Передбачення популярності по характеристикам пісні

Імпорти

In [503…
```
import pandas as pd
import numpy as np

from IPython.display import HTML

import seaborn as sns
import matplotlib.pyplot as plt

import warnings
```

In [504…
```
warnings.filterwarnings("ignore")
```

In [505…
```
df = pd.read_csv("data/spotify_songs_dataset.csv")
```

In [506…
```
df.head()
```

Out[506…

	song_id	song_title	artist	album	genre	release_date	duration	populari
0	SP0001	Space executive series.	Sydney Clark	What.	Electronic	1997-11-08	282.0	
1	SP0002	Price last painting.	Connor Peters DDS	Nature politics.	Electronic	2015-05-10	127.0	
2	SP0003	Piece.	Anna Keith	Visit.	Pop	2024-07-08	NaN	
3	SP0004	Power industry your.	Zachary Simpson	Behavior evening.	Hip-Hop	2022-08-15	214.0	
4	SP0005	Food animal second.	Christopher Mcgee	Front.	Pop	2023-03-05	273.0	

In [507…
```
df.shape
```

Out[507… `(7000, 15)`

In [508…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7000 entries, 0 to 6999
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   song_id           7000 non-null   object
 1   song_title        7000 non-null   object
 2   artist            7000 non-null   object
 3   album             7000 non-null   object
 4   genre             7000 non-null   object
 5   release_date      7000 non-null   object
 6   duration          6326 non-null   float64
 7   popularity        7000 non-null   int64
 8   stream            7000 non-null   int64
 9   language          6630 non-null   object
 10  explicit_content  7000 non-null   object
 11  label             7000 non-null   object
 12  composer          7000 non-null   object
 13  producer          7000 non-null   object
 14  collaboration     2125 non-null   object
dtypes: float64(1), int64(2), object(12)
memory usage: 820.4+ KB
```

Descriptive Statistics

In [509…

```
print("\nCategorical Variables Summary:")
df.describe(include="O")
```

```
Categorical Variables Summary:
```

Out[509…

	song_id	song_title	artist	album	genre	release_date	language	explicit
count	7000	7000	7000	7000	7000	7000	6630	
unique	7000	6854	6672	4520	9	5207	7	
top	SP7000	Explain.	Christopher Johnson	Star.	Pop	2004-03-10	English	
freq	1	4	7	10	1778	5	4650	

In [510…

```
print("\nNumerical Variables Summary:")
df.describe(exclude="O")
```

```
Numerical Variables Summary:
```

Out[510…

	duration	popularity	stream
count	6326.000000	7000.000000	7.000000e+03
mean	239.684951	50.583143	5.041078e+07
std	50.076064	28.905272	2.891778e+07
min	45.000000	1.000000	1.978800e+04
25%	206.000000	25.000000	2.576459e+07
50%	240.000000	51.000000	5.056644e+07
75%	273.000000	76.000000	7.549213e+07
max	428.000000	100.000000	9.999386e+07

In [511…
```
df.isnull().sum()
```

Out[511…
```
song_id              0
song_title           0
artist               0
album                0
genre                0
release_date         0
duration           674
popularity           0
stream               0
language           370
explicit_content     0
label                0
composer             0
producer             0
collaboration     4875
dtype: int64
```

Dropping the id

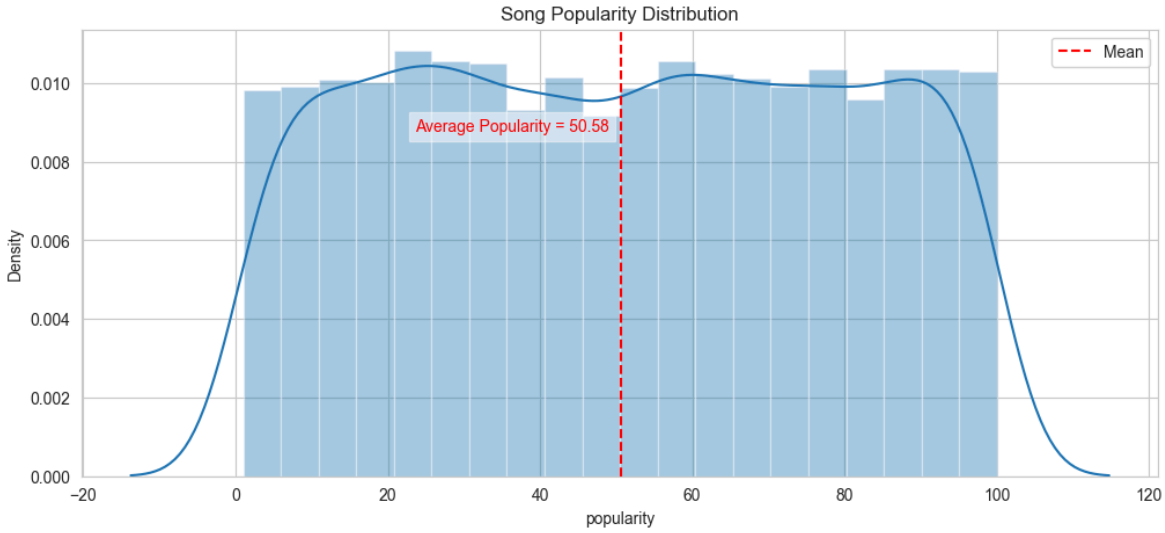
In [512…
```
df.drop(columns=['song_id'], inplace=True)
```

Data visualization

1. Song Popularity Distribution

In [513…
```
sns.set_style("whitegrid")
plt.figure(figsize=(12, 5))
plotd = sns.distplot(df["popularity"], kde=True, bins=20)
mean_line = plt.axvline(
    df["popularity"].mean(), c="red", linestyle="dashed", label="Mean"
)
plt.text(
    0.49,
    0.8,
    f'Average Popularity = {df["popularity"].mean():.2f}',
    transform=plt.gca().transAxes,
    color="red",
    fontsize=10,
```

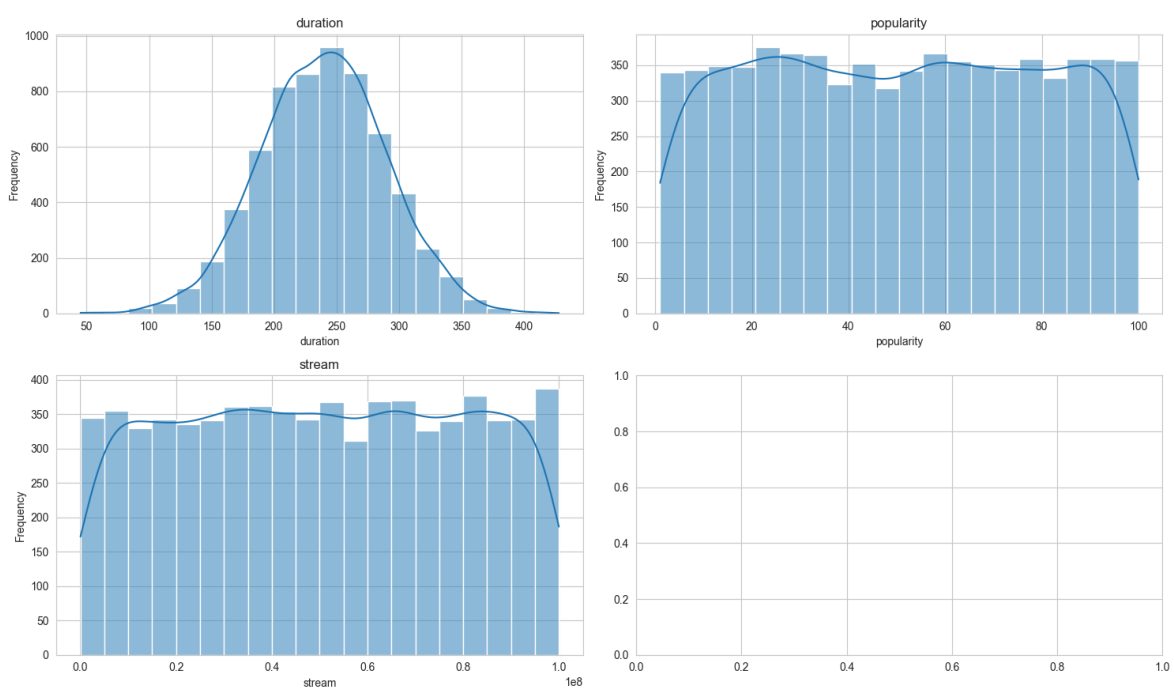
```
    verticalalignment="top",
    horizontalalignment="right",
    bbox=dict(facecolor="white", alpha=0.5),
)
plt.legend()
plt.title("Song Popularity Distribution")
plt.show()
```



2. Numerical Features Distribution

In [514…

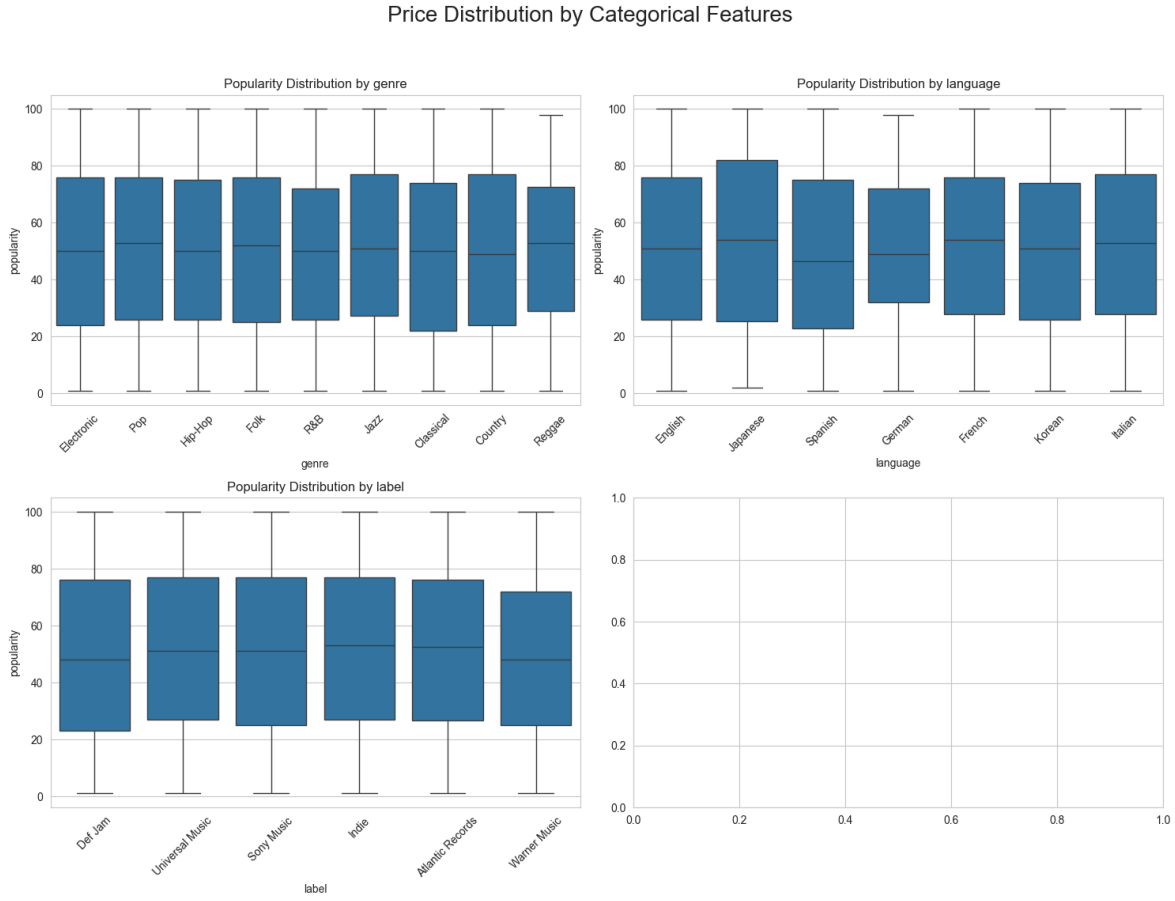
```
numerical_cols = [
    "duration",
    "popularity",
    "stream"
]
num_subplots = len(numerical_cols)
num_rows = (num_subplots - 1) // 2 + 1
num_cols = min(2, num_subplots)

fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, num_rows * 5))
fig.suptitle("Histograms of Numerical Features", size=20)

for idx, col in enumerate(numerical_cols):
    i, j = idx // num_cols, idx % num_cols
    sns.histplot(df[col], ax=axes[i, j], kde=True, bins=20)
    axes[i, j].set_title(col)
    axes[i, j].set_xlabel(col)
    axes[i, j].set_ylabel("Frequency")

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



3. Categorical Features vs Price

In [515…

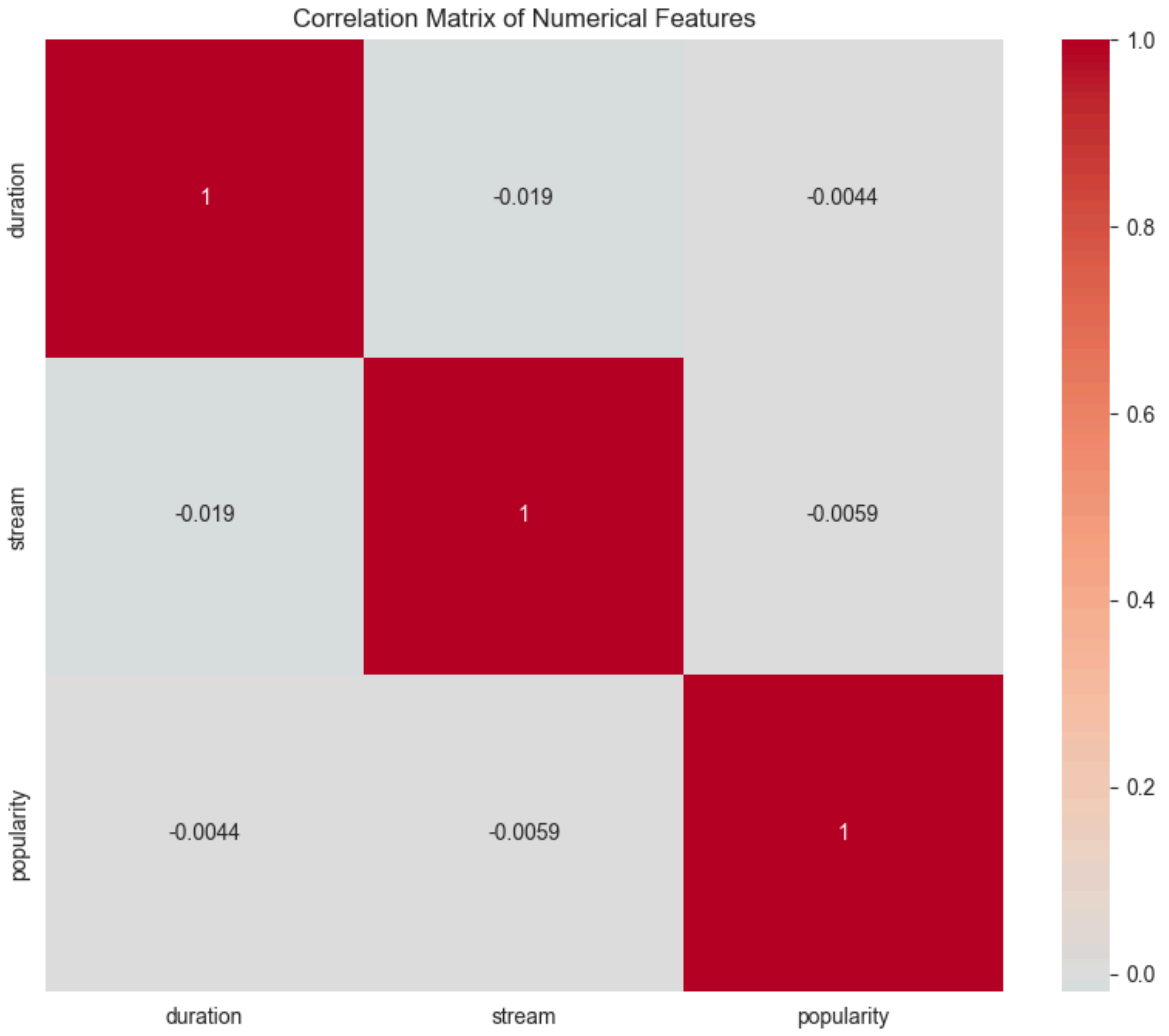
```
categorical_cols = ["genre", "language", "label"]
# , "release_date", "language", "explicit_content", "label", "composer", "produc
num_subplots = len(categorical_cols)
num_rows = (num_subplots - 1) // 2 + 1
num_cols = min(2, num_subplots)

fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, num_rows * 6))
fig.suptitle("Price Distribution by Categorical Features", size=20)

for idx, col in enumerate(categorical_cols):
    i, j = idx // num_cols, idx % num_cols
    sns.boxplot(x=col, y="popularity", data=df, ax=axes[i, j])
    axes[i, j].set_xticklabels(axes[i, j].get_xticklabels(), rotation=45)
    axes[i, j].set_title(f"Popularity Distribution by {col}")

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



4. Correlation Analysis for Numerical Features

In [516…

```
numerical_cols = [
    "duration",
    "stream",
    "popularity"
]
correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", center=0)
plt.title("Correlation Matrix of Numerical Features")
plt.show()
```

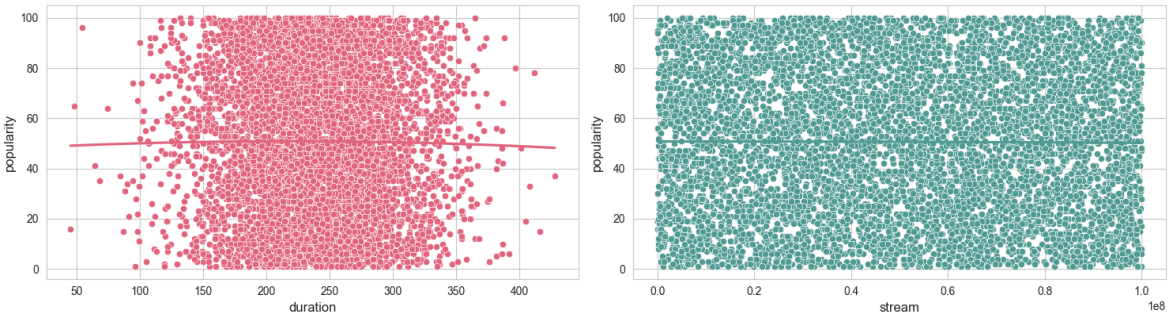


In [517…

```
# Вибір числових колонок (без цільової змінної)
numerical_cols = df.select_dtypes(include=["float64", "int64"]).columns

# Виключаємо цільову змінну та ідентифікатори, якщо вони є
selected_cols = [col for col in numerical_cols if col not in ["popularity"]]

# Визначаємо кількість графіків
num_subplots = len(selected_cols)
num_rows = (num_subplots - 1) // 2 + 1
num_cols = min(2, num_subplots)

# Побудова матриці графіків
fig, axes = plt.subplots(
    num_rows, num_cols, figsize=(15, num_rows * 5), facecolor="white"
)
fig.suptitle(
    "Scatter Plots of Numerical Features vs Popularity with Polynomial Lines", s
)

# Перетворимо axes на 2D масив, навіть якщо subplot один
if num_rows * num_cols == 1:
    axes = np.array([axes])
elif num_rows == 1 or num_cols == 1:
    axes = np.reshape(axes, (num_rows, num_cols))

# Палітра кольорів
palette = sns.husl_palette(n_colors=len(selected_cols), s=0.7, l=0.6)
```

```
# Побудова окремих графіків
for i in range(num_rows):
    for j in range(num_cols):
        idx = i * num_cols + j
        if idx < num_subplots:
            sns.scatterplot(
                x=selected_cols[idx],
                y="popularity",
                data=df,
                ax=axes[i, j],
                color=palette[idx],
            )

            sns.regplot(
                x=selected_cols[idx],
                y="popularity",
                data=df,
                ax=axes[i, j],
                scatter=False,
                order=2,
                color=palette[idx],
                ci=None,
            )

            axes[i, j].set_xlabel(selected_cols[idx], fontsize=12)
            axes[i, j].set_ylabel("popularity", fontsize=12)

# Оформлення
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Scatter Plots of Numerical Features vs Popularity with Polynomial Lines



