Current Industry Practices and the Role of Logistic Regression in Mental Health

```
Logistic regression is a widely used machine learning technique in the healthcare industry for predicting binary outcomes, such as the presence or absence of depression. It is favored for its simplicity, interpretability, and effectiveness in handling structured clinical data. In practice, logistic regression models utilize patient demographics, lifestyle factors, medical history, and psychological assessments to predict depression risk, supporting early diagnosis and personalized treatment plans in clinical settings.
```

Research Landscape and Recent Developments in Depression Prediction

```
Recent studies demonstrate the effectiveness of logistic regression models in accurately predicting depression by identifying significant risk factors such as age, sleep patterns, and physical activity. For instance, research by Smith et al. (2022) and Zhang et al. (2023) showed logistic regression models achieving high accuracy in depression classification. These models are praised for their ability to provide clear, interpretable results that can be easily communicated to healthcare professionals for clinical decision-making.
```

Challenges, Limitations, and Future Directions

```
While logistic regression is a robust tool for depression prediction, challenges such as handling multicollinearity, feature scaling, and dealing with missing data can affect model performance. Recent literature explores techniques like regularization and data imputation to overcome these limitations. The field is also moving towards integrating logistic regression with advanced machine learning methods to enhance predictive accuracy while maintaining interpretability, addressing the evolving needs of precision mental health care.
```

	Variable Name	Type	Description	Values
0	Age	Numerical	The age of the individual.	Numbers
1	Marital Status	Categorical	The marital status of the individual.	Married, Single, Widowed, Divorced
2	Education Level	Categorical	The highest level of education attained by the individual.	Bachelor's Degree, High School, Associate Degree, Master's Degree, PhD
3	Number of Children	Numerical	The number of children the individual has.	0, 1, 2, 3, 4
4	Smoking Status	Categorical	Whether the individual smokes.	Non-smoker, Former, Current
5	Physical Activity Level	Categorical	The level of physical activity engaged in by the individual.	Sedentary, Moderate, Active
6	Employment Status	Categorical	The current employment status of the individual.	Employed, Unemployed
7	Income	Numerical	The annual income of the individual.	Numbers
8	Alcohol Consumption	Categorical	The level of alcohol consumption by the individual.	Moderate, Low, High
9	Dietary Habits	Categorical	The eating habits of the individual.	Unhealthy, Moderate, Healthy
10	Sleep Patterns	Categorical	The sleep patterns of the individual.	Fair, Poor, Good
11	History of Mental Illness	Categorical	Whether the individual has a history of mental illness.	No, Yes
12	History of Substance Abuse	Categorical	Whether the individual has a history of substance abuse.	No, Yes
13	Family History of Depression	Categorical	Whether there is a family history of depression.	No, Yes
14	Depression	Categorical	Whether the individual has any chronic medical conditions.	No, Yes
15	Name	Object	Name of Individual	—

Libraries

In [55]:
```
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import OneHotEncoder

import time
```

In [2]:
```
import pyarrow as pa
import pyarrow.parquet as pq

file = pd.read_csv("depression_data.csv")
table = pa.Table.from_pandas(file)
pq.write_table(table, "depression_data.parquet")
```

Tell Time - Decorator for calculating the time

In [3]:
```
def tell_time(function, *args, **kwargs):
    def wrapper(*args, **kwargs):
        start = time.time()
        done = function(*args, **kwargs)
        print(f"{function.__name__}() function took - {(time.time()-start)//60} Mins {(time.time()-start)%60:.3f} Sec")
        return done
    return wrapper
```

In [4]:
```
@tell_time
def read_csv_data(file):
    return pd.read_csv(file)
```

In [5]:
```
@tell_time
def read_parquet_data(file):
    return pd.read_parquet(file)
```

Reading the Dataset

In [6]:
```python
csv_df = read_csv_data("depression_data.csv")
```
```
read_csv_data() function took - 0.0 Mins 0.785 Sec
```

In [7]:
```python
parquet_df = read_parquet_data("depression_data.parquet")
```
```
read_parquet_data() function took - 0.0 Mins 0.463 Sec
```

In [8]:
```python
df = parquet_df
```

In [9]:
```python
df.rename(columns={"Chronic Medical Conditions":"Depression"}, inplace=True)
```

In [10]:
```python
df.head(3)
```

Out[10]:

	Name	Age	Marital Status	Education Level	Number of Children	Smoking Status	Physical Activity Level	Employment Status	Income	Alcohol Consumption	Dietary Habits	Sleep Patterns	History of Mental Illness	History of Substance Abuse	Family History of Depression	Depression
0	Christine Barker	31	Married	Bachelor's Degree	2	Non-smoker	Active	Unemployed	26265.67	Moderate	Moderate	Fair	Yes	No	Yes	Yes
1	Jacqueline Lewis	55	Married	High School	1	Non-smoker	Sedentary	Employed	42710.36	High	Unhealthy	Fair	Yes	No	No	Yes
2	Shannon Church	78	Widowed	Master's Degree	1	Non-smoker	Sedentary	Employed	125332.79	Low	Unhealthy	Good	No	No	Yes	No

Inspection of Dataset

In [11]:
```python
df.shape
```

Out[11]:
```
(413768, 16)
```

In [12]:
```python
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 413768 entries, 0 to 413767
Data columns (total 16 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   Name                          413768 non-null  object 
 1   Age                           413768 non-null  int64  
 2   Marital Status                413768 non-null  object 
 3   Education Level               413768 non-null  object 
 4   Number of Children            413768 non-null  int64  
 5   Smoking Status                413768 non-null  object 
 6   Physical Activity Level       413768 non-null  object 
 7   Employment Status             413768 non-null  object 
 8   Income                        413768 non-null  float64
 9   Alcohol Consumption           413768 non-null  object 
 10  Dietary Habits                413768 non-null  object 
 11  Sleep Patterns                413768 non-null  object 
 12  History of Mental Illness     413768 non-null  object 
 13  History of Substance Abuse    413768 non-null  object 
 14  Family History of Depression  413768 non-null  object 
 15  Depression                    413768 non-null  object 
dtypes: float64(1), int64(2), object(13)
memory usage: 50.5+ MB
```

In [13]:
```python
df["Number of Children"].value_counts()
```

Out[13]:
```
Number of Children
0    155232
2     83961
1     83925
3     76974
4     13676
Name: count, dtype: int64
```

In [14]:
```python
df["Number of Children"] = df["Number of Children"].astype("object")
```

Statistical Inferences - 5 Point Summary & describe

In [15]:
```python
df.describe(include="number")
```

Out[15]:

	Age	Income
count	413768.000000	413768.000000
mean	49.000713	50661.707971
std	18.158759	40624.100565
min	18.000000	0.410000
25%	33.000000	21001.030000
50%	49.000000	37520.135000
75%	65.000000	76616.300000
max	80.000000	209995.220000

In [16]:
```python
df.describe(exclude="number")
```

Out[16]:

	Name	Marital Status	Education Level	Number of Children	Smoking Status	Physical Activity Level	Employment Status	Alcohol Consumption	Dietary Habits	Sleep Patterns	History of Mental Illness	History of Substance Abuse	Family History of Depression	Depression
count	413768	413768	413768	413768	413768	413768	413768	413768	413768	413768	413768	413768	413768	413768
unique	196851	4	5	5	3	3	2	3	3	3	2	2	2	2
top	Michael Smith	Married	Bachelor's Degree	0	Non-smoker	Sedentary	Employed	Moderate	Unhealthy	Fair	No	No	No	No
freq	198	240444	124329	155232	247416	176850	265659	173440	170817	196789	287943	284880	302515	277561

