**Detailed Report**

**Azure Spotify ML Pipeline and Data Warehousing Project**

**Part 1**

Data Ingestion from Kaggle Api to Azure Blob Storage using Airflow

Before starting the Data Ingestion process , Airflow must be setup in our system using wsl commands or Dockerized application,

Here's a step-by-step tutorial to set up and run an Airflow DAG for transferring Spotify data from Kaggle to Azure Storage:

The Kaggle dataset which we are using is   
<https://www.kaggle.com/datasets/dhruvildave/spotify-charts>

**1.Inital Setup**

**Create and Activate Virtual Environment**

Bash

Python -m venv airflow\_env

source airflow\_env/bin/activate

go to nano/airflow\_env/bin/activate file,

then

add your azure storage credentials

export AZURE\_STORAGE\_ACCOUNT\_NAME="your\_account\_name"

export AZURE\_STORAGE\_CONTAINER\_NAME="your\_container"

export AZURE\_STORAGE\_CONNECTION\_STRING="your\_storage\_connection\_string"

**Install Required Packages**

Bash

pip install apache-airflow

pip install kagglehub

**2.Initialize Airflow**

**Setup Airflow Home**

Bash

mkdir -p /home/shushilgirish/airflow/dags

cd airflow/dags

**3. Configure Airflow**

**Edit Airflow Configuration**

bash

nano airflow.cfg

Key configurations to modify:

- Set the correct dags\_folder path

- Configure the database connection

- Set the appropriate timezone

A screenshot of a computer

Description automatically generated

**4. Start Airflow Services**

bash

airflow webserver --port 8081

Start Scheduler in a New Terminal

source airflow\_env/bin/activate

airflow scheduler

A screenshot of a computer screen

Description automatically generated

**5. Create Admin User**

airflow users create \

--username admin \

--firstname Peter \

--lastname Parker \

--role Admin \

--email [spiderman@superhero.org](mailto:spiderman@superhero.org)

**6. Verify Setup**

Check Running DAGs

bash

airflow dags list

Check for error: bash

airflow dags list-import-errors

**7. DAG Management**

bash

airflow dags detail spotify\_dagfile.py

**8. Troubleshooting(Check Port Conflicts)**

bash

lsof -i tcp:8080

**Stop Running Processes**

Kill specific processes

kill 20003 44800 44801

Kill scheduler using PID file

kill $(cat ~/airflow/airflow-scheduler.pid)

This setup allows you to create a DAG that can fetch spotify data from Kaggle api ,process the data as needed,uploads it to Azure Blob Storage and runs on a schedules basis

Once everything mentioned above is done, we can use the success status in your airflow running in your local website

A screenshot of a computer

Description automatically generated

And the data is loaded into the rawdata folder in the Azure Data Storage account.

A screenshot of a computer

Description automatically generated

Part 2

Basic Data Transformation:

Tools Used: Azure Data Factory and Databricks Notebook

After loading in Azure Storage container, data is transformed in the Azure Databricks notebook in schedule mode using Data Factory as the orchestrator. Before processing with the transformation, for the safe authentication , an new service principal called **databricks-access** is create app registry of the azure , and then a certificate is created from which id and secret values is noted for integrating it with azure key vault. Using the service principal created , various roles are assigned to it based on the azure resource in the access control section of the workspace

A screenshot of a computer

Description automatically generated

And then in the Databricks workspace , go to compute and create the cluster with small size such as 1Driver with 16Gb memory,4 cores and runtime 15.4.x-scala2.12

A screenshot of a computer

Description automatically generated

This is sufficient compute engine which moderately consumes your Azure credits , so you don’t have to be much cautious about the compute . And after creating the cluster , note the cluster id and cluster name , to Linkedservice connection in data factory.The Data Factory is mainly used for the batch processing of the large Spotfiy dataset which contains almost 20 million records .