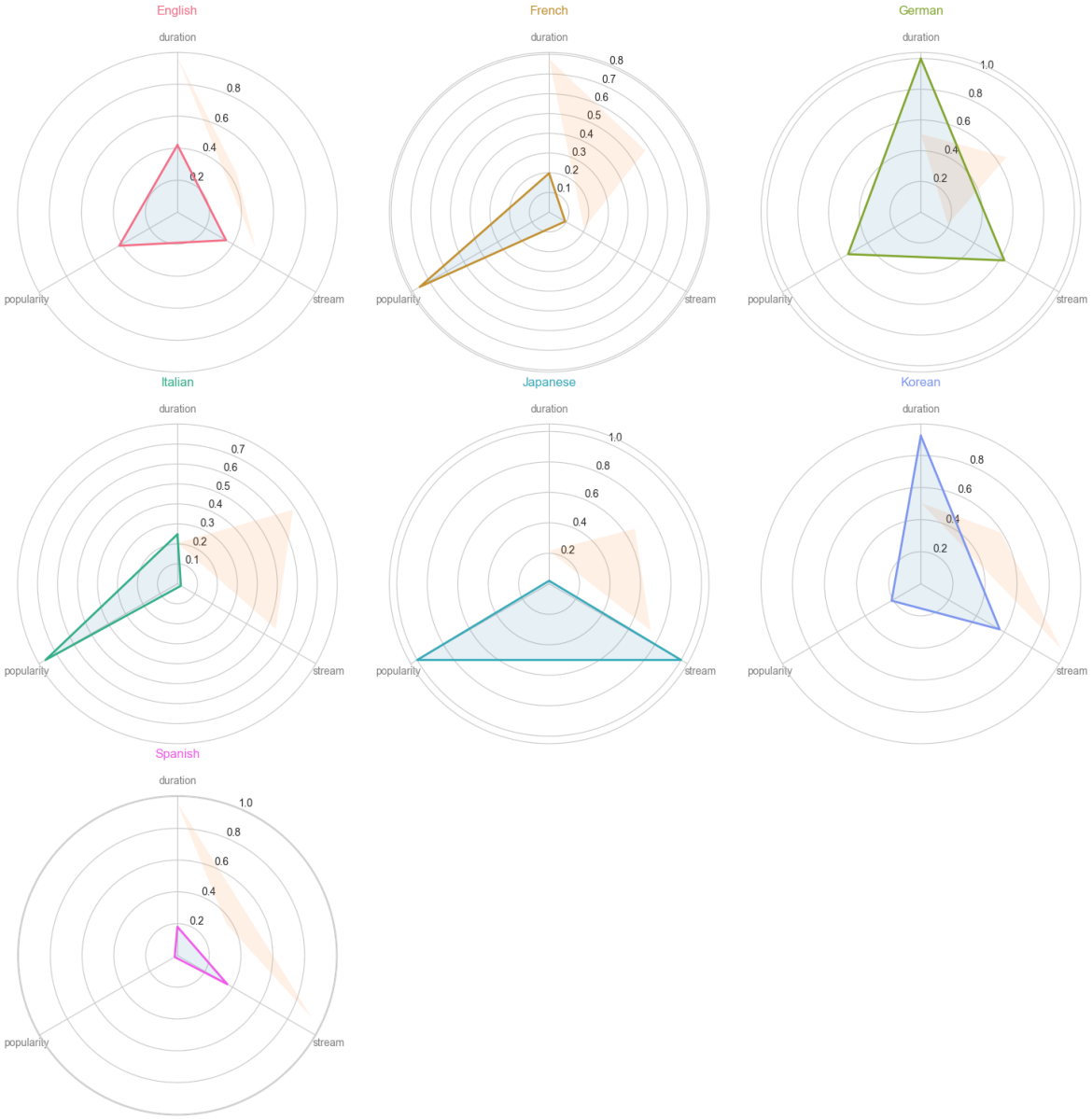
Радіальні графіки характеристик для кожної мови

In [518…

```
from math import pi
from sklearn.preprocessing import MinMaxScaler

# Обчислення середніх значень характеристик для кожної компанії
company_skills = (
    df.groupby("language")[
        ["duration", "stream", "popularity"]
    ]
    .mean()
    .reset_index()
)

# Нормалізація даних
scaler = MinMaxScaler((0.02, 1))
```

```
df_normalized = pd.DataFrame(
    scaler.fit_transform(company_skills.iloc[:, 1:]), columns=company_skills.col
)
df_normalized["language"] = company_skills["language"]
df_normalized = df_normalized[
    [
        "language",
        "duration",
        "stream",
        "popularity"
    ]
]

# Підготовка даних для побудови графіка
categories = df_normalized.columns.tolist()[1:]
N = len(categories)
colors = sns.color_palette(
    "husl", len(df_normalized)
)  # Динамічне визначення кольорів для всіх компаній

# Налаштування кількості рядків і стовпців для графіків
num_subplots = len(df_normalized)
num_rows = (num_subplots - 1) // 3 + 1
num_cols = min(3, num_subplots)

# Побудова радіальних графіків
fig, axes = plt.subplots(
    figsize=(15, num_rows * 5),
    nrows=num_rows,
    ncols=num_cols,
    subplot_kw=dict(polar=True),
)
axes = axes.flatten()  # Перетворюємо матрицю осей у список

for ax, (idx, row), color in zip(axes, df_normalized.iterrows(), colors):
    values = row.drop("language").values.flatten().tolist()
    values += values[:1]  # Замикаємо радіус
    angles = [n / float(N) * 2 * pi for n in range(N)]
    angles += angles[:1]  # Замикаємо кут

    ax.set_theta_offset(pi / 2)
    ax.set_theta_direction(-1)

    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(categories, color="grey", size=10)

    ax.plot(
        angles,
        values,
        linewidth=2,
        linestyle="solid",
        label=row["language"],
        color=color,
    )
    ax.fill(angles, values, color, alpha=0.1)

    ax.set_title(row["language"], size=12, color=color, y=1.1)

# Видаляємо зайві осі, якщо компаній менше, ніж кількість осей
for ax in axes[len(df_normalized) :]:
```

```
    fig.delaxes(ax)

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



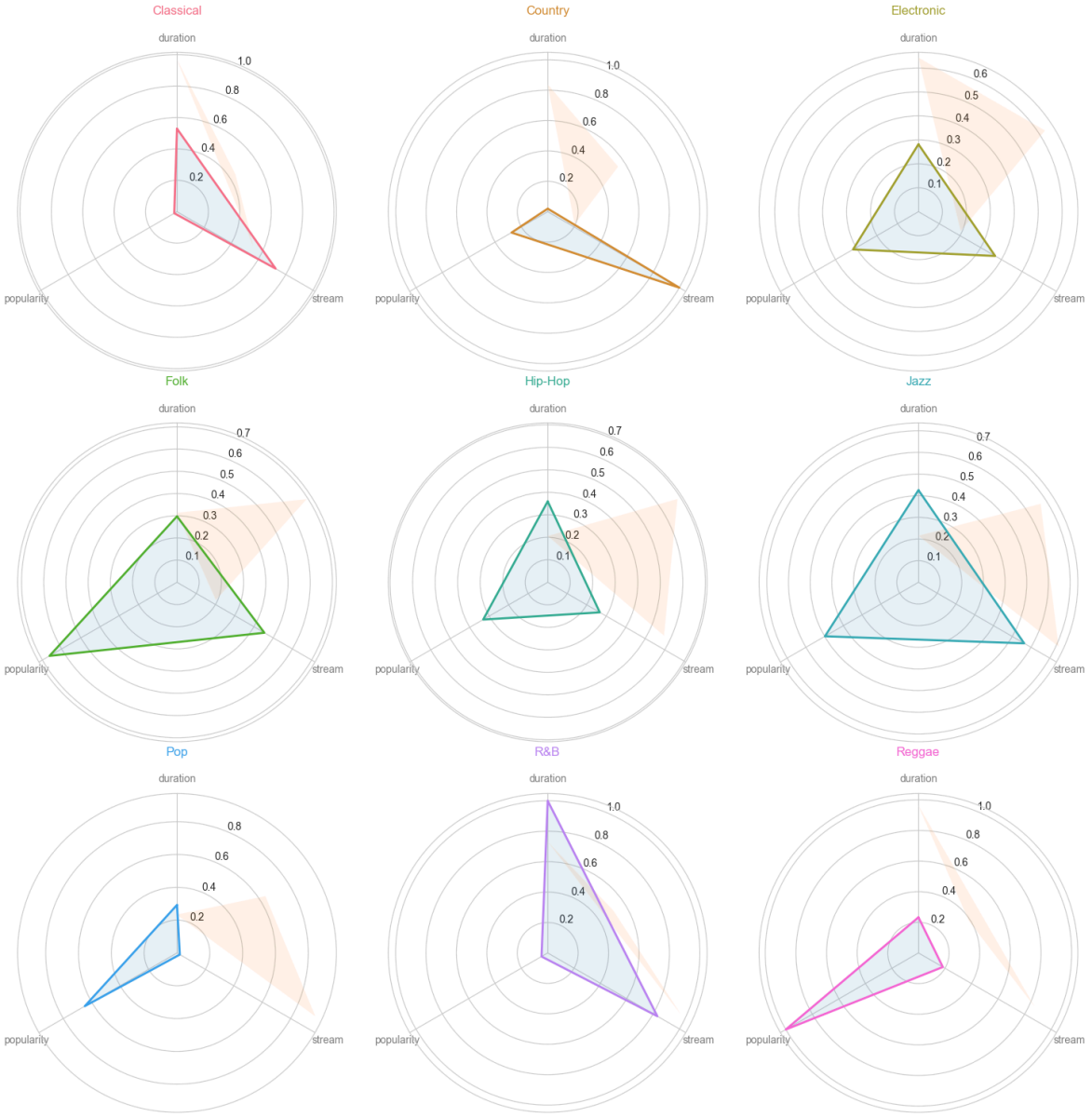
Радіальні графіки характеристик для кожного жанру

In [519…

