Smart Internz

Machine Learning-Based Music Genre Classification on Spotify Data

Team 155

Team Members:

Pavan Harsha	20BCD7036
Nalluri Shweta	20BCD7038
Rampam Greeshma Geethika	20BCD7094
Sehej Soni	20BCI7100

Campus: VIT-AP

Smart Internz

Introduction:

Overview

The popularity of the music industry has led to a rapid change in the way people listen to music. The growth of music streaming services has raised the need for autonomous music classification and recommendation systems. One of the top music streaming services in the world, Spotify, has millions of users and a vast song library. However, Spotify needs to suggest songs that match users' preferences if it wants to give them a personalized music experience. Spotify guides and classifies music based on Genre using machine learning algorithms.

Purpose

This project will focus on the Spotify Genre classification problem. The main goal is to develop a model that classifies the genre that can accurately predict the genre of a music track on Spotify.

Literature Survey:

Existing problem

Spotify is a music streaming service that was established in 2006 in Sweden and later expanded to other parts of Europe and the US in 2008 and 2011. Spotify does not create original content; instead, it exclusively displays copyrighted material that has been licensed by big media companies. As music genres frequently overlap or change over time, defining a music genre can be arbitrary and fundamentally confusing. These frequent changes are sometimes difficult for machine learning methods to effectively capture, leading to incorrect classifications or just focusing on one rather than all.

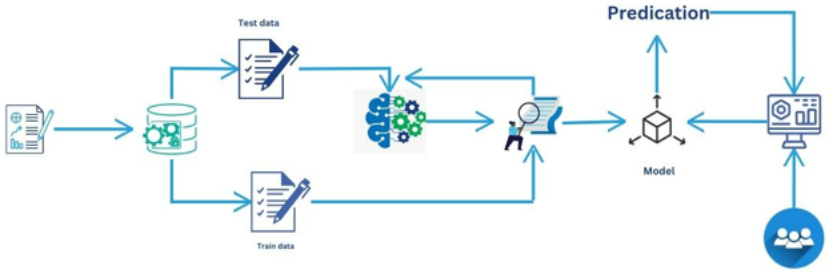
Proposed Solution

Spotify uses big data to generate personalized playlists and subscription suggestions based on customer preferences and interests. As it already has an enormous user base, Spotify is known to use the most advanced and accurate machine learning algorithms to match the taste of the already existing users as well as to attract new users.

People listen to music according to their moods, so the genre can be predicted on the basis of the current mood or the current song being played. Here we are using Random Forest and Support Vector Machine Algorithm and ensemble learning models like bagging and XgBoost to precisely and accurately predict the genre of a user. **Random Forest** takes less training time as compared to other algorithms, it predicts output with high accuracy, even for a large dataset. It can also maintain accuracy when a large proportion of data is missing. **SVM** works relatively well when there is a clear margin of separation between classes. It is more effective in high-dimensional spaces and is relatively memory efficient. For the case of our project, amongst the several algorithms used, Random forest has worked the best upon performing hyperparameter tuning. This model was used to deploy the system using Flask.

Theoretical Analysis:

Block diagram



Hardware/Software Designing

For completing the project, the hardware used was our laptops(Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz 64-bit operating system, x64-based processor.) Anaconda played a primary role, as it provided us with a jupyter notebook and spyder which enabled us to complete the project.

Experimental Investigations:

The different steps in the process of investigation include:

(i) Data set Collection: For this project, we have used the dataset provided to us by SmartInternz which includes a dataset with 22 columns having a target attribute ‘genre’.

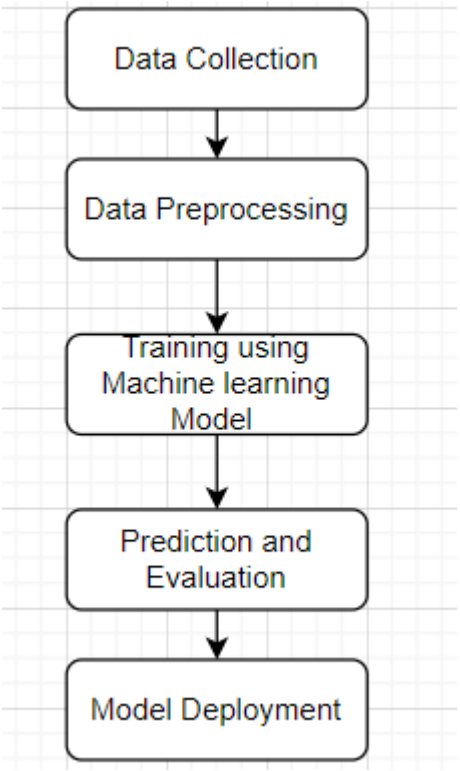
(ii) Data Preprocessing: The data was explored and exploratory data analysis was performed and various plots were generated. The presence of null values was checked. Label Encoding was done to convert the String attributes to numerical columns. SMOTE was applied to improve the accuracy. A correlation coefficient was also generated to know the dependency between the various attributes.

