MACHINE LEARNING FOR DATA ANALYTICS(CS985)

Assignment Report – Classification

CS985 MLDA Group 9

Members:

HARSHAN RETHINAVELU SELVAKUMAR – 202480548

MANOJ KUMAR DHARMARAJ – 202468855

JHANSI VELURI – 202384615

PRATHUSHA PUNJARLA – 202351330

MAISAM BARKAT ALI DOHTA - 202359049

Problem Statement:

The aim of the assignment is to develop a machine learning model to predict the genre of a song using classification techniques. The test and training datasets are provided.

Libraries and models used:

Before we start execution, we need to install and load various libraries. The following libraries are required:

1. **Pandas** : This set of software is intended for data processing and evaluation. This code uses pandas' pd.read_csv() function to import a CSV file into a pandas DataFrame, and df.head() to present the initial several rows of the DataFrame.
2. **Numpy** : This library is used for numerical calculations. A scientific computer software designed for matrix calculations.
3. **Sklearn.model_selection.train_test_split :** This package is used to separate the dataset into training and testing sets. It is necessary for assessing the results of machine learning methods. The train_test_split() method is used to partition the information into traing and testing sets.
4. **Sklearn.metrics.accuracy_score :** This library includes utilities for measuring the correctness of an identification model. The accuracy_score() function is used to determine the accuracy of model.
5. **Sklearn.tree.DecisionTreeClassifier :** This library includes a decision tree classifier, an important machine learning method utilized in identification problems. In this programme, a decision tree classifier is used to create a classification system.

Explanation of the code:

Firstly, we need to download the data set from the Kaggle. To perform our task, we must first import the required libraries.

Import pandas as pd

- This line imported the pandas library and assigns it the shorthand alias ‘pd’, following a common used convention.

File_path = ‘ML2.csv’

- This line provides the dataset’s filename “ML2.csv” to a parameter called file_path. This dataset most likely involves music-related information.

df = pd.read_csv(file_path, encoding = ‘latin1’)

- The pd.read_csv() tool is employed for importing the dataset from the CSV file supplied by the file_path. This information is imported using a pandas DataFrame termed df. The encoding=’latin1’ argument is employed to deal with any unique letters that may appear in the dataset.

print(df.head())

- The following line uses the head() method to display the primary few rows of the DataFrame df. This gives a brief description about the dataset’s format and information.

df.columns = df.columns.str.strip()

- Starting and ending blank spaces are eliminated from the DataFrame df’s column titles using the str.strip()ntechnique. This maintains continuity in column names.

target_variable = ‘top genre’

- This line initiates the targeted variable’s name, ‘top genre’, for the variable target_variable. This variable is going to be applied subsequently to train and evaluate the model.

X = df.drop(columns =[‘top genre’])   # Features

- This line of code extracts the ‘top genre’ column from the initial DataFrame df, resulting in a DataFrame X contains the algorithms’s training parameters.

Y = df[‘top genre]

- Here, a sequence of values y is formed with the target variable ‘top genre’. This will ultimately be applied to train and evaluate the algorithms.

train_data, test_data = train_test_split(file_path, test_size=0.2, random_state=42)

- The following line divides the dataset into training and testing sets employing scikit-learn’s train_test_split() technique. The test_size=0.2 value indicates that 20% of the information will be utilised for testing and the remaining part for training. The argument that split is reproducible.

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

- Identical to the before line, this one divides the characteristics (x) and target variable(y) into training and testing sets for model development and verification.

print("Training set shape:", X_train.shape, y_train.shape)

print("Testing set shape:", X_test.shape, y_test.shape)

- The following lines display the shapes(dimensions) of the training and testing sets, allowing you to comprehend the amount of information required to train and test the algorithms.

from sklearn.preprocessing import LabelEncoder, standardScaler

- This line includes particular preparation techniques taken from the scikit-learn library. LabelEncoder encodes classified labels using integers, while StandardScaler scales statistical characteristics.

clf = DecisionTreeClassifier()

- The determination tree classification model is created utilising scikit-learn's Decision Tree Classifier() class. The resilting model will be developed to predict the desired variable.

clf.fit(X_train_imputed, y_train)

- The following line uses the training data(X train imputed and Y train) for develop the decision tree classifier(clf)

predictions = clf.predict(X_test_imputed)

- The trained decision tree algorithm can be utilised to develop results using the testing data(X_test_imputed), with the suggested that are expected saved in the predictions variables.

accuracy = accuracy_score(y_test, predictions)

print("Accuracy:", accuracy)

- The accuracy of selecting the tree classifier algorithm is determined by evaluating expected values (predictions) to true labels within the testing set (y_test). The accuracy score us immediately published out.

precision = precision_score(y_test, y_pred, average='micro', zero_division=1)

recall = recall_score(y_test, y_pred, average='macro', zero_division=1)

f1 = f1_score(y_test, y_pred, average='weighted', zero_division=1)

- These lines compute the accuracy, recall, and F1-score metrics to assess the effectiveness of the selection tree classifier system. Those indicators include information about the algorithm's capability to accurately categorise individuals of each class.

print("Precision:", precision)

print("Recall:", recall)

print("F1-score:", f1)

- The previously estimated precision, recall, and F1-score metrics are written down to evaluate the final outcome of the decision tree algorithm model.

print("\nClassification Report:")

print(classification_report(y_test, y_pred, zero_division=0))

- This component generates a complete classification report with precision, recall, F1-score, and assistance for every category in the target variable. It offers an increased overview of the model’s efficiency throughout classifications.

Classification Methods used in the code:

1.Decision tree classification method

The Decision Tree Classifier is an algorithm of supervised learning designed for tasks like classification. It operates by repeatedly splitting data into subgroups depending on characteristics. At every phase, it chooses the characteristic that effectively divided the data, ensuring that the categories under any subset are as consistent as feasible.Decision tree makes no assumptions regarding the arrangement of data. They can work for both numerical and quantitative data, in addition to linear and nonlinear correlations among characteristics and the target variable.

1.2. Comparision od decision tree with other classification methods

By comparing decision tree method with other classification methods such as Logistic Regression, Support Vector Machines(SVM), Random Forests, the decision tree provide a combination of understanding and flexibility. While Decision Trees are simpler to understand and visualise than sophisticated models such as SVM or GBM, they can fail to accomplish the same degree of accuracy and resilience, particularly on complicated data sets that have high density and unpredictable interactions. RandomForest, a collective approach built on top of Decision Trees, overcomes some of Decision Trees’ failings by combining several trees and decreasing excessive fitting.

2. Random Forest classifier

Random Forest often generates highly accurate forecasts. Integrating predictions from numerous decision trees decreases overfitting and variance, leading to accurate models that generalise well to new data. Random Forest is able to manage datasets with numerous attributes and instances. Random Forest can reduce the overfitting problem by using predictions from multiple trees, this leads to better selection boundaries and better generalisation accuracy. Random Forest may be constructed in parallel, making it perfect for huge database and dispersed technology platforms. Every decision tree in the collection might be developed separately, allowing for faster training periods.

2.1. Comparing Random Forest classifier with other classification method

By comparing Random Forest with Decision tree, Random Forest solves some disadvantages of decision trees, especially overfitting and large variation, by combining predictions from several trees. In overall, it outperforms an individual decision tree in terms of accuracy and generalisation. And coming to Logistic Regression, it is easy and accessible, although it might have trouble with irregular patterns in the data. Random Forest may identify complicated correlations among characteristics and the target variable, resulting in a better fit for non-linear classifying issues. SVM is good for tasks involving binary classification that can manage complicated choices limits. But SVM may necessitate thoughtful choice of hyperparameters and kernel features, and it cannot expand well to huge database. Random Forest provides an easier technique to identification that eliminates the requirement to tune sophisticated characteristics.

3.Super Vector Machine(SVM)

Support Vector Machines(SVMS) are an advanced identification technique that can manage complicated selection thresholds and data with high dimensions. SVMs feature a regularisation parameter (C) that serves to prevent overfitting. By altering the number of C, you can strike a compromise among maximising the range(decision boundary) and minimising the identification failure. This enables SVMs to accurately generalise to previously unknown datasets. SVMs are especially useful whenever a set of inputs is minimal to medium in size. These models are less likely to get overfit in certain situations and can nevertheless make precise recommendations.

3.1. Comparing SVM method with other classification methods:

Decision Trees and Random Forests provide specific categorization regulations, making them quicker to comprehend than SVMs. But they are more likely to fitted the initial data, particularly for stochastic or complex data. SVMs, on the opposite hand, offer an extra reliable remedy for overfitting by maximising the variance among groups. Logistic Regression is an easier and better understandable technique than SVMs. But Logistic Regression presupposes linear correlations among characteristics and the target variable, this might not be valid for complicated situations. SVMs, which use irregular kernel functions, may identify greater complexity interactions in datasets.

Conclusion:

In conclusion, we can conclude that the accuracy of the SVM model Accuracy: 0.8461538461
538461 and is higher than the decision Tree and Random Forest methods. However, SVM m
odel gives errors, hence it is not an advisable
method to be followed. And there is still space for enhancement across all scenarios, as see
n by average achievement indicators. Additional characteristic technology, parameter modif
ication, or research of complex models may improve prediction accuracy. It is crucial to rem
ember that outcomes may differ based on the database and characteristics employed, henc
e additional testing and refining are required for improved model accuracy.