```
from math import pi
from sklearn.preprocessing import MinMaxScaler

# Обчислення середніх значень характеристик для кожної компанії
company_skills = (
    df.groupby("genre")[
        ["duration", "stream", "popularity"]
    ]
    .mean()
    .reset_index()
)

# Нормалізація даних
scaler = MinMaxScaler((0.02, 1))
df_normalized = pd.DataFrame(
    scaler.fit_transform(company_skills.iloc[:, 1:]), columns=company_skills.col
```

```
)
df_normalized["genre"] = company_skills["genre"]
df_normalized = df_normalized[
    [
        "genre",
        "duration",
        "stream",
        "popularity"
    ]
]

# Підготовка даних для побудови графіка
categories = df_normalized.columns.tolist()[1:]
N = len(categories)
colors = sns.color_palette(
    "husl", len(df_normalized)
)  # Динамічне визначення кольорів для всіх компаній

# Налаштування кількості рядків і стовпців для графіків
num_subplots = len(df_normalized)
num_rows = (num_subplots - 1) // 3 + 1
num_cols = min(3, num_subplots)

# Побудова радіальних графіків
fig, axes = plt.subplots(
    figsize=(15, num_rows * 5),
    nrows=num_rows,
    ncols=num_cols,
    subplot_kw=dict(polar=True),
)
axes = axes.flatten()  # Перетворюємо матрицю осей у список

for ax, (idx, row), color in zip(axes, df_normalized.iterrows(), colors):
    values = row.drop("genre").values.flatten().tolist()
    values += values[:1]  # Замикаємо радіус
    angles = [n / float(N) * 2 * pi for n in range(N)]
    angles += angles[:1]  # Замикаємо кут

    ax.set_theta_offset(pi / 2)
    ax.set_theta_direction(-1)

    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(categories, color="grey", size=10)

    ax.plot(
        angles,
        values,
        linewidth=2,
        linestyle="solid",
        label=row["genre"],
        color=color,
    )
    ax.fill(angles, values, color, alpha=0.1)

    ax.set_title(row["genre"], size=12, color=color, y=1.1)

# Видаляємо зайві осі, якщо компаній менше, ніж кількість осей
for ax in axes[len(df_normalized) :]:
    fig.delaxes(ax)
```

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

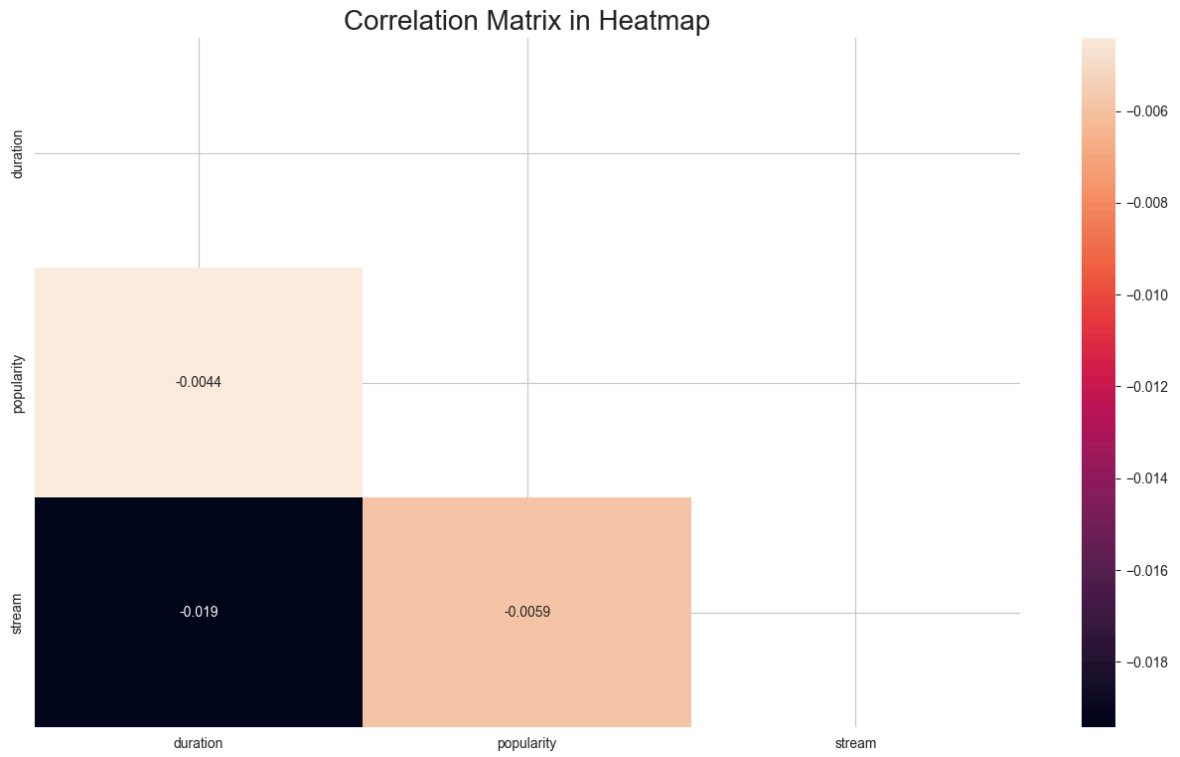


In [520…