(iii) The data was split into independent and dependent variables which were later split into training and testing data in the ratio 80:20

(iv) Various Machine Learning algorithms were applied and the best one was selected for deployment

(v) The best model was deployed using Flask and the webpage was built which takes values from users and predicts the genre as the output.

Flowchart:



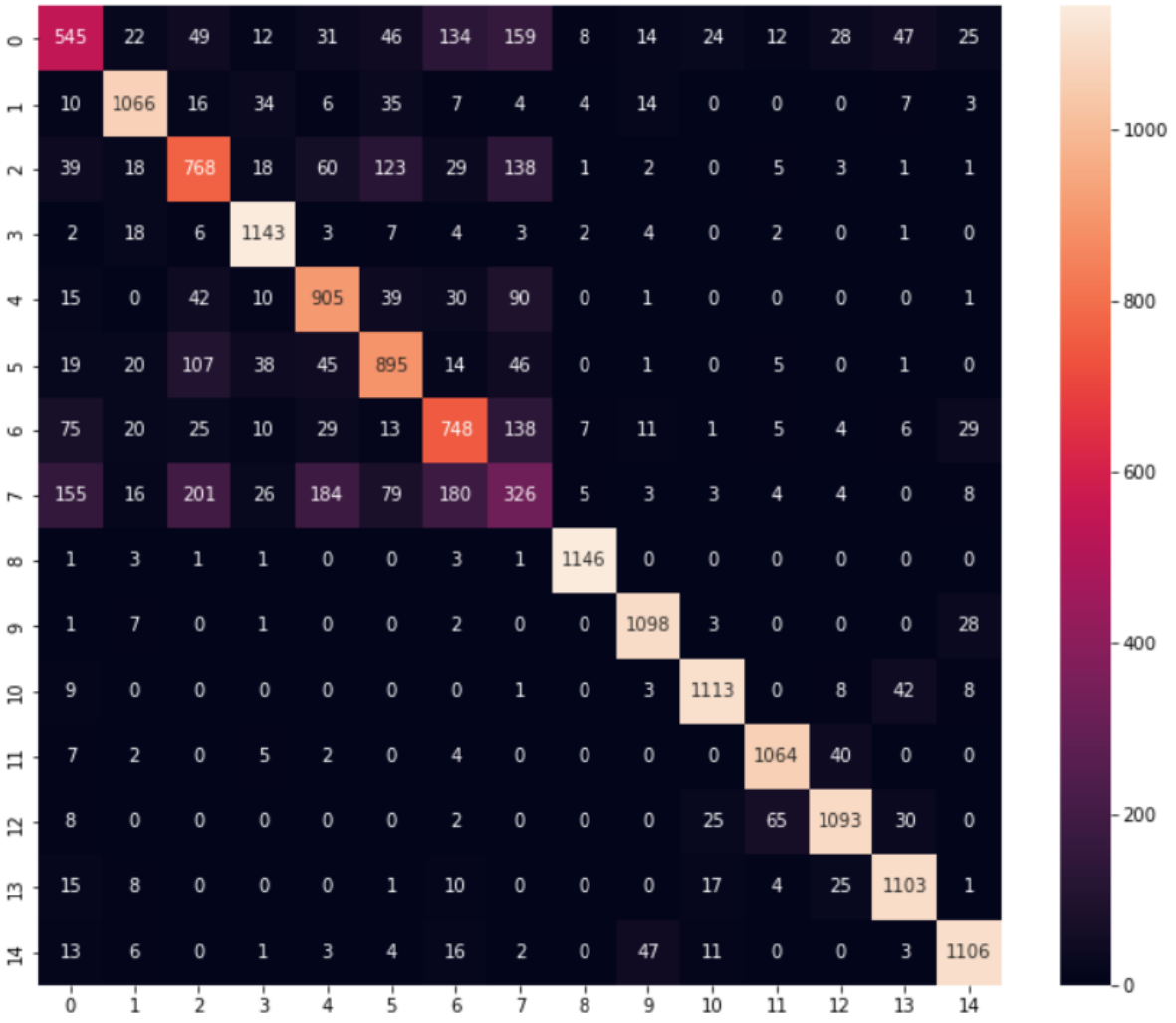
Results:

We come to a conclusion after implementing 4 different algorithms, namely SVM, Random Forest, XGBoost, and Bagging that Random Forest has performed the best.
The screenshots of the output are as follows:

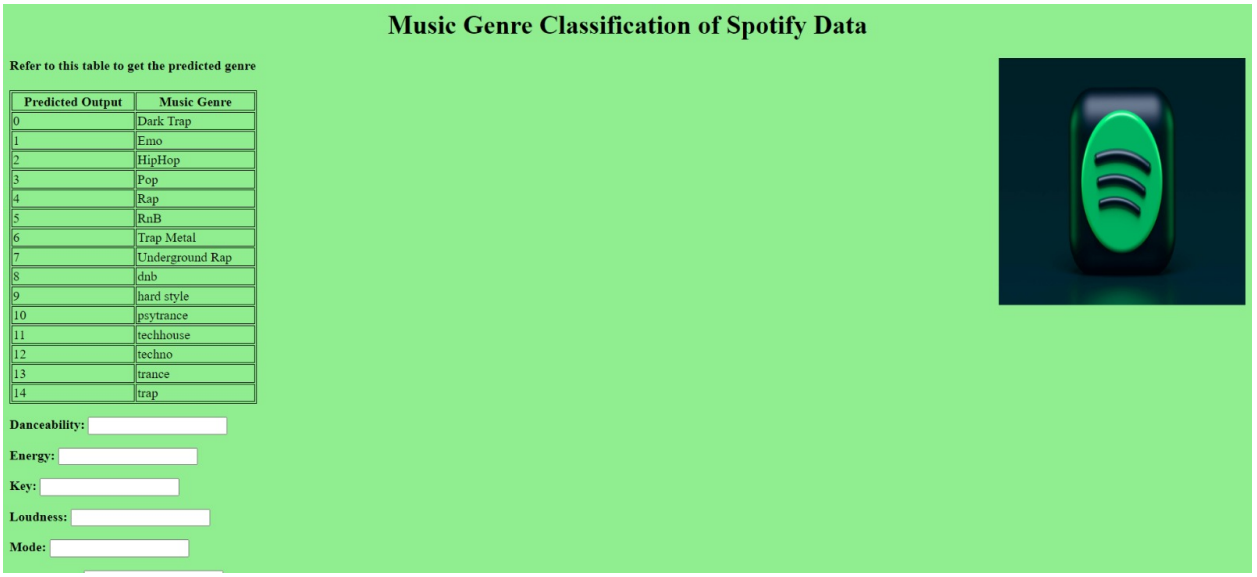
```
Best Parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 300}
Accuracy: 0.8010780141843972
```

	precision	recall	f1-score	support
0	0.60	0.47	0.53	1156
1	0.88	0.88	0.88	1206
2	0.63	0.64	0.63	1206
3	0.88	0.96	0.92	1195
4	0.71	0.80	0.75	1133
5	0.72	0.75	0.74	1191
6	0.63	0.67	0.65	1121
7	0.36	0.27	0.31	1194
8	0.98	0.99	0.98	1156
9	0.92	0.96	0.94	1140
10	0.93	0.94	0.93	1184
11	0.91	0.95	0.93	1124
12	0.91	0.89	0.90	1223
13	0.89	0.93	0.91	1184
14	0.91	0.91	0.91	1212
accuracy			0.80	17625
macro avg	0.79	0.80	0.79	17625
weighted avg	0.79	0.80	0.79	17625

```
Precision is 0.7911370299393888
Recall is 0.8010780141843972
```



Upon deployment using Flask, the output webpage looks like the following:



Advantages & Disadvantages:

ADVANTAGES

Machine learning techniques offer several advantages when it comes to music genre classification. Here are some key benefits:

- Automated Classification: Machine learning enables the automated classification of music into different genres without the need for manual annotation. This saves time and effort, especially when dealing with large music databases or streaming platforms with vast catalogs.
- Objective and Consistent Results: Machine learning algorithms apply consistent rules and criteria to classify music genres, reducing subjective biases that can arise from human classification. This ensures more objective and reliable genre classification results across different instances.
- Scalability: Machine learning algorithms can handle large volumes of music data efficiently. They can process and classify music genres at scale, making it feasible to analyze and classify vast music collections or continuously classify incoming music on streaming platforms.
- Adaptability and Flexibility: Machine learning models can adapt and improve over time by learning from new data. As more music becomes available, the models can continuously update their classification capabilities, ensuring accurate genre classification even for emerging or evolving genres.
- Feature Extraction: Machine learning models can automatically extract relevant features from music audio signals, such as tempo, spectral characteristics, rhythm patterns, and harmonic content. These features capture essential aspects of different music genres, facilitating accurate classification.
- Multimodal Classification: Machine learning techniques can combine audio features with other modalities, such as textual metadata, album art, or user

listening behavior, to enhance genre classification accuracy. This multimodal approach provides a more comprehensive understanding of the music and its genre characteristics.

- Genre Discovery and Exploration: Machine learning can help discover and explore new or niche music genres by analyzing patterns and similarities in the music data. This can lead to the discovery of emerging trends or subgenres that may not be immediately apparent to human classifiers.
- Personalization and Recommendation: Accurate genre classification powered by machine learning can improve personalized music recommendations. By understanding users' genre preferences and behavior patterns, recommendation systems can suggest relevant songs or playlists that align with their music tastes.

It's worth noting that machine learning models for music genre classification have limitations. They may struggle with specific ambiguous or hybrid genres, and the quality of classification heavily relies on the availability and quality of labeled training data. However, overall, machine learning offers significant advantages in automating and enhancing the music genre classification process.

DISADVANTAGES

While machine learning brings numerous benefits to music genre classification, there are also some potential disadvantages to consider:

- Subjectivity and Ambiguity: Music genre classification can be subjective and inherently ambiguous, as genres often overlap or evolve over time. Machine learning models may struggle to capture these nuances accurately, leading to misclassifications or oversimplifications of complex genres.
- Lack of Contextual Understanding: Machine learning models primarily rely on pattern recognition from audio features and metadata. They may not fully understand the cultural, historical, or contextual aspects that shape genres. As a result, the models may miss important contextual cues and make inaccurate genre classifications.
- Bias in Training Data: Machine learning models heavily rely on training data, which can introduce bias if it is not diverse or representative enough. If the training dataset is skewed towards certain genres or lacks diversity in cultural perspectives, the models may exhibit bias in genre classification.
- Inability to Capture Individual Preferences: Music genre preferences can be highly subjective and vary significantly among individuals. Machine learning models may struggle to capture the individual nuances and complexities of personal music tastes, leading to less accurate genre recommendations for specific users.
- Difficulty with Emerging or Niche Genres: Machine learning models rely on existing labeled data for training. If there is a lack of labeled data for emerging or niche genres, the models may struggle to classify or recognize these genres accurately, as they may not have sufficient examples to learn from.
- Limited Adaptability to Genre Evolution: Genres constantly evolve and blend with one another. Machine learning models may face challenges in adapting quickly to emerging genres or capturing the nuances of genre fusion, as they

typically require retraining with updated datasets to incorporate new genre characteristics.

- Computational Demands: Training and deploying machine learning models for music genre classification can require significant computational resources, especially for large-scale music databases. This can be a limitation in terms of computational power, storage, and processing time for real-time or resource-constrained applications.
- Lack of Interpretability: Some machine learning models, such as deep neural networks, can be considered black boxes, making it challenging to interpret their decision-making process. This lack of interpretability can limit the transparency and understandability of genre classification results.

It's important to address these disadvantages by incorporating human expertise, considering diverse perspectives, continuously updating and evaluating models, and ensuring responsible and ethical use of machine learning in music genre classification.

Applications:

Machine learning-based music genre classification on Spotify data can be implemented in various areas and applications. Some notable areas include:

- Music Recommendation Systems: Music genre classification can be used to improve the accuracy and personalization of music recommendation systems on Spotify. By analyzing the genre of a user's listening history and preferences, the system can suggest similar songs or artists from the same genre or even recommend songs from different genres to diversify the user's music taste.

- Playlist Generation: Machine learning algorithms can classify songs into different genres and use this information to create playlists that cater to specific moods, activities, or genres. For example, an algorithm can generate a "Relaxing Jazz" playlist by selecting songs classified as jazz and with a low tempo.

- Music Tagging and Organization: Machine learning algorithms can automatically tag and organize music tracks in a user's library based on their genre. This can help users easily search and navigate their music collection based on specific genres, allowing for efficient music management.

- Music Streaming Quality Enhancement: Machine learning can be used to improve the quality of music streaming by classifying songs into different genres and adapting the audio streaming parameters accordingly. For example, the streaming quality for a classical music track can be optimized differently than that for a hip-hop track, considering the specific characteristics of each genre.

- Music Analysis and Insights: Machine learning algorithms can be applied to analyze Spotify's music data to gain insights into music trends, patterns, and

preferences across different genres. This can be valuable for music industry professionals, researchers, and marketers to understand user behavior, popularity trends, and genre-specific dynamics.

- Music Genre Classification for Copyright and Licensing: Accurate genre classification can be crucial for copyright and licensing purposes. Machine learning algorithms can help classify songs into specific genres, making it easier for copyright agencies, music labels, and artists to manage and protect their intellectual property.

These are just a few examples of the areas where machine learning-based music genre classification on Spotify data can be implemented. The potential applications are broad and diverse, highlighting the versatility and usefulness of such systems in the music industry.

Conclusion:

In conclusion, utilizing the research and modeling performed in this study, we were able to categorize Spotify music genres with an accuracy of 88%. This is a respectable level of accuracy given the difficulty and subjectivity involved in categorizing musical genres. However, there's always room for improvement, and our analysis has certain drawbacks.

One drawback is the need for more diversity in our Dataset, particularly in terms of Spotify's selection of rap and hip-hop music. This favored particular genres in our study and modeling. The Dataset must include a greater range of musical subgenres to enhance our model.

Another restriction is the possibility of human error during data classification, which may have led to inconsistencies in genre categorization. We might make use of more advanced techniques, including

Future Scope:

The main thing to identify and divide the audio into different features is amplitude and frequency which changes within a short span of time. We can visualize the audio frequency wave of amplitude and frequency with respect to time in the form of a wave plot that can be easily plotted using librosa. We have restricted the project to machine learning. We can take it up a notch and make it challenging by using deep learning algorithms and also go with hybridization.

Bibliography:

(i) poonia, sahil & verma, chetan & Malik, Nikita. (2022). Music Genre Classification using Machine Learning: A Comparative Study. 13. 15-21.

(ii) Hareesh Bahuleyan. 2018. Music genre classification using machine learning techniques. arXiv preprint arXiv:1804.01149(2018).

(iii) Snigdha Chillara, AS Kavitha, Shwetha A Neginhal, Shreya Haldia, and KS Vidyullatha. 2019. Music genre classification using machine learning algorithms: a comparison. Int Res J Eng Technol 6, 5 (2019), 851–858

(iv) J. Bergstra; N. Casagrande; D. Erhan; D. Eck; B. Kégl. Aggregate features and ADABOOST for music classification.*Machine Learning*, 65(2–3):473–484, 2006.

(v) Brunner G, Konrad A, Wang Y, Wattenhofer R. Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer. In: 19th International Society for Music Information Retrieval Conference (ISMIR 2018) 2018.

Appendix:

SOURCE CODE:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
from sklearn import metrics
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.decomposition import PCA, KernelPCA, TruncatedSVD
from sklearn.manifold import Isomap, TSNE, MDS
import random
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis
from matplotlib import rcParams
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from itertools import product
import time
%matplotlib inline


data=pd.read_csv('C:/Users/Shweta/Downloads/genres_v2.csv')
Data
data.head()
data.info()
data.describe()
Data.shape
data.columns
df=data.drop(['type','id','uri','track_href','analysis_url','duration_ms','time_signature'
```

```
df.isnull()
df.isnull().sum()
df['genre'].value_counts()

#EDA
import plotly.express as px
px.histogram(df.genre)

plt.boxplot(df['danceability'])

plt.hist(df.liveness)

plt.figure(figsize = (13, 8))
plt.xticks(rotation=25)
sns.barplot(data=df, x='genre', y='danceability')
plt.show()

plt.figure(figsize = (13, 8))
plt.xticks(rotation=25)
sns.barplot(data=df, x='genre', y='energy')
plt.show()

plt.figure(figsize = (13, 8))
plt.xticks(rotation=25)
sns.barplot(data=df, x='genre', y='speechiness')
plt.show()

df.hist(bins=80)
plt.figure(figsize=(10,10))
plt.show

df1=df.drop(['genre'],axis=1)
corr_mat = df1.corr()
plt.figure(figsize = (15, 15))
sns.heatmap(corr_mat,
            annot=True,
            fmt='.2f',
            cmap=plt.cm.coolwarm_r,
           )
plt.show()

