МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Московский государственный технический университет им. Н.Э. Баумана
(МГТУ им. Н.Э. Баумана)

Кафедра «Информационная безопасность» (ИУ8)

Лабораторная работа № 4

По дисциплине: «Машинное обучение»

Тема: «Предобработка данных»

Выполнил: Лагов С.П.,
Студент группы ИУ8-92

Проверила: Коннова Н.С.,
Преподаватель каф. ИУ8

г. Москва, 2024 г.

Практическая часть лабораторной работы 4

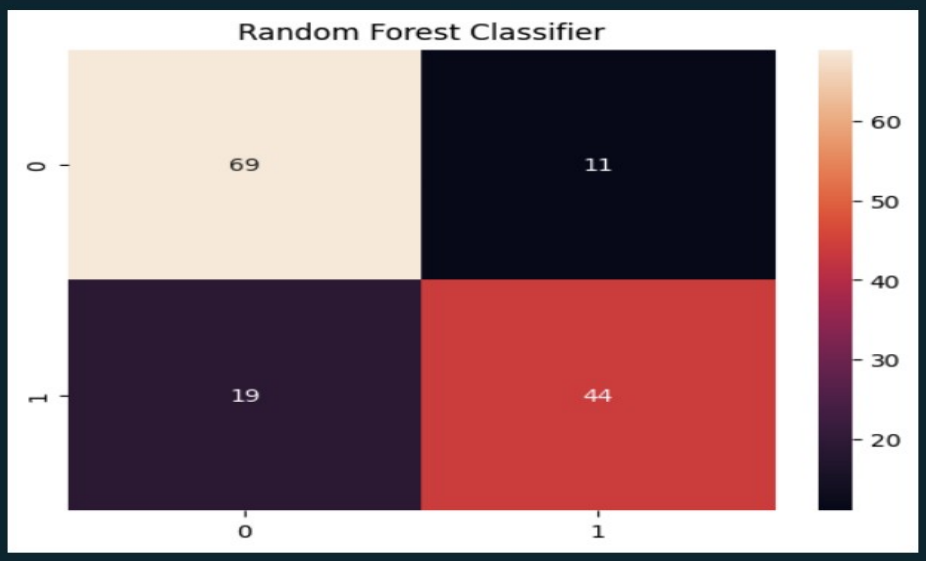
Цель работы: познакомиться с основными задачами и приемами предварительного анализа и обработки данных для целей машинного обучения.

Предварительная обработка данных является неотъемлемым этапом машинного обучения, поскольку качество данных и полезная информация, которую можно извлечь из них, напрямую влияют на способность нашей модели к обучению; поэтому чрезвычайно важно, чтобы мы предварительно обработали наши данные, прежде чем вводить их в нашу модель.

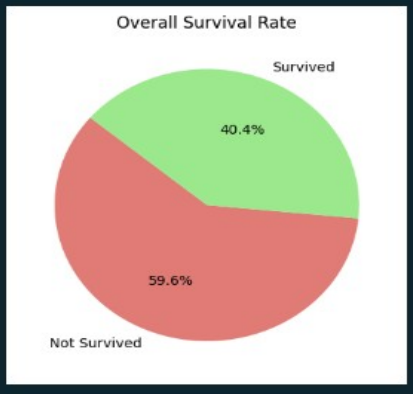
Ход работы

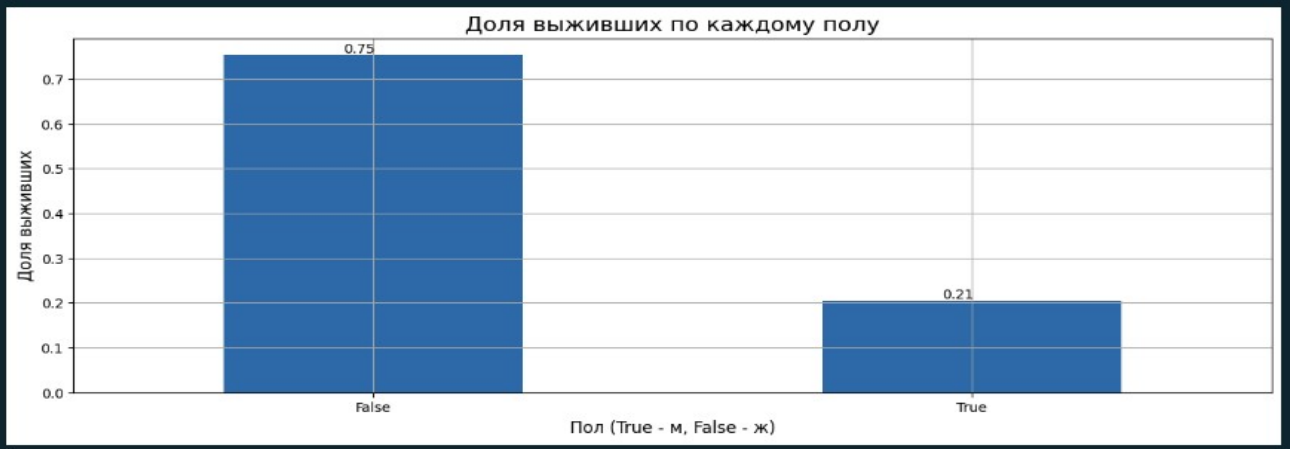
```
# 1 Постройте по получившемуся набору данных простую модель машинного обучения и оцените ее эффективность.

X = training_set[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'male', 'Q', 'S']]
y = training_set['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
perform_prediction(RandomForestClassifier(), X_train, y_train, X_test, y_test, 'Random Forest Classifier')
```

```
Score: 0.7902097902097902
```