Establish a databricks connection in data factory manage services section

A screenshot of a computer

Description automatically generated

After that use the databricks component in the pipeline section

A screenshot of a computer

Description automatically generated

When running the debug, it triggers the cluster mentioned in the linkedservice , making the pipeline to run. It can be monitored in the monitor section of the workspace .

Databricks Transformations:

Before running the Adf pipeline, you need to create a key vault certificate for the storage account key for safe authorization with datarbricks notebook or create secret scope.

After creating , use the certificate id for connecting inside the databricks notebook ,

And run the spark configuration commands , after that copy the wasb file path of the rawdata which starts like ‘wasb://rawdata@storage\_name.blob.core.windows.net/file.csv’

The data comprises of 9 columns:

Title- music tracks of each artists (String)

Rank-music which ranked in the chart ,ranging from 1 to 200

Date-from 2017 to 2021

Artist-artist names

url- spotify link of the song

region-Countries where the music charts were recognized

chart- have values as top200 or viral50

trend – have 4 values , new\_entry,same\_position,move\_up and move\_down

and

streams – Songs streamed across a region for a track

and it contains 20 millions rows , The data is about Top 200 Spotify music tracks from the year 2017-2021 across all regions including from global charts .

Here are the basic transformation covered in the notebook:

a.date column is converted to datetime format and rank/streams to integer

A screen shot of a computer program

Description automatically generated

b.removed punctuations,special characters ,lowercased and removed all the string columns except url

c.used regex to find patterns and remove redundant components

d.found 10k + rows where there rank is more than 50 and the chart is viral50, Also those rows doesn’t have any value in streams column. Imputed those streams column with the one from the rows which are top200 .

A screenshot of a computer program

Description automatically generated

e. There were multipled artist names in a single row. So , the names were exploded using pyspark function and appended into separate rows for each artist names in the previous row.

A screen shot of a computer program

Description automatically generated

Finally , the transformed data is then sent back to the account in the staggingdata container in csv format.

A screenshot of a computer program

Description automatically generated

Part 3

Data Warehousing and Analytics using Synapse

Firstly , dedicated sql pool is created with configuration of DW300c performance level , 240TB max size .Later on the staggingdata is copied from the storage account to the dedicated sql pool as table using copy data tool in synapse analytics workspace.

A screenshot of a computer

Description automatically generated

After the stagging table(spotify\_stagging) is created , Kimball Dimensional model is created for fast and seamless query purpose  
A diagram of a data stream

Description automatically generated with medium confidence

The model comprises of :  
**Dimensions:**

1.dim\_track-contains song tracks populated in charts

2.dim\_artist- All the artist which were exploded in the rawdata,are listed in this table

3.dim\_chart-these refers to the ranking based on top200 or viral50

4.dim\_region-countries are allocated in this table for offering spatial capabilities

5.dim\_date - This table, which has fields for day, week, month, quarter, and year, allows for analysis by date and hence supports temporal trend analysis.

6.dim\_trend

**Fact:**

1.fact\_streams-it contains the streams and rank values of each tracks and each artists

This is the sql ddl script

DROP TABLE dbo.spotify\_stagging;

CREATE TABLE dbo.spotify\_stagging (

    title NVARCHAR(255),

    rank INT,

    date DATE,

    artist NVARCHAR(255),

    url NVARCHAR(500),

    region NVARCHAR(50),

    chart NVARCHAR(100),

    trend NVARCHAR(50),

    streams BIGINT,

    track\_id VARCHAR(255) -- Changed to VARCHAR for compatibility

)

WITH (

    DISTRIBUTION = ROUND\_ROBIN,

    HEAP

);

--artists

CREATE TABLE dbo.dim\_artist (

    artist\_id INT IDENTITY(1, 1),

    artist NVARCHAR(255)

)

WITH (

    DISTRIBUTION = REPLICATE,

    HEAP

);

CREATE CLUSTERED INDEX CI\_dim\_artist ON dbo.dim\_artist (artist\_id);

--dim track

CREATE TABLE dbo.dim\_track (

    track\_id VARCHAR(255),

    title NVARCHAR(255),

    url NVARCHAR(500)

)

WITH (

    DISTRIBUTION = REPLICATE,

    HEAP

);

CREATE CLUSTERED INDEX CI\_dim\_track ON dbo.dim\_track (track\_id);

--dim date

CREATE TABLE dbo.dim\_date (

    date\_id INT IDENTITY(1, 1),

    date DATE,

    year INT,

    month INT,

    day INT

)

WITH (

    DISTRIBUTION = REPLICATE,

    HEAP

);

CREATE CLUSTERED INDEX CI\_dim\_date ON dbo.dim\_date (date\_id);

--dim region

CREATE TABLE dbo.dim\_region (

    region\_id INT IDENTITY(1, 1),

    region NVARCHAR(50)

)

WITH (

    DISTRIBUTION = REPLICATE,

    HEAP

);

CREATE CLUSTERED INDEX CI\_dim\_region ON dbo.dim\_region (region\_id);

CREATE TABLE dbo.dim\_trend (

    trend\_id INT IDENTITY(1, 1),

    trend NVARCHAR(50)

)

WITH (

    DISTRIBUTION = REPLICATE,

    HEAP

);

CREATE CLUSTERED INDEX CI\_dim\_trend ON dbo.dim\_trend (trend\_id);

CREATE TABLE dbo.dim\_chart (

    chart\_id INT IDENTITY(1, 1),

    chart NVARCHAR(50)

)

WITH (

    DISTRIBUTION = REPLICATE,

    HEAP

);

CREATE CLUSTERED INDEX CI\_dim\_chart ON dbo.dim\_chart (chart\_id);

--fact table

CREATE TABLE dbo.fact\_streams (

    fact\_id INT IDENTITY(1, 1),

    track\_id VARCHAR(255),  -- FK reference to dim\_track

    artist\_id INT,          -- FK reference to dim\_artist

    region\_id INT,          -- FK reference to dim\_region

    date\_id INT,            -- FK reference to dim\_date

    trend\_id INT, -- FK reference to dim\_trend

    chart\_id INT, -- FK reference to dim\_chart

    streams BIGINT,

    rank INT

)WITH (

    DISTRIBUTION = HASH(track\_id),

    HEAP

);

CREATE CLUSTERED COLUMNSTORE INDEX CCI\_fact\_streams ON dbo.fact\_streams;

After creating the tables , the dimension table are inputted with the values from the spotify\_stagging table  
-- Insert into dim\_artist

INSERT INTO dbo.dim\_artist (artist)

SELECT DISTINCT artist

FROM dbo.spotify\_stagging;

-- Insert into dim\_track

INSERT INTO dbo.dim\_track (track\_id, title, url)

SELECT DISTINCT track\_id, title, url

FROM dbo.spotify\_stagging;

-- Insert into dim\_date

INSERT INTO dbo.dim\_date (date, year, month, day)

SELECT DISTINCT

    date,

    YEAR(date) AS year,

    MONTH(date) AS month,

    DAY(date) AS day

FROM dbo.spotify\_stagging;

-- Insert into dim\_region

INSERT INTO dbo.dim\_region (region)

SELECT DISTINCT region

FROM dbo.spotify\_stagging;

-- Populate dim\_trend

INSERT INTO dbo.dim\_trend (trend)

SELECT DISTINCT trend

FROM dbo.spotify\_stagging;

-- Populate dim\_chart

INSERT INTO dbo.dim\_chart (chart)

SELECT DISTINCT chart

FROM dbo.spotify\_stagging;

And the stream table is creacted using the dimension tables and spotify\_stagging table  
-- Add trend and chart relationships during population

INSERT INTO dbo.fact\_streams (track\_id, artist\_id, region\_id, date\_id, trend\_id, chart\_id, streams, rank)

SELECT

    st.track\_id,

    da.artist\_id,

    dr.region\_id,

    dd.date\_id,

    dt.trend\_id,

    dc.chart\_id,

    st.streams,

    st.rank

FROM dbo.spotify\_stagging st

JOIN dbo.dim\_artist da ON st.artist = da.artist

JOIN dbo.dim\_region dr ON st.region = dr.region

JOIN dbo.dim\_date dd ON st.date = dd.date

JOIN dbo.dim\_trend dt ON st.trend = dt.trend

JOIN dbo.dim\_chart dc ON st.chart = dc.chart;

Later , on the tables created were used for query using Synapse notebook ran in spark cluster ,

Query 1:Trend Analysis Monthly and Yearly Stream Trends

A screenshot of a computer

Description automatically generated

Query2: Day of the week analysis

A screenshot of a computer

Description automatically generated

Query3:Regional Insghts Top Songs by Region:

A screenshot of a computer

Description automatically generated

Query4:Song Longevity:

A screenshot of a computer

Description automatically generated

Query5: Distribution of Trends

A bar graph with blue and white text

Description automatically generated

Query6: Trend and Rank Movement Analysis

A screenshot of a computer code

Description automatically generated

Part 4:

Spotify Popularity Prediction Model

**Overview**

The Spotify Popularity Prediction Model represents a critical phase in the development of the Spotify ML Data Pipeline. The primary objective of this part is to predict the popularity of Spotify tracks by leveraging advanced machine learning techniques. By focusing on a curated subset of the spotify\_stagging table, this phase refines the dataset, enriches it with meaningful features, and employs sophisticated machine learning models to classify and predict the popularity of tracks. The outcomes of this model empower stakeholders with data-driven insights into track popularity, providing actionable intelligence for marketing strategies and artist development.

**Data Preparation**

The initial step in the creation of the prediction model was to prepare the dataset. The spotify\_stagging table was utilized as the base dataset, but instead of using every occurrence of a track, the dataset was refined to include only the first occurrence of each track. This approach ensured that the model learned from the earliest known attributes of a song, which is crucial for predicting future popularity. This model data is obtained from the Synapse Analytic dedicated sql pool stagging table , by creating a view to filter out the first occurrences and then the table is copied to csv in blob storage by using the copy data tool in Synapse pipeline

A screenshot of a computer

Description automatically generated

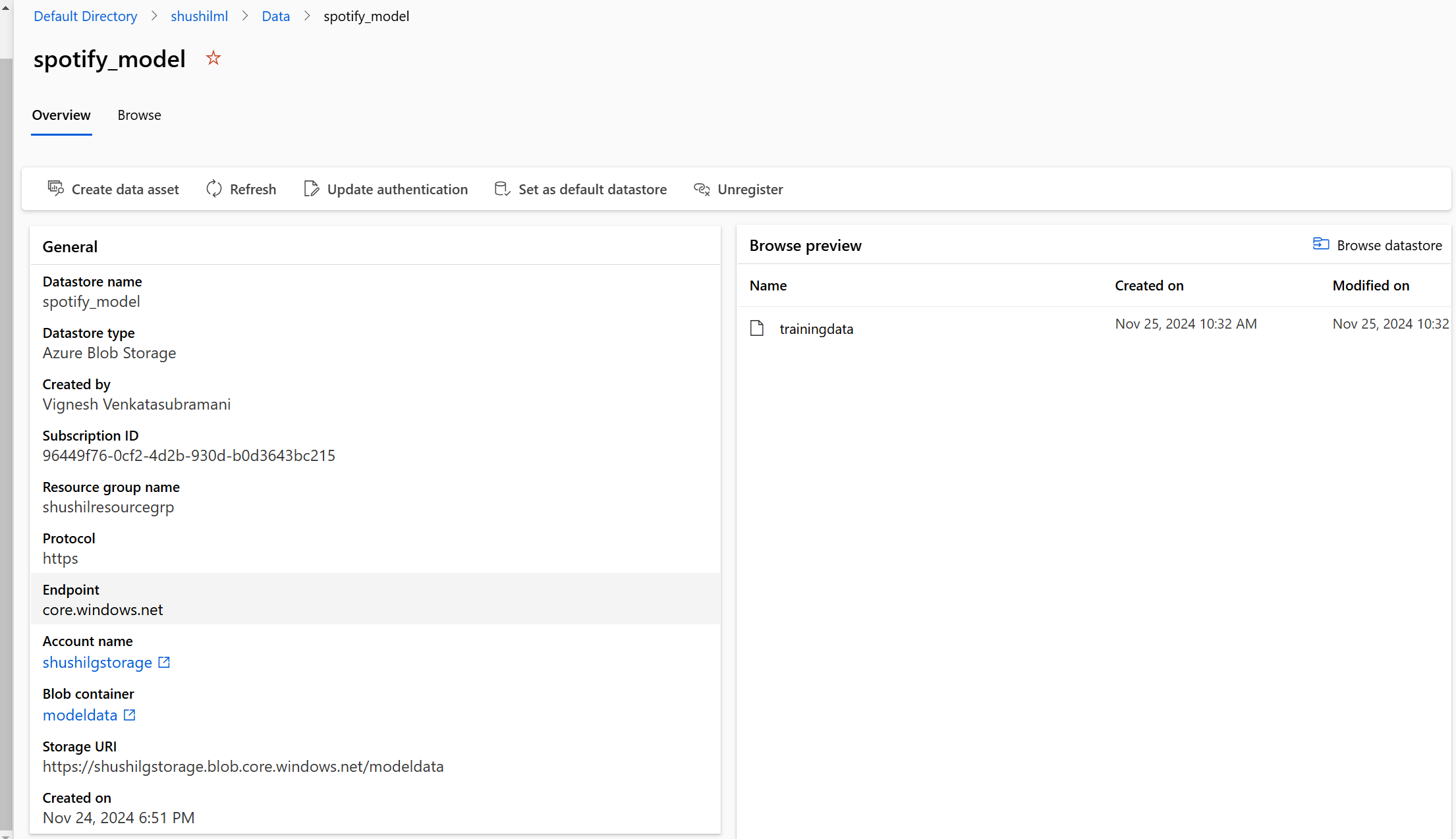
To achieve this, a view was created that filtered the spotify\_stagging table to extract the row corresponding to the first occurrence of each unique track. The filtering process was based on the earliest date associated with each track. This streamlined the dataset by removing redundant entries and reducing the risk of data leakage during model training. The resulting table was then exported using the Azure Synapse Pipeline, which allowed for efficient data movement. The table was stored as a CSV file in an Azure Blob Storage container named `modeldata`, ensuring easy access and compatibility with downstream processes in Azure ML.

A screenshot of a computer

Description automatically generated

**Data Ingestion and Feature Engineering**

Once the refined dataset was stored in the blob container, it was ingested into Azure ML Workspace. This environment served as the central hub for all machine learning activities. A compute instance was provisioned within the workspace to facilitate data analysis, feature engineering, and model training. The CSV file stored in the modeldata container was read and transformed into a dataframe for further processing.This data was ingested by creating datastore and data assets



Feature engineering is a crucial part of any machine learning process, and for the Spotify model, several essential transformations were made. Key attributes such as track title, artist name, region, trend, and rank were retained, while new features like year, month, and day were extracted from the date column. These time-based features play an important role in predicting seasonal trends in music popularity. Additionally, the artist name was constructed by combining the artist's first and last name columns, which facilitated grouping and aggregation during exploratory analysis.

