Detect Lung Cancer using patient diagnosis data

Objective

The goal of this project is to develop a machine learning model that predicts the likelihood of lung cancer in patients based on various medical and lifestyle factors. The project involves **data preprocessing, exploratory data analysis (EDA), model training, and performance evaluation** using different classification algorithms, including **Support Vector Classifier (SVC), Random Forest Classifier, and Decision Tree Classifier**.

Dataset Description

The dataset used in this project contains information about patients, including their **demographic details, medical conditions, and lifestyle choices**. The key attributes include:

- **GENDER**: Male (0) / Female (1)
- **AGE**: Patient's age
- **SMOKING**: Smoking habit (0 - No, 1 - Yes)
- **YELLOW_FINGERS**: Presence of yellow fingers due to smoking (0 - No, 1 - Yes)
- **ANXIETY**, **PEER_PRESSURE**, **CHRONIC DISEASE**, **FATIGUE**, **ALLERGY**, **WHEEZING**, **ALCOHOL CONSUMING**, **COUGHING**, **SHORTNESS OF BREATH**, **SWALLOWING DIFFICULTY**, **CHEST PAIN**: Various medical and behavioral factors (0 - No, 1 - Yes)
- **LUNG_CANCER**: Target variable (0 - No, 1 - Yes)

Methodology

1. Data Preprocessing

- **Dataset Import & Initial Exploration**: The dataset was loaded using *pd.read_csv('dataset_med.csv')* and examined using *df.head(), df.tail(), df.shape,* and *df.dtypes*.
- **Handling Missing & Duplicate Data**:
    - Missing values were checked using *df.isnull().sum().*
    - Duplicate records were identified using *df.duplicated().sum()* and removed with *df.drop_duplicates(inplace=True).*
- **Data Encoding**:
    - The *GENDER* column was converted from categorical ('M' and 'F') to numerical (0 and 1) using *df['GENDER']=df['GENDER'].replace(['M', 'F'],[0,1]).*
    - The *LUNG_CANCER* column was converted from categorical ('YES' and 'NO') to numerical (1 and 0) using *df['LUNG_CANCER']=df['LUNG_CANCER'].replace(['YES', 'NO'],[1,0]).*

2. Exploratory Data Analysis (EDA)

- **Basic Statistical Insights**:
    - Summary statistics of numerical features were obtained using *df.describe().*
    - Categorical data was analyzed using *df.describe(include="object").*
- **Feature Distributions & Target Analysis**:
    - **Target Variable (*LUNG_CANCER*)**:
        - Count distribution plotted using *sns.countplot(x=df['LUNG_CANCER']).*
        - Pie chart representation:

```
plt.pie(df.groupby(by=["LUNG_CANCER"]).size(), labels=df["LUNG_CANCER"].unique(), autopct="%0.2f")
plt.title('Lung Cancer Distribution')
plt.show()
```

    - **Feature-wise Analysis**:
        - **Gender Distribution**: *plt.pie(df.groupby(by=["GENDER"]).size(), labels=df["GENDER"].unique(), autopct="%0.2f")*
        - **Age Distribution**: *sns.distplot(df['AGE'])*
        - **Alcohol Consumption vs. Lung Cancer**:

```
sns.countplot(x='ALCOHOL CONSUMING', hue="LUNG_CANCER", data=df, palette='Pastel1')
```

        - **Smoking vs. Lung Cancer**:

```
sns.countplot(x='SMOKING', hue="LUNG_CANCER", data=df, palette='Pastel1')
```

        - **Chronic Disease vs. Lung Cancer**:

```
sns.countplot(x='CHRONIC DISEASE', hue="LUNG_CANCER", data=df, palette='Pastel1')
```

    - **Correlation Analysis**:
        - A heatmap was generated to visualize feature correlations using *sns.heatmap(df.corr(), annot=True, cbar=True, cmap='RdYlGn', fmt='.1f').*

3. Model Training and Evaluation

Train-Test Split

- The dataset was split into training and testing sets using *train_test_split(),* with 80% training data and 20% testing data:

```
X = df.drop('LUNG_CANCER', axis=1)
y = df['LUNG_CANCER']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

A. Support Vector Classifier (SVC)

- Model trained using a linear kernel:

```
svc = SVC(kernel='linear')
svc.fit(X_train, y_train)
```

- Accuracy score: *svc.score(X_train, y_train)*
- Predictions & evaluation:

```
svc_pred = svc.predict(X_test)
svc_acc = accuracy_score(y_test, svc_pred)
print(classification_report(y_test, svc_pred))
```

- Confusion matrix visualization:

```
cm1 = confusion_matrix(y_test, svc_pred)
sns.heatmap(cm1, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

B. Random Forest Classifier

- Model trained with **100 estimators**:

```
RF_model = RandomForestClassifier(n_estimators=100)
RF_model.fit(X_train, y_train)
```

- Accuracy score: *RF_model.score(X_train, y_train)*
- Predictions & evaluation:

```
RF_pred = RF_model.predict(X_test)
RF_acc = accuracy_score(y_test, RF_pred)
print(classification_report(y_test, RF_pred))
```

- Confusion matrix visualization:

```
cm2 = confusion_matrix(y_test, RF_pred)
sns.heatmap(cm2, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

C. Decision Tree Classifier

- Model trained using default parameters:

```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

- Accuracy score: *dt.score(X_train, y_train)*
- Predictions & evaluation:

```
dt_pred = dt.predict(X_test)
DT_acc = accuracy_score(y_test, dt_pred)
print(classification_report(y_test, dt_pred))
```

- Confusion matrix visualization:

```
cm3 = confusion_matrix(y_test, dt_pred)
sns.heatmap(cm3, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

4. Model Comparison

A comparison of model performance was conducted using a DataFrame, ranking models based on their accuracy scores:

```
models = pd.DataFrame({
    'Model': ['Random Forest', 'SVC', 'Decision Tree'],
    'Accuracy': [RF_acc, svc_acc, DT_acc]
})
models.sort_values(by='Accuracy', ascending=True)
```

Model	Accuracy Score
Random Forest	**RF_acc**
Support Vector Classifier (SVC)	**svc_acc**
Decision Tree	**DT_acc**

The Random Forest Classifier outperformed the other models, making it the best choice for lung cancer prediction.

Conclusion

This project successfully applied **machine learning** techniques to predict lung cancer. **Random Forest** provided the best accuracy. Future improvements may include **hyperparameter tuning**, **feature selection**, and **deep learning models** for enhanced performance.