EE 375 Final Project on Jhayco Machine Learning Song Analysis -- Cristian Hernandez

Project Topic: Predicting Music Streaming Success Using Machine Learning

The modern music industry heavily relies on data analytics to predict the success of new tracks. With millions of songs being released annually, predicting a track's potential performance can help artists and record labels make strategic decisions. My project focused on building a machine learning model that forecasts the number of streams an unreleased song could receive based on audio features extracted from MP3 files. By analyzing patterns from Jhayco's historical music catalog, I aimed to create a predictive model using Python libraries such as Librosa, Scikit-learn, and XGBoost.

The first step was gathering relevant data from Jhayco's discography. This included track details like release dates and total streams from platforms like Spotify. Additionally, I extracted audio features using the Librosa library, which analyzes MP3 files to produce various audio characteristics, such as tempo, spectral centroid, spectral bandwidth, root mean square energy (RMSE), MFCC (Mel-frequency cepstral coefficients), and zero-crossing rate. Since these features exist on different scales, such as streams measured in billions while audio features have much smaller ranges, I normalized the dataset using StandardScaler from Scikit-learn and applied a logarithmic transformation to the stream count target variable. This ensured that features were comparable and the model was less sensitive to extreme values.

Data preprocessing was an essential stage of the project. I carefully reviewed the extracted audio features to identify any potential outliers or inconsistencies. Missing data points were addressed using imputation techniques, ensuring the dataset remained as comprehensive as possible. After preprocessing, I examined the distribution of streams and audio features, confirming that the scaling and transformations had improved feature comparability. This step was critical because even minor inconsistencies could negatively impact the model's performance.

Initially, I used a Random Forest Regressor, a robust ensemble learning method. However, the model struggled with the large-scale differences between audio features and stream counts. Random Forest's lack of sensitivity to feature scaling became problematic, producing high Mean Squared Errors (MSE). After reviewing feature importance, I decided to switch to XGBoost, a more advanced boosting algorithm that handles complex datasets better. The final model pipeline included feature scaling using StandardScaler, model training with XGBoost using hyperparameter tuning via GridSearchCV, and evaluation based on MSE and feature importance analysis.

Model evaluation revealed both strengths and weaknesses. While XGBoost provided significant improvements in prediction accuracy, the model occasionally overestimated stream counts. For example, the song "Mami Chula" was predicted to receive 793 million

streams when it had only 179 million. This discrepancy likely stemmed from several limitations. First, the dataset only included part of Jhayco's discography, reducing the model's ability to generalize. Second, the model couldn't account for external influences like marketing efforts, playlist placements, and fan engagement, which heavily affect streaming success. Lastly, despite adding features like tempo variance and MFCC coefficients, the features captured only the song's inherent qualities, not industry-specific variables.

To enhance the model's predictive power, future improvements could include expanding the dataset to cover Jhayco's entire discography and tracks from similar artists, integrating social media engagement data from platforms like Instagram and Twitter, incorporating marketing campaign metrics such as ad spend and playlist inclusion, and using natural language processing (NLP) to assess lyrical themes. Additionally, considering the effect of featured artists and genre popularity could further refine the model's accuracy.

The implementation pipeline was executed in Python using libraries such as Librosa for audio feature extraction, Pandas and Numpy for data management and numerical operations, Scikit-learn for data preprocessing, scaling, and evaluation, and XGBoost for model training and prediction. The code's key stages included loading and preprocessing the data, extracting audio features, engineering new features, and performing a comprehensive evaluation using MSE calculations and feature importance visualization.

