Лабораторная работа: Анализ и визуализация данных с использованием Python

1. Цели лабораторной работы

Основной целью данной лабораторной работы является формирование у студентов практических навыков работы с данными в среде Python с использованием современных библиотек для анализа данных. Студенты должны освоить полный цикл работы с данными: от первичной предобработки до создания информативных визуализаций и формулирования выводов на основе проведенного анализа.

2. Задачи лабораторной работы

1. Изучить основные методы загрузки и предварительного анализа данных с помощью библиотеки pandas
2. Освоить техники очистки и предобработки данных (обработка пропусков, выбросов, дубликатов)
3. Применить методы статистического анализа данных с использованием numpy и pandas
4. Создать информативные визуализации с помощью matplotlib и seaborn
5. Научиться интерпретировать результаты анализа и формулировать выводы
6. Выполнить индивидуальное задание по анализу предложенного набора данных

3. Теоретические сведения

3.1 Предобработка данных

Предобработка данных является критически важным этапом любого аналитического проекта. От качества предобработки зависит достоверность и точность последующего анализа.

Основные этапы предобработки:

Загрузка данных:

```
import pandas as pd
import numpy as np

# Загрузка из различных форматов
df = pd.read_csv('data.csv')
df = pd.read_excel('data.xlsx')
df = pd.read_json('data.json')
```

Первичный анализ структуры данных:

```
# Основная информация о датасете
print(df.info())
print(df.describe())
print(df.head())
# Проверка размерности
print(f"Размер датасета: {df.shape}")
# Типы данных
print(df.dtypes)
```

Обработка пропущенных значений:

```
# Поиск пропусков
print(df.isnull().sum())
print(df.isnull().sum() / len(df) * 100)  # в процентах
# Методы обработки пропусков
df_cleaned = df.dropna()  # удаление строк с пропусками
df['column'] = df['column'].fillna(df['column'].mean())  # заполнение средним
df['column'] = df['column'].fillna(method='forward')  # прямое заполнение
```

Обработка дубликатов:

```
# Поиск и удаление дубликатов
print(f"Количество дубликатов: {df.duplicated().sum()}")
df_unique = df.drop_duplicates()
```

Обработка выбросов:

```
# Метод межквартильного размаха (IQR)
Q1 = df['column'].quantile(0.25)
Q3 = df['column'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Фильтрация выбросов
df_filtered = df[(df['column'] >= lower_bound) & (df['column'] <= upper_bound)]
```

Преобразование типов данных:

```
# Преобразование строк в даты
df['date_column'] = pd.to_datetime(df['date_column'])

# Преобразование категориальных переменных
df['category'] = df['category'].astype('category')

# Кодирование категориальных переменных
df_encoded = pd.get_dummies(df, columns=['category_column'])
```

3.2 Анализ данных

Статистический анализ позволяет выявить закономерности, тренды и взаимосвязи в данных.

Описательная статистика:

```
# Основные статистические показатели
stats = df.describe()

# Дополнительные меры
median = df['column'].median()
mode = df['column'].mode()[0]
variance = df['column'].var()
std_dev = df['column'].std()
skewness = df['column'].skew()
kurtosis = df['column'].kurtosis()
```

Корреляционный анализ:

```
# Матрица корреляций
correlation_matrix = df.corr()

# Корреляция между двумя переменными
correlation = df['var1'].corr(df['var2'])

# Проверка значимости корреляции
from scipy.stats import pearsonr
corr_coef, p_value = pearsonr(df['var1'], df['var2'])
```

Группировка и агрегация:

```
# Группировка по категории
grouped = df.groupby('category')
# Агрегирующие функции
group_stats = df.groupby('category').agg({
    'numeric_column': ['mean', 'std', 'count'],
    'another_column': 'sum'
})
# Сводные таблицы
pivot_table = df.pivot_table(
    values='value_column',
    index='row_category',
    columns='col_category',
    aggfunc='mean'
)
```

Временные ряды:

```
# Установка индекса времени
df.set_index('date_column', inplace=True)
# Ресемплинг
monthly_data = df.resample('M').mean()
# Скользящие средние
df['rolling_mean'] = df['value'].rolling(window=7).mean()
```

3.3 Визуализация данных

Визуализация данных является ключевым инструментом для понимания структуры данных, выявления закономерностей и эффективной презентации результатов анализа.

Настройка среды визуализации:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# Настройка стиля
plt.style.use('seaborn-v0_8')
sns.set_palette("husl")
plt.rcParams['figure.figsize'] = (10, 6)
plt.rcParams['font.size'] = 12
```

Одномерная визуализация

Количественные признаки

Гистограммы и графики плотности показывают распределение значений количественной переменной, позволяют определить форму распределения (нормальное, асимметричное, мультимодальное), выявить выбросы и аномалии. Используются при изучении распределения непрерывных переменных (цены, доходы, возраст), проверке нормальности распределения, поиске оптимального количества интервалов для дискретизации.

```
# Простая гистограмма
plt.figure(figsize=(10, 6))
plt.hist(df['price'], bins=30, alpha=0.7, edgecolor='black')
plt.title('Распределение цен')
plt.xlabel('Цена')
plt.ylabel('Частота')
plt.grid(True, alpha=0.3)

# Гистограмма с кривой плотности (seaborn)
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='price', kde=True, bins=30)
plt.title('Распределение цен с кривой плотности')

# График плотности
plt.figure(figsize=(10, 6))
sns.kdeplot(data=df, x='price', fill=True)
plt.title('График плотности распределения цен')
```

Box plot компактно показывает основные статистические характеристики распределения: медиану, квартили, размах и выбросы. Позволяет быстро оценить симметричность распределения и наличие аномальных значений. Используется для сравнения распределений между группами, выявления выбросов, оценки разброса данных, когда важна медиана, а не среднее значение.

```
# Простой boxplot
plt.figure(figsize=(8, 6))
plt.boxplot(df['price'])
plt.title('Коробчатая диаграмма цен')
plt.ylabel('Цена')

# Seaborn boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(y=df['price'])
plt.title('Распределение цен (boxplot)')
```

Violin plot сочетает возможности boxplot и графика плотности. Показывает не только квартили и выбросы, но и форму распределения на разных уровнях значений. Используется когда нужна детальная информация о форме распределения, при сравнении мультимодальных распределений между группами, для презентации данных с высокой детализацией.

```
# Violin plot для одной переменной
plt.figure(figsize=(8, 6))
sns.violinplot(y=df['price'])
plt.title('Скрипичная диаграмма цен')
```

Describe предоставляет численные характеристики распределения: среднее, медиану, стандартное отклонение, квартили, минимум и максимум. Используется как первый шаг анализа данных, для получения общего представления о переменной, при подготовке отчетов с численными характеристиками.

```
# Подробная статистика
print("Описательная статистика:")
print(df['price'].describe())

# Дополнительные статистики
print(f"Медиана: {df['price'].median()}")
print(f"Мода: {df['price'].mode().values}")
print(f"Коэффициент асимметрии: {df['price'].skew()}")
print(f"Коэффициент эксцесса: {df['price'].kurtosis()}")
```

Категориальные и бинарные признаки

Frequency table показывает, как часто встречается каждое значение категориальной переменной в абсолютных числах и процентах. Используется для анализа распределения категориальных переменных, выявления редких категорий, подготовки данных для дальнейшего анализа.

```
# Абсолютные частоты
freq_table = df['category'].value_counts()
print("Таблица абсолютных частот:")
print(freq_table)

# Относительные частоты
rel_freq = df['category'].value_counts(normalize=True)
print("\nТаблица относительных частот:")
print(rel_freq)

# Сводная таблица
summary_table = pd.DataFrame({
    'Частота': freq_table,
    'Процент': rel_freq * 100
})
print(summary_table)
```

Bar plot визуально отображает частоты категорий, позволяет легко сравнить популярность разных категорий. Используется для визуализации распределения категориальных переменных, сравнения частот между категориями, в презентациях и отчетах.

```
# Простая столбчатая диаграмма
plt.figure(figsize=(10, 6))
df['category'].value_counts().plot(kind='bar')
plt.title('Распределение по категориям')
plt.xlabel('Категория')
plt.ylabel('Количество')
plt.xticks(rotation=45)

# Seaborn countplot
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='category')
plt.title('Количество наблюдений по категориям')
plt.xticks(rotation=45)

# Горизонтальная диаграмма
plt.figure(figsize=(10, 6))
sns.countplot(data=df, y='category', order=df['category'].value_counts().index)
plt.title('Распределение по категориям (горизонтальное)')
```

Многомерная визуализация

Correlation matrix показывает силу и направление линейной связи между всеми парами количественных переменных одновременно. Используется при исследовательском анализе данных для выявления взаимосвязей, отборе признаков для моделирования, поиске мультиколлинеарности.

```
# Вычисление корреляций
corr_matrix = df.select_dtypes(include=[np.number]).corr()

# Тепловая карта корреляций
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0,
            square=True, fmt='.2f', cbar_kws={'label': 'Коэффициент
корреляции'})
plt.title('Матрица корреляций')

# Маска для верхнего треугольника (избежание дублирования)
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, mask=mask, annot=True, cmap='coolwarm', center=0)
plt.title('Матрица корреляций (нижний треугольник)')
```

Scatter plot показывает взаимосвязь между двумя количественными переменными, позволяет выявить линейные и нелинейные зависимости, выбросы. Используется для изучения связи между парами переменных, проверки предположений о линейности, выявления аномальных наблюдений, демонстрации трендов.

```
# Простая диаграмма рассеяния
plt.figure(figsize=(10, 6))
plt.scatter(df['area'], df['price'], alpha=0.6)
plt.xlabel('Площадь')
plt.ylabel('Цена')
plt.title('Зависимость цены от площади')

# Seaborn scatterplot с дополнительными возможностями
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='area', y='price', hue='rooms', size='floor')
plt.title('Зависимость цены от площади (с учетом комнат и этажа)')

# С линией тренда
plt.figure(figsize=(10, 6))
sns.regplot(data=df, x='area', y='price', scatter_kws={'alpha':0.6})
plt.title('Зависимость цены от площади с линией тренда')
```

Scatterplot matrix показывает все возможные парные связи между множеством количественных переменных в одном графике. Используется при исследовательском анализе многомерных данных, для быстрого обзора всех взаимосвязей, поиска интересных паттернов между переменными.

```
# Pairplot для всех количественных переменных
numeric_cols = df.select_dtypes(include=[np.number]).columns
sns.pairplot(df[numeric_cols], diag_kind='hist')
plt.suptitle('Матрица диаграмм рассеяния', y=1.02)

# Pairplot с группировкой по категориальной переменной
sns.pairplot(df, vars=['price', 'area', 'rooms'], hue='district')
plt.suptitle('Матрица рассеяния по районам', y=1.02)
```

Количественные против категориальных

Этот тип визуализации показывает, как распределяется количественная переменная в разных категориях, позволяет сравнить группы по центральной тенденции и разбросу. Используется для сравнения групп, проверки гипотез о различиях между группами, анализа влияния категориального фактора на количественную переменную.

```
# Box plot для группированных данных
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='category', y='price')
plt.title('Распределение цен по категориям')
plt.xticks(rotation=45)

# Violin plot для сравнения распределений
plt.figure(figsize=(12, 6))
sns.violinplot(data=df, x='category', y='price')
plt.title('Сравнение распределений цен по категориям')
plt.xticks(rotation=45)

# Strip plot (точечная диаграмма)
plt.figure(figsize=(12, 6))
sns.stripplot(data=df, x='category', y='price', size=4, alpha=0.7)
plt.title('Точечная диаграмма цен по категориям')
plt.xticks(rotation=45)

# Swarm plot (роевая диаграмма)
plt.figure(figsize=(12, 6))
sns.swarmplot(data=df, x='category', y='price', size=3)
plt.title('Роевая диаграмма цен по категориям')
plt.xticks(rotation=45)

# Группированная статистика
grouped_stats = df.groupby('category')['price'].agg(['mean', 'median', 'std'])
print("Статистика по группам:")
print(grouped_stats)
```

Contingency table показывает совместное распределение двух категориальных переменных, позволяет выявить ассоциации между категориями. Используется для анализа связи между категориальными переменными, проверки независимости признаков, подготовки данных для анализа соответствий.

```
# Создание таблицы сопряженности
contingency_table = pd.crosstab(df['category1'], df['category2'])
print("Таблица сопряженности:")
print(contingency_table)

# Таблица с процентами
contingency_percent = pd.crosstab(df['category1'], df['category2'],
normalize='index') * 100
print("\nТаблица сопряженности (%):")
print(contingency_percent.round(1))

# Тепловая карта таблицы сопряженности
plt.figure(figsize=(10, 6))
sns.heatmap(contingency_table, annot=True, fmt='d', cmap='Blues')
plt.title('Тепловая карта таблицы сопряженности')

# Stacked bar chart
contingency_table.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Столбчатая диаграмма с накоплением')
plt.xlabel('Категория 1')
plt.ylabel('Количество')
plt.legend(title='Категория 2', bbox_to_anchor=(1.05, 1), loc='upper left')

# Grouped bar chart
contingency_table.plot(kind='bar', figsize=(12, 6))
plt.title('Групповая столбчатая диаграмма')
plt.xlabel('Категория 1')
plt.ylabel('Количество')
plt.legend(title='Категория 2')
plt.xticks(rotation=45)
```

Создание многопанельных графиков:

```
# Комплексная визуализация в одном окне
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# Гистограмма
axes[0, 0].hist(df['price'], bins=30, alpha=0.7, edgecolor='black')
axes[0, 0].set_title('Распределение цен')
axes[0, 0].set_xlabel('Цена')
axes[0, 0].set_ylabel('Частота')

# Диаграмма рассеяния
axes[0, 1].scatter(df['area'], df['price'], alpha=0.6)
axes[0, 1].set_title('Цена vs Площадь')
axes[0, 1].set_xlabel('Площадь')
axes[0, 1].set_ylabel('Цена')

# Box plot
df.boxplot(column='price', by='category', ax=axes[1, 0])
axes[1, 0].set_title('Цены по категориям')
axes[1, 0].set_xlabel('Категория')

# Столбчатая диаграмма
df['category'].value_counts().plot(kind='bar', ax=axes[1, 1])
axes[1, 1].set_title('Частота категорий')
axes[1, 1].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```

4. Описание индивидуального задания

Выбрать с сайта kaggle.com набор данных в формате csv, загрузить и подготовить его к дальнейшей обработке. Наборы данных не должны повторяться внутри группы. Задание индивидуальное. Требования:

1. **Выбор и загрузка данных**
    - Выбрать набор данных с сайта kaggle.com в формате csv
    - Размер датасета должен быть не менее 1000 строк и содержать не менее 5 столбцов
    - Данные должны содержать как количественные, так и категориальные переменные
    - Зарегистрировать выбранный датасет у преподавателя во избежание дублирования
2. **Изучение структуры данных**
    - Загрузить данные с помощью pandas
    - Изучить размерность датасета (количество строк и столбцов)
    - Проанализировать типы данных каждого столбца
    - Изучить первые и последние строки данных
    - Получить общую информацию о датасете с помощью методов info() и describe()

3. **Предобработка данных**

    - Проверить наличие пропущенных значений и принять обоснованное решение по их обработке
    - Выявить и обработать дубликаты
    - Найти и проанализировать выбросы, принять решение о методах их обработки
    - При необходимости преобразовать типы данных (например, строки в даты)
    - Создать новые переменные на основе существующих (если это целесообразно)
    - Подготовить данные для анализа (кодирование категориальных переменных при необходимости)

4. **Формулирование и выполнение аналитических запросов**

    - Сформулировать на естественном языке не менее 10 исследовательских вопросов к данным
    - Примеры вопросов: "Какая категория товаров наиболее популярна?", "Есть ли связь между возрастом и доходом?", "Как изменяется показатель во времени?"
    - Реализовать каждый запрос с помощью методов pandas и numpy
    - Получить численные ответы на поставленные вопросы
    - Интерпретировать полученные результаты

5. **Создание визуализаций**
   Создать следующие типы графиков (согласно пункту 3.3):

   Одномерная визуализация:

    - Гистограммы для количественных переменных (минимум 2)
    - Box plot для количественных переменных (минимум 2)
    - Bar plot для категориальных переменных (минимум 2)

   Многомерная визуализация:

    - Correlation matrix для количественных переменных
    - Scatter plot для пар количественных переменных (минимум 2)
    - Contingency table с визуализацией для категориальных переменных

6. **Анализ и выводы**

    - Проанализировать каждую созданную визуализацию
    - Выявить основные закономерности в данных
    - Сформулировать выводы о структуре данных, распределениях переменных и взаимосвязях между ними
    - Определить наиболее интересные находки в данных