Autor: Tomasz Mazur
Index: 401133

Wstęp - opis zbioru danych

Do analizy został wykorzystany zbiór danych zawierający informacje na temat pacjentów, którym wykonano badanie kości pod względem możliwej osteoporozy. Zbiór składa się z 169 rekordów. Każdy z nich zawiera dane jednej osoby tj. dane personalne jak wiek, płeć oraz dane związane z ich zdrowiem tj. waga, stopień złamania itp. Najważniejszymi informacjami w tym zbiorze są:

- wskaźnik `bmd` (z ang. *bone mineral density*) - określa poziom wapnia oraz innych typów minerałów w obszarze kości. Jego wartość pozwala wykryć osteoporozę a także przewidzieć ryzyko złamania kości.
- zmienna `fracture` - określa czy u danego pacjenta doszło do złamania, czy nie
- pozostałe zmienne dotyczące samego pacjenta (tj. wiek, płeć itp) oraz sposobu leczenia

Wczytanie zbioru danych i wyświetlenie pierwszych 5 wierszy

In [ ]:
```
import pandas as pd
df = pd.read_csv('data/bmd_prep.csv')
df.head()
```

Out[ ]:

	age	sex	fracture	weight_kg	height_cm	medication	waiting_time	bmd
0	57.052768	F	no fracture	64.0	155.5	Anticonvulsant	18	0.8793
1	75.741225	F	no fracture	78.0	162.0	No medication	56	0.7946
2	70.778900	M	no fracture	73.0	170.5	No medication	10	0.9067
3	78.247175	F	no fracture	60.0	148.0	No medication	14	0.7112
4	54.191877	M	no fracture	55.0	161.0	No medication	20	0.7909

W ramach analizy zostały sformułowane dwie hipotezy badawcze:

1. Czy płeć, wiek i waga pacjenta wpływają na wartość wskaźnika `bmd`?

- zmienne opisujące (niezależne): `sex` (jak.), `age` (il.), `weight_kg` (il.)
- zmienna objaśniana (zależna): `bmd` (il.)

2. Czy rodzaj leczenia, wiek i wartość wskaźnika `bmd` pozwalają przewidzieć, czy wystąpi złamanie (`fracture`) czy nie?

- zmienne opisujące (niezależne): `medication` (jak.), `age` (il.), `bmd` (il.)
- zmienna objaśniana (zależna): `fracture` (jak.)

Hipoteza 1 - badanie zależności płci, wieku i wagi na wskaźnik `bmd` osoby dorosłej

Badanie struktury populacji

In [ ]:
```
# Potrzebne biblioteki
import numpy as np
import matplotlib.pyplot as plt
```

1. Statystyki opisowe - zmienne ilościowe

In [ ]:
```
numeric_vars = ['age', 'weight_kg', 'bmd']
np.round(df[numeric_vars].describe(),3)
```

Out[ ]:

	age	weight_kg	bmd
count	169.000	169.000	169.000
mean	63.632	64.666	0.783
std	12.357	11.537	0.167
min	35.814	36.000	0.408
25%	54.424	56.000	0.671
50%	63.488	64.500	0.786
75%	72.081	73.000	0.889
max	88.754	96.000	1.362

Najstarsza osoba w zbiorze ma niemal 89 lat natomiast najmłodsza niecałe 36. Waga pacjentów mieści się w zakresie od 36 do 96 kg. Średnia wartość wskaźnika `bmd` wynosi 0.783 a mediana 0.786.

wariancja

In [ ]:
```
np.round(df[numeric_vars].var(), 3)
```

Out[ ]:
```
age          152.694
weight_kg    133.106
bmd            0.028
dtype: float64
```

moda

In [ ]:
```
np.round(df[numeric_vars].mode(), 3)
```

Out[ ]:

	age	weight_kg	bmd
0	35.814	55.0	0.690
1	36.648	59.0	0.713
2	37.462	62.0	0.889
3	39.255	NaN	NaN
4	40.232	NaN	NaN
...	...	...	...
164	85.902	NaN	NaN
165	86.860	NaN	NaN
166	87.935	NaN	NaN
167	88.024	NaN	NaN
168	88.754	NaN	NaN

169 rows × 3 columns

Najczęsciej występujące w zbiorze wagi pacjentów to 55, 59 i 62 kg.