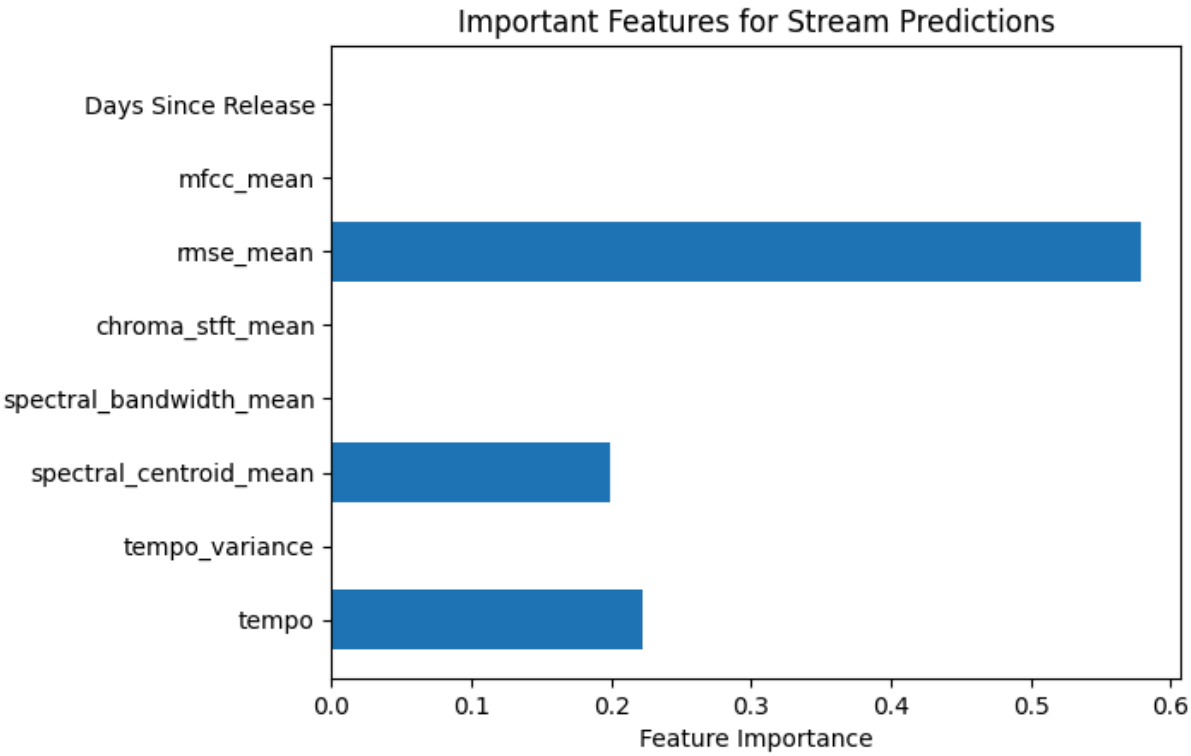
This project demonstrated the complexities of predictive modeling in the music industry. Accurate predictions require extensive feature engineering, proper data scaling, and careful model selection. Expanding the dataset, adding social and marketing metrics, and incorporating more sophisticated audio features would further improve its predictive capabilities. Overall, this project showcased the intersection of technology and music, highlighting the potential of machine learning to transform how the music industry anticipates a song's commercial success. The insights gained from this project underscored the ever-evolving relationship between data science and creative industries, offering new ways to understand and predict cultural trends.