#Encoding
from sklearn.preprocessing import LabelEncoder
l = LabelEncoder()
df['genre']=l.fit_transform(df['genre'])

#Splitting the data
X=df.drop("genre",axis=1)
```

```
y=df["genre"]
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_norm=scaler.fit_transform(X)

#SMOTE
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_r, y_r = smote.fit_resample(X_norm, y)

#Splitting
X_train,X_test,y_train,y_test=train_test_split(X_r,y_r,test_size=0.2,random_state=25)

#SVM
from sklearn.svm import SVC
svm_classifier = SVC(kernel='rbf',C=1100,gamma='scale')
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y_test,y_pred))
from sklearn.metrics import precision_score,recall_score
print("Precision is", precision_score(y_test,y_pred,average='weighted'))
print("Recall is", recall_score(y_test,y_pred,average='weighted'))

# XGBoos
from sklearn.model_selection import GridSearchCV
dtrain = xgb.DMatrix(X_train, label=y_train)

# Define the XGBoost classifier
xgb_classifier = xgb.XGBClassifier()

# Define the parameter grid for grid search
param_grid = {
   'max_depth': [3, 5, 7],
   'learning_rate': [0.1, 0.01, 0.001],
   'n_estimators': [100, 200, 300],
}

# Perform grid search
grid_search = GridSearchCV(estimator=xgb_classifier, param_grid=param_grid,
scoring='accuracy', cv=3)
grid_search.fit(X_train, y_train)

# Get the best parameters and model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
```

```
# Make predictions on the test set using the best model
y_pred = best_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Print the best parameters and accuracy
print("Best Parameters:", best_params)
print("Accuracy:", accuracy)

#Random forest with hyperparameter tuning
from sklearn.model_selection import GridSearchCV
param_grid = {
   'n_estimators': [100, 200, 300],  # Number of trees in the forest
   'max_depth': [None, 5, 10],  # Maximum depth of the trees
    'min_samples_split': [2, 5, 10]  # Minimum number of samples required to split an internal node
}

# Create the Random Forest classifier
rf_classifier = RandomForestClassifier()

# Perform Grid Search with cross-validation
grid_search = GridSearchCV(rf_classifier, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Get the best parameters and best model from Grid Search
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

# Make predictions on the test set using the best model
y_pred = best_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Print the best parameters and accuracy
print("Best Parameters:", best_params)
print("Accuracy:", accuracy)

from sklearn.metrics import confusion_matrix,classification_report
con = confusion_matrix(y_test, y_pred)
fig = plt.subplots(figsize=(12, 10))
ax = sns.heatmap(con, annot=True, fmt="d")
print(classification_report(y_test,y_pred))

from sklearn.metrics import precision_score,recall_score
print("Precision is", precision_score(y_test,y_pred,average='weighted'))
```

```
print("Recall is", recall_score(y_test,y_pred,average='weighted'))

#Bagging
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

# Define the base classifier
base_classifier = DecisionTreeClassifier()

# Define the BaggingClassifier
bagging_classifier = BaggingClassifier(base_classifier)

# Define the hyperparameters to tune
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_samples': [0.5, 0.7, 0.9],
    'max_features': [0.5, 0.7, 0.9]
}

# Perform grid search with cross-validation
grid_search = GridSearchCV(bagging_classifier, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Get the best parameters and best estimator
best_params = grid_search.best_params_
best_estimator = grid_search.best_estimator_

# Evaluate the best estimator
y_pred = best_estimator.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Best Parameters:", best_params)
print("Accuracy:", accuracy)

from sklearn.metrics import confusion_matrix,classification_report
con = confusion_matrix(y_test, y_pred)
fig = plt.subplots(figsize=(12, 10))
ax = sns.heatmap(con, annot=True, fmt="d")

print(classification_report(y_test,y_pred))
from sklearn.metrics import precision_score,recall_score
print("Precision is", precision_score(y_test,y_pred,average='weighted'))
print("Recall is", recall_score(y_test,y_pred,average='weighted'))

import pickle
pickle.dump(best_model,open("model.pkl","wb"))
```