In [2]:

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

# Importing data
data = pd.read_csv("/Users/kachu/Desktop/DS/smoking_driking_dataset_Ver01.cs
data
```

```
/Users/kachu/anaconda3/lib/python3.11/site-packages/pandas/core/arrays/maske
d.py:60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottlenec
k' (version '1.3.5' currently installed).
  from pandas.core import (
```

Out[2]:

	sex	age	height	weight	waistline	sight_left	sight_right	hear_left	hear_right
0	Male	35	170	75	90.0	1.0	1.0	1.0	1.0
1	Male	30	180	80	89.0	0.9	1.2	1.0	1.0
2	Male	40	165	75	91.0	1.2	1.5	1.0	1.0
3	Male	50	175	80	91.0	1.5	1.2	1.0	1.0
4	Male	50	165	60	80.0	1.0	1.2	1.0	1.0
...	...	...	...	...	...	...	...	...	...
991341	Male	45	175	80	92.1	1.5	1.5	1.0	1.0
991342	Male	35	170	75	86.0	1.0	1.5	1.0	1.0
991343	Female	40	155	50	68.0	1.0	0.7	1.0	1.0
991344	Male	25	175	60	72.0	1.5	1.0	1.0	1.0
991345	Male	50	160	70	90.5	1.0	1.5	1.0	1.0

991346 rows × 24 columns

This is a supervised learning for the question "is he drinker or not?".

About Dataset

Colunm Description(US)

Sex male, female

age round up to 5 years

height round up to 5 cm[cm]

weight [kg] 몸무게

sight_left eyesight(left)

sight_right eyesight(right)

hear_left hearing left, 1(normal), 2(abnormal)

hear_right hearing right, 1(normal), 2(abnormal)

SBP Systolic blood pressure[mmHg]

DBP Diastolic blood pressure[mmHg]

BLDS BLDS or FSG(fasting blood glucose)[mg/dL]

tot_chole total cholesterol[mg/dL]

HDL_chole HDL cholesterol[mg/dL]

LDL_chole LDL cholesterol[mg/dL]

triglyceride triglyceride[mg/dL]

hemoglobin hemoglobin[g/dL]

urine_protein protein in urine, 1(-), 2(+/-), 3(+1), 4(+2), 5(+3), 6(+4)

serum_creatinine serum(blood) creatinine[mg/dL]

SGOT_AST SGOT(Glutamate-oxaloacetate transaminase) AST(Aspartate transaminase) [IU/L]

SGOT_ALT ALT(Alanine transaminase)[IU/L]

gamma_GTP y-glutamyl transpeptidase[IU/L]

SMK_stat_type_cd Smoking state, 1(never), 2(used to smoke but quit), 3(still smoke)

DRK_YN Drinker or Not

In [ ]:
```
#Converting string values to numeric values.
```

In [3]:
```
data = data.drop(['SMK_stat_type_cd'], axis = 1)
data.DRK_YN = [1 if each == 'Y' else 0 for each in data.DRK_YN]
data.sex = [1 if each == 'Male' else 0 for each in data.sex]
```

In [4]:
```
x_data = data.drop(['DRK_YN'], axis = 1)
y_data = data.DRK_YN.values
```

In [5]:
```
x = (x_data - np.min(x_data))/(np.max(x_data) - np.min(x_data))
x = x.values
```

In [ ]:
```
#Train-Test split.
```

In [6]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y_data, test_size = 0


# transpose
x_train = x_train.T
x_test = x_test.T
y_train = y_train.T
y_test = y_test.T

# check the shapes
print('x_train :', x_train.shape)
print('x_test :', x_test.shape)
print('y_train :', y_train.shape)
print('y_test :', y_test.shape)
```

```
x_train : (22, 793076)
x_test : (22, 198270)
y_train : (793076,)
y_test : (198270,)
```

In [ ]:
```
#Linear Regression
```

In [7]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(x_train.T,y_train.T)
```

Out[7]:
```
▾ LinearRegression
LinearRegression()
```

In [8]:
```
# Accuracy

print("Accuracy of Linear Regression:",reg.score(x_test.T,y_test.T))
```

```
Accuracy of Linear Regression: 0.23906246267589037
```

In [9]:
```
from sklearn.model_selection import cross_val_score
reg = LinearRegression()
k = 3
cv_result = cross_val_score(reg,x_train.T,y_train.T,cv=k) # uses R^2 as scor
print('CV Scores: ',cv_result)
print('CV scores average: ',np.sum(cv_result)/k)
```

```
CV Scores:  [0.24274318 0.23072655 0.24097592]
CV scores average:  0.2381485477081533
```

In [ ]:
```
#Decision Tree Regression
```

In [10]:
```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(x_train.T,y_train.T)
```

Out[10]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [11]:
```
print("Accuracy of Decision Tree Regression:",dt.score(x_test.T,y_test.T))
```

```
Accuracy of Decision Tree Regression: -0.44319443632701794
```

In [ ]:
```
#Random Forest Regression
```

In [13]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=30, random_state=42)
rf.fit(x_train.T,y_train.T)
```

Out[13]:
```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=30, random_state=42)
```

In [14]:
```
print("Accuracy of Random Forest Regression:",rf.score(x_test.T,y_test.T))
```

```
Accuracy of Random Forest Regression: 0.25856794266589667
```

In [ ]:
```
#Logistic Regression
```

In [15]:
```
from sklearn.linear_model import LogisticRegression

logReg = LogisticRegression(max_iter= 50)
logReg.fit(x_train.T, y_train.T)
```

```
/Users/kachu/anaconda3/lib/python3.11/site-packages/sklearn/linear_model/_lo
gistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
```

Out[15]:

```
▾ LogisticRegression
LogisticRegression(max_iter=50)
```

In [16]:

```python
print("Accuracy of Logistic Regression:".format(logReg.score(x_test.T, y_tes
```

```
Accuracy of Logistic Regression:
```

In [17]:

```python
from sklearn.linear_model import SGDRegressor

svm = SGDRegressor()
svm.fit(x_train.T,y_train.T)
```

Out[17]:

```
▾ SGDRegressor
SGDRegressor()
```

In [18]:

```python
print("Accuracy of SVM:",svm.score(x_test.T,y_test.T))
```

```
Accuracy of SVM: 0.006278944532603470
```

In [ ]:

```python
pip uninstall scikit-learn threadpoolctl --yes
```

In [ ]:

```python

```

In [ ]:

```python
pip install scikit-learn threadpoolctl
```

In [ ]:

```python
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors=7) #n_neighbors=k
knn.fit(x_train.T,y_train.T)
prediction = knn.predict(x_test.T)
```

In [ ]:

```python
print("{} nn score: {}".format(3,knn.score(x_test.T,y_test.T)))
```

In [ ]:

```python
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train.T, y_train.T)
```

In [ ]:
```
print('Decision Tree Classifier accuracy -> ', dt.score(x_test.T, y_test.T))
```

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators = 100, random_state = 1)
rfc.fit(x_train.T, y_train.T)
```

In [ ]:
```
print('Random Forest Classifier accuracy -> ', rfc.score(x_test.T, y_test.T)
```

Problem Explanation:

The problem revolves around predicting whether an individual is a drinker or not based on various demographic and health-related features. This classification task is crucial for understanding the prevalence of drinking habits within a population and can inform public health interventions and policies.

Approach:

1. Data Preprocessing: Initially, the dataset was loaded and explored. String values were converted into numeric values, and the data was normalized to ensure uniformity across features.

2. Model Training and Evaluation: Several regression and classification algorithms were applied and evaluated:

    - Regression: Linear Regression, Decision Tree Regression, Random Forest Regression, SVM, and KNN.
    - Classification: Logistic Regression, Decision Tree Classifier, and Random Forest Classifier.
3. Evaluation Metrics: The accuracy score was used to evaluate the performance of the models. Additionally, cross-validation was employed to assess model robustness.

Findings:

1. Regression Models: Linear Regression and Random Forest Regression achieved moderate accuracies of approximately 24% and 26%, respectively. However, Decision Tree Regression and SVM performed poorly.

2. Classification Models: Decision Tree Classifier achieved an accuracy of about 64%, indicating good predictive performance. Random Forest Classifier outperformed all other models with an accuracy of approximately 73%.

Further Research Ideas:

1. Feature Engineering: Exploring additional features or deriving new ones from existing ones could enhance model performance. For example, combining certain health indicators or creating interaction terms might capture more nuanced relationships.
2. Ensemble Techniques: Investigating advanced ensemble methods, such as Gradient Boosting or Stacking, could potentially yield even better results by leveraging the strengths of multiple models.
3. Data Augmentation: Collecting more diverse and comprehensive data, particularly regarding drinking habits and associated behaviors, could improve model generalization and robustness.

Recommendations:

1. Public Health Interventions: Use the insights from the Random Forest Classifier to identify high-risk groups for targeted interventions, such as educational campaigns or counseling services, aimed at reducing alcohol consumption.
2. Policy Development: Incorporate the findings into policy-making processes to implement regulations or initiatives addressing alcohol-related health issues, considering the demographics and health indicators identified as significant predictors.
3. Individual Health Management: Develop personalized health management strategies based on the Decision Tree Classifier's predictions to offer tailored advice and support for individuals seeking to modify their drinking behaviors.

In [ ]: