САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1
по курсу «Современные инструменты анализа данных»
Тема: «Уменьшение размерности данных и сравнение методов»

Выполнила:
Скоринцева Т.А
К3260

Проверила:

Санкт-Петербург
2025 г.

Содержание отчета

Введение с описанием целей и задач работы

Цель работы: Освоить практические навыки работы с методами уменьшения размерности данных, такими как PCA и t-SNE, а также интерпретации их результатов.

Задачи работы: сбор и предобработка данных, интерпретация и описание собранных данных, применение к данным PCA и t-SNE, визуализация и анализ результатов, анализ используемых методов уменьшения размерности данных и выводы по задаче.

Описание исходных данных и процесса их предобработки

Мною было выбрано реализовывать данный проект в Google Collab для простоты визуализации и структурирования блоков задач.

Исходные данные:

Были загружены данные о жанрах музыки (датасет)

```
%%capture

!pip install opendatasets

import opendatasets as od

import pandas as pd

import warnings

warnings.filterwarnings("ignore")

dataset_url = 'https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre'

od.download(dataset_url)

df = pd.read_csv('./prediction-of-music-genre/music_genre.csv')

display(df.head())

print(df.shape)
```

Исходно 50000 наблюдений (строк) и 17 признаков (столбцов)

Категориальные признаки: *instance_id* (ID экземпляра), *artist_name* (имя исполнителя), *track_name* (название трека), *key* (тональность), *mode* (лад), *obtained_date* (дата получения из API), *music_genre* (музыкальный жанр – целевая переменная, которую хотим предсказать).

Числовые признаки: *popularity* (Популярность), *acousticness* (Акустичность), *danceability* (Танцевальность), *duration_ms* (Длительность трека в миллисекундах), *energy* (Мера интенсивности и активности), *instrumentalness* (Инструментальность), *liveness* (Вероятность, что трек был записан при live-выступлении), *loudness* (Общая

громкость трека в децибелах (дБ)), *speechiness* (Наличие в треке произнесенных слов), *tempo* (Темп трека в ударах в минуту), *valence* (Позитивность трека).

Предобработка данных:

Заменяем нечисловые данные на числовые:

- key (тональность): так как тональности имеют музыкальную иерархию, применим для записи тональностей порядковое кодирование;
- mode (лад): преобразуем в 1 и 0;
- music_genre (целевая переменная): присвоим каждому жанру уникальное число;
- artist_name и track_name: слишком много уникальных значений, они не несут полезной информации и загрузят модель, поэтому удаляем их;
- obtained_date: дата получения данных также не влияет на предсказание.

```
df_processed = df.copy()

cols_to_drop = ['instance_id', 'artist_name', 'track_name', 'obtained_date']

df_processed = df_processed.drop(columns=cols_to_drop)



TONALITIES = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']

tonality_to_number = {tonality: idx for idx, tonality in enumerate(TONALITIES)}

df_processed['key_encoded'] = df_processed['key'].map(tonality_to_number)

df_processed = df_processed.drop(columns=['key'])



df_processed['mode_encoded'] = df_processed['mode'].map({'Major': 1, 'Minor': 0})

df_processed = df_processed.drop(columns=['mode'])



from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df_processed['music_genre_encoded'] = le.fit_transform(df_processed['music_genre']) # все
уникальные жанры закодировали в числа

genre_classes = le.classes_

df_processed = df_processed.drop(columns=['music_genre'])
```

Создаём шкалу соответствия:
Чтобы правильно обратно интерпретировать результаты модели, со отнесем закодированные категориальные признаки с названиями.

Все переменные имеют ненулевую дисперсию, ничего не удаляем.

