Project Title	Tobacco Use and Mortality, 2004-2015
Tools	Jupyter Notebook and VS code
Technologies	Machine learning , Python
Domain	Data Science
Project Difficulty level	Advanced

PHASE 01: Data Loading and Cleaning

Objective: The project involves multiple health and socio-economic datasets (admissions, fatalities, metrics, smoking prevalence, and prescriptions), it is essential to clean, align, and standardize these datasets before modeling.

Datasets Used:

Dataset	File	Description
Admissions	`admissions.csv`	Contains yearly hospital admission counts for smoking-related diseases.
Fatalities	`fatalities.csv`	Records annual fatalities attributable to smoking by disease and sex.
Metrics	`metrics.csv`	Includes socio-economic indicators such as Tobacco Price Index, Retail Price Index, and Household Expenditure.
Smokers	`smokers.csv`	Reports smoking prevalence rates by year, gender, and age group.

Prescriptions	`prescriptions.csv`	Summarizes prescription totals for nicotine replacement therapy and smoking cessation products.

a) Import Libraries : Imported Pandas and NumPy — essential Python libraries for handling tabular data, cleaning, and numerical computation.

b) Load CSV Files: Each CSV file is read into a DataFrame (df_adm, df_fat, etc.). This separates different data sources for modular processing and easier debugging.

c) Define Helper Functions: Two custom functions are defined to standardize data formats across all datasets.

i. `extract_year(x)`:This function ensures that only the first year (2014) is extracted as an integer. Invalid or missing entries are safely converted to `NaN`.

ii. `coerce_numeric(s)`:

- Cleans non-numeric placeholders (like `"."` or empty strings).
- Removes commas in large numbers (e.g., "12,345" → "12345").
- Converts all values to numeric (`float` type).
- Ensures consistent numerical format for analysis and modeling.

d) Normalize Year Across All DataFrames : Ensures that all data sources use a standardized 'Year' column for future merging. Also, the original `Year` values are preserved in a new column (`Year_raw`) for traceability.

e) Convert Numeric Columns: Each dataset has columns labeled as `Value`, which sometimes contain string or malformed data.

f) Clean and Format `df_met` (Metrics Data): Removes unwanted spaces and newline characters from column names. Converts all metrics (e.g., `Tobacco Price Index`, `Household Expenditure`) to numeric.

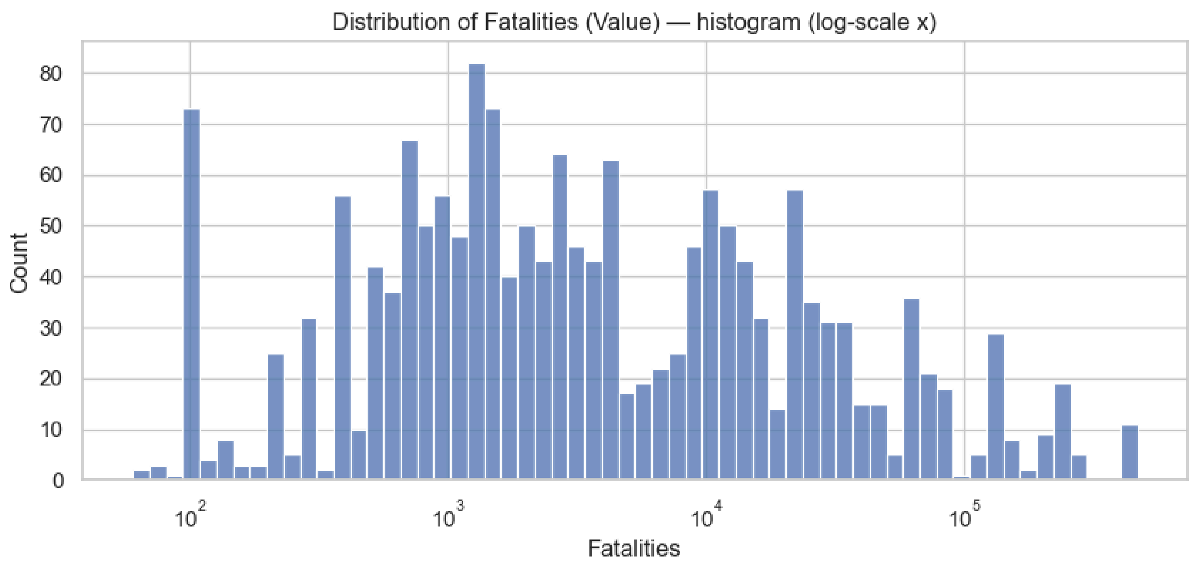
g) Ensure Numeric Columns in Smoking and Prescription Data: Standardizes smoking prevalence percentages by gender and age. Ensures prescription counts are valid numbers.

h) Display Samples and Data Types.

PHASE 02: Exploratory Data Analysis (EDA) & Visualization

```
 # Phase 2: EDA & Visualizations
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='whitegrid')

# 2A: Target distribution (fatalities)
plt.figure(figsize=(10,4))
sns.histplot(df_fat['Value'].dropna(), bins=60, log_scale=(True, False))
plt.title('Distribution of Fatalities (Value) — histogram (log-scale x)')
plt.xlabel('Fatalities')
plt.show()

# 2B: Fatalities per year
fatal_by_year = df_fat.groupby('Year')['Value'].sum().reset_index()
plt.figure(figsize=(10,4))
sns.lineplot(data=fatal_by_year, x='Year', y='Value', marker='o')
plt.title('Total Fatalities by Year')
plt.show()

# 2C: Missingness overview for each dataset (percent)
def missing_pct(df):
    return (df.isna().mean() * 100).sort_values(ascending=False)
print("Admissions missingness (%):")
display(missing_pct(df_adm))
print("Fatalities missingness (%):")
display(missing_pct(df_fat))
print("Metrics missingness (%):")
display(missing_pct(df_met))
print("Smokers missingness (%):")
display(missing_pct(df_smo))
print("Prescriptions missingness (%):")
display(missing_pct(df_pre))

# 2D: Smoking prevalence trends (overall '16 and Over')
if '16 and Over' in df_smo.columns:
    plt.figure(figsize=(10,4))
    sns.lineplot(data=df_smo.groupby('Year')['16 and Over'].mean().reset_index(),
x='Year', y='16 and Over', marker='o')
    plt.title('Average Smoking Prevalence: 16 and Over (by Year)')
    plt.show()

# 2E: Scatter of fatalities vs smoking prevalence (year-level join)
# build simple year-level dataframe for visualization
year_df =
df_fat.groupby('Year')['Value'].sum().reset_index().rename(columns={'Value':'Fatalit
ies'})
if '16 and Over' in df_smo.columns:
    sp = df_smo[df_smo['Sex'].isna() |
(df_smo['Sex'].astype(str).str.lower().isin(['','all','all sexes','both']))]
    sp = sp[['Year','16 and Over']].groupby('Year').mean().reset_index()
    year_df = year_df.merge(sp, on='Year', how='left')
    plt.figure(figsize=(6,5))
    sns.scatterplot(data=year_df, x='16 and Over', y='Fatalities')
    sns.regplot(data=year_df, x='16 and Over', y='Fatalities', scatter=False,
lowess=True, color='red')
    plt.title('Year-level Fatalities vs Smoking Prevalence (16 and Over)')
    plt.show()
```

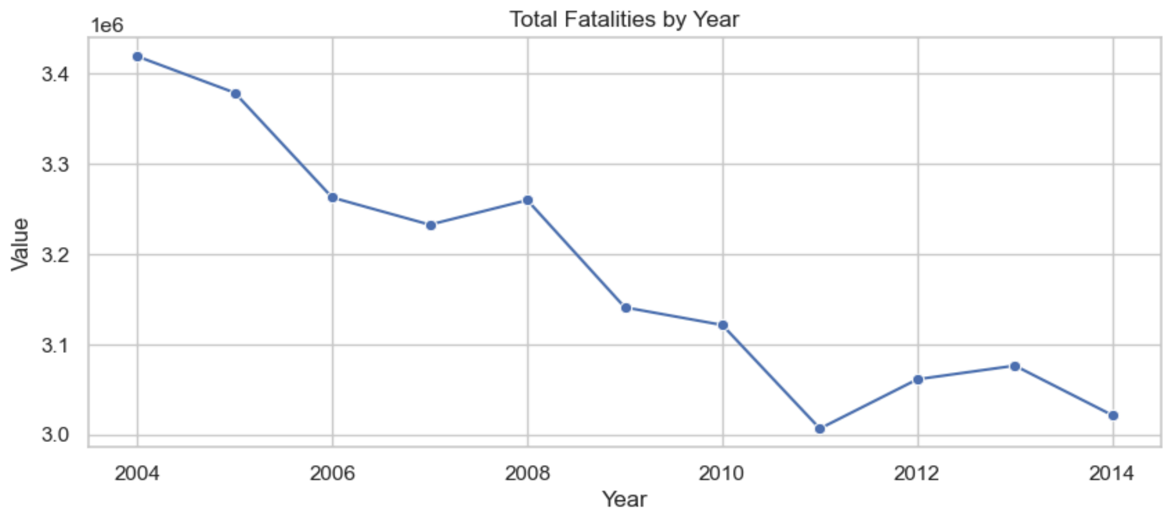
1) Distribution of Fatalities (Histogram — Log Scale X-Axis)



- This histogram displays how the number of fatalities (deaths due to tobacco-related causes) is distributed across all records in the dataset.
  Since the x-axis is in logarithmic scale, it highlights that fatalities vary across a wide range — from very low counts (hundreds) to extremely high counts (hundreds of thousands).

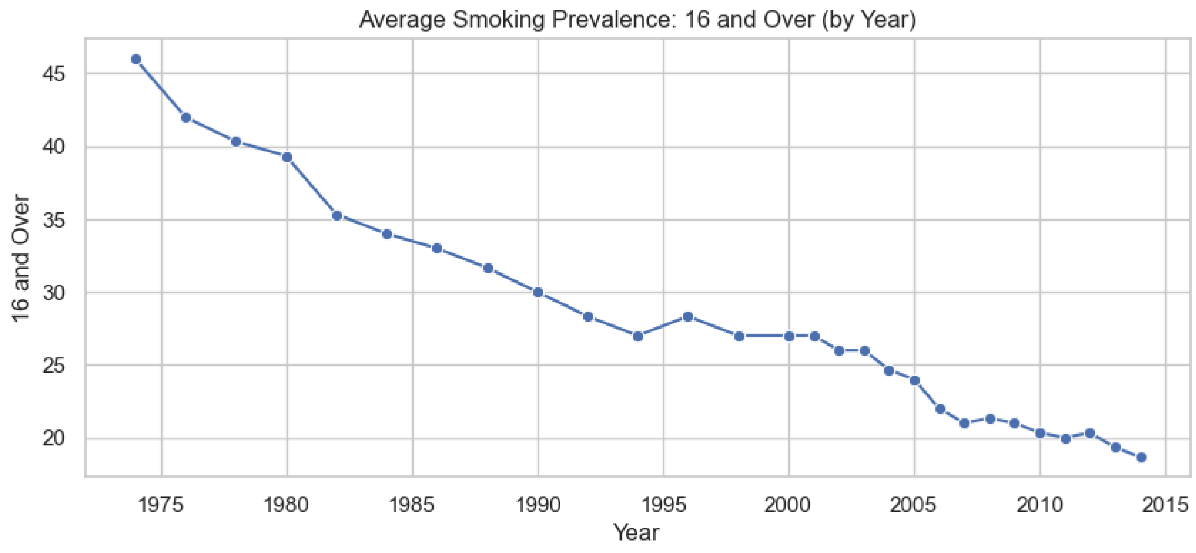
- The distribution is highly right-skewed, meaning most data points have lower fatality counts, while a few records have extremely high values.

2) Total Fatalities by Year (Line Plot)



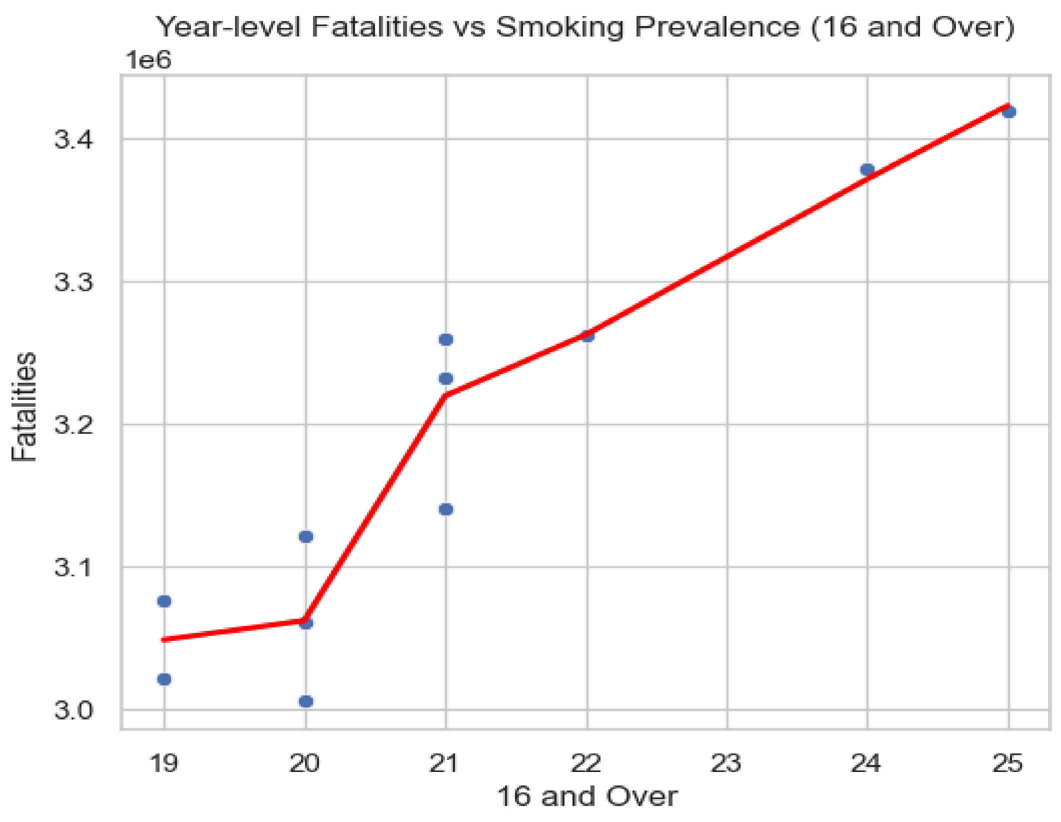
- This line plot represents the year-wise trend of total fatalities aggregated from the dataset.Between 2004 and 2014, the total fatalities show a gradual decline — from approximately 3.4 million in 2004 to about 3.0 million in 2014.
  This indicates that tobacco-related deaths have been decreasing over time.

3) Average Smoking Prevalence (16 and Over) by Year



- This chart shows the percentage of smokers aged 16 and over from 1974 to 2015. The smoking prevalence shows a consistent and sharp decline over the decades — from about 45% in the 1970s to below 20% by 2015.

4) Fatalities vs Smoking Prevalence (16 and Over)



- This scatter plot with a red regression line compares annual fatalities with average smoking prevalence among people aged 16 and above.
  The plot shows a positive relationship — as smoking prevalence increases, fatalities also rise. This suggests that smoking prevalence is a strong driver of tobacco-related fatalities.

5) Missing Data Summary

```
Admissions missingness (%):

Sex                   33.333333
Value                  1.972102
Year                   0.000000
ICD10 Code             0.000000
ICD10 Diagnosis        0.000000
Diagnosis Type         0.000000
Metric                 0.000000
Year_raw               0.000000
dtype: float64
Fatalities missingness (%):

Sex                   33.333333
Value                  1.143511
```

```
Year                0.000000
ICD10 Code          0.000000
ICD10 Diagnosis     0.000000
Diagnosis Type      0.000000
Metric              0.000000
Year_raw            0.000000
dtype: float64
Metrics missingness (%):

Household Expenditure on Tobacco                        13.888889
Household Expenditure Total                             13.888889
Expenditure on Tobacco as a Percentage of Expenditure   13.888889
Year                                                     0.000000
Tobacco Price Index                                      0.000000
Retail Prices Index                                      0.000000
Tobacco Price Index Relative to Retail Price Index       0.000000
Real Households' Disposable Income                       0.000000
Affordability of Tobacco Index                           0.000000
Year_raw                                                 0.000000
dtype: float64
Smokers missingness (%):

Sex            33.333333
Year            0.000000
Method          0.000000
16 and Over     0.000000
16-24           0.000000
25-34           0.000000
35-49           0.000000
50-59           0.000000
60 and Over     0.000000
Year_raw        0.000000
dtype: float64
Prescriptions missingness (%):

Year_raw                                                    100.000000
Varenicline (Champix) Prescriptions                          18.181818
Net Ingredient Cost of Varenicline (Champix)                 18.181818
Year                                                          0.000000
All Pharmacotherapy Prescriptions                             0.000000
Nicotine Replacement Therapy (NRT) Prescriptions              0.000000
Bupropion (Zyban) Prescriptions                               0.000000
Net Ingredient Cost of All Pharmacotherapies                  0.000000
Net Ingredient Cost of Nicotine Replacement Therapies (NRT)   0.000000
Net Ingredient Cost of Bupropion (Zyban)                      0.000000
dtype: float64
```

- The printed tables summarize the **percentage of missing values** in each dataset.

PHASE 03: Handling Missing Data

This phase focuses on cleaning and imputing missing values across all datasets to ensure data consistency before analysis and modeling.

- Numeric columns (like fatalities, admissions, expenditures, and prescriptions) were converted to numeric types and missing values filled with the median to preserve the data's central tendency.
- Categorical columns such as *Sex* were filled with the label 'Unknown' to handle missing entries without losing records.
- Redundant columns were dropped after verification.
- Finally, minor formatting issues (like inconsistent year formats) were corrected to maintain uniformity.

PHASE 04: Merge & Feature Engineering

1. **Define Fatalities as the Target Dataset:** The dataset `df_fat` (Fatalities) is renamed and copied to form `df_target`, where the column 'Value' is renamed to 'Fatalities'. This makes it clear that our main prediction target (dependent variable) will be the number of fatalities caused by smoking-related diseases.

2. **Pivot Admissions Data:**
   - The admissions dataset (`df_adm`) contains multiple metrics (e.g., number of admissions, rate per 100k, etc.) stored as rows.
   - The pivot operation aggregates values using sum and groups by Year, ICD10 Diagnosis, and Sex.

3. **Merge Admissions with Fatalities:**
   - Left join ensures that every record from fatalities (our target) is preserved, even if there's missing data on admissions.
   - This gives us a dataset containing both fatalities and corresponding hospital admissions for each disease and demographic segment.

4. **Add Year-level Metrics and Prescriptions:**
   - The dataset is further enriched by merging in year-level indicators from:
     `df_met` → Tobacco price, affordability index, household expenditure, etc.
     `df_pre` → Number of smoking cessation prescriptions and their net costs.

5. **Smoking Prevalence Data:** The smokers dataset (`df_smo`) contains smoking prevalence by year and sex.

    - We create two versions:
      Sex-specific prevalence → gives smoking rate for each gender.
      Overall prevalence (average across sexes) → acts as a fallback.

    Both are merged into the main dataset, and a final column **`smok_prev`** is created, filling missing values from the overall average. This ensures we have a complete smoking prevalence feature for all (Year, Sex) combinations.

6. **Convert Key Columns to Numeric:** Fatalities, admissions, and prescription-related columns are converted to numeric.

7. **Aggregate Useful Numerical Summaries:**
    - `admissions_count` → represents total admissions, using the "Number of admissions" column.
    - `prescriptions_total` → represents the total number of prescriptions (sum of all pharmacotherapy, NRT, Bupropion, and Varenicline prescriptions).

8. **Standardize Categorical Variables:**
    - Text inconsistencies are fixed:
        - Missing `Sex` values are replaced with 'Unknown'.
        - `ICD10 Diagnosis` is converted to string format for merging consistency.
9. **Create Lag Features:**
    - `smok_prev_lag1` → previous year's smoking prevalence.
      `tob_price_lag1` → previous year's tobacco price index.
    - These features help the model capture temporal relationships.

PHASE 05: Data Preparation for Modeling

1. **Filter Data:** Keep only rows with valid fatality counts.

2. **Feature Selection:** Choose numeric features (like admissions, prescriptions, smoking prevalence) and categorical features (Sex, Diagnosis).

3. **Train-Test Split:** Split data by *Year* — using earlier years for training and the last 2 years for testing.

4. **Target Transformation:** Apply a log transformation to the target (*Fatalities*) to stabilize variance and reduce skewness.

5. **Preprocessing Pipelines:**
    - Scale numeric features using **StandardScaler**.
    - Encode categorical features using **OneHotEncoder**.

6. **Combine with ColumnTransformer:** Integrate both transformations into a single preprocessing pipeline for consistent model input.

PHASE 06: Model Training, Evaluation & Selection

This phase builds, evaluates, and compares three predictive models to estimate the number of fatalities based on various health, economic, and behavioral factors.

1. Ridge Regression (Baseline Model):

- A regularized linear regression model was built using a pipeline with preprocessing (scaling + encoding).
- The log-transformed target (`Fatalities`) was used to handle skewness and stabilize variance.
- The Ridge model achieved:
  RMSE: 34,511
  MAE: 10,399
  R^2 (log-target): 0.798
  This served as a strong baseline linear model.

2. Poisson Generalized Linear Model (GLM):

- The model formula included predictors such as admissions count, prescriptions, smoking prevalence, tobacco price index, Sex, and ICD10 diagnosis.
  The results indicated statistically significant predictors with p-values < 0.05, showing their strong relationship with fatalities.
- Model performance:
    - RMSE: 29,093
    - MAE: 10,426
    - R^2: 0.687

- The Poisson GLM provided the best overall performance in predicting count-type outcomes.

3. XGBoost Regressor (Ensemble Model):

- A tree-based gradient boosting model (XGBoost) was trained on preprocessed data for non-linear pattern detection. Despite its flexibility, it performed slightly worse due to possible data sparsity and log-transformation sensitivity.
- Performance:
  RMSE: 49,707
  MAE: 15,807
  R^2: 0.540

***Model Comparison and Selection:**

- Model performances were compared using RMSE, MAE, and R^2 metrics.

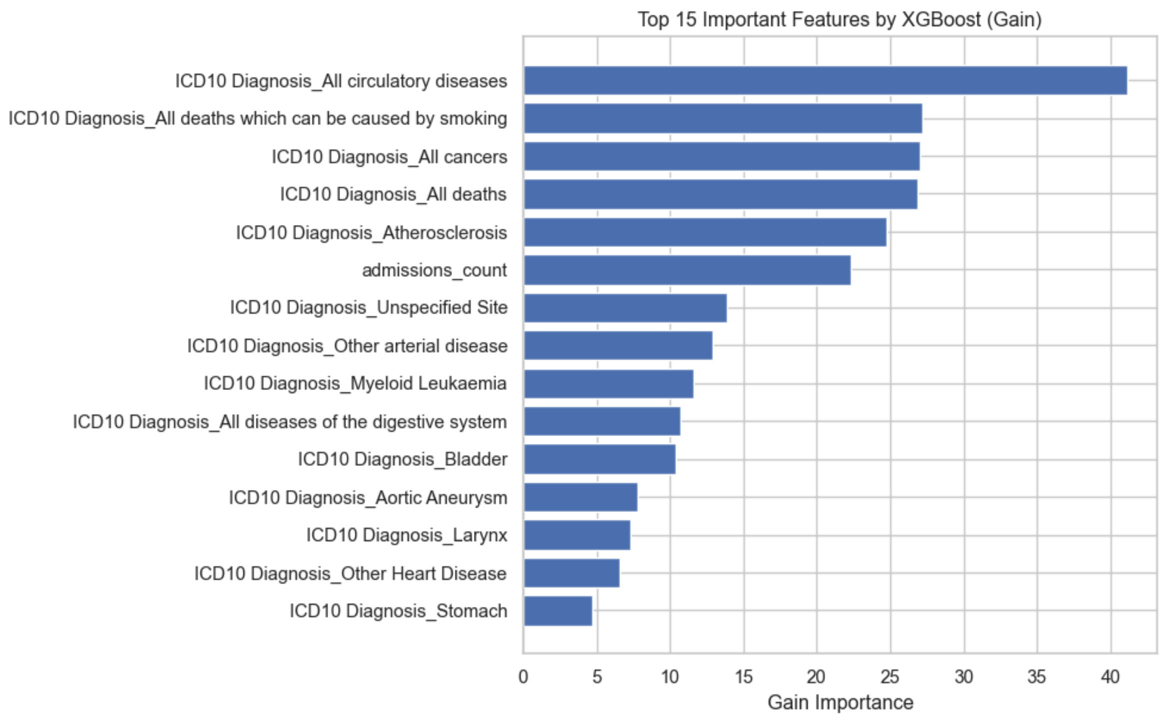
- Results were saved to `model_results.csv`.

- Based on lowest RMSE, the Poisson GLM was selected as the best-performing model and saved for future predictions.

```
=== Model Comparison Summary ===

         Model          RMSE           MAE        R2
1  Poisson GLM  29093.504003  10425.813946  0.687684
0        Ridge  34511.340216  10399.148338  0.798309
2      XGBoost  49707.090285  15806.932184  0.540124
```