Cleaning the Data - Nulls & Duplicates

In [17]:
```
df.duplicated().sum()
```

Out[17]:
```
np.int64(0)
```

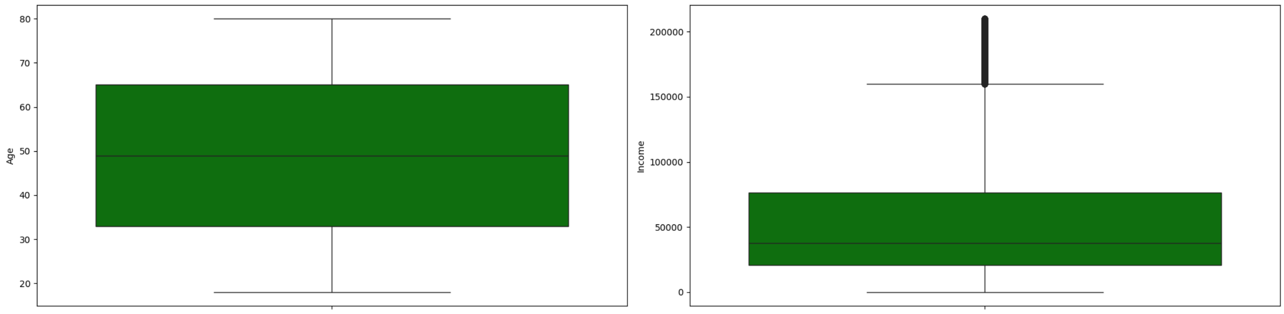
In [18]:
```
df.isnull().sum()
```

Out[18]:
```
Name                            0
Age                             0
Marital Status                  0
Education Level                 0
Number of Children              0
Smoking Status                  0
Physical Activity Level         0
Employment Status               0
Income                          0
Alcohol Consumption             0
Dietary Habits                  0
Sleep Patterns                  0
History of Mental Illness       0
History of Substance Abuse      0
Family History of Depression    0
Depression                      0
dtype: int64
```

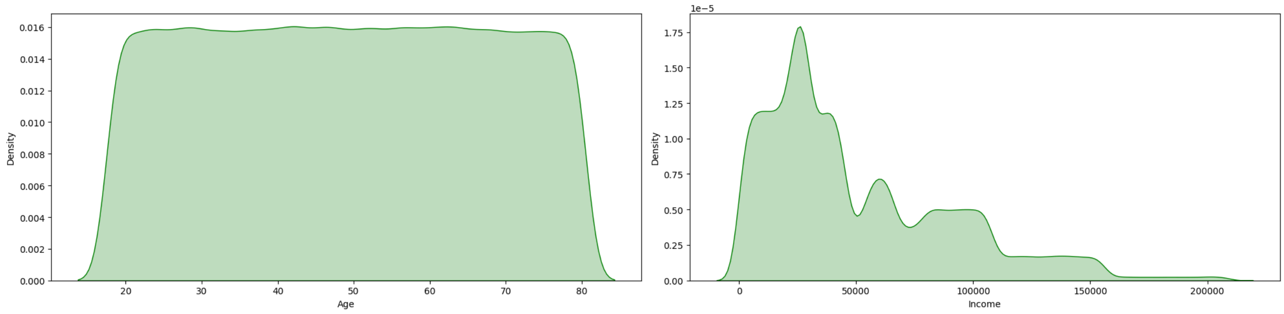
In [19]:
```
df.drop("Name", axis=1, inplace=True)
```

Plots

In [20]:
```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20,5))

for feature, ax_object in zip(df.select_dtypes(include="number").columns, ax.flatten()):
    sns.boxplot(df[feature], ax=ax_object, color="green")


plt.tight_layout()
plt.show()
```



In [25]:
```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20,5))

for feature, ax_object in zip(x_train.select_dtypes(include="number").columns, ax.flatten()):
    sns.kdeplot(x_train[feature], ax=ax_object, color="green", fill=True)


plt.tight_layout()
plt.show()
```



In [21]:
```
fig, axes = plt.subplots(nrows=4, ncols=4, figsize=(20, 7))

for i in range(13):

    feature = ['Marital Status', 'Education Level', 'Number of Children',
       'Smoking Status', 'Physical Activity Level', 'Employment Status',
       'Alcohol Consumption', 'Dietary Habits', 'Sleep Patterns',
       'History of Mental Illness', 'History of Substance Abuse',
       'Family History of Depression', 'Depression']

    category = df[feature[i]].value_counts().index
    percentage = [f"{i:.2f}%" for i in (df[feature[i]].value_counts().values/df.shape[0])*100]

    f_table = pd.DataFrame({"Category":category, "Percentage":percentage})

    axes[i//4, i%4].axis('tight')
    axes[i//4, i%4].axis('off')
    axes[i//4, i%4].table(cellText=f_table.values, colLabels=f_table.columns, loc="center", cellLoc="center", fontsize=15, colColours=["yellow"]*13)
    axes[i//4, i%4].set_title(feature[i])



axes[3, 1].axis('off')
axes[3, 2].axis('off')
axes[3, 3].axis('off')
```

```python
plt.tight_layout()
plt.show()
```

Marital Status

Category	Percentage
Married	58.11%
Single	17.43%
Widowed	16.55%
Divorced	7.91%

Education Level

Category	Percentage
Bachelor's Degree	30.05%
High School	28.74%
Associate Degree	19.33%
Master's Degree	17.83%
PhD	4.05%

Number of Children

Category	Percentage
0	37.52%
2	20.29%
1	20.28%
3	18.60%
4	3.31%

Smoking Status

Category	Percentage
Non-smoker	59.80%
Former	28.08%
Current	12.12%

Physical Activity Level

Category	Percentage
Sedentary	42.74%
Moderate	38.19%
Active	19.07%

Employment Status

Category	Percentage
Employed	64.20%
Unemployed	35.80%

Alcohol Consumption

Category	Percentage
Moderate	41.92%
Low	33.65%
High	24.43%

Dietary Habits

Category	Percentage
Unhealthy	41.28%
Moderate	41.19%
Healthy	17.52%

Sleep Patterns

Category	Percentage
Fair	47.56%
Poor	31.32%
Good	21.12%

History of Mental Illness

Category	Percentage
No	69.59%
Yes	30.41%

History of Substance Abuse

Category	Percentage
No	68.85%
Yes	31.15%

Family History of Depression

Category	Percentage
No	73.11%
Yes	26.89%

Depression

Category	Percentage
No	67.08%
Yes	32.92%

Spliting the data

In [22]:
```python
le_object = LabelEncoder()
df["Depression"] = le_object.fit_transform(df["Depression"])
target = df["Depression"]
df = df.drop(columns="Depression")
```

In [23]:
```python
x_train, x_test, y_train, y_test = train_test_split(df, target, test_size=0.2, random_state=5)
```

In [24]:
```python
print(f'''
x_train shape : {x_train.shape}
x_test shape : {x_test.shape}
y_train shape : {y_train.shape}
y_test shape : {y_test.shape}
''')
```

```
x_train shape : (331014, 14)
x_test shape : (82754, 14)
y_train shape : (331014,)
y_test shape : (82754,)
```

Transformation

Scaling

In [26]:
```python
scaler_objects = {}

for i in x_train.select_dtypes(include="number").columns:
    scaler_objects[i] = MinMaxScaler()
    scaler_objects[i].fit(x_train[[i]])
    x_train[i] = scaler_objects[i].transform(x_train[[i]])
```

In [28]:
```python
for i in x_test.select_dtypes(include="number").columns:
    x_test[i] = scaler_objects[i].transform(x_test[[i]])
```

Encoding

In [31]:
```python
new_df = x_train.select_dtypes(include="number")
new_df.reset_index(inplace=True)
encoder_objects = {}

for i in x_train.select_dtypes(exclude="number").columns:
    # print(i)
    encoder_objects[i] = OneHotEncoder(dtype='int', drop="first")
    dummy_df = encoder_objects[i].fit_transform(x_train[[i]]).toarray()
    new_df = pd.concat([new_df, pd.DataFrame(dummy_df, columns=encoder_objects[i].get_feature_names_out())], axis=1)
    # print(new_df)

x_train = new_df
```

In [32]:
```python
new_df = x_test.select_dtypes(include="number")
new_df.reset_index(inplace=True)

for i in x_test.select_dtypes(exclude="number").columns:
    dummy_df = encoder_objects[i].transform(x_test[[i]]).toarray()
    new_df = pd.concat([new_df, pd.DataFrame(dummy_df, columns=encoder_objects[i].get_feature_names_out())], axis=1)
    # print(new_df)

x_test = new_df
```

In [37]:
```python
x_train.drop(columns="index", inplace=True)
x_test.drop(columns="index", inplace=True)
```

Base Model - Logistic Regression

In [52]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
```

Out[52]:
```
▾ LogisticRegression
LogisticRegression()
```

In [53]:
```
y_pred = model.predict(x_test)
```

In [60]:
```
np.unique(y_pred, return_counts=True)
```

Out[60]:
```
(array([0]), array([82754]))
```

In [56]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.67      1.00      0.80     55526
           1       0.00      0.00      0.00     27228

    accuracy                           0.67     82754
   macro avg       0.34      0.50      0.40     82754
weighted avg       0.45      0.67      0.54     82754
```

In [ ]: