Beta Try the new code view

main

Music-Genre / FinalCode.ipynb

Parthib25 Created using Colaboratory History

1 contributor

623 lines (623 sloc) | 20 KB

Open in Colab

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
df=pd.read_csv("features_30_sec.csv")
data=df.iloc[:,2:-1]
labels=df.iloc[:,-1]
from sklearn import tree
```

In [3]:
```
x_train,x_test,y_train,y_test=train_test_split(data,labels,test_size=0.28,random_stat
```

In [4]:
```
import pydot
def model_to_png(model, file_name):
  graph_data = tree.export_graphviz(model)
  graph = pydot.graph_from_dot_data(graph_data)
  graph[0].write_png(file_name)
```

In [5]:
```
model=tree.DecisionTreeClassifier(random_state=100,max_depth=3)
model.fit(x_train,y_train)
model_to_png(model,'model3.png')
```

In [6]:
```
model=tree.DecisionTreeClassifier(random_state=100,max_depth=5)
model.fit(x_train,y_train)

model_to_png(model, 'model5.png')
```

In [7]:
```
model=tree.DecisionTreeClassifier(random_state=100,max_depth=10)
model.fit(data,labels)
model_to_png(model,'model10.png')
```

In [8]:
```
model=tree.DecisionTreeClassifier(random_state=100,max_depth=15)
model.fit(x_train,y_train)
model_to_png(model, 'model15.png')
```

In [9]:
```
df_to_train=x_train[["perceptr_var","rolloff_var","spectral_bandwidth_mean","mfcc4_me
df_to_test=x_test[["perceptr_var","rolloff_var","spectral_bandwidth_mean","mfcc4_mean
```

In [10]:
```
from sklearn.metrics import accuracy_score

clf=RandomForestClassifier(random_state=79)
clf.fit(df_to_train,y_train)
y_pred=clf.predict(df_to_test)
acc = accuracy_score(y_test, y_pred)

print("Accuracy:", acc)
```

```
Accuracy: 0.5107142857142857
```

In [22]:
```
df_to_train=x_train[["perceptr_var","rolloff_var","spectral_bandwidth_mean","mfcc4_me
df_to_test=x_test[["perceptr_var","rolloff_var","spectral_bandwidth_mean","mfcc4_mean
```

In [23]:
```
from sklearn.metrics import accuracy_score

clf=RandomForestClassifier(random_state=79)
clf.fit(df_to_train,y_train)
y_pred=clf.predict(df_to_test)
acc = accuracy_score(y_test, y_pred)

print("Accuracy:", acc)
```

```
Accuracy: 0.5928571428571429
```

In [24]:
```
df_to_train=x_train[["perceptr_var","rolloff_var","spectral_bandwidth_mean","mfcc4_me
df_to_test=x_test[["perceptr_var","rolloff_var","spectral_bandwidth_mean","mfcc4_mean
```

In [25]:
```
from sklearn.metrics import accuracy_score

clf=RandomForestClassifier(random_state=79)
clf.fit(df_to_train,y_train)
y_pred=clf.predict(df_to_test)
acc = accuracy_score(y_test, y_pred)

print("Accuracy:", acc)
```

```
Accuracy: 0.6357142857142857
```

In [16]:
```
from sklearn.feature_selection import VarianceThreshold
from sklearn.model_selection import GridSearchCV
selector = VarianceThreshold(threshold=0.01)
X_selected = selector.fit_transform(x_train)
indices = selector.get_support(indices=True)[:5]
X_top5 = X_selected[:, indices]
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
rf = RandomForestClassifier(random_state=79)
grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_top5, y_train)
print("Best parameters:", grid_search.best_params_)
print("Best accuracy score:", grid_search.best_score_)
```

```
Best parameters: {'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 1, 'min_sample
s_split': 10, 'n_estimators': 100}
Best accuracy score: 0.4805555555555555
```

In [19]:
```
x_selected_test=selector.transform(x_test)
indices = selector.get_support(indices=True)[:5]
X_top5 = x_selected_test[:, indices]
```

```
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
rf = RandomForestClassifier(random_state=79)
grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_top5, y_test)
print("Best parameters:", grid_search.best_params_)
print("Best accuracy score:", grid_search.best_score_)
```

```
Best parameters: {'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 1, 'min_sample
s_split': 5, 'n_estimators': 100}
Best accuracy score: 0.4821428571428571
```

In [17]:

```
X_selected = selector.fit_transform(x_train)
indices = selector.get_support(indices=True)[:10]
X_top5 = X_selected[:, indices]
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
rf = RandomForestClassifier(random_state=79)
grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_top5, y_train)
print("Best parameters:", grid_search.best_params_)
print("Best accuracy score:", grid_search.best_score_)
```

```
Best parameters: {'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 1, 'min_sample
s_split': 5, 'n_estimators': 100}
Best accuracy score: 0.5708333333333334
```

In [20]:

```
x_selected_test=selector.transform(x_test)
indices = selector.get_support(indices=True)[:10]
X_top5 = x_selected_test[:, indices]
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
rf = RandomForestClassifier(random_state=79)
grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_top5, y_test)
print("Best parameters:", grid_search.best_params_)
print("Best accuracy score:", grid_search.best_score_)
```

```
Best parameters: {'bootstrap': False, 'max_depth': 10, 'min_samples_leaf': 2, 'min_sampl
es_split': 5, 'n_estimators': 50}
Best accuracy score: 0.5142857142857142
```

In [18]:

```
X_selected = selector.fit_transform(x_train)
indices = selector.get_support(indices=True)[:15]
```

```
X_top5 = X_selected[:, indices]
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
rf = RandomForestClassifier(random_state=79)
grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_top5, y_train)
print("Best parameters:", grid_search.best_params_)
print("Best accuracy score:", grid_search.best_score_)
```

```
Best parameters: {'bootstrap': False, 'max_depth': 10, 'min_samples_leaf': 1, 'min_sampl
es_split': 5, 'n_estimators': 100}
Best accuracy score: 0.6069444444444445
```

In [21]:

```
x_selected_test=selector.transform(x_test)
indices = selector.get_support(indices=True)[:15]
X_top5 = x_selected_test[:, indices]
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [3, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
rf = RandomForestClassifier(random_state=79)
grid_search = GridSearchCV(rf, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_top5, y_test)
print("Best parameters:", grid_search.best_params_)
print("Best accuracy score:", grid_search.best_score_)
```

```
Best parameters: {'bootstrap': False, 'max_depth': 10, 'min_samples_leaf': 1, 'min_sampl
es_split': 2, 'n_estimators': 50}
Best accuracy score: 0.5642857142857143
```

In [30]:

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
selector = SelectKBest(score_func=mutual_info_classif, k=5)
X_new = selector.fit_transform(x_train, y_train)
rfc = RandomForestClassifier(n_estimators=100, random_state=79)
rfc.fit(X_new, y_train)
X_new_test=selector.transform(x_test)
# evaluate performance on test set
accuracy = rfc.score(X_new_test, y_test)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.4857142857142857
```

In [31]:

```
selector = SelectKBest(score_func=mutual_info_classif, k=10)
X_new = selector.fit_transform(x_train, y_train)
rfc = RandomForestClassifier(n_estimators=100, random_state=79)
rfc.fit(X_new, y_train)
X_new_test=selector.transform(x_test)
# evaluate performance on test set
accuracy = rfc.score(X_new_test, y_test)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.5785714285714286
```

In [32]:

```
selector = SelectKBest(score_func=mutual_info_classif, k=15)
X_new = selector.fit_transform(x_train, y_train)
rfc = RandomForestClassifier(n_estimators=100, random_state=79)
rfc.fit(X_new, y_train)
X_new_test=selector.transform(x_test)
# evaluate performance on test set
accuracy = rfc.score(X_new_test, y_test)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.5892857142857143
```

In [33]:

```
    #The accuracy in the test set that we have achieved for the
    #three different pipelines is listed as follows:

    #The Decision tree :
   # 5 features:51.0714
   # 10 features:59.2857
   # 15 features:63.5714

   # The VarianceTreshold and Grid Search:
   # 5 features:48.214
   # 10 features:51.42857
   # 15 features:56.428

   # The SelectKbest Feature selection using mutual_info_classif

   # 5 features:48.5714
   # 10 features:57.85714
   # 15 features:58.92857

#The study thus show the best way to do feature selection for music genre
```