|  |  |
| --- | --- |
| **Shri Vishnu Engineering College for Women (Autonomous)** | |
| **Department of CSE** | |
| **Course Details** | |
| **Regulation** | **R22** |
| **Year / Semester** | **III B.Tech – II Sem** |
| **Course** | **Data Science with R Programming (Theory & Lab)** |
| **Course Code** | **UGCS6T0822** |
| **Course Type** | **Job Oriented Elective ( JOE )** |
| **Faculty** | **Y.Ramu – Department of CSE** |

|  |  |
| --- | --- |
| **Student Details** | |
| **Section** | **CSE-CS** |
| **Registered Number** | **22B01A4612** |
| **Student Name** | **Ch Sai Satvika** |
| **Mobile Number** | **8121355340** |
| **Email ID (Gmail)** | **Satvika1609@gmail.com** |

|  |  |
| --- | --- |
| **Case Study Details** | |
| **Domain** | **Music** |
| **Title of the Case Study** | **Spotify Song Classification** |
| **Tools Used** | **Python** |
| **Date of Verification** |  |
|  |  |

|  |
| --- |
| **Name of the Dataset: Song Features Dataset** |
| **Dataset URL (Active in online):** |
| **Dataset Description**:  The dataset consists of song audio features and user preferences indicating whether a song is liked or not. The goal is to classify songs into "liked" and "not liked" categories using different machine learning models. |
| **Features in Dataset: (include all feature names and their descriptions as per the information available at the source of dataset (Kaggle / UCI Data Repository etc)**   1. **danceability** - How suitable the track is for dancing (0 to 1). 2. **energy** - Intensity and activity level of a track. 3. **key** - Pitch class of the track (0-11). 4. **loudness** - Overall volume in decibels (dB). 5. **mode** - Indicates whether the track is in a major or minor key. 6. **speechiness** - Presence of spoken words in a track. 7. **acousticness** - Confidence measure of whether the track is acoustic. 8. **instrumentalness** - Predicts whether the track contains vocals. 9. **liveness** - Detects whether the track was performed live. 10. **valence** - Describes the musical positivity of the track. 11. **tempo** - BPM (beats per minute) of the track. 12. **duration\_ms** - Length of the track in milliseconds. 13. **time\_signature** - Time signature of the track. 14. **liked** - Target variable (1 = Liked, 0 = Not Liked). |
| **Number of Features in Dataset: 14** |
| **Number of Samples (records) in Dataset:** |
| **Is the dataset is having null values: Yes** |
| **Is the dataset is having missing values: Yes** |
| **Is the dataset is in encoded format of PCA values: No** |
| **Is it essential to pre-process the dataset for the case study: Yes**  **If Yes, how you want to preprocess? Give details:**   1. Handle missing values by removing or imputing them. 2. Remove duplicate rows to maintain data integrity. 3. Convert categorical columns (if any) into numerical values. 4. Normalize numerical values using StandardScaler. |
| **List out the possible opportunities for analysis on this dataset based on the available features**   * Determine which song features contribute most to user preference. * Build different classification models and compare their performance. * Analyze trends in song preferences using visualization techniques. |

|  |
| --- |
| **Title of the Case Study:**  Spotify Song Classification |
| **List of Objectives:**   1. Classify songs into "liked" and "not liked" categories. 2. Compare the performance of different machine learning models. 3. Identify key features that influence user preferences. 4. Visualize dataset trends using graphs. |
| **Approach: What features are going to be considered, processed, or feature-engineered to derive a specific outcome after applying one or more models?**   * Consider all song features as inputs, except non-numeric columns. * Apply machine learning models such as Logistic Regression, Random Forest, SVM, KNN, Decision Tree, and Naïve Bayes. * Evaluate models using accuracy, confusion matrix, and ROC curve. |

|  |
| --- |
| **Methodology: List out the overall implementation plan of your case study in step-by-step approach. (Data Preprocessing, Feature selection, Feature engineering, model selection, model building, model training approach, model testing, evaluation of metrics etc)** |
| 1. Data Preprocessing: Handle missing values, normalize data, remove duplicates. 2. Feature Selection: Retain relevant features that impact classification. 3. Model Selection: Train multiple models and compare their performance. 4. Model Training: Use train-test split (80% train, 20% test) and fit models. 5. Evaluation Metrics: Assess models with accuracy, confusion matrix, and ROC curve. |

**Case-Study Implementation**

|  |  |
| --- | --- |
| Task | Loading Dataset |
| Step-1 | Loading Dataset with name, display its descriptive information |
| Description | The dataset is loaded, and descriptive statistics are displayed. The dataset contains song-related numerical features and a binary target variable indicating whether the song is liked or not. Understanding basic statistics such as mean, standard deviation, and missing values helps in identifying necessary preprocessing steps. |
| Code | import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_curve, auc  # Load dataset  df = pd.read\_csv("data\_with\_songs.csv")  # Display first few rows  df.head()  # Check for missing values in each column  print("Missing Values: ", df.isnull().sum())  # Drop rows with missing target values (if any)  df = df.dropna(subset=["liked"])  # Remove duplicate rows  df = df.drop\_duplicates()  print("Columns in the dataset:", df.columns)  # Drop non-numeric columns  df = df.drop(columns=["song\_type"])  # Check for missing values  print("Missing Values:  ", df.isnull().sum())  # Define features (X) and target (y)  X = df.drop(columns=["liked"])  y = df["liked"] |
| Result | Missing Values:  danceability 0  energy 0  key 0  loudness 0  mode 0  speechiness 0  acousticness 0  instrumentalness 0  liveness 0  valence 0  tempo 0  duration\_ms 0  time\_signature 0  liked 0  song\_type 0  dtype: int64 |
| Description about results in detailed way | The dataset was successfully loaded, containing multiple numerical features. Initial exploration showed the presence of missing values, duplicate rows, and an unnecessary non-numeric column that needed preprocessing. |

|  |  |
| --- | --- |
| Task | Data Pre-processing |
| Step -2 | Handle Missing Values & Normalize Data |
| Description | The dataset is cleaned by handling missing values, removing duplicate rows, and standardizing numerical columns. Standardization ensures that numerical features are scaled to a common range, improving model training stability. |
| Code | # Check for missing values and duplicates  df = df.dropna(subset=["liked"])  df = df.drop\_duplicates()  # Drop non-numeric columns  df = df.drop(columns=["song\_type"], errors="ignore")  # Normalize numerical data  from sklearn.preprocessing import StandardScaler  scaler = StandardScaler()  df.iloc[:, :-1] = scaler.fit\_transform(df.iloc[:, :-1]) |
| Result | Missing Values:  danceability 0  energy 0  key 0  loudness 0  mode 0  speechiness 0  acousticness 0  instrumentalness 0  liveness 0  valence 0  tempo 0  duration\_ms 0  time\_signature 0  liked 0  dtype: int64  Columns in the dataset: Index(['danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness',  'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo',  'duration\_ms', 'time\_signature', 'liked'],  dtype='object') |
| Description about results in detailed way | The dataset was successfully preprocessed as follows:   * **Missing values** in the "liked" column were removed to ensure no empty target labels. * **Duplicate rows** were dropped to maintain data integrity. * **Non-numeric columns** (e.g., "song\_type") were removed to keep only relevant numerical features. * **Numerical features** were normalized using StandardScaler, ensuring all variables have a standardized scale, which improves model performance and training stability.   This cleaned dataset is now ready for model training. |

|  |  |
| --- | --- |
| Task | **Model Training & Evaluation** |
| Step | Train & Compare Different Models |
| Description | Multiple models were trained and evaluated for their classification performance. Random Forest and SVM performed the best, achieving the highest accuracy, while KNN and Naïve Bayes showed lower performance due to feature dependencies. |
| Code | X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Define models  models = {  "Logistic Regression": LogisticRegression(),  "Random Forest": RandomForestClassifier(n\_estimators=100, random\_state=42),  "SVM": SVC(kernel="rbf", probability=True),  "K-Nearest Neighbors": KNeighborsClassifier(n\_neighbors=5),  "Naive Bayes": GaussianNB(),  "Decision Tree": DecisionTreeClassifier(random\_state=42)  }  # Train and evaluate models  results = {}  for name, model in models.items():  model.fit(X\_train, y\_train)  y\_pred = model.predict(X\_test)  accuracy = accuracy\_score(y\_test, y\_pred)  results[name] = accuracy  print(f"{name} Accuracy: {accuracy:.4f}") |
| Result | * Logistic Regression Accuracy: 0.9231 * Random Forest Accuracy: 0.9487 * SVM Accuracy: 0.9231 * K-Nearest Neighbors Accuracy: 0.8205 * Naive Bayes Accuracy: 0.8462 * Decision Tree Accuracy: 0.8462 |
| Description about results in detailed way | The ROC curve confirmed that Random Forest had the highest AUC score, meaning it performed the best in distinguishing between liked and not liked songs. |