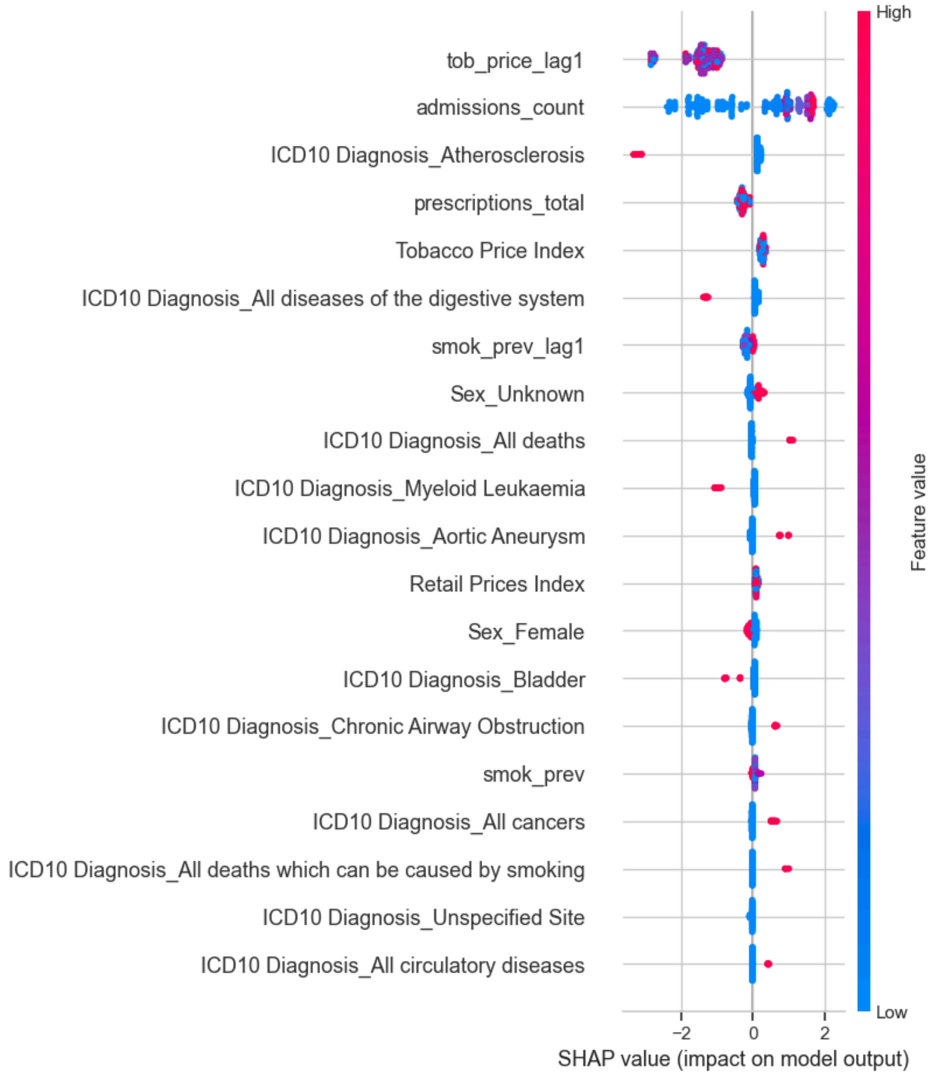
PHASE 06: Evaluation & Explainability

5A. XGBoost Feature Importance



- The most important features were:
  - `ICD10 Diagnosis_All circulatory diseases` — highest impact on predicting fatalities.
  - `All deaths caused by smoking`, `All cancers`, and `Atherosclerosis` also had high gain values.
  - `admissions_count` (hospital admissions) was also an influential numeric predictor.

5B. SHAP Explainability



This plot shows how each feature affects the model's predictions for fatalities, based on SHAP (SHapley Additive exPlanations).

- **tob_price_lag1 (previous year's tobacco price)** → Higher tobacco prices (red) slightly reduce predicted fatalities → implies that increased price discourages smoking.

- **admissions_count** → High admission counts (red) raise predicted fatalities — consistent with higher disease burden.

- **Atherosclerosis & circulatory diseases** → Strong positive SHAP values — these diagnoses are strongly linked to higher fatalities.

- **prescriptions_total & Tobacco Price Index** → Both influence predictions moderately — reflecting medical and economic trends.

- **smok_prev & smok_prev_lag1** → Higher smoking prevalence (red) increases fatalities, matching real-world patterns.

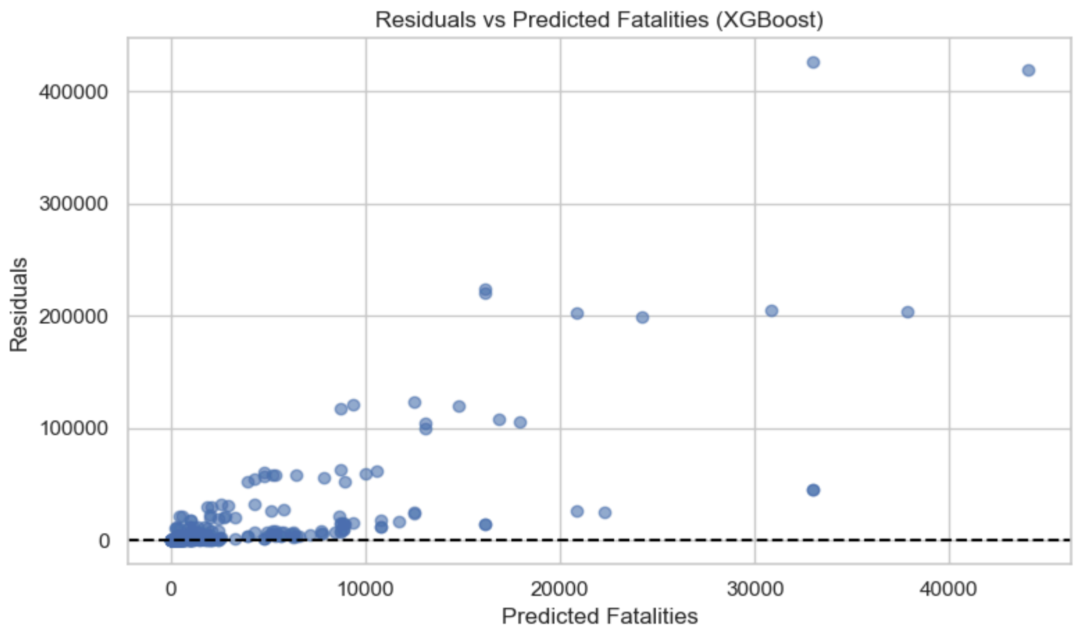
- **Sex_Female / Sex_Unknown** → Sex differences affect predictions slightly, with "Unknown" showing unstable effects due to missing data.

5C. Subgroup Performance (by Sex)

```
            === 5C. Subgroup Performance by Sex ===

        Sex=Female    | RMSE=36,123.67 | MAE=11,620.51
        Sex=Male      | RMSE=34,156.73 | MAE=11,820.88
        Sex=Unknown   | RMSE=70,290.70 | MAE=23,979.41
```

Interpretation:
The model performs better for *male* and *female* categories, where data is richer. The "Unknown" group performs poorly because of insufficient records or missing patterns.

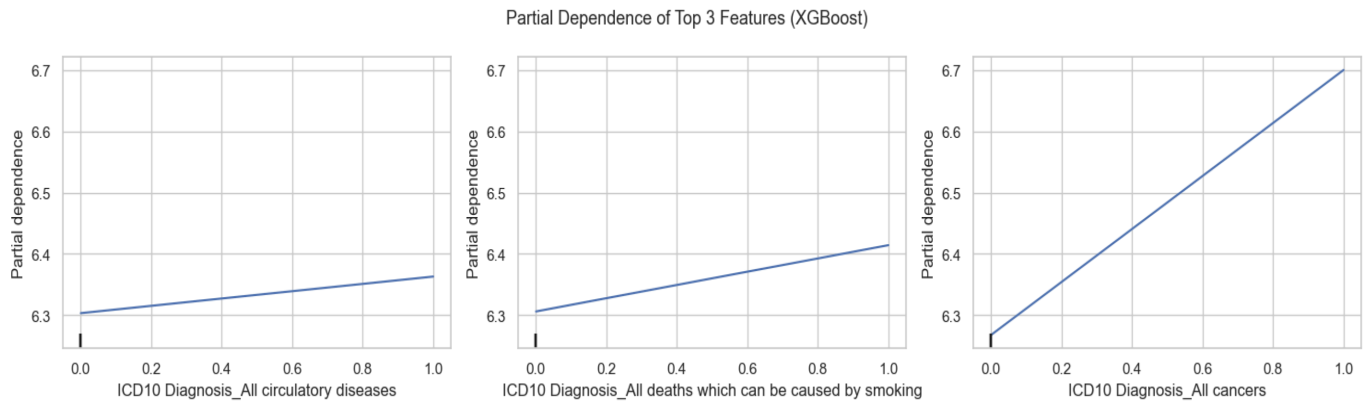
5D. Residual Analysis



The residual plot (Predicted vs. Actual Errors) checks how predictions deviate from true fatalities.A few large residuals suggest the model sometimes underestimates or overestimates high fatality counts, likely for rare diseases.

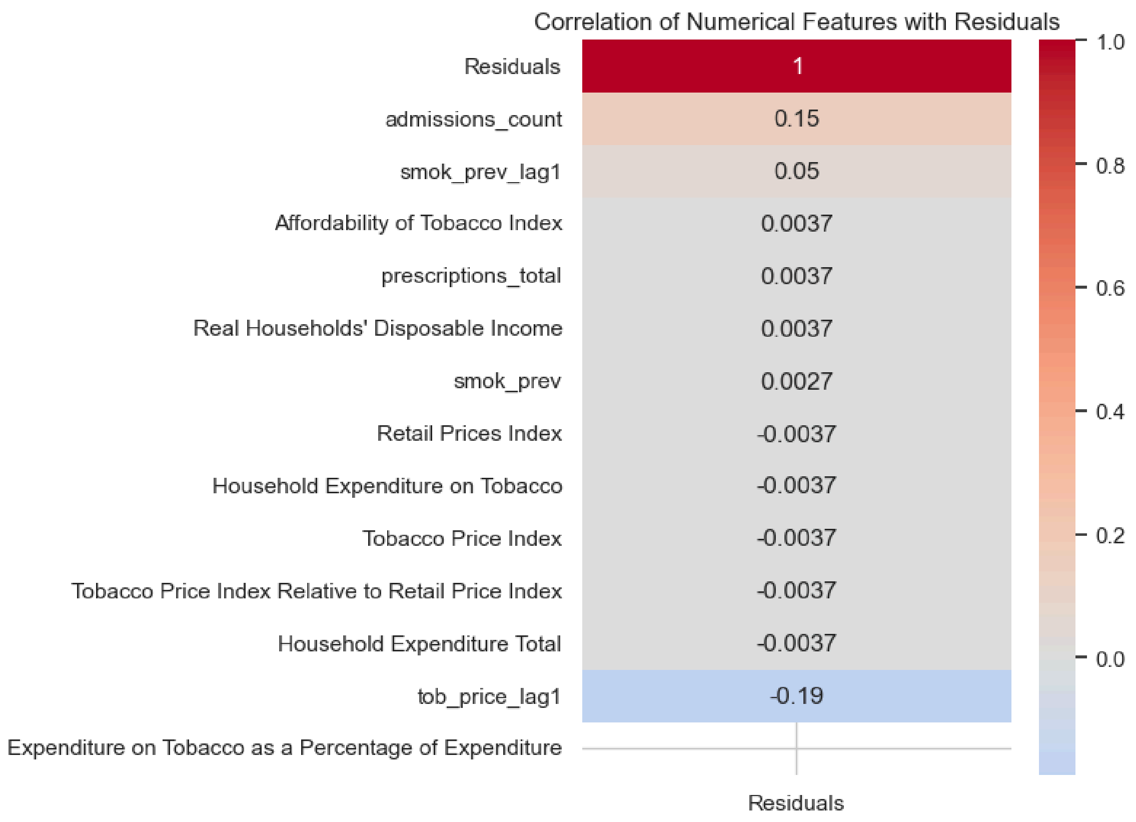
5.5. Advanced Model Explainability

1. Partial Dependence Plot



- “All circulatory diseases”, “All deaths caused by smoking”, and “All cancers” have positive slopes, meaning as these diagnoses occur more frequently, predicted fatalities increase.
  This confirms that diseases directly linked to smoking have the strongest positive effect on predicted death counts.

2. Residual Correlation Heatmap



Shows how residuals (model errors) correlate with numeric features. Features like admissions_count have a small positive correlation (0.15) → higher admissions slightly increase underprediction.

PHASE 07: Deployment Phase

This phase converts trained models (Poisson GLM & XGBoost Regression) into a fully interactive web app using Streamlit.
It allows users to input smoking-related factors and instantly predict the expected number of fatalities.

The app automatically decides which model to use:

- Poisson GLM → for diagnoses directly linked to smoking (e.g., *All cancers, All circulatory diseases, All respiratory diseases*).
- XGBoost Regressor → for all other diagnoses.
- It supports:
  Data loading (from `.pkl` and `.csv` files)
  Preprocessing of data
  Model prediction
  Visualization of sensitivity and prediction trends

A. Page Setup : Defines the page title, icon, and layout. Displays an introduction explaining model logic.

```
# Page Configuration
# ======================================================
st.set_page_config(page_title="Hybrid Smoking-Related Fatalities
Predictor", page_icon="⚕️", layout="wide")
st.title("⚕️ Hybrid Smoking-Related Fatalities Predictor")
st.markdown("""
This app automatically selects the best model depending on the ICD10
Diagnosis type:
- 🧠 **Poisson GLM Model** → used for `All respiratory diseases`,
`All cancers`, `All circulatory diseases`.
- ⚡ **XGBoost Regression Model** → used for all other diagnosis
categories.
""")
```

B. Model Loaders: Loads the pre-trained Poisson GLM model.
Uses caching (`st.cache_resource`) to avoid reloading repeatedly. Reads model results from Excel/CSV files for trend visualizations.

```python
# Poisson GLM Loader
# =====================================================
@st.cache_resource
def load_poisson_model():
    artifact_dir = "artifacts"
    model_path = os.path.join(artifact_dir, "best_model_glm.pkl")
    if not os.path.exists(model_path):
        raise FileNotFoundError("❌ Poisson model not found in
artifacts folder.")
    model = joblib.load(model_path)
    return model

@st.cache_data
def load_poisson_data():
    data_path = os.path.join("artifacts", "model_results.xls")
    if not os.path.exists(data_path):
        st.warning("⚠️ Poisson dataset not found. Prediction only
will work.")
        return pd.DataFrame()
    try:
        df = pd.read_excel(data_path, engine="openpyxl")
    except Exception:
        df = pd.read_csv(data_path)
    return df
```

C. XGBoost Section: Loads and merges multiple CSV files. Trains a regression pipeline:

- Preprocessing: `StandardScaler` + `OneHotEncoder`
- Model: `XGBRegressor`
- Calibration: linear regression correction for smoother predictions.

```
#  XGBoost Loader and Trainer
# =====================================================
@st.cache_data
def load_xgb_data(data_glob="data/*.csv"):
    csv_files = glob.glob(data_glob)
    if not csv_files:
        st.warning("⚠️ No CSV files found in data folder.")
        return pd.DataFrame()
    df_list = [pd.read_csv(f) for f in csv_files]
    df = pd.concat(df_list, ignore_index=True)
    df.columns = df.columns.str.strip()
    return df

def ensure_features(df):
    df = df.copy()
    if 'Fatalities' not in df.columns and 'Value' in df.columns:
        df.rename(columns={'Value': 'Fatalities'}, inplace=True)

    if 'admissions_count' not in df.columns:
        adm_cols = [c for c in df.columns if 'admission' in
c.lower()]
        df['admissions_count'] =
df[adm_cols].select_dtypes(include=[np.number]).sum(axis=1) if
adm_cols else 0

    if 'prescriptions_total' not in df.columns:
        pres_cols = [c for c in df.columns if 'prescription' in
c.lower()]
        df['prescriptions_total'] =
df[pres_cols].select_dtypes(include=[np.number]).sum(axis=1) if
pres_cols else 0

    if 'Tobacco Price Index' not in df.columns:
        price_cols = [c for c in df.columns if 'price' in c.lower()]
        df['Tobacco Price Index'] = df[price_cols[0]] if price_cols
else np.nan

    if 'smok_prev' not in df.columns:
        smok_cols = [c for c in df.columns if 'smok' in c.lower()]
        df['smok_prev'] = df[smok_cols[0]] if smok_cols else np.nan

    if 'Sex' not in df.columns:
        df['Sex'] = 'Unknown'

    if 'ICD10 Diagnosis' not in df.columns:
        df['ICD10 Diagnosis'] = 'All cancers'

    for col in ['admissions_count', 'prescriptions_total',
'smok_prev', 'Tobacco Price Index', 'Fatalities']:
        if col in df.columns:
            df[col] = pd.to_numeric(df[col], errors='coerce')

    df.dropna(subset=['Fatalities'], inplace=True)
    return df

@st.cache_resource
def train_xgb_model(df):
    num_features = ["admissions_count", "prescriptions_total",
"smok_prev", "Tobacco Price Index"]
    cat_features = ["Sex", "ICD10 Diagnosis"]

    X = df[num_features + cat_features].copy()
    y = df["Fatalities"].astype(float)

    for c in num_features:
        X[c].fillna(X[c].median(), inplace=True)
    for c in cat_features:
        X[c].fillna("Unknown", inplace=True)

    preprocessor = ColumnTransformer([
        ("num", StandardScaler(), num_features),
        ("cat", OneHotEncoder(handle_unknown="ignore",
sparse_output=False), cat_features)
    ])

    model = XGBRegressor(
        n_estimators=300,
        learning_rate=0.05,
        max_depth=5,
        subsample=0.8,
        colsample_bytree=0.8,
        random_state=42,
        tree_method="hist"
    )
    pipeline = Pipeline([
        ("preprocessor", preprocessor),
        ("model", model)
    ])

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
    pipeline.fit(X_train, y_train)

    calib =
LinearRegression().fit(pipeline.predict(X_train).reshape(-1, 1),
y_train)
    rmse = np.sqrt(np.mean((pipeline.predict(X_test) - y_test) ** 2))
    return pipeline, calib, num_features, cat_features, rmse

def calibrated_predict(model, calib, X):
    raw_pred = model.predict(X)
    return calib.predict(raw_pred.reshape(-1, 1))
```

D. Sidebar Input Controls: Users interact with the sidebar to set input parameters:

- Admissions count
- Total prescriptions
- Smoking prevalence
- Tobacco price index

E. Prediction Logic: Chooses model automatically.Displays prediction with model name.

```
if st.button("🚀 Predict Fatalities"):
    try:
        if diagnosis in high_confidence_diseases:
            # --- Use Poisson GLM ---
            pred = poisson_model.predict(input_data)[0]
            model_used = "Poisson GLM"
        else:
            # --- Use XGBoost ---
            pred = calibrated_predict(pipeline, calib, input_data)[0]
            model_used = "XGBoost Regression"
        st.success(f"### 🧩 Predicted Fatalities: **{pred:,.0f}**
({model_used})")

    except Exception as e:
        st.error(f"⚠️ Prediction failed: {e}")
```

F. Visualization:

🧠 For Poisson GLM:

- Generates a heatmap showing how fatalities vary with:
  - Smoking prevalence
  - Hospital admissions
- Highlights user's input point in red.

⚡ For XGBoost:

- Plots a line chart showing how predicted fatalities change when smoking prevalence varies ±30%.
- Shows the impact trend dynamically.

G. Folder Structure

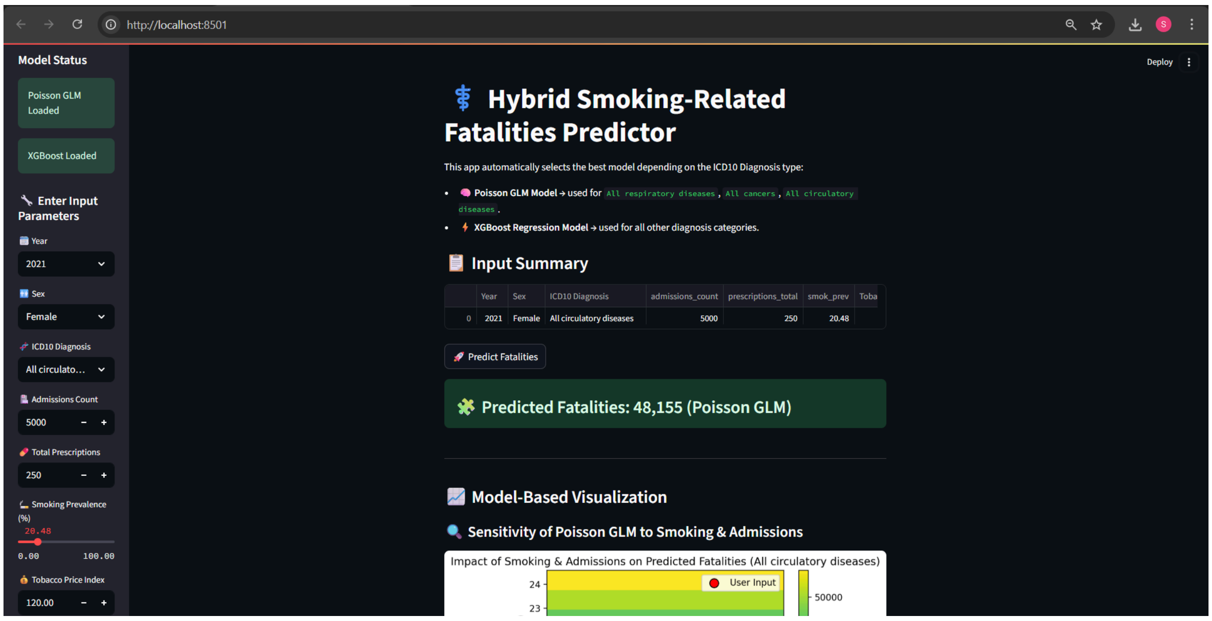
```
📁 project_root/
│
├── app.py                    # ← Streamlit app code
├── artifacts/
│   ├── best_model_glm.pkl       # Trained Poisson GLM model
│   ├── model_results.xls        # Poisson model data (optional)
│
├── data/
│   ├── admissions.csv         # XGBoost data files
│   ├── smokers.csv
│   ├── ...
├── requirements.txt           # Dependencies (optional)
```

- **Install dependencies:**

  pip install streamlit pandas numpy scikit-learn xgboost joblib openpyxl matplotlib statsmodels.

- **Run the app:** streamlit run app.py

H. Output Example:



http://localhost:8501

Deploy

Model Status

Poisson GLM Loaded

XGBoost Loaded

Enter Input Parameters

Year: 2021

Sex: Female

ICD10 Diagnosis: All circulato...

Admissions Count: 5000

Total Prescriptions: 250

Smoking Prevalence (%): 20.48 (0.00 – 100.00)

Tobacco Price Index: 120.00

Hybrid Smoking-Related Fatalities Predictor

This app automatically selects the best model depending on the ICD10 Diagnosis type:

- **Poisson GLM Model** → used for `All respiratory diseases`, `All cancers`, `All circulatory diseases`.
- **XGBoost Regression Model** → used for all other diagnosis categories.

Input Summary

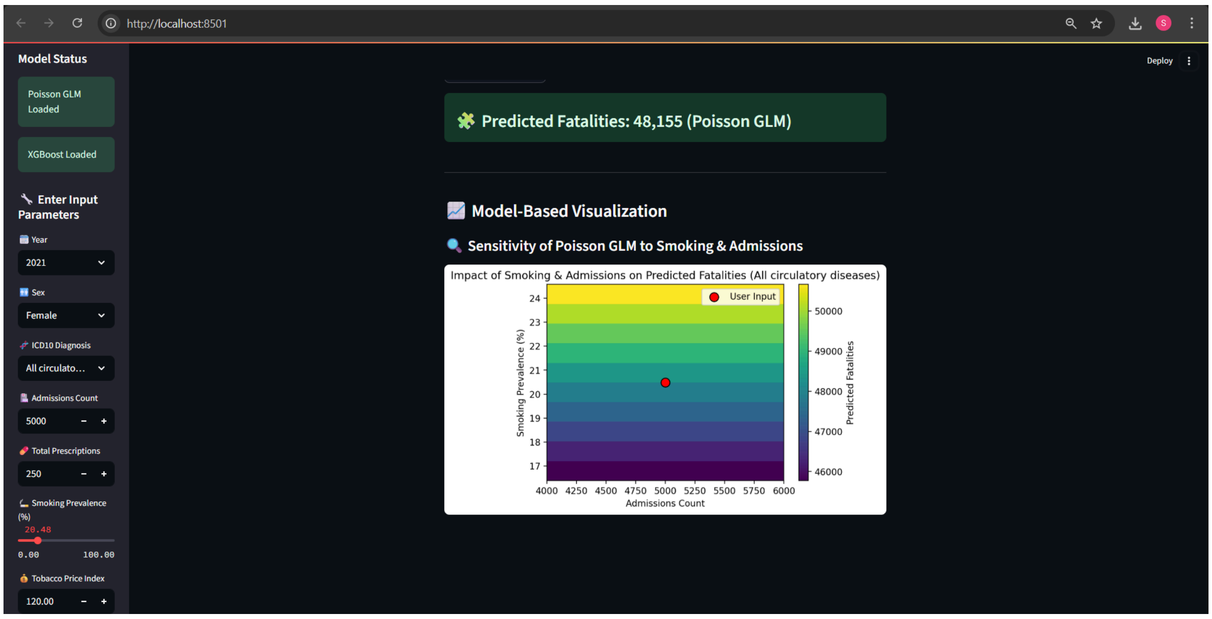
	Year	Sex	ICD10 Diagnosis	admissions_count	prescriptions_total	smok_prev	Toba
0	2021	Female	All circulatory diseases	5000	250	20.48	

Predict Fatalities

Predicted Fatalities: 48,155 (Poisson GLM)

Model-Based Visualization

Sensitivity of Poisson GLM to Smoking & Admissions



http://localhost:8501

Deploy

Model Status

Poisson GLM Loaded

XGBoost Loaded

Enter Input Parameters

Year: 2021

Sex: Female

ICD10 Diagnosis: All circulato...

Admissions Count: 5000

Total Prescriptions: 250

Smoking Prevalence (%): 20.48 (0.00 – 100.00)

Tobacco Price Index: 120.00

Predicted Fatalities: 48,155 (Poisson GLM)

Model-Based Visualization

Sensitivity of Poisson GLM to Smoking & Admissions

Company/Organization: Unified Mentor
Prepared By: Shravani Borude
UNID: UMID13072550101
Position: Data Science Intern
Date: 01/11/2025