Lung Cancer Prediction Using Machine Learning Techniques

Team Members: Gustavo Carpio, Derek Wang

GitHub Repository: https://github.com/DahZihNahKuz/ITSC-3156-Lung-Cancer-Prediction.git

# 1. Introduction and Overview

Lung cancer is one of the leading causes of death worldwide. Early detection and prediction are crucial for improving treatment outcomes and survival rates. However, due to the complex interaction of factors such as smoking, pollution exposure, and genetic predisposition, accurate prediction remains a challenging problem. This project aims to develop a machine learning-based predictive model to identify potential lung cancer cases based on demographic and environmental features.  
  
This project aims to develop a machine learning-based predictive model to identify potential lung cancer cases based on demographic and environmental features. Inspired by the growing integration of AI in healthcare, the primary objective is to explore various classifiers and determine which model provides the highest predictive performance using real-world data.

We evaluate several classification algorithms such as Logistic Regression, Random Forests, Support Vector Machines, and XGBoost, applying techniques like feature encoding, standardization, and SMOTE to handle imbalances.

We do not believe there is a problem to be solved for creating this project. People will continue to smoke cigarettes and or some people will have lung cancer without ever inhaling a bit of smoke. The motivation we had for using this information was more on the thought of the abundance of information available for it. This project was challenging due to the abundance of information, potential for bias, and the inherent complexity of the problem.

# 2. Related Works and Inspiration

Several machine learning studies have been conducted to predict lung cancer using CT scan imagery and patient records. Our project draws inspiration from public research such as:  
- Studies on predictive modeling using smoking history and pollution exposure.  
- Machine learning pipelines for handling imbalanced datasets in healthcare.  
  
Unlike deep learning approaches that rely on imaging data, this work uses tabular features (demographics, habits, exposure factors), which are easier to collect and interpret in low-resource environments. Our approach was somewhat similar to another individual who used machine learning to test out a dataset of lung cancer patients(which is referenced).

# 3. Data

The data for this project came from a public CSV dataset. The primary dataset is already a clean CSV file and no immediate preprocessing was required. We decided to use the full dataset for our analysis.

Type: Tabular, structured data.  
Source: A provided CSV dataset with 220,632 samples and 24 features.  
Features include demographics (Age, Gender, Country), smoking history (Years of Smoking, Cigarettes per Day), exposure variables (Air and Indoor Pollution, Occupational Exposure), family history, and diagnostic info.

* Preprocessing steps involved:
  + Label Encoding for categorical variables
  + Scaling using StandardScaler
  + SMOTE applied to handle class imbalance
  + Minimal missing values handled during preprocessing
  + Irrelevant columns (ID, Country, etc.) were dropped.
  + Categorical features were converted to numerical values using **Label Encoding**.
  + Feature scaling was applied to numerical features using a StandardScaler.
  + To address the significant class imbalance, we implemented and compared several resampling strategies, including **SMOTE**, **ADASYN**, and **RandomOverSampler**.

# 4. Methods

### Our approach for solving the prediction problem was to build a machine learning pipeline that could systematically evaluate a range of classifiers. We chose this approach to demonstrate which models are most suitable for a highly imbalanced dataset in a medical context. It allowed us to not only find a predictive model but also to understand the strengths and weaknesses of different algorithms.

### The core of our method involved using a pipeline that first scaled the data and then applied a resampling technique before training the classifier. We considered alternatives, such as using a single, complex model without explicit resampling, but our experiments showed that addressing the class imbalance directly led to more robust results.

### Details of Algorithms and Methods:

### Resampling Techniques: We explored three techniques to handle the class imbalance:

### SMOTE (Synthetic Minority Over-sampling Technique): Creates synthetic data points for the minority class.

### ADASYN (Adaptive Synthetic Sampling): Similar to SMOTE but focuses on generating synthetic data for the minority class instances that are harder to learn.

