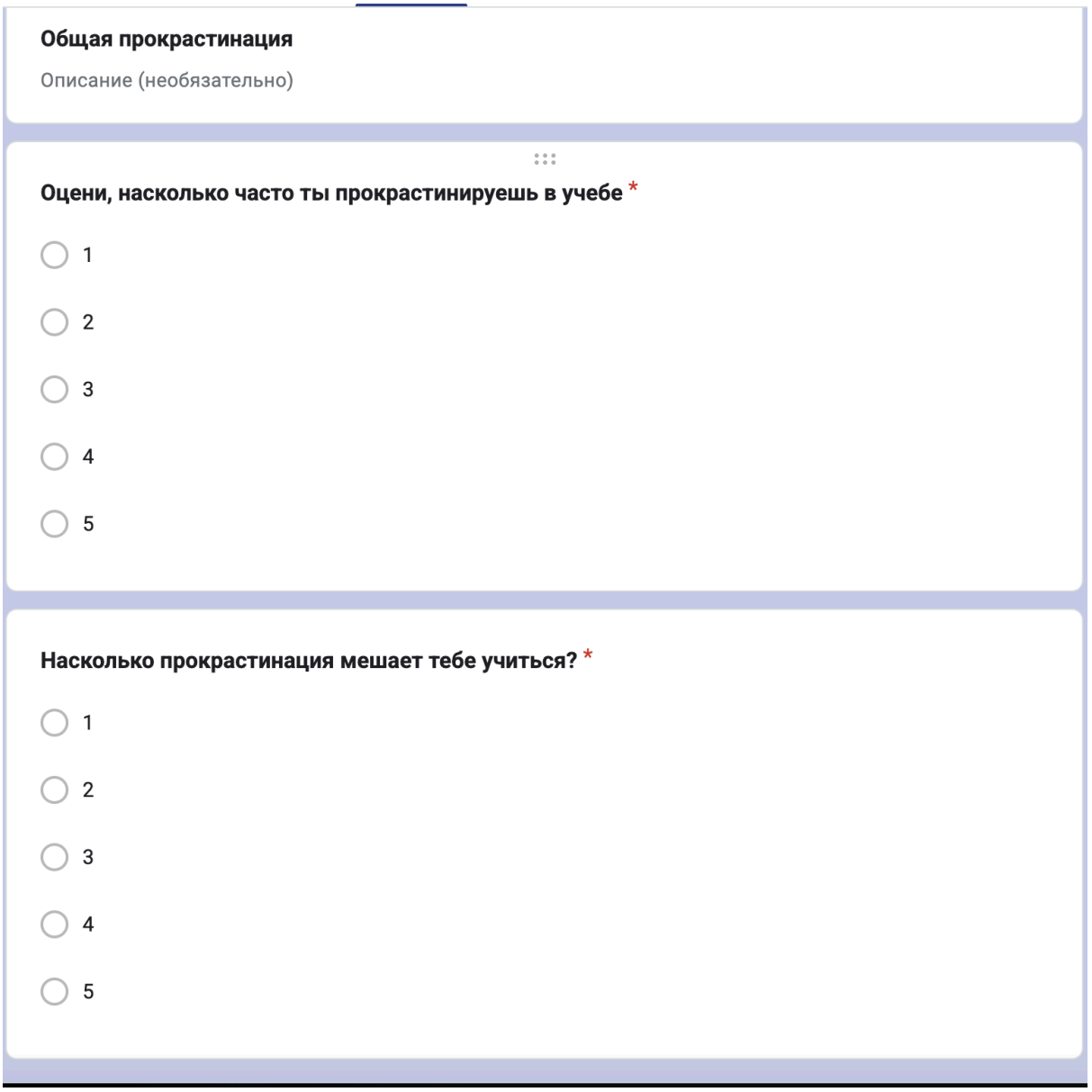
На первом этапе были собраны данные посредством анонимного онлайн-опроса студентов. Особое внимание было уделено обработке колонки procrastination_reason, содержащей текстовые ответы. Для преобразования качественных данных в количественные был применен метод кластеризации.

Пример анкеты изображены на скриншотах. Полная анкета находится в https://docs.google.com/forms/d/1qizXCRKdHtMHeAXkXHiJ50P5WFOJAfmjxUI44vURaTc/edit

На первом скриншоте показана часть с основной информацией о студенте: его уровень обучение, возраст, направление и т.д.



Основная часть с оценкой общей прокрастинации и влияющих на нее факторов.



Факторы-причины

Описание (необязательно)

Тревожность и стресс: *

Я избегаю задач, потому что чувствую тревогу или стресс от них.

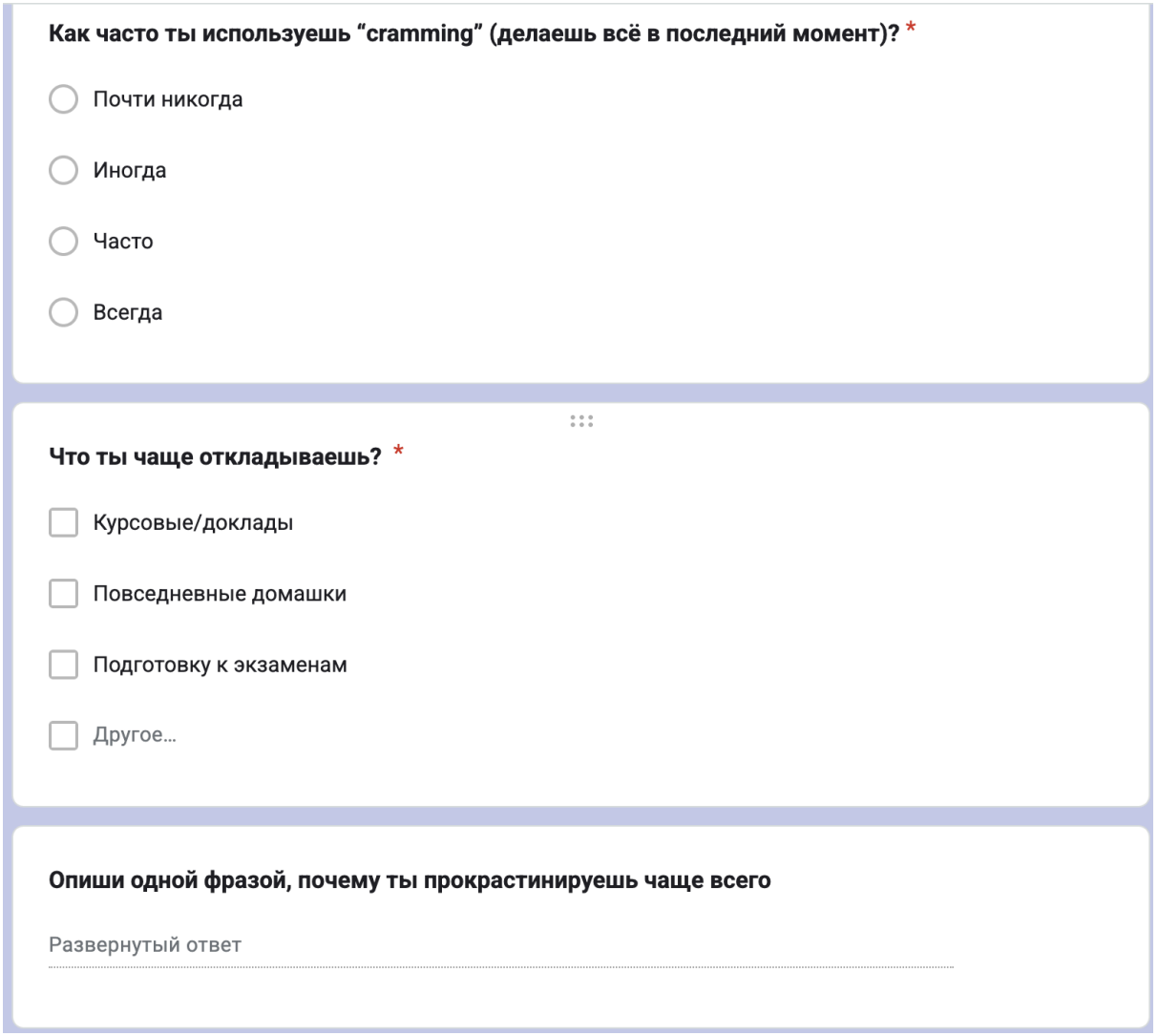
- ○ 1
- ○ 2
- ○ 3
- ○ 4
- ○ 5

Мотивация: *

Я прокрастинирую, когда не вижу смысла в задании.

- ○ 1
- ○ 2
- ○ 3

Также вопросы о заданиях и комментарий своего состояния.



Полная работа и весь проект с данными находится в репозитории: https://github.com/crissyro/4-sem-university/blob/main/system_modeling/sys_mod_finish.ipynb

По итогу тестирования были выделены следующие концепты:

In [4]:
```
column_mapping = {
    'Уровень обучения': 'education_level',
    'Направление': 'major',
    'Пол': 'gender',
    'Возраст': 'age',
    'Оцени, насколько часто ты прокрастинируешь в учебе': 'procrastination_freq',
    'Насколько прокрастинация мешает тебе учиться?': 'procrastination_impact',
    'Тревожность и стресс: Я избегаю задач, потому что чувствую тревогу или стресс от них.': 'anxiety_stress',
    'Мотивация: Я прокрастинирую, когда не вижу смысла в задании.': 'motivation',
    'Наличие дедлайна: Я работаю только когда приближается дедлайн.': 'deadline_dependency',
    'Сон: Недостаток сна снижает мою продуктивность и усиливает прокрастинацию.': 'sleep_impact',
    'Социальные сети: Я часто откладываю дела, уходя в соцсети.': 'social_media',
    'Кофеин: Кофе/энергетики помогают мне начать работу.': 'caffeine',
    'Рабочая среда: Мне сложно сосредоточиться в общежитии/доме/кафе и т.д.': 'environment',
    'Самооценка: Я прокрастинирую, потому что боюсь сделать плохо.': 'self_esteem',
    'Усталость: Я слишком устаю, чтобы начать делать задания.': 'fatigue',
    'Как часто ты используешь “cramming” (делаешь всё в последний момент)?': 'cramming_freq',
    'Что ты чаще откладываешь?': 'postponed_tasks',
    'Опиши одной фразой, почему ты прокрастинируешь чаще всего': 'procrastination_reason'
}

df = df.rename(columns=column_mapping)
```

In [ ]:
```
df.info()
```

Процесс кластеризации:

1. **Препроцессинг текста:** Текстовые ответы были очищены: приведены к нижнему регистру, избавлены от знаков препинания и стоп-слов. Был применен стемминг для приведения слов к их основам (например, "усталость", "устал" → "уста").

```
def preprocess_text(text):
    """
    Функция для очистки и предобработки текста:
    - Удаляет все, кроме букв кириллицы
    - Приводит к нижнему регистру
    - Удаляет стоп-слова
    - Применяет стемминг
    """
    try:
        if pd.isna(text):
            return np.nan
```

```
        # Оставляем только кириллицу и пробелы
        text = re.sub(r'[^а-яё\s]', '', str(text).lower())
        # Стемминг и удаление стоп-слов
        words = [stemmer.stem(w) for w in text.split()
                 if w not in stop_words and len(w) > 2]

        return ' '.join(words) if words else np.nan
    except:
        return np.nan
```

2. **Векторизация (TF-IDF):** Очищенный текст был преобразован в числовые векторы с помощью метода TF-IDF, который оценивает важность каждого слова в контексте ответа и всей совокупности данных.

```
# Векторизация текста с помощью TF-IDF
        tfidf = TfidfVectorizer(
            max_features=200,     # Ограничиваем количество самых частых слов/фраз
            ngram_range=(1, 2),   # Учитываем как отдельные слова, так и пары слов
            min_df=2              # Игнорируем слова, которые встречаются реже, чем в 2 документах
        )
        X = tfidf.fit_transform(valid_data['cleaned_text'])
```

3. **Кластеризация (K-Means):** Полученные векторы были сгруппированы с помощью алгоритма K-Means, который автоматически выделил несколько семантически целостных кластеров (тем) причин прокрастинации, таких как "Страх неудачи и

перфекционизм", "Отсутствие мотивации и энергии" и т.д.

```
# Определяем количество кластеров (не менее 2, не более 5)
        n_clusters = max(2, min(5, len(valid_data) // 5))
        kmeans = KMeans(n_clusters=n_clusters, random_state=42,
n_init='auto')

        # Добавляем номер кластера к каждой причине
        valid_data['cluster'] = kmeans.fit_predict(X)

        # Переносим номера кластеров в основной DataFrame
        df['cluster'] = valid_data['cluster']
```

В результате этого этапа неструктурированные текстовые данные были преобразованы в четкие категориальные факторы, готовые для дальнейшего анализа.

Общий код этапа:

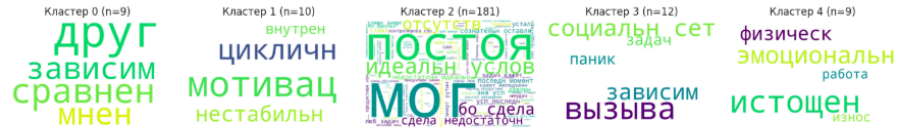
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from wordcloud import WordCloud

import nltk
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer

nltk.download('stopwords')
stop_words = stopwords.words('russian')
stemmer = SnowballStemmer('russian')

def preprocess_text(text):
    """
    Функция для очистки и предобработки текста:
```

```
    - Удаляет все, кроме букв кириллицы
    - Приводит к нижнему регистру
    - Удаляет стоп-слова
    - Применяет стемминг
    """
    try:
        if pd.isna(text):
            return np.nan

        # Оставляем только кириллицу и пробелы
        text = re.sub(r'[^а-яё\s]', '', str(text).lower())
        # Стемминг и удаление стоп-слов
        words = [stemmer.stem(w) for w in text.split()
                 if w not in stop_words and len(w) > 2]

        return ' '.join(words) if words else np.nan
    except:
        return np.nan

def cluster_and_visualize_reasons(df):
    """
    Анализирует текстовые причины прокрастинации, кластеризует их
    и визуализирует результаты в виде облаков слов.
    Не вычисляет веса, а только определяет тематические группы.
    """
    if 'procrastination_reason' not in df.columns:
        raise ValueError("В DataFrame отсутствует необходимая колонка:
'procrastination_reason'")