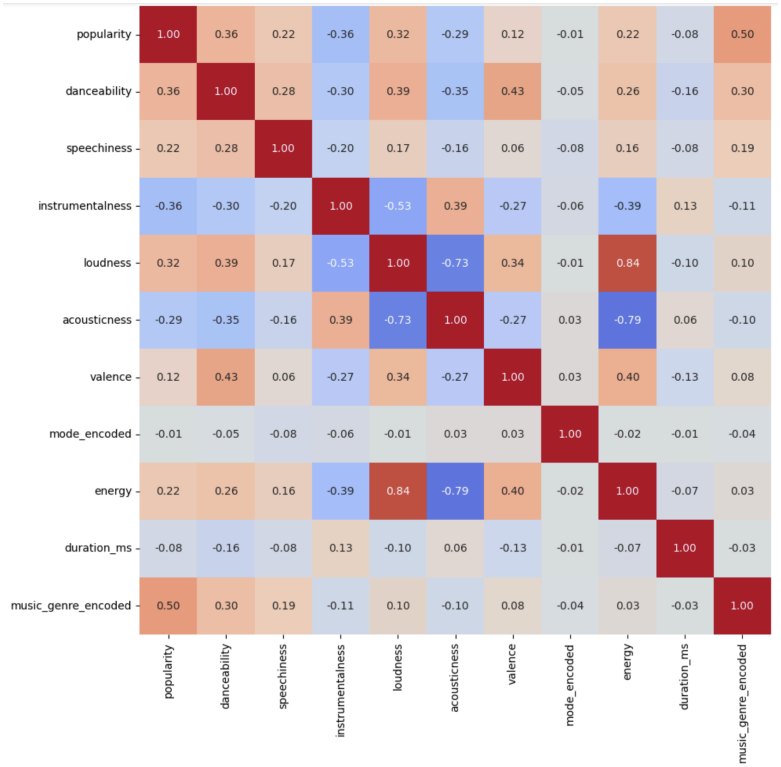
Оставляем 10 – 12 важных параметров (через подсчет корреляции с целевой переменной):

```
correlation = df_processed.corr()['music_genre_encoded'].abs().sort_values(ascending=False)

correlation = correlation.drop('music_genre_encoded')
print(correlation)

top_10_features = correlation.head(10).index
df_processed = df_processed[list(top_10_features) + ['music_genre_encoded']]
```

Увидели какие признаки оказывают наибольшее влияние на предсказание музыкального жанра:

```
popularity          0.502133
danceability        0.303463
speechiness         0.190536
instrumentalness    0.110432
loudness            0.102520
acousticness        0.097969
valence             0.075065
mode_encoded        0.039832
energy              0.034738
duration_ms         0.028517
Name: music_genre_encoded, dtype: float64
```

Построили корреляционную матрицу и дали интерпретацию выявленным зависимостям: если два признака сильно коррелируют, то один из них можно удалить для упрощения модели.



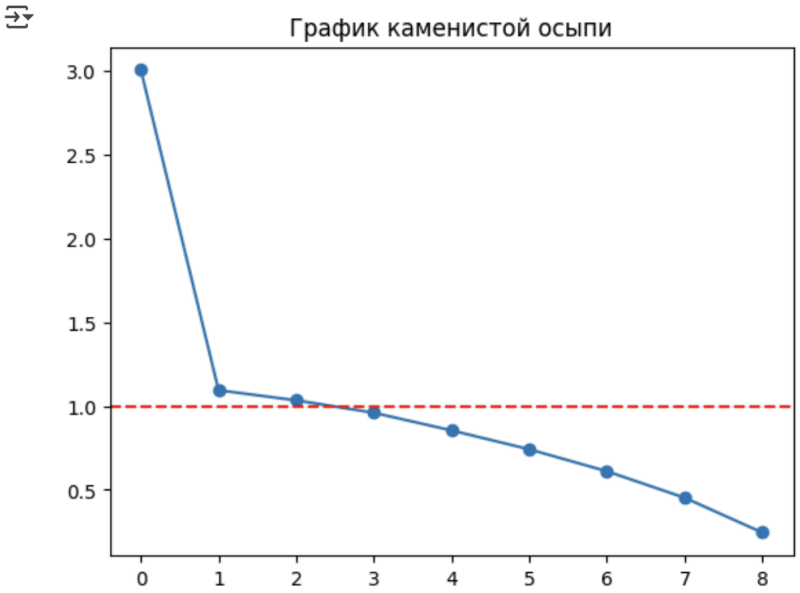

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 10))
corr_matrix = df_processed.corr()
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', center=0, square=True)
```

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

Вывод из матрицы корреляции: признак loudness и energy имеют высокий коэффициент корреляции, что логично предположить, тк громкая музыка вероятнее всего энергичная.

Построим график каменистой осыпи:

```
eigenvalues = np.linalg.eigvals(df_processed.drop(columns=['music_genre_encoded', 'energy']).corr())

plt.plot(np.sort(eigenvalues)[::-1], 'o-')

plt.axhline(y=1, color='red', linestyle='--')

plt.title('График каменистой осыпи')

plt.show()
```



По “методу локтя” наблюдаем резкий спад после 3-х компонент.

Описание реализации и результатов применения PCA

Реализуем алгоритм PCA

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import numpy as np

X = df_processed.drop(columns=['music_genre_encoded', 'energy'])
imputer = SimpleImputer(strategy='median')
X_imputed = imputer.fit_transform(X)
X_imputed = pd.DataFrame(X_imputed, columns=X.columns, index=X.index)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)
```

```
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)
```

Вычислим объясненную дисперсию для каждой компоненты

```
for i, variance in enumerate(pca.explained_variance_ratio_, 1):

   print(f"Компонента {i}: {variance:.4f} ({variance*100:.2f}%)")
```

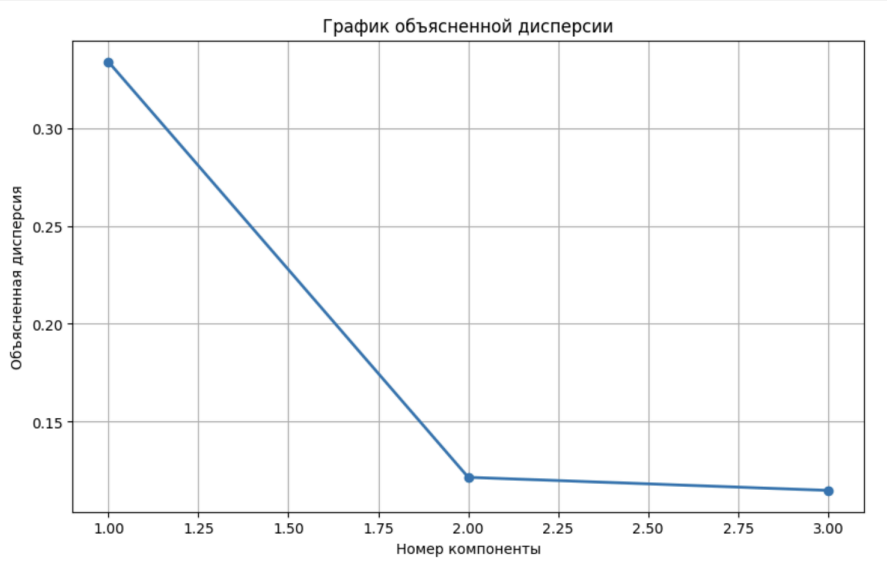
```
Объясненная дисперсия для каждой компоненты:
Компонента 1: 0.3338 (33.38%)
Компонента 2: 0.1216 (12.16%)
Компонента 3: 0.1149 (11.49%)
```

Вычислить процент объясненной дисперсии для первых n компонент

```
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
print(f"\nНакопленная объясненная дисперсия:")
print(f"Первая компонента: {cumulative_variance[0]:.4f} ({cumulative_variance[0]*100:.2f}%)")
print(f"Первые две компоненты: {cumulative_variance[1]:.4f} ({cumulative_variance[1]*100:.2f}%)")
print(f"Все три компоненты: {cumulative_variance[2]:.4f} ({cumulative_variance[2]*100:.2f}%)")
```

```
Накопленная объясненная дисперсия:
Первая компонента: 0.3338 (33.38%)
Первые две компоненты: 0.4554 (45.54%)
Все три компоненты: 0.5704 (57.04%)
```

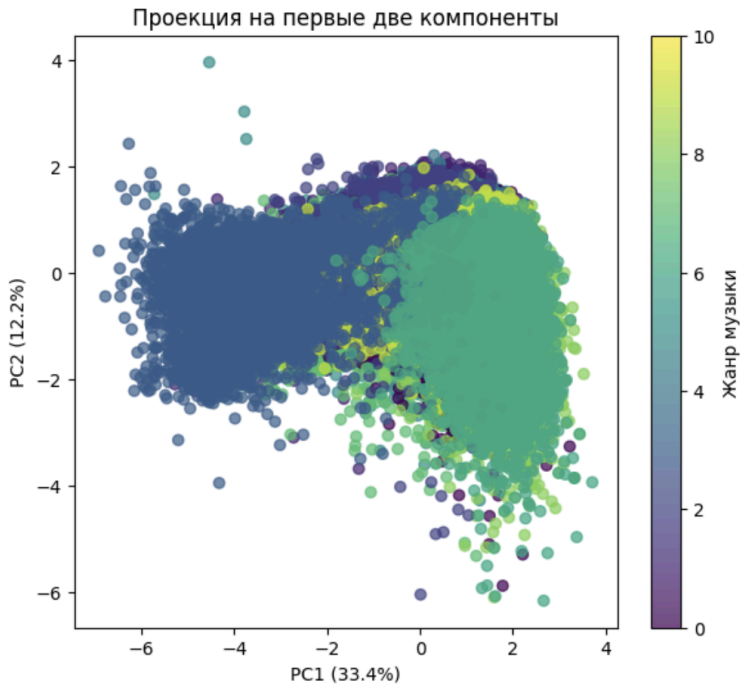
График объясненной дисперсии

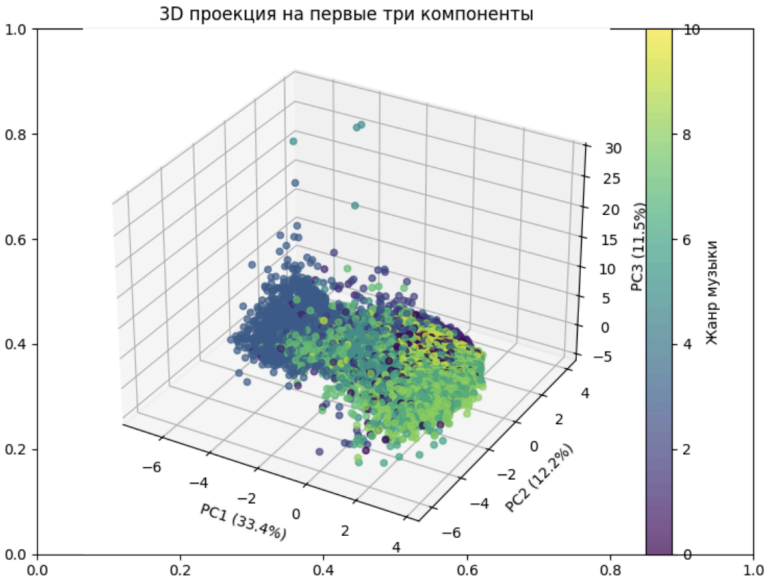


Интерпретация результатов:

Первая компонента захватывает значительную часть дисперсии (36%). Три компоненты объясняют почти 60% информации.

Построим проекцию данных на новое пространство:





Интерпретация: хорошая разделимость музыкальных жанров на выбранных компонентах.

Описание реализации и результатов применения t-SNE:

Реализация алгоритма t-SNE:

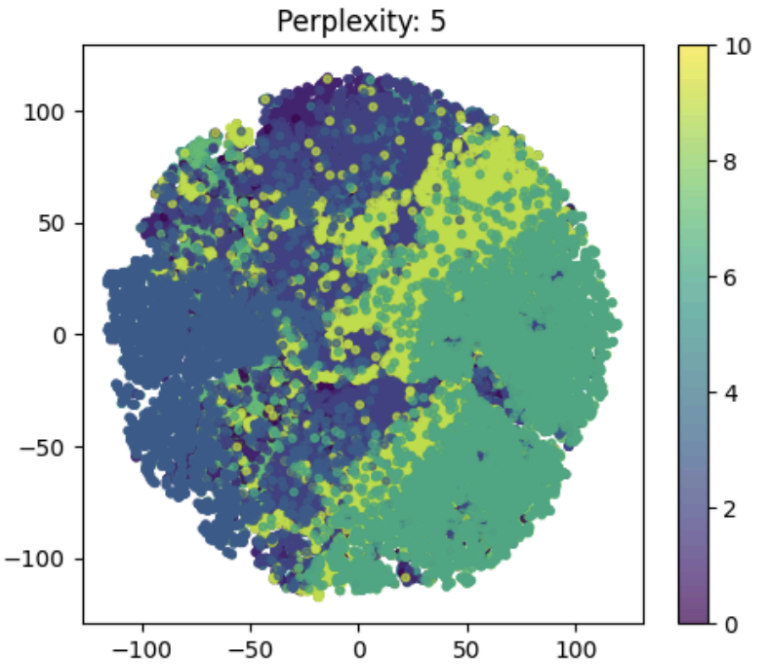
```
X = df_processed.drop(columns=['music_genre_encoded', 'energy'])

imputer = SimpleImputer(strategy='median')

X_imputed = imputer.fit_transform(X)

X_imputed = pd.DataFrame(X_imputed, columns=X.columns, index=X.index)
```

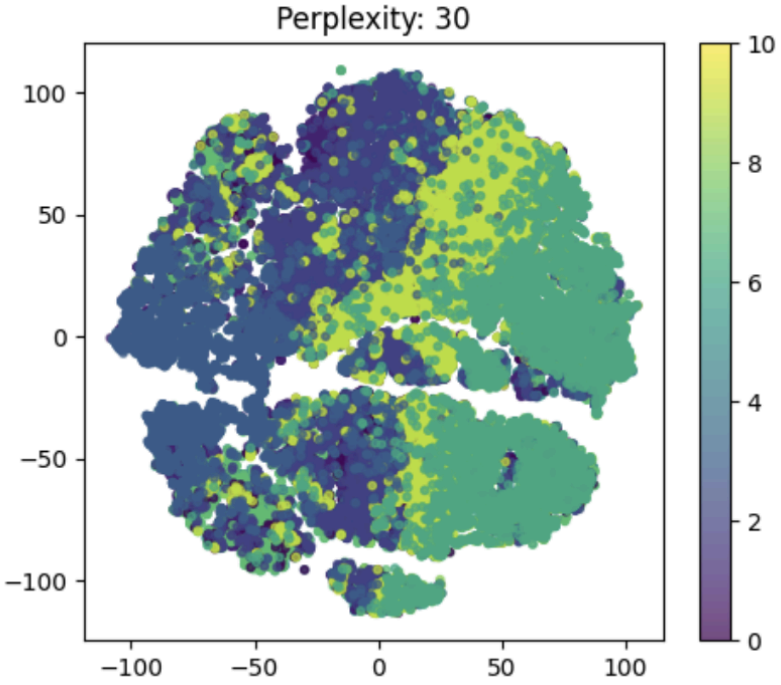
```
scaler = StandardScaler()

X_scaled = scaler.fit_transform(X_imputed)

y = df_processed['music_genre_encoded']

perplexities = [5, 30, 50, 100]

fig, axes = plt.subplots(2, 2, figsize=(12, 10))

for i, perplexity in enumerate(perplexities):

    tsne = TSNE(n_components=2, perplexity=perplexity, random_state=42)

    X_tsne = tsne.fit_transform(X_scaled)

      scatter = axes[i//2, i%2].scatter(X_tsne[:, 0], X_tsne[:, 1], c=y, cmap='viridis',
alpha=0.7, s=10)

    axes[i//2, i%2].set_title(f'Perplexity: {perplexity}')

    plt.colorbar(scatter, ax=axes[i//2, i%2])

plt.tight_layout()

plt.show()
```