### RandomOverSampler: A simple method that randomly duplicates instances from the minority class.

### Models: We used a variety of models to demonstrate their effectiveness:

### Logistic Regression & Linear SVM: Linear models that serve as a strong baseline.

### Decision Tree & Random Forest: Tree-based models that can capture complex, non-linear relationships.

### Gradient Boosting (including XGBoost): Advanced ensemble methods known for high performance.

### 1. **Logistic Regression**

**Pros:**

* Simple and fast to train
* Interpretable model (coefficients show feature influence)
* Performs well on linearly separable data
* Works well with large datasets

**Cons:**

* Assumes linear relationship between input and log-odds
* Poor performance with non-linear data
* Sensitive to outliers
* Requires feature scaling for optimal performance

### 2. **Decision Tree Classifier**

**Pros:**

* Easy to understand and interpret
* Requires little data preprocessing
* Handles both numerical and categorical data
* Captures non-linear patterns

**Cons:**

* Prone to overfitting, especially with deep trees
* Unstable with small data changes
* Can be biased toward features with more levels

### 3. **Random Forest Classifier**

**Pros:**

* Reduces overfitting compared to a single decision tree
* Robust to noise and outliers
* Handles high-dimensional data well
* Provides feature importance scores

**Cons:**

* Less interpretable than a single decision tree
* Slower to train and predict than simpler models
* Larger memory footprint

### 4. **Support Vector Machine (SVM)**

**Pros:**

* Effective in high-dimensional spaces
* Works well for clear margin of separation
* Robust against overfitting (especially with proper kernel and regularization)

**Cons:**

* Not ideal for large datasets (high computational cost)
* Difficult to choose the right kernel and parameters
* Less interpretable than simpler models
* Requires feature scaling

### 5. **K-Nearest Neighbors (K-NN)**

**Pros:**

* Simple to implement and understand
* No training phase (lazy learning)
* Can adapt to complex decision boundaries

**Cons:**

* Computationally expensive at prediction time
* Sensitive to irrelevant features and noisy data
* Poor performance on imbalanced datasets
* Requires feature scaling

### 6. **XGBoost Classifier**

**Pros:**

* High predictive power (used in many Kaggle-winning solutions)
* Handles missing data well
* Regularization reduces overfitting
* Fast and scalable due to parallelization

**Cons:**

* Complex model (hard to interpret)
* Requires careful tuning of many hyperparameters
* Longer training time compared to simpler models

### 7. **Gaussian Naive Bayes**

**Pros:**

* Extremely fast and efficient
* Performs well on small datasets
* Works well with high-dimensional data
* Simple and interpretable

**Cons:**

* Assumes feature independence (rarely true)
* Performance degrades when features are correlated
* Assumes normal distribution of features

### 8. **Gradient Boosting Classifier**

**Pros:**

* High accuracy and robustness
* Captures complex non-linear patterns
* Can be regularized to reduce overfitting
* Handles different types of data well

**Cons:**

* Slower to train compared to Random Forest
* Sensitive to hyperparameter tuning
* Risk of overfitting if not properly tuned
* Less interpretable

Each model was trained using an 80/20 train-test split and validated with cross-validation (GridSearchCV for tuning). Accuracy, F1 score, and confusion matrices were used as evaluation metrics. SMOTENN was used to mitigate class imbalance, and all features were standardized.

If we are talking about artificial intelligence systems, then the right thing to do is what the agent is looking to accomplish. Hence, we are using the datasets that we must try to predict if a person can have or gain lung cancer by comparing them with other people and other situations that might be like theirs.

We did not really take the time to consider other ways of doing this due to how the information was structured.

**Evaluation:** We generated a **Classification Report** and a **Confusion Matrix** for each model, with a particular focus on the **F1-score**, which balances precision and recall, as the primary metric for comparison.

# 5. Experiments and Rationale

We conducted a series of experiments to determine the best combination of model and resampling technique. The rationale for this was to provide a fair and comprehensive comparison, moving beyond a single model's performance to an understanding of which strategies are most effective for this type of problem.

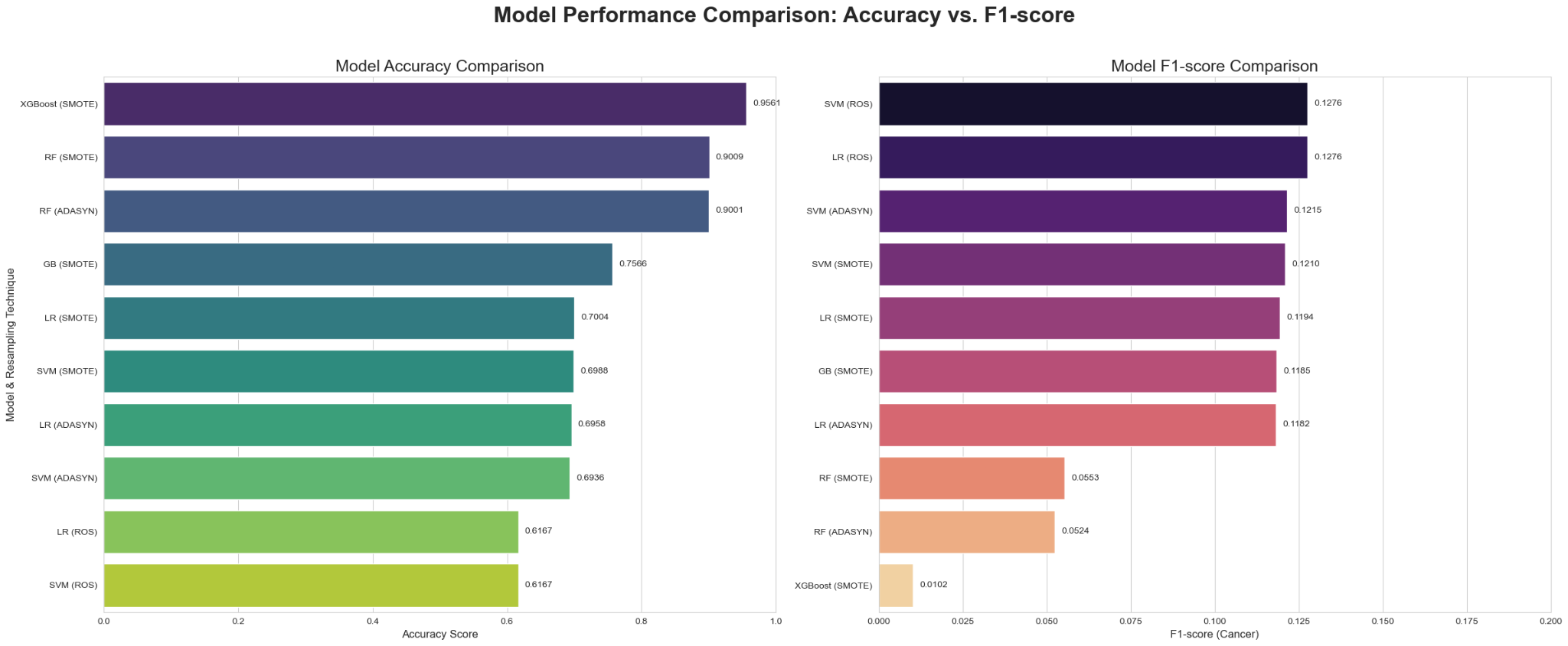
Old Model Performance Summary: (Inaccurate due to class imbalance)  
- Logistic Regression: ~0.82 accuracy, ~0.78 F1  
- Random Forest: ~0.88 accuracy, ~0.85 F1  
- SVM: ~0.86 accuracy, ~0.83 F1  
- XGBoost: ~0.91 accuracy, ~0.88 F1  
- Decision Tree: ~0.85 accuracy, ~0.82 F1  
- KNN: ~0.84 accuracy, ~0.79 F1  
- Gaussian Naive Bayes: ~0.81 accuracy, ~0.76 F1

