In [1]:

```
## ====================================
## EET-4501 - Applied Machine Learning
## Assignment 2
## ====================================
```

In [2]:

```
## Part 1: Data Loading and Exploration

import pandas as pd

inputData = pd.read_csv("Ass2_dataset.csv")

inputData.head()

inputData.tail(10)

inputData.shape

inputData['age'].describe()

trainingData = inputData.iloc[150:250]

features = trainingData[['bmi', 'charges']]

labels = trainingData.iloc[:, -1]

trainingData2F  = pd.concat([features, labels], axis=1)
trainingData2F.head()
```

Out[2]:

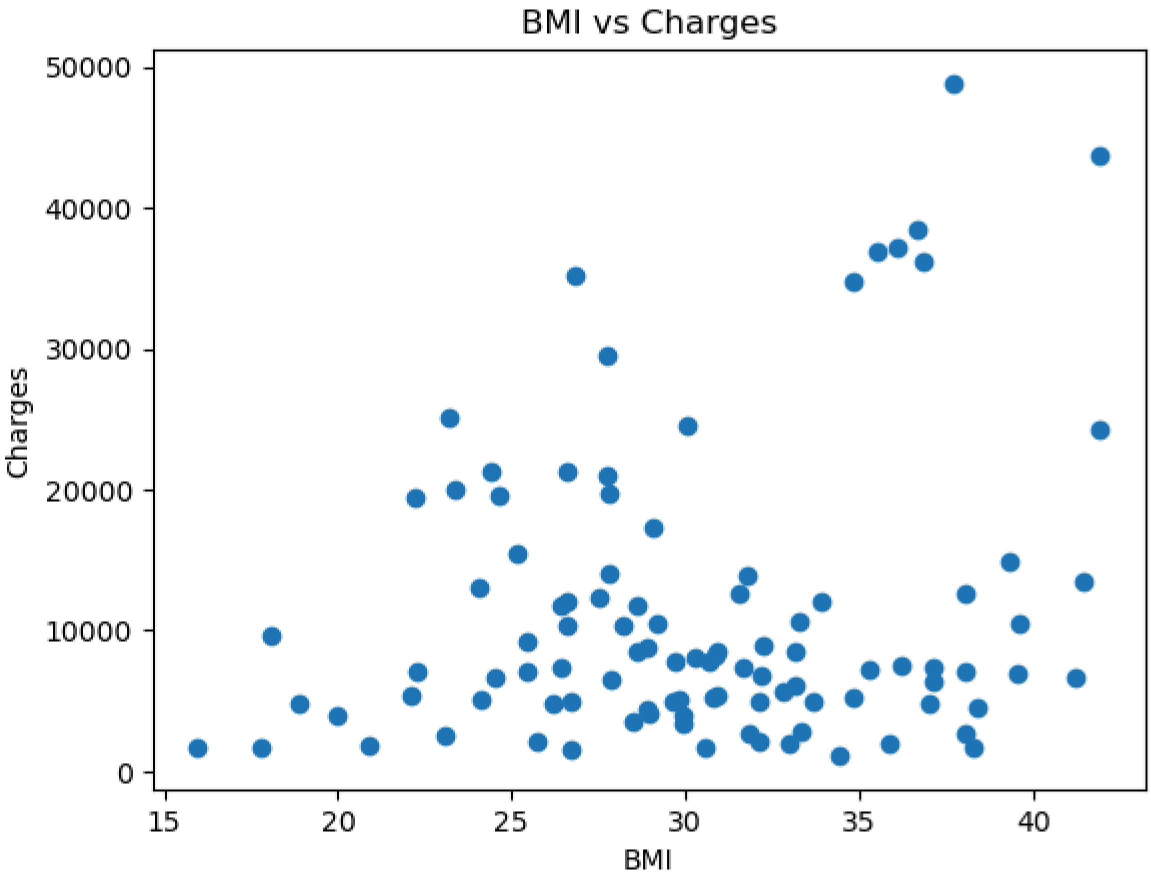
	bmi	charges	insuranceclaim
150	24.130	5125.21570	0
151	29.700	7789.63500	1
152	37.145	6334.34355	0
153	23.370	19964.74630	1
154	25.460	7077.18940	0

In [3]:

```
## Part 2: Data Visualization

import matplotlib.pyplot as plt

plt.figure()
plt.scatter(trainingData2F['bmi'], trainingData2F['charges'])
plt.xlabel('BMI')
plt.ylabel('Charges')
plt.title('BMI vs Charges')
plt.show()
```