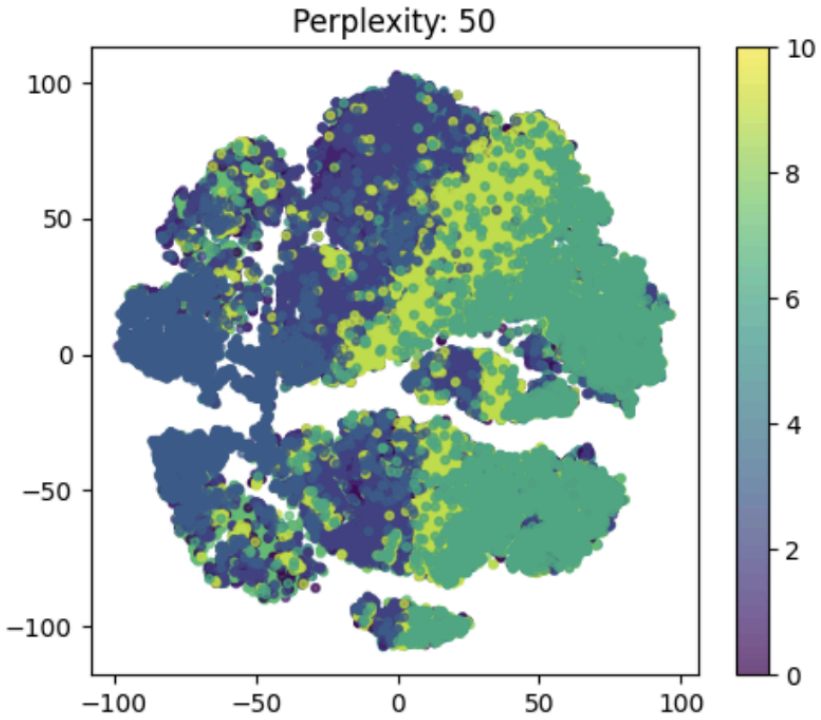
Применение t-SNE с разными значениями перплексии:

Поскольку у нас большой набор данных, для более наглядного формирования кластеров лучше выбрать большую перплексию: 30 и 50. Я также построила для маленькой перплексии, чтобы увидеть разницу при её увеличении. Чем больше параметр, тем нагляднее наши данные разделены на кластеры.

Визуализация результатов:







Вывод: отдельные скопления точек одного цвета показывают, что выбранные 10 признаков хорошо описывают соответствующие музыкальные жанры. А увеличение перплексии помогает наглядно разделить данные на кластеры.

Сравнительный анализ PCA и t-SNE:

- Преимущества и недостатки каждого метода

PCA преимущества: работает быстро при запуске, лучше для понижения размерности для обучения модели, поскольку стабильно сохраняет глобальную структуру, не искажая расстояния, проще в понимании. Алгоритм линеен, детерминирован.

PCA недостатки: чувствителен к выбросам, не улавливает нелинейные признаки, направление максимальной дисперсии может быть перпендикулярно направлению, которое лучше всего разделяет классы, поэтому качество модели может ухудшиться, так как был удален критичный для разделения признак.

t-SNE преимущества: с большей точностью визуализирует разделимость на кластеры, раскрывает нелинейные зависимости признаков, работает больше на локальные связи между признаками, нежели на глобальные.

t-SNE недостатки: так как t-SNE искажает расстояния, то при большой перплексии данные представляются слишком упрощённо и их тяжело интепретировать. Для этого нужно анализировать результаты на разных значениях перплексии и выбирать наилучший для корректной интерпретации кластеров. Также данный алгоритм работает сильно дольше при запуске. Категорически не подходит для задачи подготовки данных для обучения модели и интерпретации расстояний.

- Рекомендации по использованию методов для различных типов задач:

Для визуализации многомерных данных и обучения моделей: PCA

Для лучшего понимания структуры данных и визуального анализа: t-SNE

Вывод по работе:

В данной работе были исследованы и сравнены такие алгоритмы для уменьшения размерности данных как PCA (линейный) и t-SNE(нелинейный). С помощью визуализации результатов мы получили необходимые сведения для глубокого сравнительного анализа алгоритмов и анализа зависимостей целевой переменной (музыкального жанра) и заданных в задаче признаков. Теперь мы можем применять полученные знания и интерпретировать результаты в более прикладных задачах.