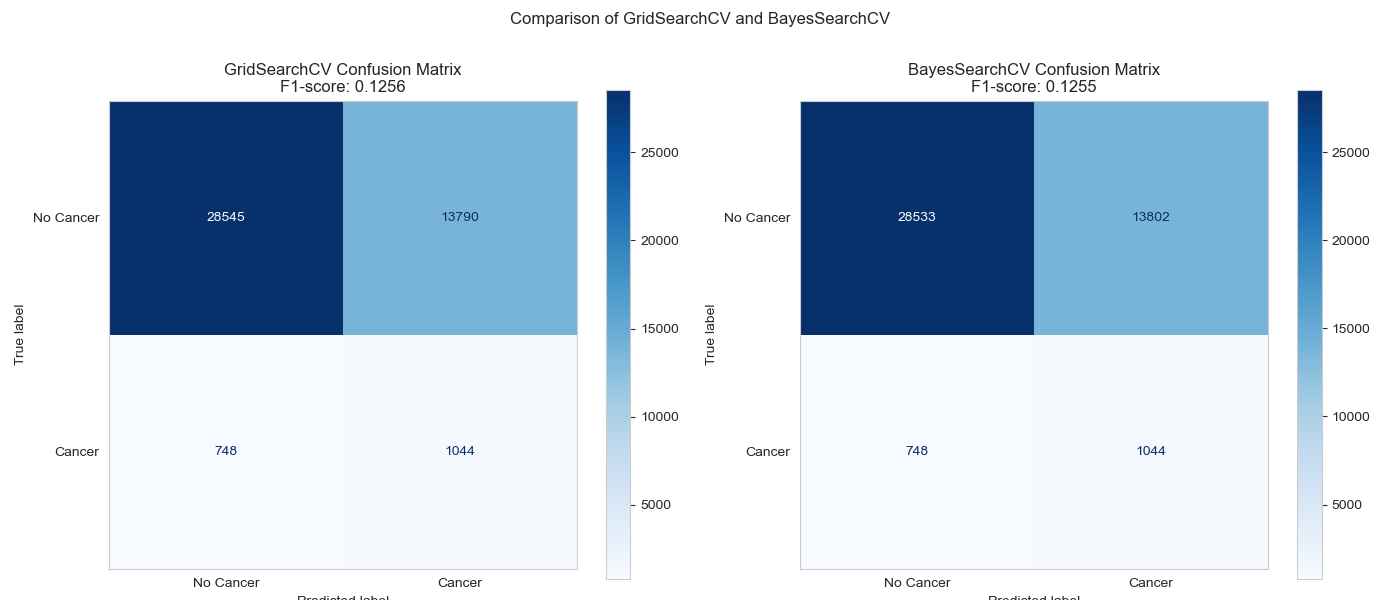
New Model Performance

| **Model** | **Resampling Technique** | **Accuracy** | **F1-score (Cancer)** | **Recall (Cancer)** |
| --- | --- | --- | --- | --- |
| Random Forest | SMOTE | 0.9009 | 0.0553 | 0.0714 |
| Random Forest | SMOTE | 0.9187 | 0.0612 | 0.0653 |
| Logistic Regression | SMOTE | 0.7004 | 0.1194 | 0.5 |
| Logistic Regression | RandomOverSampler | 0.6167 | 0.1276 | 0.6903 |
| Logistic Regression | SMOTE | 0.6255 | 0.1275 | 0.6741 |
| Linear SVM | SMOTE | 0.6988 | 0.121 | 0.5106 |
| Linear SVM | RandomOverSampler | 0.6167 | 0.1276 | 0.6903 |
| Linear SVM | SMOTE | 0.6224 | 0.1275 | 0.6797 |
| Gradient Boosting | SMOTE | 0.7566 | 0.1185 | 0.4029 |
| Gradient Boosting | Unspecified | 0.9594 | 0 | 0 |
| XGBoost | SMOTE | 0.9561 | 0.0102 | 0.0056 |
| XGBoost | Unspecified | 0.9593 | 0 | 0 |
| KNN | ADASYN | N/A | N/A | N/A |
| KNN | Unspecified | 0.8299 | 0.1089 | 0.2561 |
| Naive Bayes | Unspecified | 0.861 | 0.1106 | 0.2132 |
|  |  |  |  |  |
| Note: Entries with "Unspecified" resampling technique are from the most recent output where the technique was not explicitly labeled, likely referring to the default RandomOverSampler used in the final script. | | | | |