In [4]:
```
## Part 3: Feature Engineering & Scaling

from sklearn.preprocessing import MinMaxScaler

minmax_scaler = MinMaxScaler()
features_normalized = minmax_scaler.fit_transform(features)

features_normalized

from sklearn.preprocessing import StandardScaler

standard_scaler = StandardScaler()
features_standardized = standard_scaler.fit_transform(features)

features_standardized
```

```
Out[4]:  array([[-1.07931651, -0.59993178],
                [-0.10972722, -0.33845565],
                [ 1.18624984, -0.4812725 ],
                [-1.21161235,  0.85636415],
                [-0.84779878, -0.40837239],
                [ 1.59967436, -0.42098178],
                [-1.0288352 ,  0.97991078],
                [-0.89740972,  0.41999482],
                [ 0.90512117,  2.5232585 ],
                [-0.43524463,  0.83522924],
                [-0.64935501,  0.99218077],
                [ 1.13489817,  2.44467362],
                [ 1.61360024, -0.07732323],
                [-0.09231987, -0.59729012],
                [-0.12017163, -0.60945778],
                [-0.36822634, -0.08158883],
                [ 1.16100919, -0.62884131],
                [ 0.49169666, -0.50144403],
                [ 0.26017893, -0.83604132],
                [-1.98885044, -0.62910874],
                [ 1.93911765,  0.21265353],
                [-0.00528313, -0.30636068],
                [-2.50149684, -0.93658027],
                [ 0.77804753, -0.58807385],
                [ 0.52477062, -0.8226793 ],
                [ 1.28286063,  3.68854734],
                [-0.43437426, -0.46934707],
                [-0.19676396, -0.07874188],
                [-0.248986  , -0.2370177 ],
                [ 0.49169666, -0.26498561],
                [-0.30207841,  0.04881395],
                [ 1.38382324, -0.94276042],
                [-1.80694365, -0.70982409],
                [-0.68242897, -0.37478148],
                [ 0.06260553, -0.34416796],
                [ 2.01309888,  3.19088732],
                [-0.07143105, -0.71212496],
                [ 0.09916096, -0.58026181],
                [ 0.32545648, -0.43793382],
                [ 0.30978987, -0.61978472],
                [ 0.21579019,  0.12945509],
                [-0.7189844 , -0.62361693],
                [-0.7990582 , -0.89311993],
                [-0.64935501,  0.07908535],
                [ 0.71364034, -0.99127423],
                [ 0.04519818, -0.94200066],
                [ 0.42990057, -0.54845938],
                [-0.30120805, -0.26709156],
                [-2.13768327, -0.15645044],
                [ 1.5666004 ,  0.35947777],
                [ 0.30978987, -0.89380469],
                [ 0.33067869, -0.23231965],
                [-1.09585349,  0.17406816],
                [ 1.00086159,  2.54128039],
                [-1.39787097, -0.40151114],
                [-0.25246747, -0.67721216],
                [-0.6841697 ,  0.04954212],
                [-0.45091124,  0.95639908],
                [ 0.25582709,  0.25932308],
                [ 1.89734001, -0.45420984],
                [ 0.46471527, -0.90858459],
                [ 0.09480912, -0.30184286],
                [-0.3186154 , -0.75572433],
                [-0.62672546, -0.61194689],
                [ 0.09916096, -0.26677782],
                [ 1.17841654, -0.37946316],
                [-0.64935501, -0.08663744],
                [-1.25861219, -0.8591567 ],
                [-0.07143105, -0.76992721],
                [-1.23946411,  1.35852947],
                [ 0.5865667 , -0.61099612],
                [ 0.50823364, -0.06610306],
                [ 0.08175361, -0.58734008],
                [ 0.77804753,  2.31023986],
                [-0.99053903,  0.81228089],
                [ 0.61789993,  0.07347453],
                [ 1.34552708, -0.83896424],
                [ 2.01570998,  1.27467886],
                [ 0.22710497, -0.38079746],
                [-0.84779878, -0.19756915],
                [ 1.02088004, -0.37241   ],
                [-0.43524463,  0.27113257],
                [-2.18120164, -0.9333429 ],
                [-0.49268888,  0.10749442],
                [-1.01316858, -0.44438815],
                [-1.41179685,  0.80528615],
                [-0.62672546, -0.94433595],
                [ 1.40297133, -0.66489902],
                [-0.21939351,  0.60002578],
                [ 1.34552708, -0.40096487],
                [ 1.10356494,  2.67648566],
                [-1.4265931 , -0.57747242],
                [-0.61454031,  2.34758262],
                [ 0.86508427, -0.39662768],
                [-0.45091124,  1.79439161],
                [-0.0540237 ,  1.29912865],
                [ 1.34552708,  0.13839512],
                [ 0.96256542, -0.90791104],
                [-1.64157385, -0.9231064 ],
                [-0.23593049, -0.706376  ]])
```

In [5]:

```
## Part 4: Sampling & Cross-Validation

from sklearn.model_selection import train_test_split

X = features
y = labels

X_train, X_valandtest, y_train, y_valandtest = train_test_split(
    X, y, test_size=0.30, stratify=y, random_state=42
)
```

```
X_val, X_test, y_val, y_test = train_test_split(
    X_valandtest, y_valandtest, test_size=0.50, stratify=y_valandtest, random_state
)

from sklearn.model_selection import KFold, cross_val_score
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(random_state=42)
kf = KFold(n_splits=5, shuffle=True, random_state=42)

cv_scores = cross_val_score(model, X_train, y_train, cv=kf)

cv_scores

print("Accuracy per fold:", cv_scores)
print("Mean accuracy:", cv_scores.mean())

model.fit(X_train, y_train)
importances = model.feature_importances_

feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
})

feature_importance_df

## Stratified sampling ensures that each data split maintains the same class as the
## Feature importance helps find which input variables contribute most to the model
```

```
Accuracy per fold: [0.57142857 0.5        0.57142857 0.64285714 0.42857143]
Mean accuracy: 0.5428571428571428
```

Out[5]:

	Feature	Importance
0	bmi	0.371575
1	charges	0.628425

In [6]:

```
## Part 5: Data Export

trainingData2F.to_csv("Ass2Output_Lieu.csv", index=False)
```