```
numeric_df = df.select_dtypes(include=["number"])
mtr = numeric_df.corr()

mask = np.zeros_like(mtr)
mask[np.triu_indices_from(mask)] = True
fig, ax = plt.subplots(figsize=(16, 9))
plt.title("Correlation Matrix in Heatmap", size=20)
sns.heatmap(mtr, mask=mask, annot=True, annot_kws={"size": 10})
```

Out[520…

```
<Axes: title={'center': 'Correlation Matrix in Heatmap'}>
```



In [521…
```
df.head()
```

Out[521…

	song_title	artist	album	genre	release_date	duration	popularity	stre
0	Space executive series.	Sydney Clark	What.	Electronic	1997-11-08	282.0	42	35055
1	Price last painting.	Connor Peters DDS	Nature politics.	Electronic	2015-05-10	127.0	50	9249
2	Piece.	Anna Keith	Visit.	Pop	2024-07-08	NaN	10	76669
3	Power industry your.	Zachary Simpson	Behavior evening.	Hip-Hop	2022-08-15	214.0	86	34732
4	Food animal second.	Christopher Mcgee	Front.	Pop	2023-03-05	273.0	63	96649

In [522…
```
df.nunique()
```

Out[522…

```
song_title          6854
artist              6672
album               4520
genre                  9
release_date        5207
duration             301
popularity           100
stream              7000
language               7
explicit_content       2
label                  6
composer            6669
producer            6674
collaboration       2098
dtype: int64
```

In [523…

```python
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from feature_engine.wrappers import SklearnTransformerWrapper
from sklearn.preprocessing import MinMaxScaler
from feature_engine.encoding import OneHotEncoder

# Функція обробки відсутніх даних
def handle_missing_data(df):
    # 1. Заповнення пропусків у `duration` медіаною
    df["duration"].fillna(df["duration"].median(), inplace=True)

    # 2. Заповнення пропусків у `language` найчастішим значенням
    df["language"].fillna(df["language"].mode()[0], inplace=True)

    # 3. Додавання індикатору відсутніх значень для `collaboration`
    df["collaboration_na"] = df["collaboration"].isna().astype(int)

    # 4. Заповнення пропусків у `collaboration` значенням "Unknown"
    df["collaboration"].fillna("Unknown", inplace=True)

    return df

# Функція обробки змінної `release_date`
def handle_encoding(df):
    # Перетворення `release_date` на формат datetime
    df["release_date"] = pd.to_datetime(df["release_date"], errors="coerce")

    # Заповнення пропущених дат базовим значенням
    df["release_date"].fillna(pd.Timestamp("1900-01-01"), inplace=True)

    # Створення нових колонок з року, місяця та дня
    df["release_year"] = df["release_date"].dt.year
    df["release_month"] = df["release_date"].dt.month
    df["release_day"] = df["release_date"].dt.day

    # Видалення початкового стовпця `release_date`
    df.drop(columns=["release_date"], inplace=True)

# Обробка пропущених даних
df = handle_missing_data(df)

# Кодування змінної `release_date`
handle_encoding(df)
```

```
# Визначення ознак (X) і цільової змінної (y)
X = df[
    [
        "genre",
        "language",
        "label",
        "explicit_content",
        "collaboration_na",
        "duration",
        "stream",
        "release_year",
        "release_month",
        "release_day"
    ]
]
y = df["popularity"]

# Список категоріальних змінних
categorical_columns = [
    "genre",
    "language",
    "label",
    "explicit_content",
]

# Пайплайн обробки ознак
feature_engineering_pipeline = Pipeline(
    steps=[
        # One-hot encoding для категоріальних змінних
        (
            "encoder",
            OneHotEncoder(
                top_categories=10,  # Залишити лише 10 найчастіших категорій
                drop_last=True,
                drop_last_binary=True,
                ignore_format=True,
                variables=categorical_columns,
            ),
        ),
        # Масштабування числових змінних
        (
            "scaler",
            SklearnTransformerWrapper(MinMaxScaler()),
        ),
    ]
)
```

In [524…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7000 entries, 0 to 6999
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   song_title        7000 non-null   object
 1   artist            7000 non-null   object
 2   album             7000 non-null   object
 3   genre             7000 non-null   object
 4   duration          7000 non-null   float64
 5   popularity        7000 non-null   int64
 6   stream            7000 non-null   int64
 7   language          7000 non-null   object
 8   explicit_content  7000 non-null   object
 9   label             7000 non-null   object
 10  composer          7000 non-null   object
 11  producer          7000 non-null   object
 12  collaboration     7000 non-null   object
 13  collaboration_na  7000 non-null   int64
 14  release_year      7000 non-null   int32
 15  release_month     7000 non-null   int32
 16  release_day       7000 non-null   int32
dtypes: float64(1), int32(3), int64(3), object(10)
memory usage: 847.8+ KB
```

In [525…

```
# Split the transformed data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

X_train = feature_engineering_pipeline.fit_transform(X_train)

X_test = feature_engineering_pipeline.transform(X_test)
```