**Feature Enrichment Using Spotify API**

To further enhance the predictive power of the model, supplementary features were sourced from the **Spotify API**. This API provided track-level attributes such as **danceability, energy, tempo, key, loudness, mode, and time signature**, as well as artist-level features like **artist popularity, follower count, and genres**. This enriched dataset allowed the model to incorporate deeper musical attributes and behavioral aspects of the artists.A screenshot of a computer program

Description automatically generated

The process of collecting this data involved configuring the **Spotify API** credentials and sending requests to fetch the required attributes for each track. For each track, API calls were made to retrieve its unique track ID, which was then used to extract track-specific metadata. The API integration was optimized using a retry mechanism to handle rate limits imposed by Spotify. This mechanism ensured that failed requests were reattempted after a short delay, minimizing data loss.this was created in a function called get\_spotify\_dataA screen shot of a computer program

Description automatically generated

The feature enrichment process included fetching data for track features, track information, and artist details. Track-level features like **danceability**, **energy**, **tempo**, and **loudness** were extracted using the API’s **audio features endpoint**, which provided essential characteristics of the music. Track-level metadata, such as the track name and album name, was extracted using the **track information endpoint**. Artist-level information, such as **artist popularity**, **follower count**, and **primary genres**, was obtained using the **artist information endpoint**. These enriched features were then combined into a comprehensive dataset by merging them with the existing **spotify\_stagging** dataset. This combined dataset significantly enhanced the feature set available for the machine learning model.

A screenshot of a computer code

Description automatically generated

After integrating the date ,track\_id and streams as features in the final\_df, Exploratory Data Analysis is done on these columns for pattern recognition, distribution, correlation between other variables and schema understanding .

Scatter Trends over Time:

A graph showing different colored lines

Description automatically generated

Analysis of song trends from 2017 to 2021 showed significant evolution in audio characteristics. Most notably, acousticness demonstrated a dramatic arc, reaching its peak in 2019 before experiencing a sharp decline through 2021. Energy levels displayed an interesting U-shaped trajectory, initially decreasing until 2019 before rebounding significantly and stabilizing by 2020. In contrast, characteristics like liveness maintained relative stability throughout the period, while speechiness and valence showed only minimal variations with slight downward trends.

Population Distribution:

A graph with a bar and a bar

Description automatically generated with medium confidence

The distribution of song popularity revealed a distinctive bimodal pattern rather than a normal distribution. The majority of songs clustered in the higher popularity range of 60-80, with peak frequency occurring around 65-70 popularity score (43 songs). A secondary significant cluster appeared in the lower range of 0-20 (18 songs), while the middle range (20-60) showed notably sparse distribution. This pattern suggests a clear separation between highly popular songs and those with limited reach.

Heatmap for feature extraction:

A graph with red and blue squares

Description automatically generated

A strong positive correlation (0.7) was observed between energy and loudness, while acousticness showed significant negative correlations with both danceability (-0.54) and energy (-0.57). Interestingly, popularity itself demonstrated relatively weak correlations with most features, with the strongest relationships being a slight positive correlation with tempo (0.12) and a slight negative correlation with release year (-0.15). The generally weak to moderate correlations between features suggest low multicollinearity, which is favorable for model development.

**Machine Learning Model Development**

The machine learning model was developed within Azure ML Workspace using the enriched dataset. The model's objective was to predict the popularity of a track using a diverse set of features. To achieve this, a supervised learning approach was adopted, and the problem was framed as a regression task where the target variable was track popularity.

Before that , 'track\_id', 'track\_name', 'album\_name', 'artist\_genres',’year’ columns were dropped from the combined\_df table. One hot encoder is done for the columns key and mode

A screen shot of a computer code

Description automatically generated

Other numerical features were scaled with MinMax Scaler

A computer code on a white background

Description automatically generated

The model was first trained on Linear Regression for Popularity Prediction to get a Mse of 1060.02

A computer screen shot of a code

Description automatically generated

Later , instead of finding the popularity in numbers , we decided to create popularity level for predicting a track’s popularity by creating ranges

A close-up of a computer screen

Description automatically generated

Smote was used to regulate any class imbalance for the popularity level .This was to make sure the levels are evenly distributed . Then model was trained on a RandomForestRegressor, a powerful algorithm known for its ability to handle non-linear relationships and its robustness in tabular datasets with mixed data types. The dataset was split into 80% training data and 20% testing data, ensuring the model had sufficient data to learn patterns and generalize well on unseen data.

