MACHINE LEARNING PROJECT 2

by Taibah Shahbaz
2023-BSAI-024 4th A

Project 2: Classifying Most Streamed Spotify Songs Introduction

- **Goal:** Categorize songs using audio and popularity metrics
- **Algorithms:** Random Forest, Logistic Regression, ANN
- **Dataset:**

Source: Spotify 2023 Songs Dataset (Kaggle)

Total Records: 953

Features: 12 (track_name , artist_name, release_date, streams, in_spotify_charts ,in_apple_charts etc.)

Target Variable: Popularity Class

Preprocessing Steps (project 2)

- **Library Imports:** pandas, numpy, sklearn
- **Initial Data Exploration:** .head(), .info(), .describe()
- **Data Cleaning:** Dropped irrelevant columns (e.g., track_name, artist_name), handled missing values
- **Feature Selection:** Retained audio features and chart presence columns only
- **Encoding & Scaling:**
  - No encoding needed (features mostly numeric)
  - StandardScaler used to normalize feature values
- **Train-Test Split:** Data split into training and test sets (usually 80-20 or 70-30)

Preprocessing Steps (project 2)

Load and Preprocess Dataset

```
[4]: df = pd.read_csv("spotify-2023.csv", encoding="latin1")
```

```
[5]: df.shape
```

```
[5]: (953, 24)
```

```
[6]: df.columns
```

```
[6]: Index(['track_name', 'artist(s)_name', 'artist_count', 'released_year',
       'released_month', 'released_day', 'in_spotify_playlists',
       'in_spotify_charts', 'streams', 'in_apple_playlists', 'in_apple_charts',
       'in_deezer_playlists', 'in_deezer_charts', 'in_shazam_charts', 'bpm',
       'key', 'mode', 'danceability_%', 'valence_%', 'energy_%',
       'acousticness_%', 'instrumentalness_%', 'liveness_%', 'speechiness_%'],
      dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 953 entries, 0 to 952
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   track_name            953 non-null    object
 1   artist(s)_name        953 non-null    object
 2   artist_count          953 non-null    int64
 3   released_year         953 non-null    int64
 4   released_month        953 non-null    int64
 5   released_day          953 non-null    int64
 6   in_spotify_playlists  953 non-null    int64
 7   in_spotify_charts     953 non-null    int64
 8   streams               953 non-null    object
 9   in_apple_playlists    953 non-null    int64
 10  in_apple_charts       953 non-null    int64
 11  in_deezer_playlists   953 non-null    object
 12  in_deezer_charts      953 non-null    int64
 13  in_shazam_charts      903 non-null    object
 14  bpm                   953 non-null    int64
 15  key                   858 non-null    object
 16  mode                  953 non-null    object
```

```
[10]: columns_to_drop = ['track_name', 'artist(s)_name', 'released_year', 'released_month', 'released_day']
df.drop(columns=[col for col in columns_to_drop if col in df.columns], inplace=True)
```

```
[11]: df.dropna(inplace=True)
```

```
[12]: df['streams'] = pd.to_numeric(df['streams'].str.replace(',', ''), errors='coerce')
df.columns = df.columns.str.strip().str.lower()
print(df.columns.tolist())
```

```
['artist_count', 'in_spotify_playlists', 'in_spotify_charts', 'streams', 'in_apple_playlists', 'in_apple_charts', 'in_deezer_playlists', 'in_deezer_charts', 'in_shazam_charts', 'bpm', 'key', 'mode', 'danceability_%', 'valence_%', 'energy_%', 'acousticness_%', 'instrumentalness_%', 'liveness_%', 'speechiness_%']
```

```
[13]: df.dropna(subset=['streams'], inplace=True)
threshold = df['streams'].quantile(0.75)
df['is_popular'] = (df['streams'] >= threshold).astype(int)
df.drop(columns='streams', inplace=True)

df = pd.get_dummies(df, drop_first=True)

df.dropna(inplace=True)

print(df.columns)
```

```
Index(['artist_count', 'in_spotify_playlists', 'in_spotify_charts',
       'in_apple_playlists', 'in_apple_charts', 'in_deezer_charts', 'bpm',
       'danceability_%', 'valence_%', 'energy_%',
       ...
       'key_B', 'key_C#', 'key_D', 'key_D#', 'key_E', 'key_F', 'key_F#',
       'key_G', 'key_G#', 'mode_Minor'],
      dtype='object', length=503)
```

Modeling (Project 2)

1. **Models Used:** Random Forest, Logistic Regression, Artificial Neural Network (ANN)
2. **Libraries:** Sklearn.ensemble.RandomForestClassifier, sklearn.linear_model.LogisticRegression, keras.models.Sequential for ANN
3. **Train-Test Split:** 80:20 ratio
4. **Training:** All models trained on scaled features and ANN trained with input, hidden, and output layers
5. **Tools & Methods:** .fit(), .predict() for all models, model.evaluate() and classification_report() for evaluation

Modeling (Project 2)

Split Features and Labels

```
[15]: from sklearn.model_selection import train_test_split

X = df.drop(columns=['is_popular'])
y = df['is_popular']

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

Scale Features

```
[17]: scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Train Classification Model

```
[19]: model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
[19]: RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Logistic Regression Classifier

```
[26]: log_model = LogisticRegression()
log_model.fit(X_train_scaled, y_train)
log_preds = log_model.predict(X_test_scaled)
```

Artificial Neural Network (ANN)

```
[29]: import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
[30]: import numpy as np

X_train = np.array(X_train).astype('float32')
y_train = np.array(y_train).astype('int32')

X_test_scaled = np.array(X_test_scaled).astype('float32')
y_test = np.array(y_test).astype('int32')
```

Evaluation (project 2)

Metrics Used:

- **Accuracy Score** – Measures overall correctness
- **Precision, Recall, F1-Score** – From classification_report()
- **Confusion Matrix** – For understanding class-wise performance

Visual Evaluation:

- Confusion Matrix Heatmap
- Accuracy Comparison Across Models

Spotify project (evaluation):

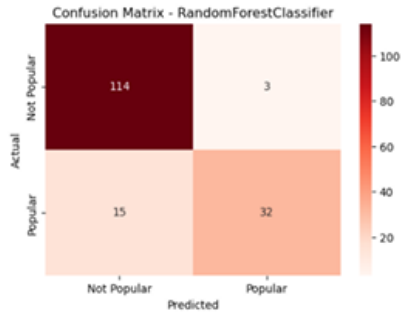
Evaluate Model

```
[21]: y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8902439024390244
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.97      0.93       117
           1       0.91      0.68      0.78        47

    accuracy                           0.89       164
   macro avg       0.90      0.83      0.85       164
weighted avg       0.89      0.89      0.88       164
```

Confusion Matrix

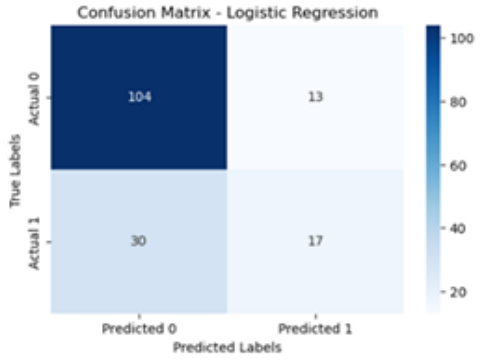
```
[58]: import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds', xticklabels=['Not Popular', 'Popular'], yticklabels=['Not Popular', 'Popular'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - RandomForestClassifier')
plt.show()
```




```
[27]: print("Logistic Regression:")
print("Accuracy:", accuracy_score(y_test, log_preds))
print(confusion_matrix(y_test, log_preds))
print(classification_report(y_test, log_preds))
```

```
Logistic Regression:
Accuracy: 0.7378048780487805
[[104  13]
 [ 30  17]]
              precision    recall  f1-score   support

           0       0.78      0.89      0.83       117
           1       0.57      0.36      0.44        47

    accuracy                           0.74       164
   macro avg       0.67      0.63      0.64       164
weighted avg       0.72      0.74      0.72       164
```

```
[56]: cm = confusion_matrix(y_test, log_preds)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Predicted 0', 'Predicted 1'], yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()
```



Spotify project (evaluation):

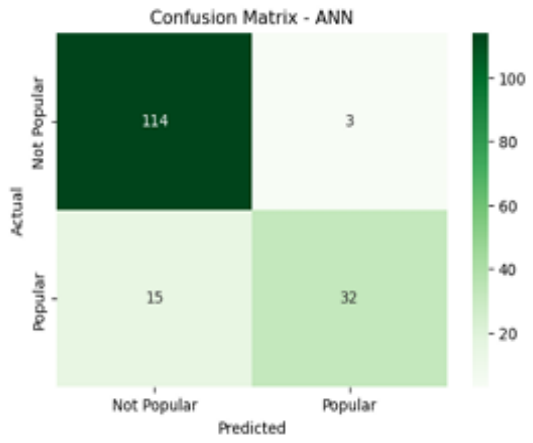
```
model = Sequential()
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2)

y_pred_ann = (model.predict(X_test_scaled) > 0.5).astype(int)

print("\nArtificial Neural Network")
print("Accuracy:", accuracy_score(y_test, y_pred_ann))
print(confusion_matrix(y_test, y_pred_ann))
print(classification_report(y_test, y_pred_ann))
```

```
Epoch 1/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 5s 36ms/step - accuracy: 0.5108 - loss: 53.3049 - val_accuracy: 0.7328 - val_loss: 14.8772
Epoch 2/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.7752 - loss: 10.9919 - val_accuracy: 0.8702 - val_loss: 1.4928
Epoch 3/20
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8725 - loss: 2.3329 - val_accuracy: 0.8855 - val_loss: 1.6298
```

```
Artificial Neural Network
Accuracy: 0.5304878048780488
[[55 62]
 [15 32]]
              precision    recall  f1-score   support

           0       0.79      0.47      0.59       117
           1       0.34      0.68      0.45        47

    accuracy                           0.53       164
   macro avg       0.56      0.58      0.52       164
weighted avg       0.66      0.53      0.55       164
```

Confusion Matrix - ANN

```
[60]: cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens', xticklabels=['Not Popular', 'Popular'], yticklabels=['Not Popular', 'Popular'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - ANN')
plt.show()
```



Key Findings

- **Random Forest** achieved the highest accuracy among the models tested
- **Logistic Regression** performed decently with faster training but slightly lower precision
- **ANN** introduced a deep learning angle but required more tuning
- **Feature scaling and model comparison** were critical to classification success
- **Business Insight:** Supports streaming platforms in identifying and promoting trending content

Conclusion

- **Random Forest outperformed Logistic Regression and ANN** in accuracy and consistency
- **Feature selection and scaling** played a crucial role in improving model results
- **Business Insight:** Enables music platforms and marketers to identify trending songs and tailor playlists or promotions based on data-driven popularity predictions

Thank you