■ Lung Cancer Risk Prediction - Machine Learning Code

Author: | B.Tech CSE (Data Science), MITS

```
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 GradientBoostingClassifier
from sklearn.metrics import classification_report, roc_auc_score, accuracy_score, roc_curve, con

# Load the dataset (replace 'lungcancer.csv' with your actual file)
data = pd.read_csv('lungcancer.csv')

# Encode categorical data if necessary
data['GENDER'] = data['GENDER'].map({'M': 1, 'F': 0})
data['LUNG_CANCER'] = data['LUNG_CANCER'].map({'YES': 1, 'NO': 0})

# Define features and target variable
X = data.drop(columns=['LUNG_CANCER'])
y = data['LUNG_CANCER']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the classifier
model = GradientBoostingClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions and evaluate the model
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Print classification metrics
print("Accuracy:", accuracy_score(y_test, y_pred))
print("AUC Score:", roc_auc_score(y_test, y_pred_proba))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Plot ROC Curve with different colors
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_test, y_pred_proba):.2f}", color='blue', linew
plt.plot([0, 1], [0, 1], linestyle='--', color='red', linewidth=1.5, label='Random Guess')
plt.fill_between(fpr, tpr, color='lightblue', alpha=0.3)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.show()

# Plot Confusion Matrix Heatmap with distinct colors
```

```
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='coolwarm', xticklabels=['No', 'Yes'], ytickl
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Feature importance
features = X.columns
feature_importances = model.feature_importances_
importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances}).sort_valu

plt.figure(figsize=(10, 6))
colors = sns.color_palette("viridis", len(importance_df))  # Generate distinct colors for each b
sns.barplot(x='Importance', y='Feature', data=importance_df, palette=colors)
plt.title('Feature Importance')
plt.xlabel('Importance Score')
plt.ylabel('Features')
plt.grid(color='gray', linestyle='--', linewidth=0.5, alpha=0.7)
plt.show()
```