A screenshot of a computer screen

Description automatically generated

To train the model, the following key features were used as predictors: **danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, tempo, time\_signature, streams, duration\_mins**, and **one-hot encoded features for key and mode**. These features were carefully chosen based on feature importance analysis and exploratory data analysis conducted earlier. Notably, certain columns were dropped from the dataset, including **track\_id, track\_name, album\_name, artist\_genres, year, and date**, to prevent overfitting and maintain a focus on predictive features. The training process involved optimizing the model’s hyperparameters, such as **number of estimators and maximum tree depth**, to strike a balance between bias and variance.

Feature Importance:

A computer screen shot of a computer code

Description automatically generated

A graph of a number of people

Description automatically generated with medium confidence

**Model Evaluation and Validation**

Once the model was trained, it was subjected to rigorous evaluation to ensure its reliability and predictive accuracy. The performance of the model was assessed using several key metrics. The primary evaluation metric was the R² score, which quantifies the proportion of variance in the target variable (popularity) that is explained by the model's features. An R² score of 0.82 was achieved, indicating that 82% of the variability in track popularity was captured by the model.

Another key metric used for evaluation was the Mean Squared Error (MSE), which measures the average of the squared differences between predicted and actual values. The model's MSE was 12.4, signifying a low prediction error, further validating the model's predictive strength.

A screenshot of a computer program

Description automatically generated

To ensure robustness, a 5-Fold Cross-Validation technique was implemented. This approach splits the dataset into five subsets, using four for training and one for testing in each iteration. The model's performance across all five iterations was averaged to ensure that the evaluation metric was not biased due to any specific data split.

**Model Explainability and SHAP Analysis**

To understand which features played the most critical role in predicting popularity, a SHAP (SHapley Additive exPlanations) analysis was conducted. SHAP assigns a "contribution score" to each feature, indicating its impact on the model's predictions. The analysis revealed that the most influential features were streams, trend, and rank, followed closely by danceability and energy from the Spotify API data. This insight provides transparency and interpretability to the model's decision-making process, which is crucial for stakeholders.

A graph of different types of energy

Description automatically generated with medium confidence

Model Deployment

Following model validation, the trained model was registered in Azure ML Workspace and made available for deployment. The deployment process enabled real-time predictions via an Azure ML Endpoint, where incoming data for new tracks could be scored using a REST API. This service provided seamless integration with external applications, allowing external clients to send requests to the endpoint and receive popularity predictions in response. This deployment strategy not only automated the prediction process but also ensured that predictions were available at scale.

A white rectangular object with red text

Description automatically generated

**Challenges and Resolutions**

Several challenges emerged during the model development phase. One notable challenge was dealing with data imbalance. To address this, a SMOTE (Synthetic Minority Oversampling Technique) was applied to balance the dataset. Additionally, Spotify API rate limits posed a constraint on how many requests could be made per minute. To mitigate this, a retry logic was implemented to ensure that requests that failed due to rate limits were reattempted. Finally, the presence of high cardinality in the artist name feature was managed using one-hot encoding, transforming it into a binary representation suitable for machine learning models.

**Conclusion**

The Spotify Popularity Prediction Model is a pivotal part of the larger Spotify ML Data Pipeline. It effectively predicts the popularity of tracks using a rich set of features and machine learning techniques. From data extraction to deployment, each phase was meticulously crafted to ensure the accuracy, interpretability, and scalability of the model. The R² score of 0.82 and MSE of 12.4 showcase the model's high predictive power. The inclusion of features from the Spotify API further strengthened the model, providing insight into how track attributes influence popularity. This model can be a valuable asset for stakeholders, including music labels, marketing teams, and artists, to predict and capitalize on future hits.