In [526…

```
import time
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge, Lasso
from sklearn.neighbors import KNeighborsRegressor
import xgboost as xgb
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Підготовка даних
X = df[
    [
        "genre",
        "language",
        "label",
        "explicit_content",
        "collaboration_na",
        "duration",
        "stream",
        "release_year",
```

```
        "release_month",
        "release_day"
    ]
]

y = df["popularity"]

# Розділення на тренувальні та тестові набори
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Опис категоріальних змінних
categorical_columns = [
    "genre",
    "language",
    "label",
    "explicit_content"
]

# Створення трансформера для категоріальних змінних (OneHotEncoder)
preprocessor = ColumnTransformer(
    transformers=[
        (
            "cat",
            OneHotEncoder(drop="first", handle_unknown="ignore"),
            categorical_columns,
        )
    ],
    remainder="passthrough",  # Залишити числові змінні без змін
)

# Ініціалізація моделей
lm = LinearRegression()
ridge = Ridge(random_state=42)
lasso = Lasso(random_state=42)
knn = KNeighborsRegressor()
xgbt = xgb.XGBRegressor(random_state=42)

# Список моделей для тестування
algo = [xgbt, lm, ridge, lasso, knn]

# Список для збереження результатів
result = []

# Тренування моделей та обчислення результатів
for model in algo:
    start = time.process_time()

    # Створення пайплайну з попередньою обробкою та моделлю
    pipeline = Pipeline(steps=[("preprocessor", preprocessor), ("model", model)])

    # Навчання моделі
    ml_model = pipeline.fit(X_train, y_train)

    # Прогнозування та обчислення метрик
    train_pred = ml_model.predict(X_train)
    test_pred = ml_model.predict(X_test)

    result.append(
```

```
        [
            str(model).split("(")[0] + "_baseline",
            ml_model.score(X_train, y_train),  # Train Score (R²)
            ml_model.score(X_test, y_test),  # Test Score (R²)
            np.sqrt(mean_squared_error(y_train, train_pred)),  # Train RMSE
            np.sqrt(mean_squared_error(y_test, test_pred)),  # Test RMSE
            mean_absolute_error(y_train, train_pred),  # Train MAE
            mean_absolute_error(y_test, test_pred),  # Test MAE
        ]
    )

    print(f"{str(model).split('(')[0]} ✓ {round(time.process_time() - start, 3)}

# Формування DataFrame з результатами
result_df = pd.DataFrame(
    result,
    columns=[
        "Algorithm",
        "Train_Score",
        "Test_Score",
        "Train_Rmse",
        "Test_Rmse",
        "Train_Mae",
        "Test_Mae",
    ],
)

# Сортування за Test RMSE
result_df = result_df.sort_values("Test_Rmse").set_index("Algorithm")
result_df
```

```
XGBRegressor ✓ 1.031 sec
LinearRegression ✓ 0.531 sec
Ridge ✓ 0.156 sec
Lasso ✓ 0.0 sec
KNeighborsRegressor ✓ 0.531 sec
```

Out[526…

Algorithm	Train_Score	Test_Score	Train_Rmse	Test_Rmse	Train_Ma
Lasso_baseline	0.000580	0.000432	28.798901	29.276418	24.98845
Ridge_baseline	0.004315	-0.005891	28.745032	29.368872	24.91202
LinearRegression_baseline	0.004315	-0.005911	28.745032	29.369151	24.91193
XGBRegressor_baseline	0.723540	-0.185307	15.146702	31.880639	11.95794
KNeighborsRegressor_baseline	0.188219	-0.191656	25.955019	31.965918	21.80082

Підбір гіперпараметрів

In [527…

```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.neighbors import KNeighborsRegressor
```

```
import xgboost as xgb

# Гіперпараметри для XGBoost
xgboost_params = {
    "model__n_estimators": [50, 100, 200],
    "model__learning_rate": [0.01, 0.1, 0.3],
    "model__max_depth": [3, 6, 9],
    "model__subsample": [0.8, 1.0],
    "model__colsample_bytree": [0.8, 1.0],
}

# Гіперпараметри для KNeighborsRegressor
knn_params = {
    "model__n_neighbors": [3, 5, 7, 9],
    "model__weights": ["uniform", "distance"],
    "model__metric": ["minkowski", "manhattan"],
}

# XGBoost Pipeline
xgb_pipeline = Pipeline(
    steps=[
        ("preprocessor", preprocessor),
        ("model", xgb.XGBRegressor(random_state=42)),
    ]
)

# KNeighborsRegressor Pipeline
knn_pipeline = Pipeline(
    steps=[
        ("preprocessor", preprocessor),
        ("model", KNeighborsRegressor()),
    ]
)

# GridSearchCV для XGBoost
xgb_grid_search = GridSearchCV(
    estimator=xgb_pipeline,
    param_grid=xgboost_params,
    cv=3,
    scoring="neg_mean_squared_error",
    n_jobs=-1,
    verbose=1,
)

# GridSearchCV для KNeighborsRegressor
knn_grid_search = GridSearchCV(
    estimator=knn_pipeline,
    param_grid=knn_params,
    cv=3,
    scoring="neg_mean_squared_error",
    n_jobs=-1,
    verbose=1,
)

# Пошук найкращих гіперпараметрів для XGBoost
print("Підбір гіперпараметрів для XGBoost...")
xgb_grid_search.fit(X_train, y_train)
print("Найкращі параметри для XGBoost:")
print(xgb_grid_search.best_params_)
print("Найкращий результат (MSE):", -xgb_grid_search.best_score_)
```

```
# Пошук найкращих гіперпараметрів для KNeighborsRegressor
print("Підбір гіперпараметрів для KNeighborsRegressor...")
knn_grid_search.fit(X_train, y_train)
print("Найкращі параметри для KNeighborsRegressor:")
print(knn_grid_search.best_params_)
print("Найкращий результат (MSE):", -knn_grid_search.best_score_)
```

```
Підбір гіперпараметрів для XGBoost...
Fitting 3 folds for each of 108 candidates, totalling 324 fits
Найкращі параметри для XGBoost:
{'model__colsample_bytree': 1.0, 'model__learning_rate': 0.01, 'model__max_dept
h': 3, 'model__n_estimators': 50, 'model__subsample': 0.8}
Найкращий результат (MSE): 830.8028631010116
Підбір гіперпараметрів для KNeighborsRegressor...
Fitting 3 folds for each of 16 candidates, totalling 48 fits
Найкращі параметри для KNeighborsRegressor:
{'model__metric': 'minkowski', 'model__n_neighbors': 9, 'model__weights': 'unifor
m'}
Найкращий результат (MSE): 921.7864302032714
```

Аналіз важливості ознак для моделей XGBoost та KNeighborsRegressor

In [528…

```
import time
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsRegressor
import xgboost as xgb

# Оновлення моделей з оптимальними гіперпараметрами
optimized_xgb = xgb.XGBRegressor(
    n_estimators=50,
    learning_rate=0.01,
    max_depth=3,
    subsample=0.8,
    colsample_bytree=1.0,
    random_state=42,
)

optimized_knn = KNeighborsRegressor(
    n_neighbors=9,
    weights="uniform",
    metric="minkowski",
)

# Список моделей із оптимальними гіперпараметрами
optimized_models = [
    ("XGBoost", optimized_xgb),
    ("KNeighbors", optimized_knn),
]

# Результати тестування
optimized_results = []

for model_name, model in optimized_models:
    start_time = time.process_time()
```

```
    # Створення пайплайну
    pipeline = Pipeline(steps=[("preprocessor", preprocessor), ("model", model)])

    # Навчання моделі
    pipeline.fit(X_train, y_train)

    # Прогнози на тестових даних
    train_pred = pipeline.predict(X_train)
    test_pred = pipeline.predict(X_test)

    # Обчислення метрик
    optimized_results.append(
        [
            model_name,
            pipeline.score(X_train, y_train),  # Train R²
            pipeline.score(X_test, y_test),  # Test R²
            np.sqrt(mean_squared_error(y_train, train_pred)),  # Train RMSE
            np.sqrt(mean_squared_error(y_test, test_pred)),  # Test RMSE
            mean_absolute_error(y_train, train_pred),  # Train MAE
            mean_absolute_error(y_test, test_pred),  # Test MAE
        ]
    )

    print(f"{model_name} ✓ {round(time.process_time() - start_time, 3)} sec")

# Формування таблиці результатів
optimized_results_df = pd.DataFrame(
    optimized_results,
    columns=[
        "Model",
        "Train_R²",
        "Test_R²",
        "Train_RMSE",
        "Test_RMSE",
        "Train_MAE",
        "Test_MAE",
    ],
)

# Відображення результатів
print(optimized_results_df)
```

```
XGBoost ✓ 0.406 sec
KNeighbors ✓ 0.891 sec
        Model  Train_R²   Test_R²  Train_RMSE  Test_RMSE  Train_MAE   Test_MAE
0     XGBoost  0.006546 -0.000856   28.712812  29.295264  24.909915  25.502445
1  KNeighbors  0.103463 -0.107064   27.276337  30.810449  23.228472  26.566429
```