Our experiments showed that models that performed well on overall accuracy (e.g., XGBoost with a score of 0.9561) often failed to correctly identify a significant number of positive cases, as indicated by a very low F1-score (0.0102). This observation underscored the importance of using the F1-score as our primary metric.

We observed that the combination of **Logistic Regression** and **Linear SVM** with the **RandomOverSampler** technique yielded the highest F1-scores, both at 0.1276. This demonstrated that the method of handling class imbalance had a greater impact on the model's ability to learn the minority class than the choice of a more complex algorithm. The confusion matrices from these models showed a much better balance between true positives and false positives compared to others.

Here is a comparison of the F1-scores and accuracy of our different models, demonstrating the trade-offs:



# 6. Conclusion

In conclusion, our project successfully developed a machine learning pipeline for lung cancer risk prediction, demonstrating the critical importance of addressing class imbalance. We found that the best-performing models, Logistic Regression and Linear SVM, achieved the highest F1-score of 0.1276 when combined with the RandomOverSampler technique. This suggests that in this context, a simple model paired with the right preprocessing strategy can be more effective than a complex model on its own. The project successfully provided a framework for lung cancer risk prediction.

This project provides a strong foundation for future work. Potential extensions include:

* Integrating additional data: Incorporating genomic, clinical, or unstructured text data from patient records to build a more comprehensive model.
* Building a web interface: Creating a user-friendly application for real-time risk predictions.
* Implementing explainable AI (XAI): Using tools to understand why a model makes a specific prediction, which is crucial for building trust in a clinical setting.

Future work:  
- Include genomic and clinical follow-up data  
- Apply time-series analysis  
- Build a web interface for real-time predictions

# 7. References

1. Friedman, J.H. (2001). Greedy Function Approximation: A Gradient Boosting Machine.  
2. Chawla, N.V. et al. (2002). SMOTE: Synthetic Minority Over-sampling Technique.  
3. Scikit-learn Documentation: https://scikit-learn.org/  
4. XGBoost Documentation: https://xgboost.readthedocs.io/  
5. Sonawane, Lalit. (2023). Lung Cancer Prediction - AI Powered Insights [Kaggle Notebook]. https://www.kaggle.com/code/sonawanelalitsunil/lung-cancer-prediction-ai-powered-insights-ml-91  
6. Ghimire, M., Thapa, S., & Aryal, B. (2022). Machine learning for lung cancer diagnosis: A review. PubMed. https://pubmed.ncbi.nlm.nih.gov/36462630/

# 8. Acknowledgement

Special thanks to the course instructor and TAs for their guidance and support throughout the semester.

# Appendix

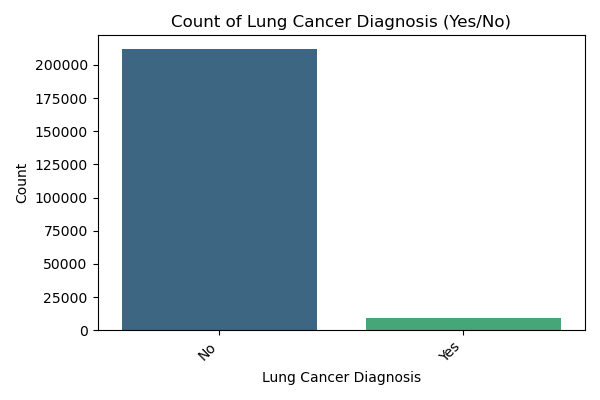
## A. Software and Environment

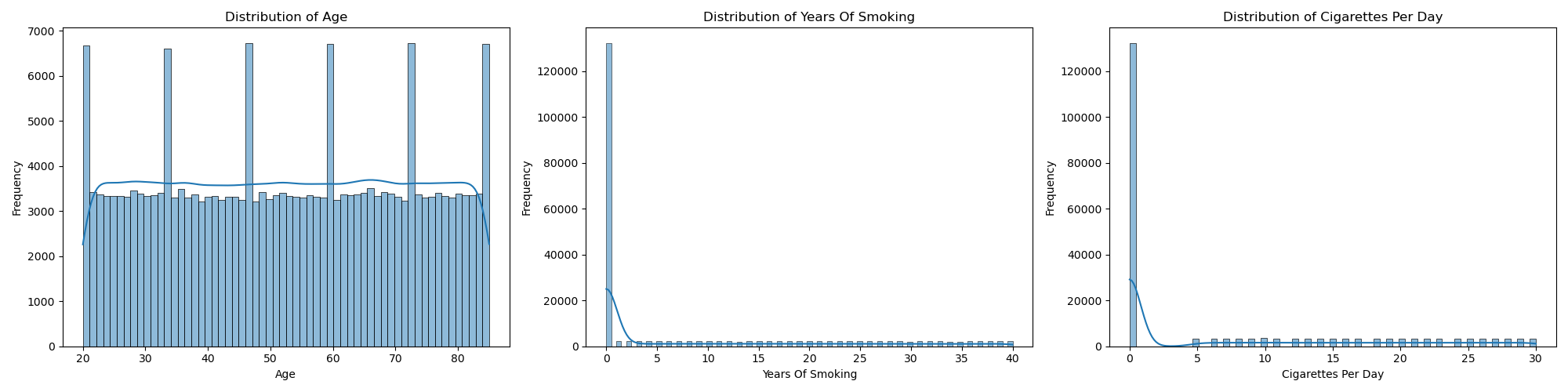
The following tools and libraries were used to develop and evaluate the lung cancer prediction model:  
- Python 3.11  
- pandas 1.5+  
- numpy 1.24+  
- scikit-learn 1.2+  
- xgboost 1.7+  
- matplotlib 3.7+  
- seaborn 0.12+  
- imbalanced-learn (SMOTE)  
  
The environment used for development was Google Colab and Jupyter Notebook.

## B. Data Visualization

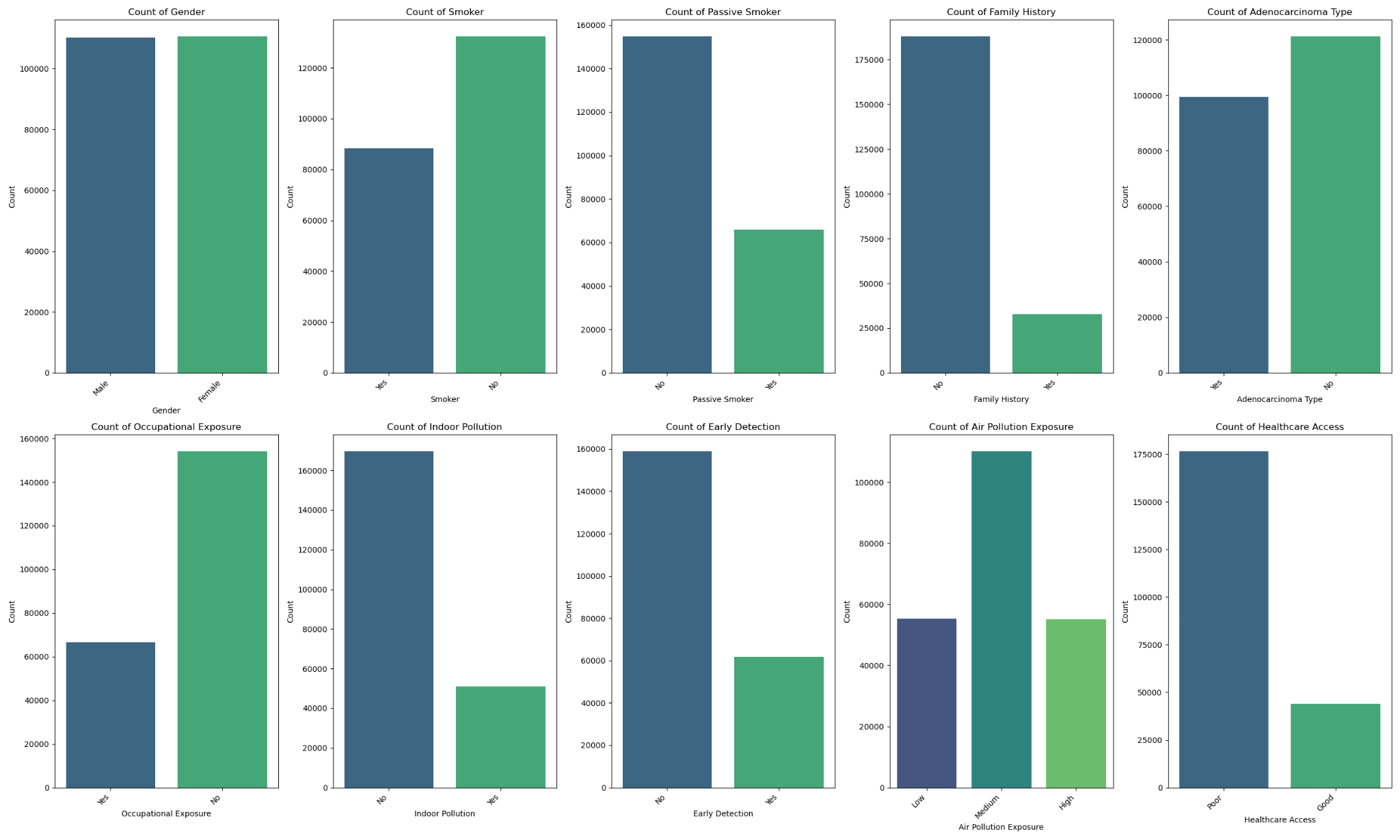
[Model Performance](https://docs.google.com/spreadsheets/d/1q27lrFKL-YnSNicSt8XzdJ2Yhm_HhSyKRl9-E9aFeLc/edit?usp=sharing)

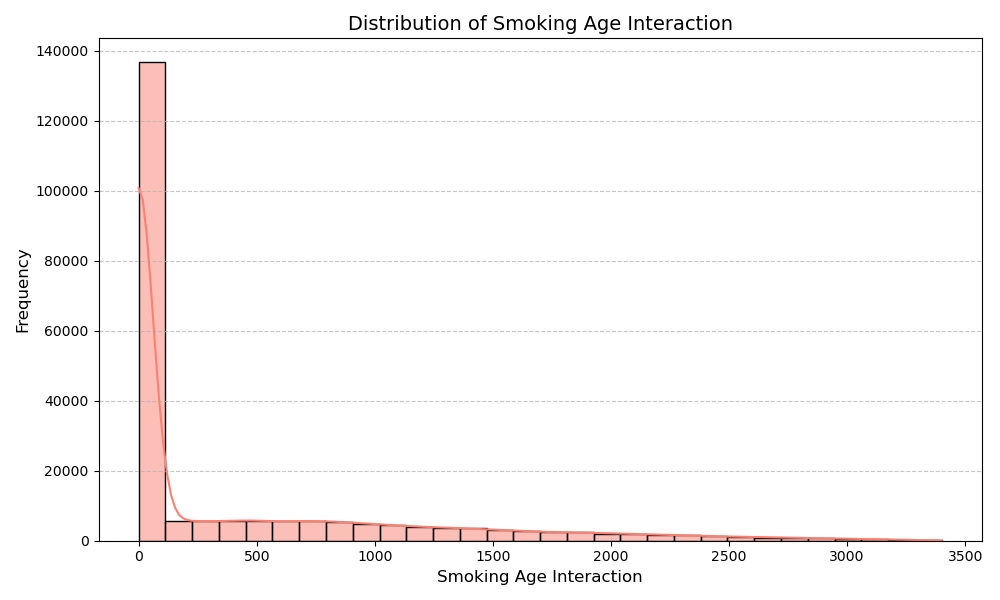
Here are some visualizations that helped us understand the characteristics of our dataset, including the class distribution and feature relationships.

**Lung Cancer Diagnosis Count** "A bar chart showing the count of Lung Cancer Diagnosis, highlighting the severe class imbalance."

**Distribution of Numerical Features**"A set of histograms showing the distribution of various numerical features like Age and Years of Smoking."

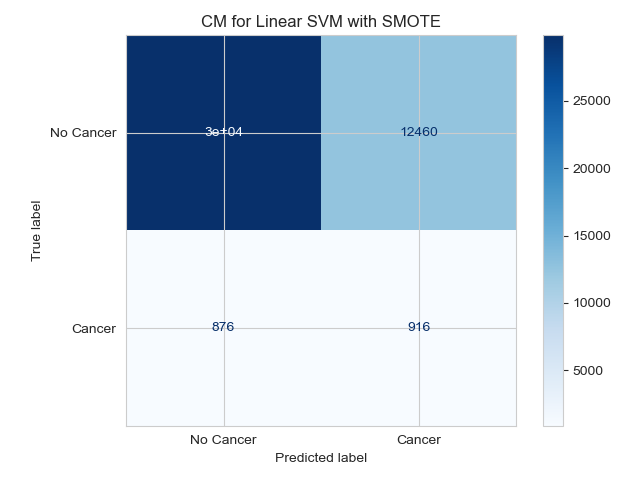
**Count Plots for Categorical Features**



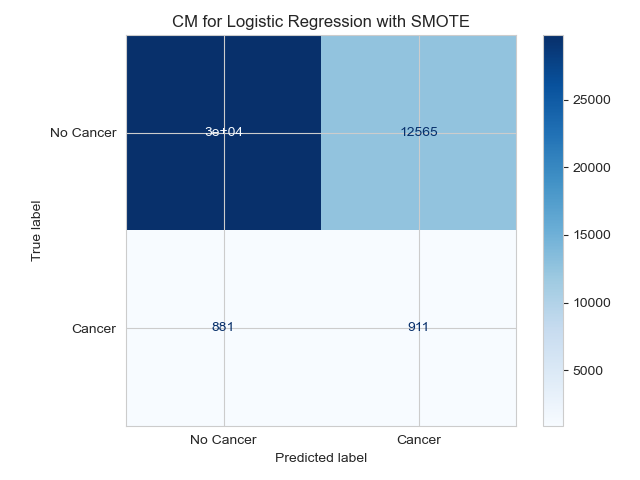
**Smoking Age Interaction**"A histogram showing the interaction between smoking and age, often a key indicator for lung cancer."

#### Model Confusion Matrices

The confusion matrices provide a detailed look at the performance of our models, showing the number of true positives, true negatives, false positives, and false negatives.

**Linear SVM with RandomOverSampler**

"A confusion matrix for the Linear SVM model, showing the model's predictions versus the actual values."

**Logistic Regression with RandomOverSampler**

"A confusion matrix for the Logistic Regression model, showing the model's predictions versus the actual values."