In [ ]:
```
import librosa
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
from xgboost import XGBRegressor


np.complex = np.complex128


file_path_excel = 'ALL JHAYCO - Copy.xlsx'
df_tracks = pd.read_excel(file_path_excel, sheet_name='Sheet2')
```

```
mp3_files = {
    'Christian Dior': 'Christian Dior.mp3',
    'Easy (Remix)': 'Easy (Remix).mp3',
    'Imaginaste': 'Imaginaste.mp3',
    'Ley Seca': 'Ley Seca.mp3',
    'Medusa': 'Medusa.mp3',
    'Holanda': 'Holanda.mp3',
    'Como Se Siente (Remix)': 'Como Se Siente (Remix).mp3',
    'No Me Conoce (Remix)': 'No Me Conoce (Remix).mp3'
}


def extract_audio_features(file_path):
    y, sr = librosa.load(file_path, sr=None)
    tempo = librosa.beat.tempo(y=y, sr=sr)[0]
    tempo_variance = np.var(librosa.beat.tempo(y=y, sr=sr))
    spectral_centroid_mean = np.mean(librosa.feature.spectral_centroid(y=y, sr=sr))
    spectral_bandwidth_mean = np.mean(librosa.feature.spectral_bandwidth(y=y, sr=sr
    chroma_stft_mean = np.mean(librosa.feature.chroma_stft(y=y, sr=sr))
    rmse_mean = np.mean(librosa.feature.rms(y=y))
    mfcc_mean = np.mean(librosa.feature.mfcc(y=y, sr=sr), axis=1).mean()

    features = {
        'tempo': tempo,
        'tempo_variance': tempo_variance,
        'spectral_centroid_mean': spectral_centroid_mean,
        'spectral_bandwidth_mean': spectral_bandwidth_mean,
        'chroma_stft_mean': chroma_stft_mean,
        'rmse_mean': rmse_mean,
        'mfcc_mean': mfcc_mean
    }
    return features


audio_features_list = []

for track, path in mp3_files.items():
    features = extract_audio_features(path)
    features['Track'] = track
    audio_features_list.append(features)

df_audio_features = pd.DataFrame(audio_features_list)


df_merged = pd.merge(df_tracks, df_audio_features, on='Track')


df_merged['Days Since Release'] = (pd.to_datetime('today') - pd.to_datetime(df_merg
df_merged['Log Total Streams'] = np.log1p(df_merged['Streams'])


X = df_merged[['tempo', 'tempo_variance', 'spectral_centroid_mean', 'spectral_bandw
               'chroma_stft_mean', 'rmse_mean', 'mfcc_mean', 'Days Since Release']]
y = df_merged['Log Total Streams']
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran


model = XGBRegressor(random_state=42, objective='reg:squarederror')
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7, 9],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.7, 0.8, 0.9, 1.0]
}


grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, n_jobs=-1,
grid_search.fit(X_train, y_train)


y_pred = grid_search.best_estimator_.predict(X_test)
mse = mean_squared_error(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'Best Model Parameters: {grid_search.best_params_}')


def evaluate_unreleased_song(file_path):
    features = extract_audio_features(file_path)
    input_data = pd.DataFrame([features])
    input_data['Days Since Release'] = 480


    input_scaled = scaler.transform(input_data)


    log_prediction = grid_search.best_estimator_.predict(input_scaled)[0]
    predicted_streams = np.expm1(log_prediction)

    print(f'Predicted Streams for Unreleased Song: {predicted_streams:.0f}')


evaluate_unreleased_song("Mami Chula.mp3")


feature_importances = grid_search.best_estimator_.feature_importances_
feature_names = ['tempo', 'tempo_variance', 'spectral_centroid_mean', 'spectral_ban

plt.barh(feature_names, feature_importances)
plt.xlabel("Feature Importance")
plt.title("Important Features for Stream Predictions")
plt.show()
```

```
C:\Users\cgher\AppData\Local\Temp\ipykernel_2660\653687737.py:34: FutureWarning: lib
rosa.beat.tempo
        This function was moved to 'librosa.feature.rhythm.tempo' in librosa version
0.10.0.
        This alias will be removed in librosa version 1.0.
  tempo = librosa.beat.tempo(y=y, sr=sr)[0]
C:\Users\cgher\AppData\Local\Temp\ipykernel_2660\653687737.py:35: FutureWarning: lib
rosa.beat.tempo
        This function was moved to 'librosa.feature.rhythm.tempo' in librosa version
0.10.0.
        This alias will be removed in librosa version 1.0.
  tempo_variance = np.var(librosa.beat.tempo(y=y, sr=sr))
Fitting 3 folds for each of 144 candidates, totalling 432 fits
C:\Users\cgher\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfr
a8p0\LocalCache\local-packages\Python311\site-packages\sklearn\model_selection\_sear
ch.py:1103: UserWarning: One or more of the test scores are non-finite: [nan nan nan
nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan]
  warnings.warn(
Mean Squared Error: 0.4704682272726376
Best Model Parameters: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 100,
'subsample': 0.7}
C:\Users\cgher\AppData\Local\Temp\ipykernel_2660\653687737.py:34: FutureWarning: lib
rosa.beat.tempo
        This function was moved to 'librosa.feature.rhythm.tempo' in librosa version
0.10.0.
        This alias will be removed in librosa version 1.0.
  tempo = librosa.beat.tempo(y=y, sr=sr)[0]
C:\Users\cgher\AppData\Local\Temp\ipykernel_2660\653687737.py:35: FutureWarning: lib
rosa.beat.tempo
        This function was moved to 'librosa.feature.rhythm.tempo' in librosa version
0.10.0.
        This alias will be removed in librosa version 1.0.
  tempo_variance = np.var(librosa.beat.tempo(y=y, sr=sr))
Predicted Streams for Unreleased Song: 793043264
```


Important Features for Stream Predictions
Days Since Release
mfcc_mean
rmse_mean
chroma_stft_mean
spectral_bandwidth_mean
spectral_centroid_mean
tempo_variance
tempo
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Feature Importance