Аналіз важливості ознак для моделей XGBoost та Random Forest

In [529…

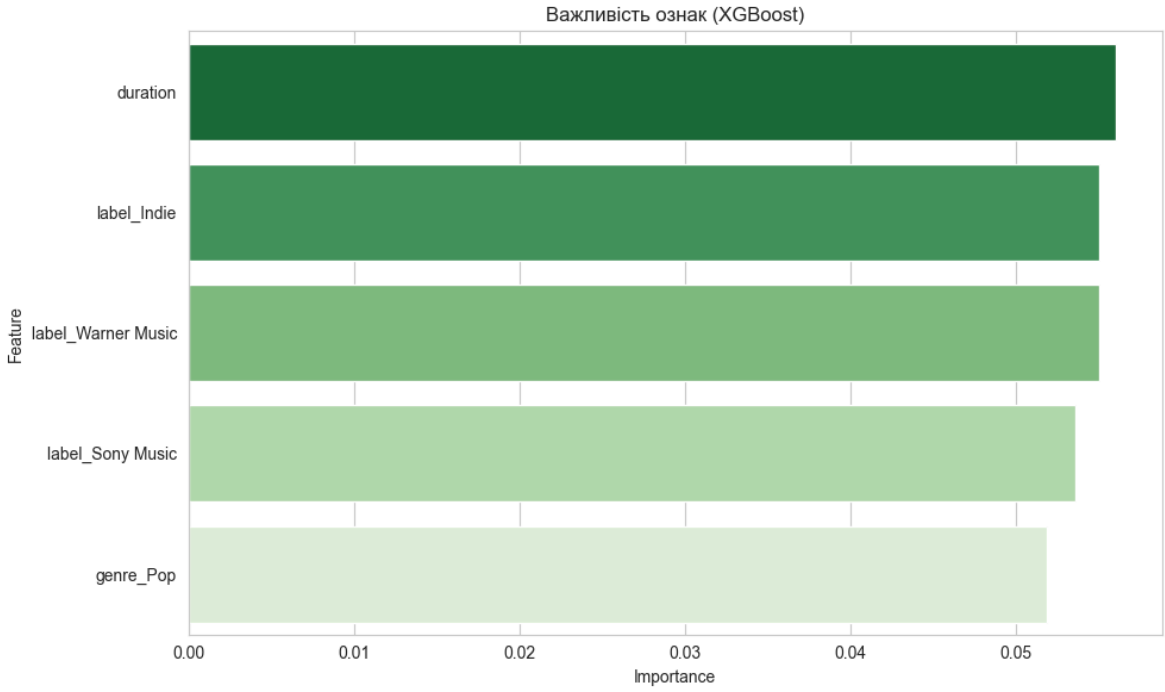
```
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsRegressor
import xgboost as xgb
import matplotlib.pyplot as plt
```

```
import seaborn as sns

# Створимо пайплайн для попередньої обробки
preprocessor = ColumnTransformer(
    transformers=[
        (
            "cat",
            OneHotEncoder(drop="first", handle_unknown="ignore"),
            categorical_columns,
        )
    ],
    remainder="passthrough",  # Залишити числові змінні без змін
)

# Ініціалізація моделей
knn_model = KNeighborsRegressor(
    n_neighbors=9,
    weights="uniform",
    metric="minkowski",
)

xgb_model = xgb.XGBRegressor(
    n_estimators=50,
    learning_rate=0.01,
    max_depth=3,
    subsample=0.8,
    colsample_bytree=1.0,
    random_state=42,
)

# Тренування моделей на тренувальних даних
knn_pipeline = Pipeline(steps=[("preprocessor", preprocessor), ("model", knn_mod
xgb_pipeline = Pipeline(steps=[("preprocessor", preprocessor), ("model", xgb_mod

# Навчання моделей
knn_pipeline.fit(X_train, y_train)
xgb_pipeline.fit(X_train, y_train)

# Важливість ознак для XGBoost
xgb_importances = xgb_pipeline.named_steps["model"].feature_importances_

# Отримання назв ознак після OneHotEncoder
one_hot_columns = preprocessor.transformers_[0][1].get_feature_names_out(
    categorical_columns
)
feature_names = list(one_hot_columns) + [
    col for col in X_train.select_dtypes(include=["float64", "int64", "int32"]).
]

# Створимо таблицю для важливості ознак
xgb_importance_df = pd.DataFrame(
    {"Feature": feature_names, "XGBoost Importance": xgb_importances}
).sort_values(by="XGBoost Importance", ascending=False)

# Виведемо таблицю
print("Важливість ознак для XGBoost:")
print(xgb_importance_df)
```

```
Важливість ознак для XGBoost:
                  Feature  XGBoost Importance
21               duration            0.056079
15            label_Indie            0.055082
18     label_Warner Music            0.055060
16       label_Sony Music            0.053661
5               genre_Pop            0.051924
19   explicit_content_Yes            0.049665
25            release_day            0.049499
24          release_month            0.049163
20       collaboration_na            0.047524
11      language_Japanese            0.047467
23           release_year            0.047450
22                 stream            0.046854
14          label_Def Jam            0.045748
8          language_French            0.045232
13       language_Spanish            0.043938
2               genre_Folk            0.038729
6                genre_R&B            0.038194
7             genre_Reggae            0.037262
17  label_Universal Music            0.035910
0           genre_Country            0.035574
10       language_Italian            0.035354
3           genre_Hip-Hop            0.034629
9          language_German            0.000000
4               genre_Jazz            0.000000
1         genre_Electronic            0.000000
12         language_Korean            0.000000
```

In [530…

```python
# Побудова графіка для важливості ознак (XGBoost)
plt.figure(figsize=(10, 6))
sns.barplot(
    y=xgb_importance_df["Feature"].head(5),
    x=xgb_importance_df["XGBoost Importance"].head(5),
    palette="Greens_r",
)
plt.title("Важливість ознак (XGBoost)")
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```


Важливість ознак (XGBoost)
duration
label_Indie
label_Warner Music
label_Sony Music
genre_Pop
Feature
0.00
0.01
0.02
0.03
0.04
0.05
Importance