Classification Task

```
# Lakukan encoding pada fitur kategori
le = preprocessing.LabelEncoder()
X['Gender'] = le.fit_transform(X['Gender'])
X_encoded = pd.get_dummies(X)

# Bagi data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, 
test_size=0.2, random_state=42)

# Pilih algoritma Decision Tree
model = DecisionTreeClassifier()

# Latih model
model.fit(X_train, y_train)

# Prediksi menggunakan data pengujian
y_pred = model.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print("CLASSIFICATION DECISION TREE")
print(f'Accuracy: {accuracy}')
print('Classification Report:\n', report)
```

Output:

```
CLASSIFICATION DECISION TREE
Accuracy                   : 1.0
Classification Report  :
              precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
         Low       1.00      1.00      1.00        55
      Medium       1.00      1.00      1.00        63
    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```

```
# Lakukan encoding pada fitur kategori
le = preprocessing.LabelEncoder()
X['Gender'] = le.fit_transform(X['Gender'])
X_encoded = pd.get_dummies(X)
```

```
# Bagi data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
test_size=0.2, random_state=42)

# Pilih algoritma Regresi Logistik
model = LogisticRegression(max_iter=1000)  # Maksimum iterasi untuk
konvergensi

# Latih model
model.fit(X_train, y_train)

# Prediksi menggunakan data pengujian
y_pred = model.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("CLASSIFICATION WITH BINARY / MULTINOMIAL LOGISTIC REGRESSION")
print(f'Accuracy: {accuracy}')
print('Classification Report:\n', report)
```

Output:

```
CLASSIFICATION WITH BINARY / MULTINOMIAL LOGISTIC REGRESSION
Accuracy                 : 1.0
Classification Report   :
                precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
         Low       1.00      1.00      1.00        55
      Medium       1.00      1.00      1.00        63
    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
```

```
# Lakukan encoding pada fitur kategori
le = preprocessing.LabelEncoder()
X['Gender'] = le.fit_transform(X['Gender'])
X_encoded = pd.get_dummies(X)

# Bagi data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
test_size=0.2, random_state=42)

# Pilih algoritma K-Nearest Neighbors
```

```
k_value = 5  # Ganti dengan nilai K yang diinginkan
model = KNeighborsClassifier(n_neighbors=k_value)

# Latih model
model.fit(X_train, y_train)

# Prediksi menggunakan data pengujian
y_pred = model.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("CLASSIFICATION WITH KNN")
print(f'Accuracy: {accuracy}')
print('Classification Report:\n', report)
```

Output:

```
CLASSIFICATION WITH KNN
Accuracy                 : 0.995
Classification Report  :
               precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
         Low       1.00      0.98      0.99        55
      Medium       0.98      1.00      0.99        63
    accuracy                           0.99       200
   macro avg       0.99      0.99      0.99       200
weighted avg       1.00      0.99      0.99       200
```

```
# Lakukan encoding pada fitur kategori
le = preprocessing.LabelEncoder()
X['Gender'] = le.fit_transform(X['Gender'])
X_encoded = pd.get_dummies(X)

# Bagi data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
test_size=0.2, random_state=42)

# Pilih algoritma Naive Bayes (Gaussian Naive Bayes untuk fitur numerik)
model = GaussianNB()

# Latih model
model.fit(X_train, y_train)
```

```
# Prediksi menggunakan data pengujian
y_pred = model.predict(X_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("CLASSIFICATION WITH NAIVE BAYES")
print(f'Accuracy: {accuracy}')
print('Classification Report:\n', report)
```

Output:

```
CLASSIFICATION WITH NAIVE BAYES
Accuracy                 : 0.895
Classification Report  :
               precision    recall  f1-score   support

        High       0.87      0.96      0.91        82
         Low       1.00      0.85      0.92        55
      Medium       0.85      0.84      0.85        63
    accuracy                           0.90       200
   macro avg       0.91      0.89      0.89       200
weighted avg       0.90      0.90      0.90       200
```

```
# Lakukan encoding pada fitur kategori
le = preprocessing.LabelEncoder()
X['Gender'] = le.fit_transform(X['Gender'])
X_encoded = pd.get_dummies(X)

# Bagi data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
test_size=0.2, random_state=42)

# Pilih algoritma Decision Tree Classifier
model = DecisionTreeClassifier()

# Latih model
model.fit(X_train, y_train)

# Menampilkan decision tree sebagai teks
tree_rules = export_text(model, feature_names=list(X_encoded.columns))
print("Decision Tree Rules:\n", tree_rules)
```

Output:

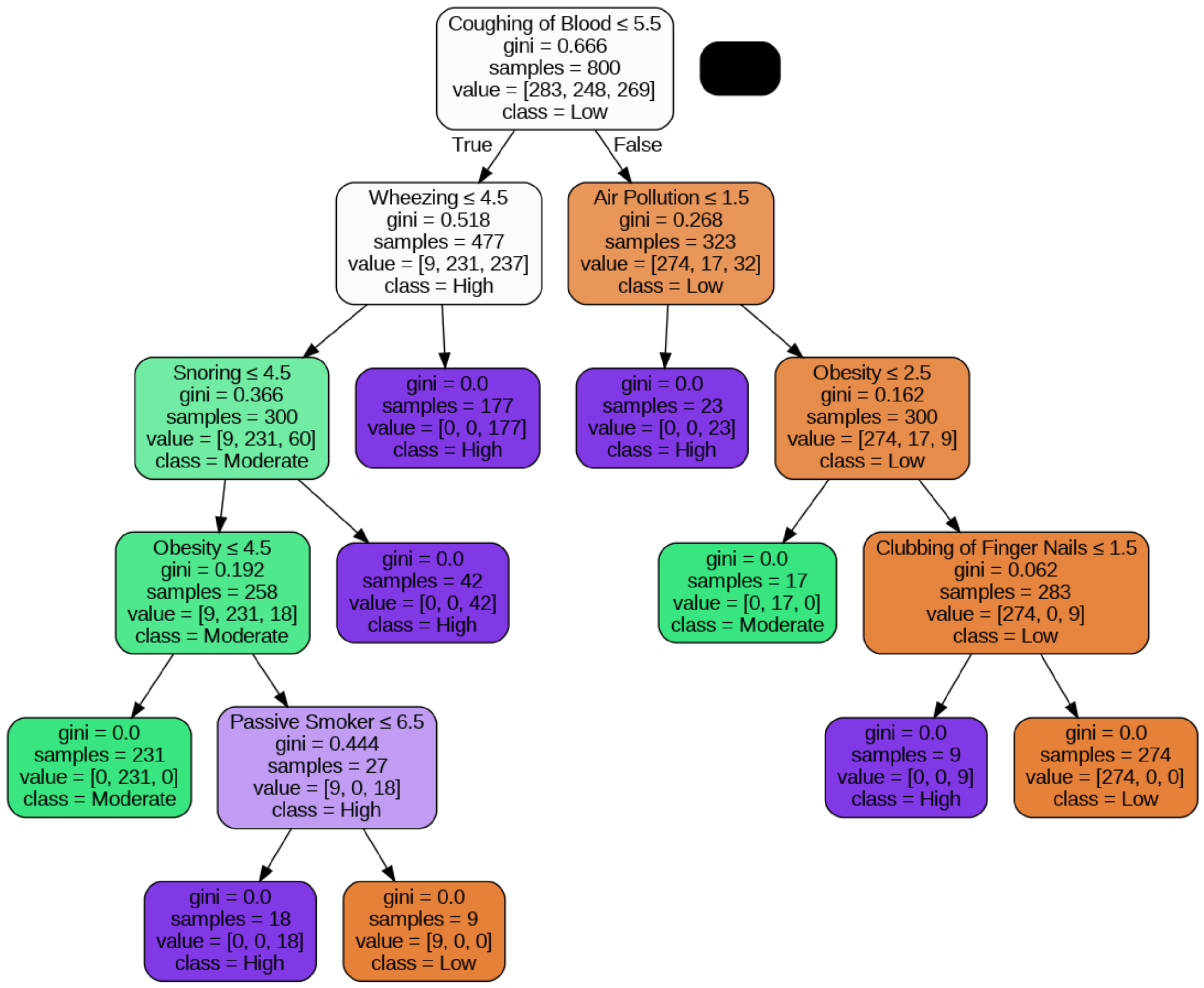
```
Decision Tree Rules:
 |--- Coughing of Blood <= 5.50
|   |--- Wheezing <= 4.50
|   |   |--- Snoring <= 4.50
|   |   |   |--- Obesity <= 4.50
|   |   |   |   |--- class: Low
|   |   |   |--- Obesity >  4.50
|   |   |   |   |--- Passive Smoker <= 6.50
|   |   |   |   |   |--- class: Medium
|   |   |   |   |--- Passive Smoker >  6.50
|   |   |   |   |   |--- class: High
|   |   |--- Snoring >  4.50
|   |   |   |--- class: Medium
|   |--- Wheezing >  4.50
|   |   |--- class: Medium
|--- Coughing of Blood >  5.50
|   |--- Air Pollution <= 1.50
|   |   |--- class: Medium
|   |--- Air Pollution >  1.50
|   |   |--- Obesity <= 2.50
|   |   |   |--- class: Low
|   |   |--- Obesity >  2.50
|   |   |   |--- Clubbing of Finger Nails <= 1.50
|   |   |   |   |--- class: Medium
|   |   |   |--- Clubbing of Finger Nails >  1.50
|   |   |   |   |--- class: High
```

```python
from sklearn.tree import export_graphviz
import pydotplus
from IPython.display import Image

# Menampilkan decision tree sebagai grafik
dot_data = export_graphviz(model, out_file=None,
feature_names=list(X_encoded.columns),
                           class_names=['Low', 'Moderate', 'High'],
                           filled=True, rounded=True,
special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Output:


Coughing of Blood ≤ 5.5
gini = 0.666
samples = 800
value = [283, 248, 269]
class = Low
[redacted]
True
False
Wheezing ≤ 4.5
gini = 0.518
samples = 477
value = [9, 231, 237]
class = High
Air Pollution ≤ 1.5
gini = 0.268
samples = 323
value = [274, 17, 32]
class = Low
Snoring ≤ 4.5
gini = 0.366
samples = 300
value = [9, 231, 60]
class = Moderate
gini = 0.0
samples = 177
value = [0, 0, 177]
class = High
gini = 0.0
samples = 23
value = [0, 0, 23]
class = High
Obesity ≤ 2.5
gini = 0.162
samples = 300
value = [274, 17, 9]
class = Low
Obesity ≤ 4.5
gini = 0.192
samples = 258
value = [9, 231, 18]
class = Moderate
gini = 0.0
samples = 42
value = [0, 0, 42]
class = High
gini = 0.0
samples = 17
value = [0, 17, 0]
class = Moderate
Clubbing of Finger Nails ≤ 1.5
gini = 0.062
samples = 283
value = [274, 0, 9]
class = Low
gini = 0.0
samples = 231
value = [0, 231, 0]
class = Moderate
Passive Smoker ≤ 6.5
gini = 0.444
samples = 27
value = [9, 0, 18]
class = High
gini = 0.0
samples = 9
value = [0, 0, 9]
class = High
gini = 0.0
samples = 274
value = [274, 0, 0]
class = Low
gini = 0.0
samples = 18
value = [0, 0, 18]
class = High
gini = 0.0
samples = 9
value = [9, 0, 0]
class = Low