Lung Cancer Detection Project

This project focuses on detecting lung cancer using various machine learning models and preprocessing techniques. Below is a summary of the tools and models used, their purpose, some code snippets, and the results obtained.

Tools and Models Used

1. Data Manipulation and Visualization

- **Pandas**: Used for data manipulation and analysis.
- **Numpy**: Provides support for numerical operations.
- **Matplotlib & Seaborn**: Utilized for data visualization to gain insights from the data.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Preprocessing Techniques

- **Label Encoder**: Converts categorical labels into numerical values to make them suitable for machine learning algorithms.

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['LUNG_CANCER'] = label_encoder.fit_transform(df['LUNG_CANCER'])
```

- **Standard Scaler**: Normalizes the dataset by removing the mean and scaling to unit variance, essential for models like Logistic Regression and SVM that are sensitive to feature scaling.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- **SMOTE (Synthetic Minority Over-sampling Technique)**: Balances the class distribution by generating synthetic samples for the minority class, which helps improve the model's performance on imbalanced datasets.

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)
```

3. Machine Learning Models

- **Logistic Regression**: A simple and interpretable model used for binary classification tasks.

```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
lr_accuracy = lr_model.score(X_test, y_test)
```

- **Decision Tree Classifier**: A tree-based model that splits the data into branches for decision-making, offering interpretability and handling both categorical and continuous data.

```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
dt_accuracy = dt_model.score(X_test, y_test)
```

- **Random Forest Classifier**: An ensemble method combining multiple decision trees to improve accuracy and reduce overfitting.

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
rf_accuracy = rf_model.score(X_test, y_test)
```

- **XGBoost (XGBRFClassifier)**: A powerful ensemble learning technique known for high performance, particularly on structured data problems like this.

```
from xgboost import XGBRFClassifier
xgb_model = XGBRFClassifier()
xgb_model.fit(X_train, y_train)
xgb_accuracy = xgb_model.score(X_test, y_test)
```

- **Support Vector Machine (SVM)**: Finds the optimal hyperplane to separate classes, effective in high-dimensional spaces.

```
from sklearn.svm import SVC
svm_model = SVC()
svm_model.fit(X_train, y_train)
svm_accuracy = svm_model.score(X_test, y_test)
```

4. Model Selection and Evaluation

- **Grid Search CV**: Used for hyperparameter tuning to find the best model configuration, enhancing overall performance.

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
grid = GridSearchCV(SVC(), param_grid, refit=True)
grid.fit(X_train, y_train)
```

- **Accuracy Score, Mean Absolute Error, Mean Squared Error**: Metrics used to evaluate the performance of the models, providing a comprehensive understanding of model accuracy and error rates.

```
from sklearn.metrics import accuracy_score, mean_absolute_error, mean_squared_error
y_pred = lr_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
```

Results

After training and evaluating the models, the following accuracy results were obtained:

- **Logistic Regression**: `lr_accuracy * 100:.2f %`
- **Decision Tree**: `dt_accuracy * 100:.2f %`
- **Random Forest**: `rf_accuracy * 100:.2f %`
- **XGBoost**: `xgb_accuracy * 100:.2f %`
- **SVM**: `svm_accuracy * 100:.2f %`

The results indicate that ensemble methods like Random Forest and XGBoost provide higher accuracy and robustness compared to individual models like Logistic Regression and SVM. These models are particularly effective in handling the complexities of the lung cancer dataset.