    # Создаем копию для безопасной работы и добавляем очищенный текст
    analysis_df = df.copy()
    analysis_df['cleaned_text'] = analysis_df['procrastination_reason'].apply(preprocess_text)

    # Отбираем только строки с валидным текстом для анализа
```

```
    valid_data = analysis_df.dropna(subset=['cleaned_text']).copy()

    # Проверяем, достаточно ли данных для кластеризации
    if len(valid_data) < 5:
        print("Недостаточно данных для кластеризации (менее 5 валидных текстовых записей).")
        return df # Возвращаем исходный DataFrame без изменений

    try:
        # Векторизация текста с помощью TF-IDF
        tfidf = TfidfVectorizer(
            max_features=200,    # Ограничиваем количество самых частых слов/фраз
            ngram_range=(1, 2),  # Учитываем как отдельные слова, так и пары слов
            min_df=2             # Игнорируем слова, которые встречаются реже, чем в 2
документах
        )
        X = tfidf.fit_transform(valid_data['cleaned_text'])

        # Определяем количество кластеров (не менее 2, не более 5)
        n_clusters = max(2, min(5, len(valid_data) // 5))
        kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init='auto')

        # Добавляем номер кластера к каждой причине
        valid_data['cluster'] = kmeans.fit_predict(X)

        # Переносим номера кластеров в основной DataFrame
        df['cluster'] = valid_data['cluster']

        # --- Визуализация кластеров ---
        plt.figure(figsize=(15, 7))
        plt.suptitle('Тематические кластеры причин прокрастинации', fontsize=16)

        sorted_clusters = sorted(valid_data['cluster'].unique())

        for i, cluster_num in enumerate(sorted_clusters):
```

```
            # Собираем весь текст из текущего кластера
            cluster_text = ' '.join(valid_data[valid_data['cluster'] ==
cluster_num]['cleaned_text'])

            # Считаем количество элементов в кластере
            cluster_size = len(valid_data[valid_data['cluster'] == cluster_num])

            # Генерируем облако слов
            wordcloud = WordCloud(width=800, height=500, background_color='white',
colormap='viridis').generate(cluster_text)

            plt.subplot(1, len(sorted_clusters), i + 1)
            plt.imshow(wordcloud, interpolation='bilinear')
            plt.axis('off')
            plt.title(f'Кластер {cluster_num} (n={cluster_size})')

        plt.tight_layout(rect=[0, 0, 1, 0.96])
        plt.show()

    except Exception as e:
        print(f"Ошибка во время анализа текста: {str(e)}")

    return df

result_df = cluster_and_visualize_reasons(df)

print("\nРезультаты кластеризации причин:")
if 'cluster' in result_df.columns:
    print(result_df[['procrastination_reason', 'cluster']].dropna().head(15))
else:
    print("Колонка 'cluster' не была добавлена.")
```

Результат работы:



Результаты кластеризации причин:

	procrastination_reason	cluster
0	Физическое истощение	4
1	Нет внешнего контроля	2
2	Осознанное откладывание	2
3	Лень	2
4	Любая задача кажется неподъемной	2
5	Цифровая перегрузка	2
6	Легко переключаюсь между задачами	2
7	Прокрастинирую только по неинтересным предметам	2
8	Боюсь пропустить что-то важное	2
9	Отсутствие энергии	2
10	FOMO (страх упущенной возможности)	2
11	Не могу контролировать свои циклы	2
12	Пик продуктивности ночью	2
13	Подавляющее чувство ответственности	2
14	Любая задача кажется неподъемной	2

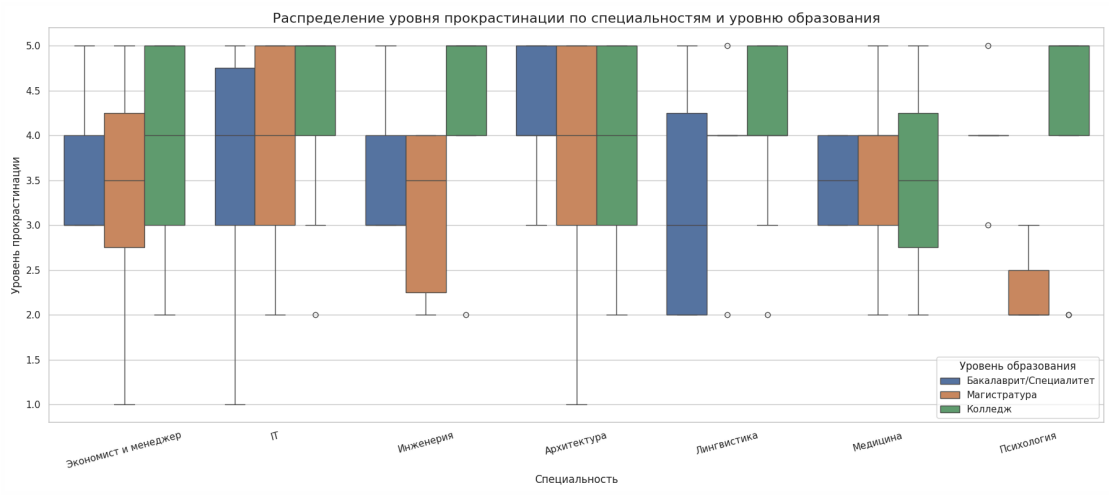
Этап 2. Исследовательский анализ и визуализация

Для глубокого понимания структуры данных и выявления неочевидных закономерностей был проведен исследовательский анализ с построением ряда визуализаций.

1. **Сравнение по группам (Box Plot)**: Были построены диаграммы размаха для сравнения уровня прокрастинации между студентами разных специальностей и уровней образования. Это позволило визуально оценить медианные значения и разброс данных, а также выявить "группы риска".

```
plt.figure(figsize=(18, 8))

sns.boxplot(

    x='major',

    y='procrastination_freq',

    data=df,

    hue='education_level',


order=df.groupby('major')['procrastination_freq'].median().sort_values
().index

)
```

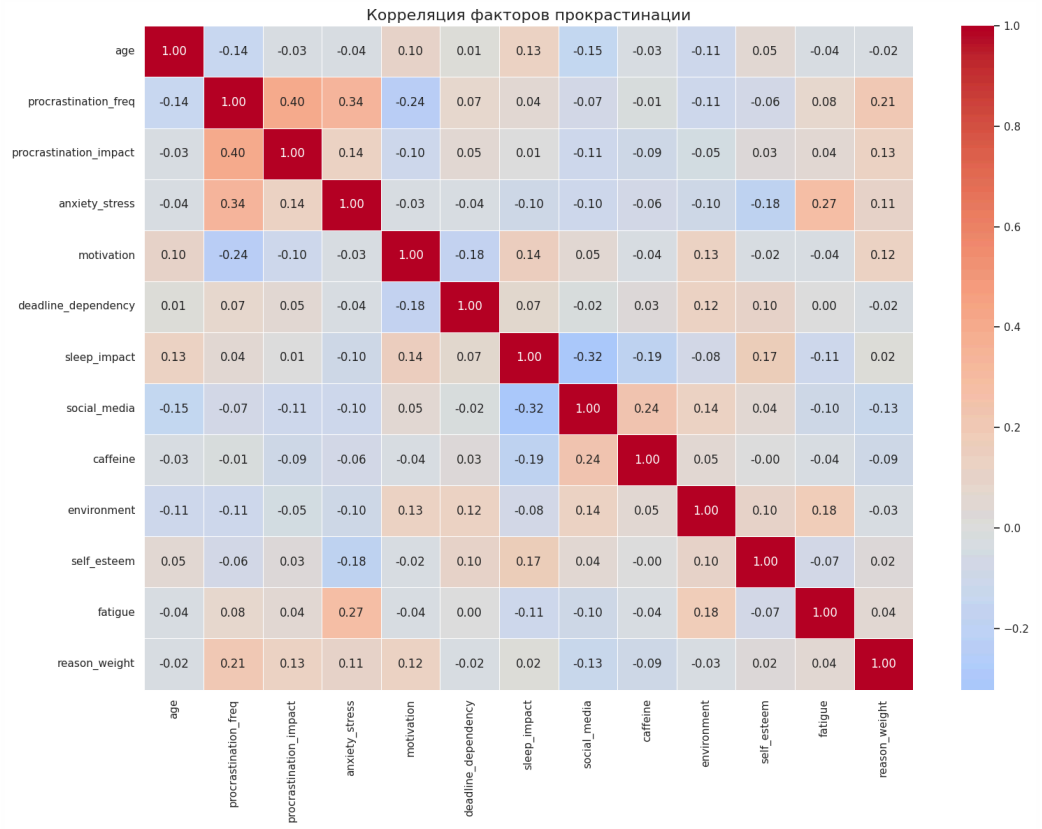
```
plt.title('Распределение уровня прокрастинации по специальностям и уровню образования', fontsize=16)

plt.xticks(rotation=15)

plt.xlabel('Специальность')

plt.ylabel('Уровень прокрастинации')

plt.legend(title='Уровень образования')

plt.tight_layout()

plt.show()
```



2. **Анализ взаимосвязей (Heatmap):** Была построена тепловая карта корреляций для всех числовых переменных. Этот анализ выявил наличие и направление линейных связей между факторами, например, положительную корреляцию между тревогой и прокрастинацией и отрицательную — между мотивацией и прокрастинацией.

```
corr_matrix = df.select_dtypes(include=np.number).corr()


plt.figure(figsize=(16, 12))
sns.heatmap(
    corr_matrix,
    annot=True,
```

```
    fmt=".2f",
    cmap="coolwarm",
    center=0,
    linewidths=.5
)
plt.title('Корреляция факторов прокрастинации', fontsize=16)
plt.tight_layout()
plt.show()
```



3. **Профилирование специальностей (Radar Chart):** Для каждой специальности были построены радиальные диаграммы, создающие уникальный "профиль" на основе средних значений по ключевым факторам (тревожность, усталость, влияние соцсетей и др.). Это позволило сравнить каждую специальность с общим средним и выявить ее специфику.

```
def plot_radar_chart(direction):
    profile_df = df[df['major'] == direction]

    if profile_df.empty:
        print(f"Нет данных для направления: {direction}")
        return

    profile_data = profile_df.mean(numeric_only=True)

    categories = [
        'procrastination_freq', 'procrastination_impact',
        'anxiety_stress', 'fatigue', 'social_media',
        'caffeine', 'sleep_impact'
    ]

    values = profile_data[categories].values.tolist()
    values += values[:1]

    num_vars = len(categories)
    angles = np.linspace(0, 2 * np.pi, num_vars,
endpoint=False).tolist()
    angles += angles[:1]

    fig = go.Figure()

    fig.add_trace(go.Scatterpolar(
        r=values,
        theta=categories + [categories[0]],
        fill='toself',
        name=direction,
```

```
        line=dict(color='royalblue', width=3),
        fillcolor='rgba(65, 105, 225, 0.4)'
    ))

    avg_values = df[categories].mean().values.tolist()
    avg_values += avg_values[:1]

    fig.add_trace(go.Scatterpolar(
        r=avg_values,
        theta=categories + [categories[0]],
        name='Средние значения',
        line=dict(color='gray', dash='dot', width=2),
        opacity=0.6
    ))

    fig.update_layout(
        polar=dict(
            radialaxis=dict(
                visible=True,
                range=[0, 5],
            ),
            angularaxis=dict(
                tickfont=dict(size=12),
                gridcolor='lightgray'
            ),
        ),
        title=dict(
            text=f'Профиль прокрастинации: {direction}',
            x=0.5,
            font=dict(size=20, family='Arial')
```

```
        ),
        showlegend=True,
        height=600,
        width=800,
        paper_bgcolor='white',
        plot_bgcolor='white'
    )

    fig.show()

unique_majors = df['major'].unique()

for direction in unique_majors:
    plot_radar_chart(direction)
```

Этап 3. Статистическая проверка гипотез

На основе наблюдений из EDA были сформулированы и проверены четыре основные гипотезы с уровнем значимости $\alpha=0.05$.

1. **Гипотеза о влиянии пола:** T-тест не выявил статистически значимых различий в уровне прокрастинации между мужчинами и женщинами ($p > 0.05$).

```
print("\n### Гипотеза 1: Различие в прокрастинации между полами ###")
male_scores = df[df['gender'] == 'Мужской']['procrastination_freq']
female_scores = df[df['gender'] == 'Женский']['procrastination_freq']

# Проводим t-тест для двух независимых выборок
t_stat, p_value_gender = stats.ttest_ind(male_scores, female_scores,
equal_var=False) # equal_var=False (тест Уэлча) более устойчив
```

```
print(f"\nСредняя частота прокрастинации (мужчины):
{male_scores.mean():.2f}")

print(f"Средняя частота прокрастинации (женщины):
{female_scores.mean():.2f}")



print(f"\nt-статистика: {t_stat:.4f}")

print(f"p-value: {p_value_gender:.4f}")



if p_value_gender < 0.05:

    print("\nРезультат: p < 0.05. Отвергаем нулевую гипотезу.
Существуют статистически значимые различия.\n")

else:

    print("\nРезультат: p >= 0.05. Не удалось отвергнуть нулевую
гипотезу. Статистически значимых различий не обнаружено.\n")
```

2. **Гипотеза о влиянии уровня образования:** Дисперсионный анализ (ANOVA) показал наличие статистически значимых различий в уровне прокрастинации между студентами разных уровней образования ($p < 0.05$).

```
print("\n### Гипотеза 2: Влияние уровня образования на прокрастинацию
###")



education_levels = df['education_level'].unique()

grouped_data = [df['procrastination_freq'][df['education_level'] ==
level] for level in education_levels]



f_stat, p_value_edu = stats.f_oneway(*grouped_data)



print(f"\nУровни образования для сравнения:
{list(education_levels)}\n")



print(f"F-статистика: {f_stat:.4f}")

print(f"p-value: {p_value_edu:.4f}")
```

```
if p_value_edu < 0.05:

    print("\nРезультат: p < 0.05. Отвергаем нулевую гипотезу.
Существуют статистически значимые различия в частоте прокрастинации
между группами.\n")

else:

    print("\nРезультат: p >= 0.05. Не удалось отвергнуть нулевую
гипотезу. Различия между группами статистически не значимы.\n")
```

3. **Гипотеза о связи с тревогой:** Ранговая корреляция Спирмена подтвердила наличие статистически значимой положительной связи между уровнем тревожности и частотой прокрастинации ($\rho = 0.29$, $p < 0.05$).

```
print("\n### Гипотеза 3: Связь между тревогой и прокрастинацией ###")


corr_anxiety, p_value_anxiety =
stats.spearmanr(df['procrastination_freq'], df['anxiety_stress'])


print(f"\nКоэффициент корреляции Спирмена: {corr_anxiety:.4f}")

print(f"p-value: {p_value_anxiety:.4f}")


if p_value_anxiety < 0.05:

    print("\nРезультат: p < 0.05. Отвергаем нулевую гипотезу.
Обнаружена статистически значимая корреляция.\n")

    if corr_anxiety > 0:

        print("Тип связи: положительная. С ростом тревожности растет и
частота прокрастинации.\n")

    else:

        print("Тип связи: отрицательная. С ростом тревожности частота
прокрастинации снижается.\n")

else:

    print("\nРезультат: p >= 0.05. Не удалось отвергнуть нулевую
гипотезу. Статистически значимой корреляции не обнаружено.\n")
```

4. **Гипотеза о связи с мотивацией:** Ранговая корреляция Спирмена подтвердила наличие статистически значимой отрицательной связи между уровнем мотивации и частотой прокрастинации ($\rho = -0.22$, $p <$

0.05).

```
print("\n### Гипотеза 4: Связь между мотивацией и прокрастинацией
###")



corr_motivation, p_value_motivation =
stats.spearmanr(df['procrastination_freq'], df['motivation'])



print(f"\nКоэффициент корреляции Спирмена: {corr_motivation:.4f}")

print(f"p-value: {p_value_motivation:.4f}")



if p_value_motivation < 0.05:

    print("\nРезультат: p < 0.05. Отвергаем нулевую гипотезу.
Обнаружена статистически значимая корреляция.\n")

    if corr_motivation > 0:

        print("Тип связи: положительная. С ростом мотивации растет и
частота прокрастинации.\n")

    else:

        print("Тип связи: отрицательная. С ростом мотивации частота
прокрастинации снижается.\n")

else:

    print("\nРезультат: p >= 0.05. Не удалось отвергнуть нулевую
гипотезу. Статистически значимой корреляции не обнаружено.\n")
```

Этап 4. Разработка когнитивной модели

Для построения итоговой карты влияния была разработана и обучена нейросетевая модель-автоэнкодер на базе PyTorch.

1. **Архитектура модели:** Модель состояла из входного слоя (11 факторов), скрытого слоя сжатия (6 нейронов, активация *ReLU*) и выходного слоя реконструкции (11 нейронов).

```
self.encoder = nn.Sequential(

            nn.Linear(num_concepts, hidden_size),
```

```
            nn.ReLU()  # Функция активации для нелинейности
        )
```

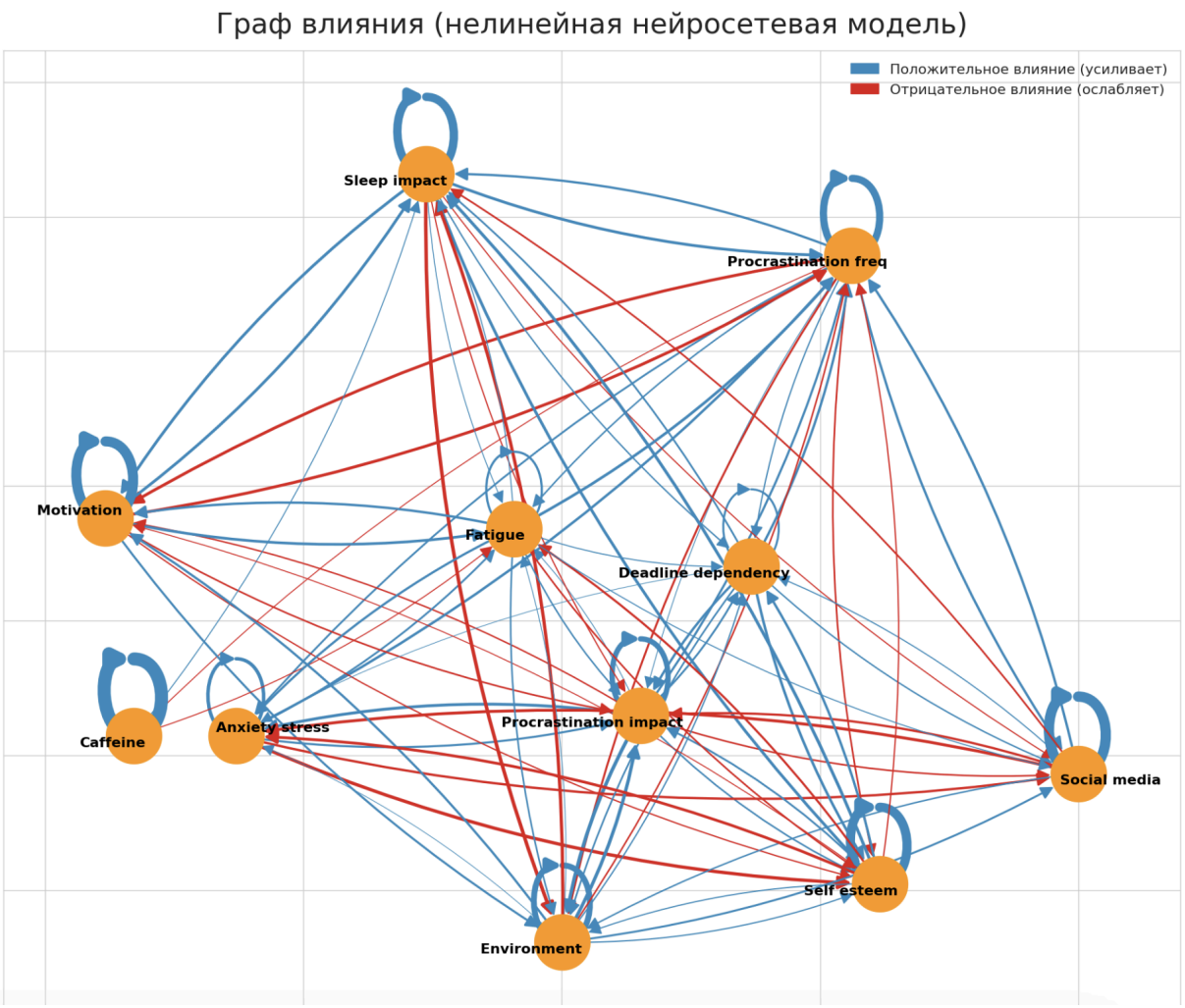
2. **Обучение:** Модель обучалась на протяжении 500 эпох с целью минимизации ошибки реконструкции (*MSE*) между входными и выходными данными.

```
# Цикл обучения
        for epoch in range(epochs):
            for x_batch, y_batch in train_loader:
                optimizer.zero_grad()  # Обнуление градиентов
                y_pred = self.model(x_batch)  # Прямой проход
                loss = criterion(y_pred, y_batch)  # Расчет потерь
                loss.backward()  # Обратное распространение
                optimizer.step()  # Обновление весов

            # Периодический вывод прогресса
            if (epoch + 1) % 50 == 0:
                print(f"Эпоха {epoch+1}/{epochs}, Потеря (Loss): {loss.item():.6f}")
```

3. **Получение связей:** Сила и направление влияния между факторами были вычислены на основе матрицы Якоби обученной модели, что позволило учесть нелинейные зависимости в системе.

```
# Вычисление матрицы влияния через якобиан
        mean_input = torch.FloatTensor(

self.scaler.transform(df[self.concepts].mean().values.reshape(1, -1))
        )
        jacobian = torch.autograd.functional.jacobian(self.model, mean_input)
        influence_matrix = jacobian.squeeze().detach().cpu().numpy()
```

Вывод программы:



Вывод: В ходе выполнения эксперимента была успешно решена поставленная цель — разработана когнитивная модель для анализа причин студенческой прокрастинации.

1. Задача 1 (сбор и кластеризация): Были собраны и обработаны данные. Применение методов NLP позволило успешно преобразовать качественные текстовые данные в структурированные факторы.
2. Задача 2 (EDA): Проведенный исследовательский анализ позволил выявить ключевые паттерны и различия в данных, которые легли в основу для последующих этапов.
3. Задача 3 (проверка гипотез): Статистически подтверждены значимые связи между прокрастинацией и такими факторами, как уровень образования, тревожность и мотивация, что придало результатам научную обоснованность.
4. Задача 4 (моделирование): Разработана и обучена нейросетевая модель-автоэнкодер с высокой объяснительной способностью ($R2=0.694$). Модель позволила построить итоговый когнитивный граф, который наглядно демонстрирует, что прокрастинация является сложным системным явлением, где ключевую роль играют каскадные эффекты, в первую очередь цепочка "Тревожность → Усталость → Прокрастинация".

Работа демонстрирует эффективность применения гибридного подхода, сочетающего статистику и машинное обучение, для анализа сложных поведенческих феноменов.