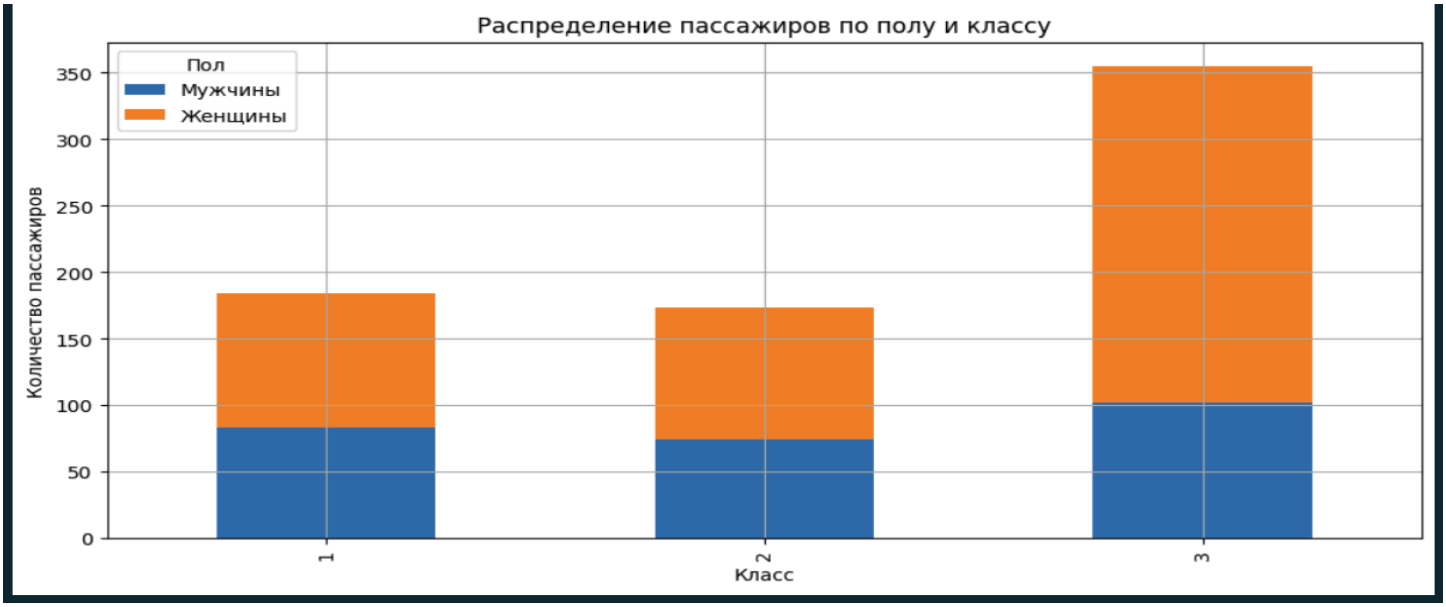


```python
# 2 Ответьте на следующие вопросы при помощи визуализации и численных данных по исходному набору данных:

# 3 Какова доля выживших после крушения пассажиров? Какова доля мужчин и женщин среди выживших?
survival_pivot = training_set.pivot_table(
    index='male',
    values='Survived',
    aggfunc=['mean', 'count']
)
survival_pivot.columns = ['Survival Rate', 'Count']

# Визуализация общей доли выживших (pie chart)
overall_survival = training_set.pivot_table(index='Survived', aggfunc='size')
survival_labels = ['Not Survived', 'Survived']

# Визуализация
plt.pie(
    overall_survival,
    labels=survival_labels,
    autopct='%1.1f%%',
    colors=['lightcoral', 'lightgreen'],
    startangle=140
)
plt.title('Overall Survival Rate')

column='male'
pivot = training_set.pivot_table(index=column, values='Survived', aggfunc='mean')

fig, ax = plt.subplots(figsize=(15,5))
ax.set_title('Доля выживших по каждому полу', fontdict={'size': 16})
ax.set_ylabel('Доля выживших', fontdict={'size': 12})
ax.set_xlabel(column, fontdict={'size': 12})

for cnt in range(pivot.shape[0]):
    value = pivot.iloc[cnt].values[0]
    ax.text(cnt - .05, value + .005, round(value, 2))

pivot.plot(kind='bar', rot=0, grid=True, legend=False, ax=ax)
ax.set_xlabel('Пол (True - м, False - ж)', fontdict={'size': 12})
plt.show()
```



