UNIFIED MENTOR
YOUR SKILL, SUCCESS & JOURNEY

Project Title	**Tobacco Use and Mortality, 2004-2015**
Tools	Jupyter Notebook and VS code
Technologies	Machine learning , python , sql
Domain	Data Science
Project Difficulties level	Advanced

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

About Dataset

Context

Conditions that could be caused by smoking resulted in 1.7 million admissions to hospitals in England, for adults aged 35 and over, in 2014-2015 -- an average of 4,700 admissions per day! These figures refer to admissions with a primary diagnosis of a disease that can be caused by smoking, but for which smoking may or may not have actually been the cause.

Content

The Statistics on Smoking in England report aims to present a broad picture of health issues relating to smoking in England and covers topics such as smoking prevalence, habits, behaviours, and attitudes, smoking-related health issues and mortality, and associated costs.

Acknowledgements

This report contains data and information previously published by the Health and Social Care Information Centre (HSCIC), Department of Health, the Office for National Statistics, and Her Majesty's Revenue and Customs

Tobacco Use and Mortality Machine Learning Project

This project involves using machine learning to analyze the relationship between tobacco use and mortality rates. The goal is to predict the likelihood of mortality based on various factors related to tobacco use. Here's a detailed outline to guide you through the project:

1. Problem Definition

- **Objective**: Predict the likelihood of mortality based on tobacco use patterns and related factors.
- **Scope**: Focus on a specific demographic or geographic region if necessary, and consider both direct and indirect factors influencing mortality.

2. Data Collection

- **Datasets**:
    - **Health Surveys**: National Health and Nutrition Examination Survey (NHANES), Behavioral Risk Factor Surveillance System (BRFSS).
    - **Mortality Data**: World Health Organization (WHO), Centers for Disease Control and Prevention (CDC).
    - **Tobacco Use Data**: Surveys on smoking habits, duration, frequency, and types of tobacco used.
    - **Socioeconomic Data**: Age, gender, income, education level, occupation.
    - **Health Data**: Pre-existing conditions, lifestyle habits, healthcare access.

3. Data Preprocessing

- **Data Cleaning**: Handle missing values, outliers, and inconsistencies.
- **Data Integration**: Merge datasets from different sources to create a comprehensive dataset.
- **Feature Engineering**: Create relevant features such as:
    - Duration of tobacco use.
    - Frequency and type of tobacco products used.
    - Demographic factors (age, gender).
    - Socioeconomic factors (income, education).
    - Health-related factors (pre-existing conditions, healthcare access).

4. Exploratory Data Analysis (EDA)

- **Visualizations**: Use histograms, scatter plots, and heatmaps to understand distributions and correlations.
- **Statistical Analysis**: Perform correlation analysis to identify significant relationships between features and mortality.

5. Model Selection

- **Supervised Learning Algorithms**: Consider algorithms suitable for classification problems such as:

    - Logistic Regression.
    - Decision Trees and Random Forests.
    - Gradient Boosting Machines (e.g., XGBoost, LightGBM).
    - Support Vector Machines (SVM).
    - Neural Networks.

6. Model Training and Evaluation

- **Training**: Split the dataset into training and testing sets. Use cross-validation to ensure robust model performance.
- **Evaluation Metrics**: Choose appropriate metrics such as accuracy, precision, recall, F1-score, and ROC-AUC score.
- **Model Tuning**: Perform hyperparameter tuning to optimize model performance using techniques like Grid Search or Random Search.

7. Model Interpretation

- **Feature Importance**: Identify which features are most influential in predicting mortality.
- **Model Explainability**: Use tools like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to explain model predictions.

8. Deployment

- **API Development**: Create an API for the model using frameworks like Flask or FastAPI.
- **Web Application**: Develop a user interface to input data and display predictions using frameworks like Streamlit or Dash.

9. Monitoring and Maintenance

- **Model Monitoring**: Continuously monitor the model's performance using new data to ensure its accuracy and reliability.

- **Regular Updates**: Update the model periodically with new data to maintain its relevance.

10. Documentation and Reporting

- **Documentation**: Maintain comprehensive documentation of the project, including data sources, preprocessing steps, model selection, and evaluation results.
- **Reporting**: Create detailed reports and visualizations to communicate findings and insights to stakeholders.

Tools and Technologies

- **Programming Language**: Python.
- **Libraries**: pandas, numpy, scikit-learn, seaborn, matplotlib, XGBoost, LightGBM, TensorFlow/Keras (for neural networks), SHAP, LIME.
- **Frameworks for Deployment**: Flask, FastAPI, Streamlit, Dash.

Potential Challenges

- **Data Quality**: Ensuring the data is clean and representative.
- **Bias**: Addressing potential biases in the data and model.
- **Interpretability**: Ensuring the model is interpretable and explainable to non-technical stakeholders.

Additional Considerations

- **Ethical Considerations**: Ensure the project adheres to ethical guidelines, especially concerning sensitive health data.
- **Privacy and Security**: Implement measures to protect the privacy and security of personal health information.

This project will provide valuable insights into the impact of tobacco use on mortality and potentially inform public health policies and interventions.

Sure! Here's a step-by-step guide with sample code snippets for a Tobacco Use and Mortality Modeling machine learning project using Python.

1. Problem Definition

   Objective: Predict mortality based on tobacco use patterns and related factors.

2. Data Collection

   For the sake of this example, let's assume you have a CSV dataset named tobacco_mortality.csv.

3. Data Preprocessing

```
import pandas as pd
import numpy as np

# Load the dataset
df = pd.read_csv('tobacco_mortality.csv')

# Display basic info and check for missing values
print(df.info())
print(df.isnull().sum())

# Fill missing values or drop rows/columns as necessary
df.fillna(method='ffill', inplace=True)  # Example method
```

4. Exploratory Data Analysis (EDA)

```
import seaborn as sns
import matplotlib.pyplot as plt

# Basic statistics
print(df.describe())

# Histograms for numeric features
df.hist(bins=30, figsize=(20, 15))
plt.show()

# Correlation matrix
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```

5. Feature Engineering

```
# Example feature engineering
df['tobacco_use_duration'] = df['age'] - df['age_started_smoking']

# One-hot encoding for categorical variables
df = pd.get_dummies(df, columns=['gender', 'income_level', 'education_level'],
drop_first=True)
```

6. Model Selection

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Define features and target variable
X = df.drop('mortality', axis=1)
y = df['mortality']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

7. Model Training and Evaluation

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score

# Initialize models
log_reg = LogisticRegression()
```

```
rf_clf = RandomForestClassifier()

# Train models
log_reg.fit(X_train, y_train)
rf_clf.fit(X_train, y_train)

# Make predictions
log_reg_pred = log_reg.predict(X_test)
rf_clf_pred = rf_clf.predict(X_test)

# Evaluate models
print("Logistic Regression Metrics")
print(f"Accuracy: {accuracy_score(y_test, log_reg_pred)}")
print(f"Precision: {precision_score(y_test, log_reg_pred)}")
print(f"Recall: {recall_score(y_test, log_reg_pred)}")
print(f"F1 Score: {f1_score(y_test, log_reg_pred)}")
print(f"ROC AUC: {roc_auc_score(y_test, log_reg_pred)}")

print("\nRandom Forest Classifier Metrics")
print(f"Accuracy: {accuracy_score(y_test, rf_clf_pred)}")
print(f"Precision: {precision_score(y_test, rf_clf_pred)}")
print(f"Recall: {recall_score(y_test, rf_clf_pred)}")
print(f"F1 Score: {f1_score(y_test, rf_clf_pred)}")
print(f"ROC AUC: {roc_auc_score(y_test, rf_clf_pred)}")
```

8. Model Interpretation

```
import shap

# Use SHAP for model interpretation
explainer = shap.Explainer(rf_clf, X_train)
shap_values = explainer(X_test)

# SHAP summary plot
shap.summary_plot(shap_values, X_test)
```

9. Deployment

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    input_data = np.array([data['feature1'], data['feature2'], ...])  # Adjust as needed
    input_data = scaler.transform([input_data])
    prediction = rf_clf.predict(input_data)
    return jsonify({'prediction': int(prediction[0])})

if __name__ == '__main__':
    app.run(debug=True)
```

10. Monitoring and Maintenance

Set up logging and monitoring tools to track model performance over time, and schedule regular retraining with new data.

Additional Considerations

- **Ethical Considerations**: Ensure the use of data is ethical and compliant with privacy regulations.
- **Privacy and Security**: Implement measures to protect sensitive health data.

This is a simplified version of a real-world project. Depending on the complexity and specifics of your dataset, you may need to adjust the steps accordingly.

Sample Project Report

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import statsmodels.api as sm
# I've added Daft to the stack so that I can visualize my graphical models beautifully
import daft
from daft import PGM

flatdata = pd.read_csv('../input/nhs-tobacco-flattened/flattened.csv').drop("Unnamed: 0", axis
=1)
```

For your reference, here's a preview of the data I've prepared.

In [2]:

```
flatdata.head(5)
```

Out[2]:

	Year	Sex	Age	Smokers	Income	Cost	In Treatment	Drug Cost	Deaths	Admissions	Spent
0	-0.786968	0	0	0.686744	-0.692975	-0.607955	-0.193109	-0.452985	0.969774	-0.285335	-0.817474
1	-0.786968	1	0	0.512484	-0.692975	-0.607955	-0.193109	-0.452985	0.969774	-0.285335	-0.817474

In [3]:

```
flatdata.describe()
```

Out[3]:

	Year	Sex	Age	Smokers	Income	Cost	In Treatment	Drug Cost	De
count	1.100000e+02	110.000000	110.000000	1.100000e+02	1.100000e+02	1.100000e+02	1.100000e+02	1.100000e+02	1.1
mean	2.018587e-18	0.500000	2.000000	9.285502e-17	1.478211e-14	-2.723074e-15	3.229740e-17	-1.433197e-16	5.7
std	5.000000e-01	0.502288	1.420686	5.000000e-01	5.000000e-01	5.000000e-01	5.000000e-01	5.000000e-01	5.0
min	-7.869677e-01	0.000000	0.000000	-1.055859e+00	-6.929748e-01	-6.079553e-01	-1.185350e+00	-8.936567e-01	-5. 01
25%	-4.721806e-01	0.000000	1.000000	-1.845575e-01	-4.287204e-01	-4.485363e-01	-1.931086e-01	-4.529852e-01	-4. 01
50%	0.000000e+00	0.500000	2.000000	7.683295e-02	3.880659e-02	-1.728573e-01	3.641846e-02	1.805771e-01	-3. 03
75%	4.721806e-01	1.000000	3.000000	3.382234e-01	5.876426e-01	4.796550e-01	4.327446e-01	5.079759e-01	2.8
max	7.869677e-01	1.000000	4.000000	1.122395e+00	7.705880e-01	9.333654e-01	5.482209e-01	6.442580e-01	9.6

Reference link