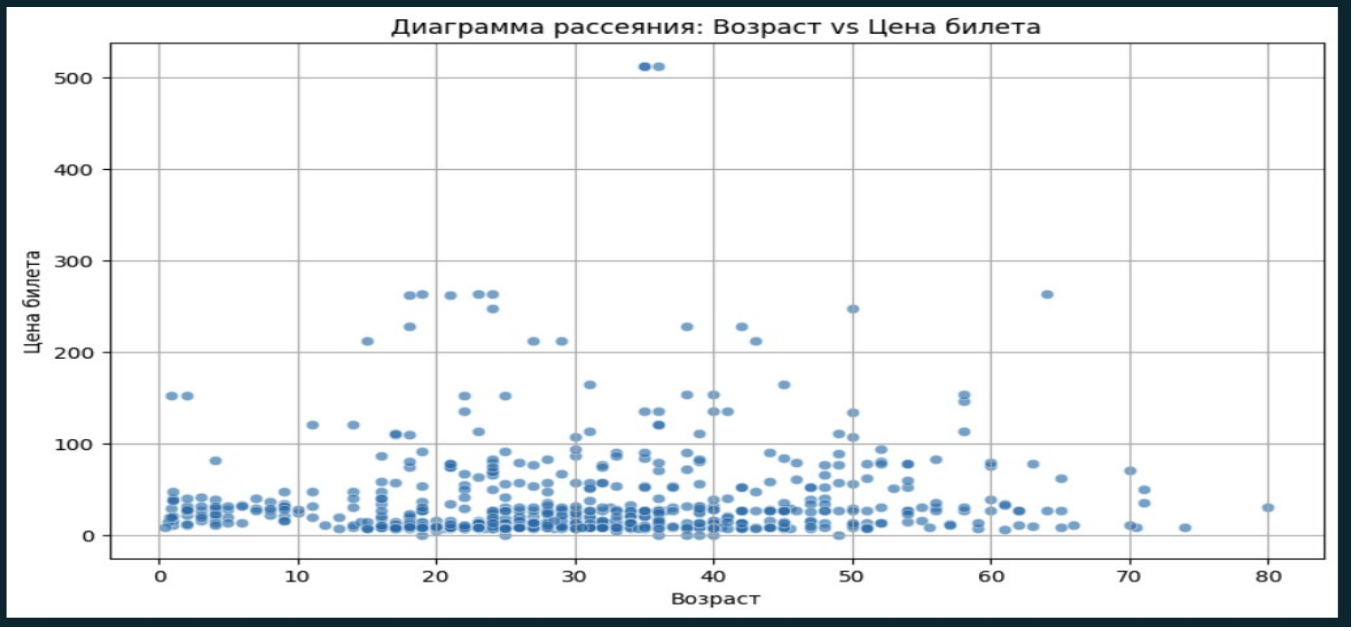


```python
# 4 Сколько пассажиров ехало в каждом классе? Кого было больше в самом многолюдном классе — мужчин или женщин?

class_counts = training_set['Pclass'].value_counts().sort_index()

class_gender_counts = training_set.groupby(['Pclass', 'male']).size().unstack()

fig, axs = plt.subplots(2, 1, figsize=(10, 10))
class_counts.plot(kind='bar', ax=axs[0])
axs[0].set_title('Количество пассажиров в каждом классе')
axs[0].set_xlabel('Класс')
axs[0].set_ylabel('Количество пассажиров')
axs[0].grid()

# График 2: Распределение пассажиров по полу и классу
class_gender_counts.plot(kind='bar', stacked=True, ax=axs[1])
axs[1].set_title('Распределение пассажиров по полу и классу')
axs[1].set_xlabel('Класс')
axs[1].set_ylabel('Количество пассажиров')
axs[1].legend(title='Пол', labels=['Мужчины', 'Женщины'])

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



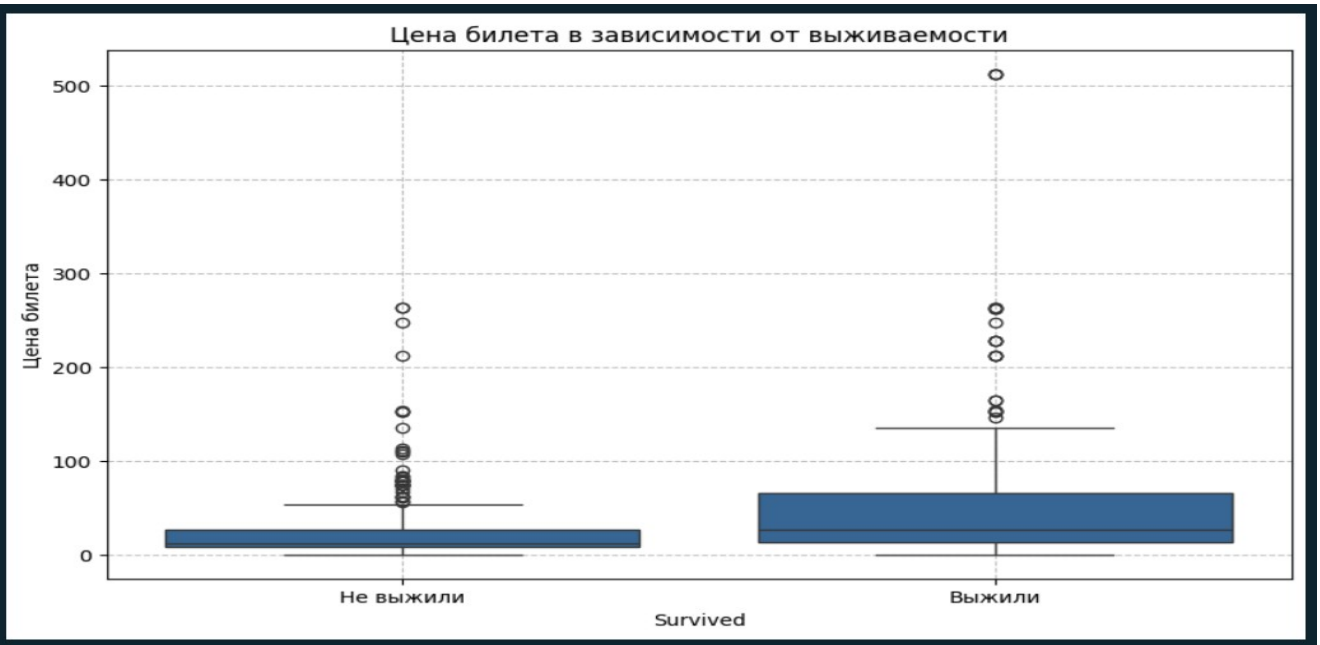
```python
# 6 Посчитайте, насколько сильно коррелируют друг с другом цена за билет и возраст пассажиров.
# Также проверьте наличие этой зависимости визуально (в этом вам поможет построение диаграммы рассеяния).

correlation = training_set['Age'].corr(training_set['Fare'])
print(f"Коэффициент корреляции между возрастом и ценой билета: {correlation:.4f}")

plt.figure(figsize=(10, 6))
sns.scatterplot(data=training_set, x='Age', y='Fare', alpha=0.6)
plt.title('Диаграмма рассеяния: Возраст vs Цена билета')
plt.xlabel('Возраст')
plt.ylabel('Цена билета')
plt.grid()
plt.show()
```

```
Коэффициент корреляции между возрастом и ценой билета: 0.0931
```

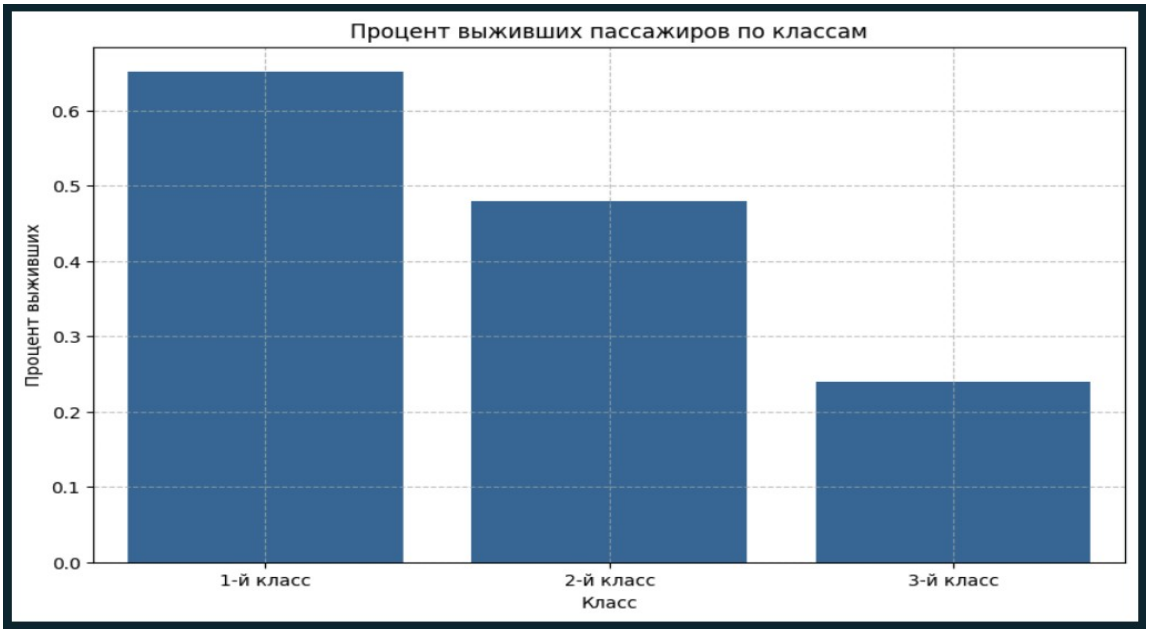



```python
# 7 Правда ли, что чаще выживали пассажиры с более дорогими билетами? А есть ли зависимость выживаемости от класса?
survival_fare_mean = training_set.groupby('Survived')['Fare'].mean()
print(f"Средняя цена билета для выживших: {survival_fare_mean[1]:.2f}")
print(f"Средняя цена билета для невыживших: {survival_fare_mean[0]:.2f}")

plt.figure(figsize=(10, 6))
sns.boxplot(data=training_set, x='Survived', y='Fare')
plt.title('Цена билета в зависимости от выживаемости')
plt.ylabel('Цена билета')
plt.xticks([0, 1], ['Не выжили', 'Выжили'])
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()

class_survival_rate = training_set.groupby('Pclass')['Survived'].mean()
print(f"Процент выживших пассажиров по классам:\n{class_survival_rate}")

plt.figure(figsize=(10, 6))
sns.barplot(x=class_survival_rate.index, y=class_survival_rate.values)
plt.title('Процент выживших пассажиров по классам')
plt.xlabel('Класс')
plt.ylabel('Процент выживших')
plt.xticks([0, 1, 2], ['1-й класс', '2-й класс', '3-й класс'])
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

```
Средняя цена билета для выживших: 51.65
Средняя цена билета для невыживших: 22.97
```

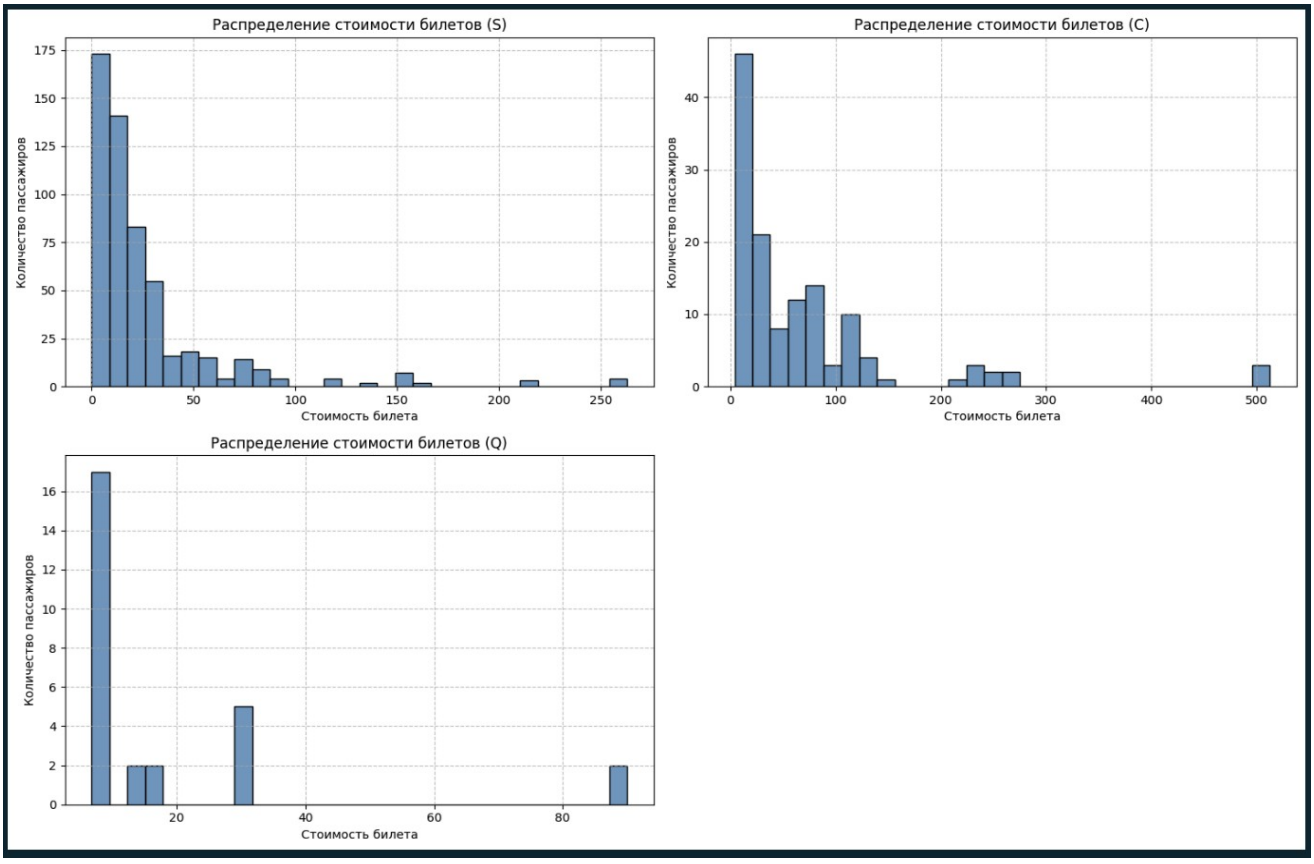



```
Процент выживших пассажиров по классам:
Pclass
1    0.652174
2    0.479769
3    0.239437
Name: Survived, dtype: float64
```



```python
# 8 Какова связь между стоимостью билета и портом отправления? Выведите минимальную, среднюю и максимальную сумму,
# которую заплатили пассажиры за проезд. Проделайте то же самое только для тех пассажиров, которые сели на
# корабль в Саутгемптоне.

training_set_cleaned = training_set[['Fare', 'Q', 'S']].dropna()

training_set_cleaned['embark_town'] = 'C'  # По умолчанию считаем, что порт — Cherbourg
training_set_cleaned.loc[training_set_cleaned['Q'], 'embark_town'] = 'Q'  # Порт Queenstown
training_set_cleaned.loc[training_set_cleaned['S'], 'embark_town'] = 'S'  # Порт Southampton

fare_stats_by_port = training_set_cleaned.groupby('embark_town')['Fare'].agg(['min', 'mean', 'max'])
print("Минимальная, средняя и максимальная стоимость билетов по портам отправления:")
print(fare_stats_by_port)

southampton_fares = training_set_cleaned[training_set_cleaned['embark_town'] == 'S']['Fare']
southampton_stats = southampton_fares.agg(['min', 'mean', 'max'])
print("\nМинимальная, средняя и максимальная стоимость билетов для пассажиров, севших в Саутгемптоне:")
print(southampton_stats)

ports = training_set_cleaned['embark_town'].unique()
plt.figure(figsize=(15, 10))

for i, port in enumerate(ports):
    plt.subplot(2, 2, i + 1)  # Делаем сетку 2x2
    port_data = training_set_cleaned[training_set_cleaned['embark_town'] == port]
    sns.histplot(port_data['Fare'], bins=30, kde=False, color='steelblue')
    plt.title(f'Распределение стоимости билетов ({port})')
    plt.xlabel('Стоимость билета')
    plt.ylabel('Количество пассажиров')
    plt.grid(True, linestyle='--', alpha=0.7)

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

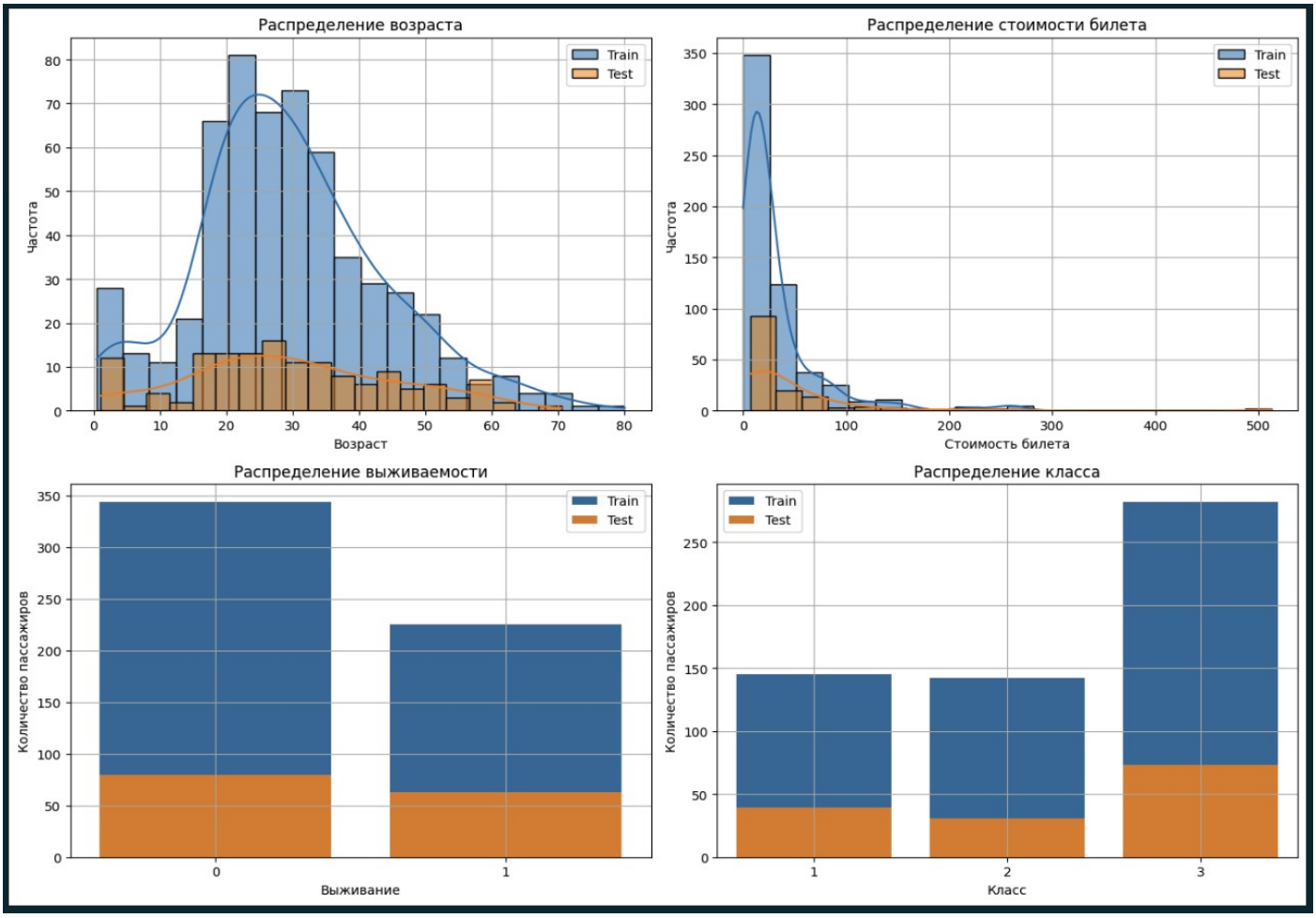
```
Минимальная, средняя и максимальная стоимость билетов по портам отправления:
                min        mean        max
embark_town
C            4.0125   68.296767   512.3292
Q            6.7500   18.265775    90.0000
S            0.0000   27.476284   263.0000

Минимальная, средняя и максимальная стоимость билетов для пассажиров, севших в Саутгемптоне:
min        0.000000
mean      27.476284
max      263.000000
Name: Fare, dtype: float64
```



```python
# 10 Оцените репрезентативность представленной выборки. Сколько всего было пассажиров Титаника? Сколько из них выжило?
# Какую долю составляет представленный набор данных от всей генеральной совокупности?

total_passengers = 887  # Общее количество пассажиров на Титанике

survived_count = training_set['Survived'].sum()

sample_size = len(training_set)
sample_fraction = sample_size / total_passengers

print(f"Общее количество пассажиров на Титанике: {total_passengers}")
print(f"Количество выживших пассажиров: {survived_count}")
print(f"Доля представленной выборки от всей генеральной совокупности: {sample_fraction:.2f}")

print(f"Общее количество пассажиров в наборе данных: {sample_size}")
```

```
Общее количество пассажиров на Титанике: 887
Количество выживших пассажиров: 288
Доля представленной выборки от всей генеральной совокупности: 0.80
Общее количество пассажиров в наборе данных: 712
```

```python
# 11 Разделите выборку на тестовую и обучающую части при помощи train_test_split(). Изобразите на графиках распре
# атрибутов и целевой переменной. Насколько однородно получившееся разбиение?

X = training_set[['Age', 'Fare', 'Pclass']]
y = training_set['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

fig, axs = plt.subplots(2, 2, figsize=(14, 10))

sns.histplot(X_train['Age'], kde=True, label='Train', ax=axs[0, 0], bins=20)
sns.histplot(X_test['Age'], kde=True, label='Test', ax=axs[0, 0], bins=20)
axs[0, 0].set_title('Распределение возраста')
axs[0, 0].set_xlabel('Возраст')
axs[0, 0].set_ylabel('Частота')
axs[0, 0].grid()
axs[0, 0].legend()

sns.histplot(X_train['Fare'], kde=True, label='Train', ax=axs[0, 1], bins=20)
sns.histplot(X_test['Fare'], kde=True, label='Test', ax=axs[0, 1], bins=20)
axs[0, 1].set_title('Распределение стоимости билета')
axs[0, 1].set_xlabel('Стоимость билета')
axs[0, 1].set_ylabel('Частота')
axs[0, 1].grid()
axs[0, 1].legend()

sns.countplot(x=y_train, ax=axs[1, 0], label='Train')
sns.countplot(x=y_test, ax=axs[1, 0], label='Test')
axs[1, 0].set_title('Распределение выживаемости')
axs[1, 0].set_xlabel('Выживание')
axs[1, 0].set_ylabel('Количество пассажиров')
axs[1, 0].grid()
axs[1, 0].legend()

sns.countplot(x=X_train['Pclass'], ax=axs[1, 1], label='Train')
sns.countplot(x=X_test['Pclass'],  ax=axs[1, 1], label='Test')
axs[1, 1].set_title('Распределение класса')
axs[1, 1].set_xlabel('Класс')
axs[1, 1].set_ylabel('Количество пассажиров')
axs[1, 1].grid()
axs[1, 1].legend()

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




```python
# 12 Сбалансируйте классы в исходном датасете двумя способами:
# 13 Удалите лишние объекты мажоритарного класса (выбранные случайно)

training_set_balanced_1 = training_set.copy()

majority_class = training_set_balanced_1['Survived'].value_counts().idxmax()
majority_class_df = training_set_balanced_1[training_set_balanced_1['Survived'] == majority_class]
majority_class_indices = majority_class_df.index
majority_class_count = len(majority_class_indices)

to_remove = majority_class_count - (len(training_set_balanced_1) - majority_class_count)

remove_indices = majority_class_df.sample(to_remove).index
training_set_balanced_1.drop(remove_indices, inplace=True)

training_set_balanced_1['Survived'].value_counts()
```

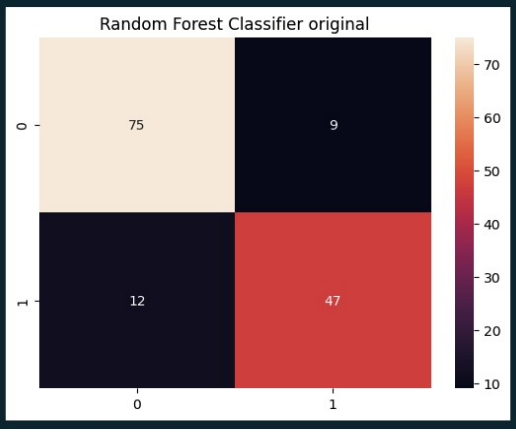
```
Survived
0    288
1    288
Name: count, dtype: int64
```

```python
# 14 Добавьте в выборку дубликаты миноритарного класса

training_set_balanced_2 = training_set.copy()

majority_class = training_set_balanced_2[training_set_balanced_2["Survived"] == 0]
minority_class = training_set_balanced_2[training_set_balanced_2["Survived"] == 1]

minority_upsampled = resample(
    minority_class,
    replace=True,  # Разрешить дублирование
    n_samples=len(majority_class)
)

training_set_balanced_2 = pd.concat([majority_class, minority_upsampled])
training_set_balanced_2['Survived'].value_counts()
```

```
Survived
0    424
1    424
Name: count, dtype: int64
```

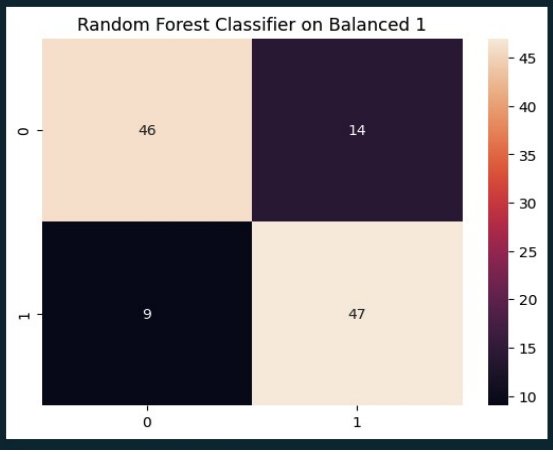
```python
# 15 Проведите исследование эффективности простой модели классификации до и после данных преобразований.
X = training_set[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'male', 'Q', 'S']]
y = training_set['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
perform_prediction(RandomForestClassifier(), X_train, y_train, X_test, y_test, 'Random Forest Classifier original')
```

```
Score: 0.8531468531468531
```



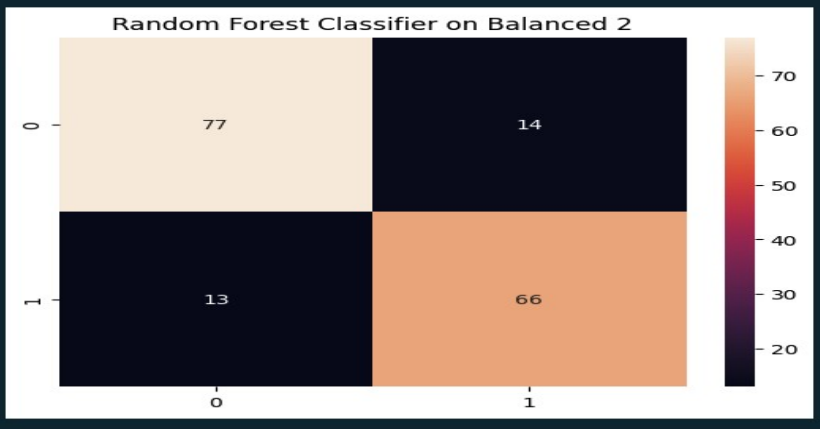

```python
X = training_set_balanced_1[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'male', 'Q', 'S']]
y = training_set_balanced_1['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
perform_prediction(RandomForestClassifier(), X_train, y_train, X_test, y_test, 'Random Forest Classifier on Balanced 1')
```

```
Score: 0.8017241379310345
```



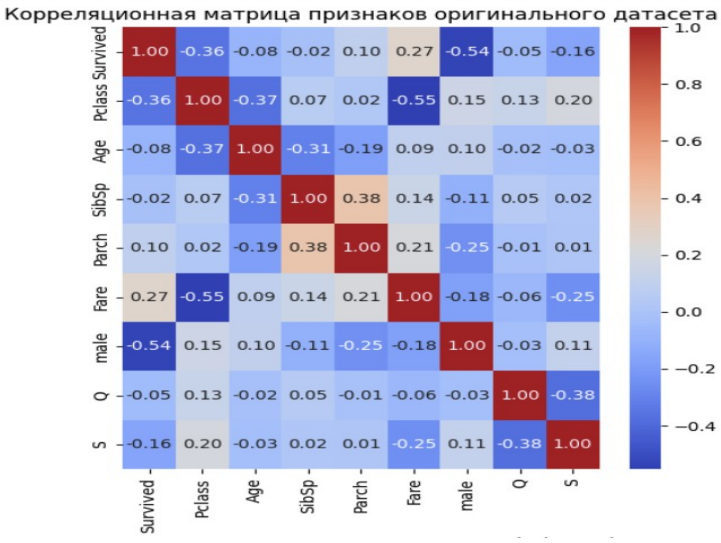
```
X = training_set_balanced_2[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'male', 'Q', 'S']]
y = training_set_balanced_2['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
perform_prediction(RandomForestClassifier(), X_train, y_train, X_test, y_test, 'Random Forest Classifier on Balanced 2')
```

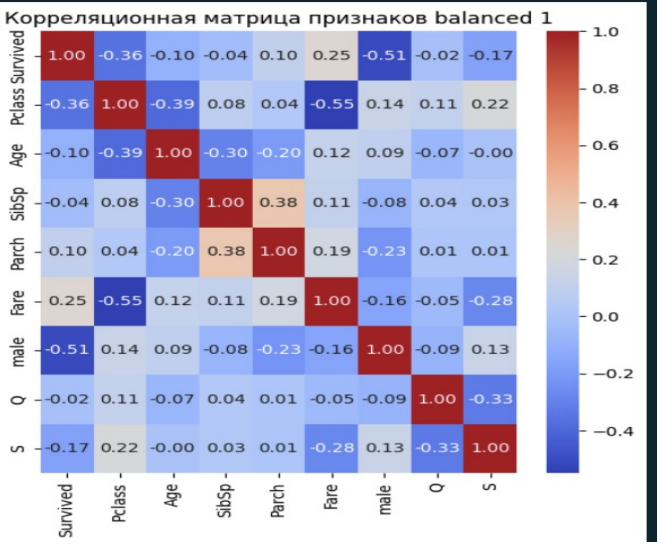
```
Score: 0.8411764705882353
```

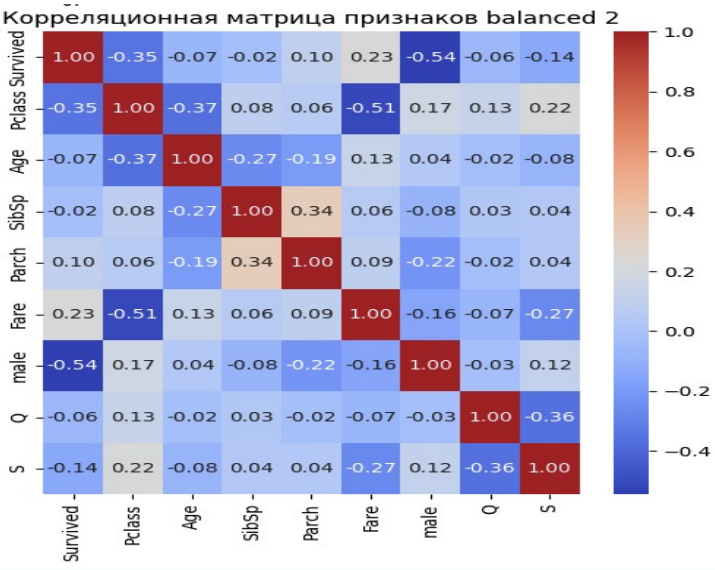



```
# 16 Постройте корреляционную матрицу признаков после преобразования данных.
# Сделайте вывод о наличии либо отсутствии мультиколлинеарности признаков.

plt.figure(figsize=(13, 14))
plt.subplot(2, 2, 1)
sns.heatmap(training_set.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Корреляционная матрица признаков оригинального датасета")

plt.subplot(2, 2, 2)
sns.heatmap(training_set_balanced_1.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Корреляционная матрица признаков balanced 1")

plt.subplot(2, 2, 3)
sns.heatmap(training_set_balanced_2.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Корреляционная матрица признаков balanced 2")

plt.show()
```

Python

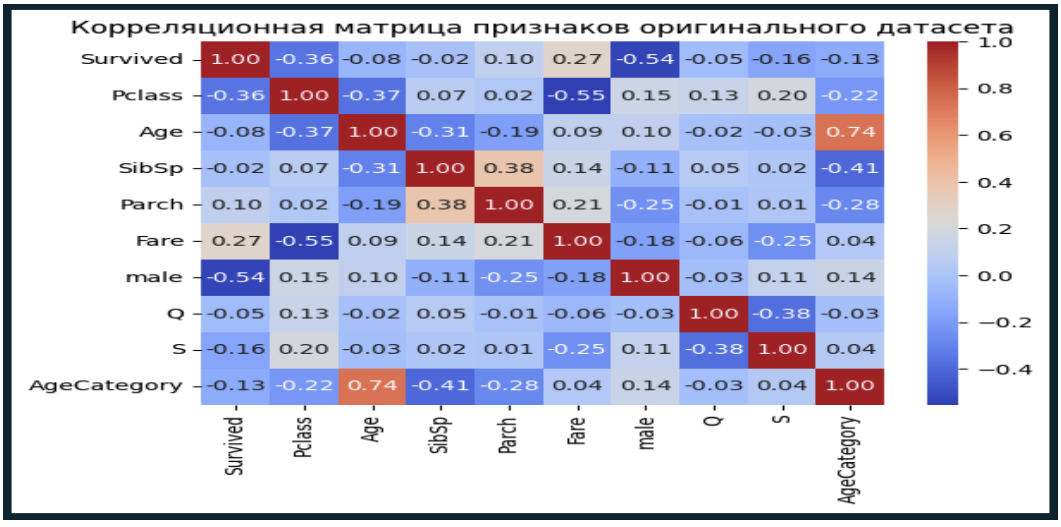







```python
# 17 Проведите группировку данных по значению возраста. Введите новый признак "возрастная категория", значениями которой будут
# "ребенок", "взрослый", "старик". Проведите анализ эффективности данного признака.

def categorize_age(age):
    if age < 18:
        return 0 # ребенок
    elif age < 60:
        return 1 # взрослый
    else:
        return 2 # старик

training_set["AgeCategory"] = training_set["Age"].apply(categorize_age)

survival_analysis = training_set.groupby("AgeCategory")["Survived"].mean().reset_index()
survival_analysis.rename(columns={"Survived": "SurvivalRate"}, inplace=True)

print("Анализ выживаемости по возрастным категориям:")
print(survival_analysis)

sns.heatmap(training_set.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Корреляционная матрица признаков оригинального датасета")
```

```
Анализ выживаемости по возрастным категориям:
   AgeCategory  SurvivalRate
0            0      0.539823
1            1      0.385017
2            2      0.240000
```



Вывод

В ходе анализа датасета Titanic было выявлено несколько ключевых закономерностей, связанных с вероятностью выживания пассажиров.

Женщины имели значительно больше шансов на выживание по сравнению с мужчинами, что объясняется приоритетной эвакуацией женщин и детей на шлюпки. Также было замечено, что пассажиры с более дорогими билетами имели выше шансы на выживание, вероятно, из-за лучших условий обслуживания и преимуществ при эвакуации. Кроме того, существовала связь с портом отправления Саутгемптон: пассажиры, отправлявшиеся из этого порта, обладали немного более высокой вероятностью выживания по сравнению с другими портами, такими как Куинстаун и Йорк.

Контрольные вопросы

1. Какие основные виды визуализации вы знаете? Какие у них области применения?

- Гистограммы (Histograms). Позволяют анализировать распределение переменных и частоты значений. Применяются для изучения распределения числовых данных и выявления выбросов
- Диаграммы разброса (Scatter Plots). Используются для выявления зависимости между двумя переменными и определения их корреляции
- Линейные графики (Line Charts). Применяются для анализа изменений данных во времени, например, тренды и динамику во временных рядах
- Круговые диаграммы (Pie Charts). Наиболее удобны для отображения пропорций или долей данных. Часто используются в задачах, где нужно показать процентное соотношение категорий
- Столбчатые диаграммы (Bar Charts). Позволяют сравнивать категории данных по величине. Применяются для сравнения значений между группами данных

- Ящико-боксы (Box Plots). Отображают распределение данных, медиану, межквартильный диапазон и выбросы. Применяются для статистического анализа и проверки гипотез
- Тепловые карты (Heatmaps). Используются для отображения матричных данных и выявления закономерностей между переменными. Например, в анализе корреляции или визуализации таблиц данных

2. Какие типы визуализации больше всего подходят для анализа совместного распределения двух непрерывных переменных?

- Scatter Plot показывает корреляцию и тренды между переменными
- Density Plots демонстрируют вероятность распределения данных
- Contour Plots визуализируют зоны высокой плотности данных

3. Какие типы визуализации больше всего подходят для анализа совместного распределения двух дискретных переменных?

- Столбчатые диаграммы сравнивают частоты категорий переменных
- Кросс-таблицы (Cross-tabulations): отображают частотное распределение двух переменных
- Heatmaps визуализируют частоты пар категорий с помощью цветовой шкалы
- Диаграммы точек показывают распределение пар значений и их частоты

4. Как лучше всего построить совместное распределение дискретной и непрерывной переменной?

- Box Plot позволяет показать распределение непрерывной переменной для каждой категории дискретной переменной
- Bar + Density Plot позволяют сравнить средние значения и распределения
- График точек (Violin Plot) комбинирует элементы ящика с усами и плотности данных, показывая распределение непрерывной переменной по группам дискретной переменной

- Круговые диаграммы и разреженные плотности визуализируют группы дискретных категорий и их соответствующие распределения

5. Как лучше всего построить совместное распределение двух непрерывных и одной дискретной переменной?

- Графики “Violin Plot” с разбивкой по категориям позволяют сравнить распределение двух непрерывных переменных по группам, определяемым дискретной переменной
- Параллельные координаты (Parallel Coordinates) позволяют визуализировать многомерные зависимости между переменными

6. Как лучше всего построить совместное распределение двух дискретных и одной непрерывной переменной?

- Box Plot позволяет показать распределение непрерывной переменной по каждой комбинации категорий двух дискретных переменных
- Графики с наложением плотности (Density / Violin Plots) позволяют визуализировать, как непрерывная переменная распределяется по группам, определяемым двумя дискретными переменными
- Столбчатые диаграммы с наложением позволяют визуально сравнить распределения непрерывной переменной между группами двух категориальных переменных