2. Tabele liczności - zmienne jakościowe

In [ ]:

```
categorical_vars = 'sex'
df[categorical_vars].value_counts()
```

Out[ ]:

```
M    86
F    83
Name: sex, dtype: int64
```

Dane cechują się równomiernym rozkładem pod względem płci - podobna ilość mężczyzn i kobiet

3. Tabela wielodzielcza

In [ ]:

```
# group values in numeric features to equal ranges
df = pd.read_csv('data/bmd_prep.csv')
df['age'] = pd.cut(df['age'], 6, precision=0)
df['weight_kg'] = pd.cut(df['weight_kg'], 3, precision=0)

pivot = np.round(pd.pivot_table(df, values='bmd',
                                index=['sex','weight_kg', ],
                                columns=['age'],
                                aggfunc=['mean'],
                                margins=True,
                                margins_name='Checksum',
                                fill_value='-'),2)
pivot
```

Out[ ]:

		mean						
	age	(36.0, 45.0]	(45.0, 53.0]	(53.0, 62.0]	(62.0, 71.0]	(71.0, 80.0]	(80.0, 89.0]	Checksum
sex	weight_kg							
F	(36.0, 56.0]	0.8888	0.71702	0.671383	0.700929	0.584114	0.551625	0.6568
	(56.0, 76.0]	-	0.825856	0.828788	0.798714	0.5757	0.8171	0.790464
	(76.0, 96.0]	-	-	0.85175	0.720533	0.6518	0.9061	0.75935
M	(36.0, 56.0]	0.762	-	0.756567	0.64615	0.5287	0.61185	0.675612
	(56.0, 76.0]	1.027325	0.822356	0.860767	0.807264	0.839067	0.7478	0.838392
	(76.0, 96.0]	0.88595	0.89415	0.92668	0.9745	1.0015	0.70525	0.917216
Checksum		0.892545	0.808292	0.823289	0.800995	0.694108	0.674905	0.783104

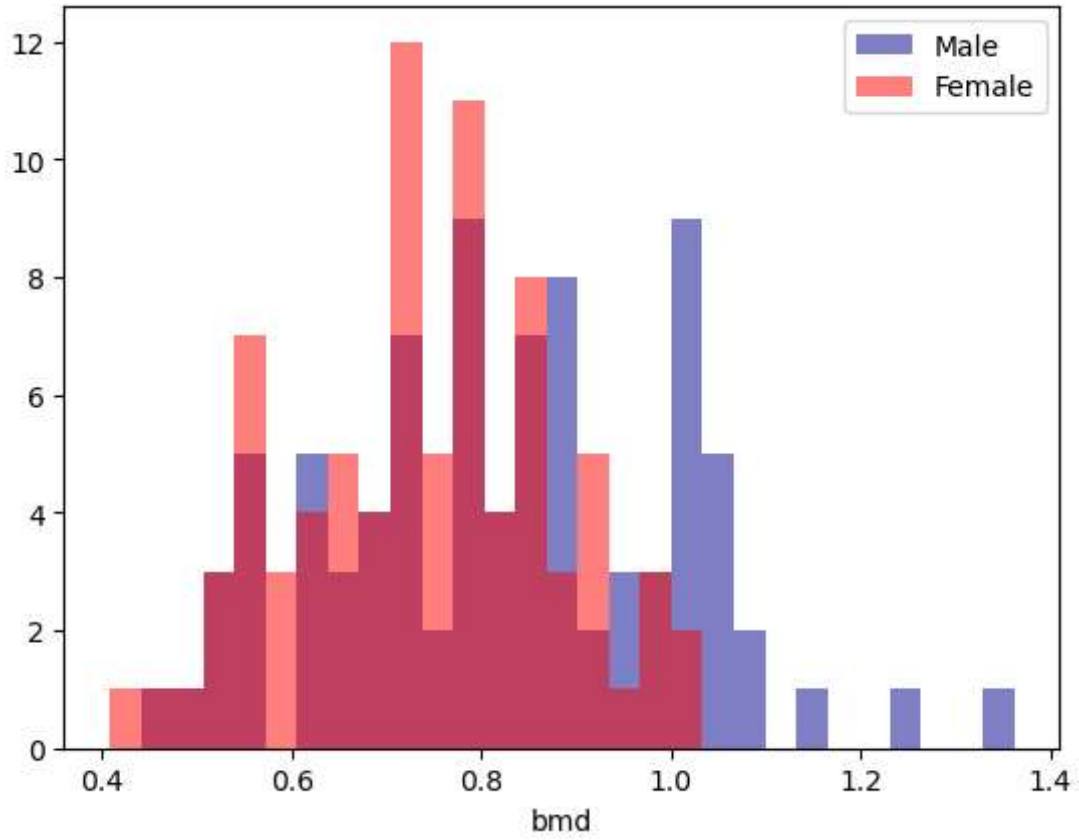
Na otrzymanej tabeli widzimy, że:

- Wraz z wiekiem kości są coraz słabsze (średnio coraz niższy wskaźnik `bmd`).
- W przypadku mężczyzn, nie uwzględniając zakresu wiekowego, obserwujemy wzrost wskaźnika `bmd` wraz z masą ciała (`weigth_kg`).
- W przypadku kobiet obserwujemy, że średnio więcej minerałów w kościach mają kobiety w wieku 56-76 niż 76-96.
- Na ogół mężczyźni mają średnio wytrzymalsze kości niż kobiety (różnice ukazują się w poszczególnych grupach wiekowych)

4. Histogram dla zmiennej `bmd` skategoryzowany względem płci (`sex`)

In [ ]:

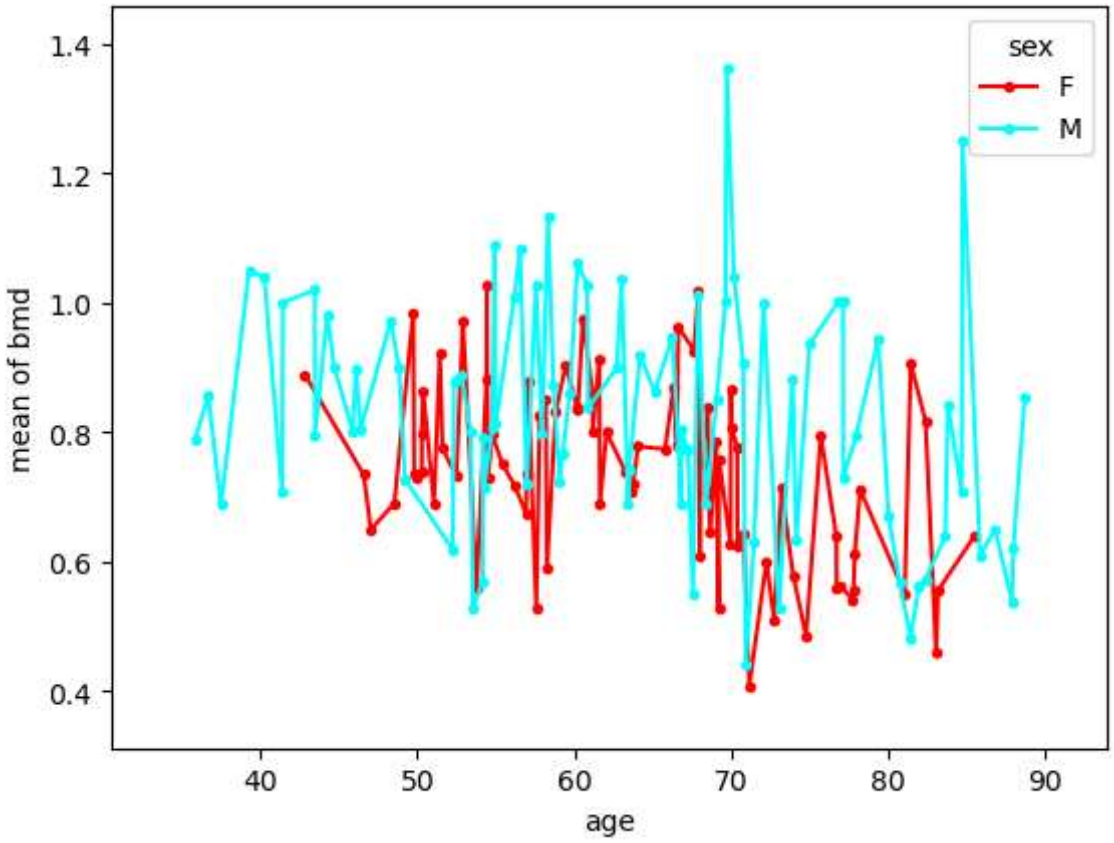
```
df = pd.read_csv('data/bmd_prep.csv')
dependent_var = 'bmd'
grouping_var = 'sex'

min = df[dependent_var].min()
max = df[dependent_var].max()
bins = np.linspace(min, max, 30)

bmd_male = df.loc[df[grouping_var]=='M', dependent_var]
bmd_female = df.loc[df[grouping_var]=='F', dependent_var]

plt.hist(bmd_male, bins, alpha=0.5, label='Male', color='darkblue')
plt.hist(bmd_female, bins, alpha=0.5, label='Female', color='red')
plt.legend(loc='upper right')
plt.xlabel(dependent_var)
plt.show()
```



Na otrzymanym histogramie możemy zauważyć, że rozkład wartości wskaźnika `bmd` dla kobiet jest bardziej jednorodny niż w przypadku mężczyzn. Wartości mieszczą się w zakresie od 0.42 do 1.02. Natomiast u mężczyzn rozkład ten jest większy - rozkład jest bardziej przesunięty w prawo, gdzie mężczyźni mają wskaźnik `bmd` większy niż 1. Modalna wartość `bmd` dla kobiet wynosi około 0.75, a dla mężczyzn 0.78 oraz 1.04.

5. Wykres interakcji (wykres średnich `bmd` w zależnosi od wieku i płci)

In [ ]:
```
from statsmodels.graphics.factorplots import interaction_plot
df = pd.read_csv('data/bmd_prep.csv')
fig = interaction_plot(x=df['age'], trace=df['sex'], response=df['bmd'])
plt.show()
```



Średnie wartości wskaźnika `bmd` dla mężczyzn są na ogół większe niż u kobiet. W wieku ok. 50 lat można jednak zauważyć, że kobiety mają średnio większy ten wskaźnik. Wśród mężczyzn możemy zaobserwować wartości odstające - 2 piki ilustrujące wyjątkowo wysokie średnie wartości wskaźnika `bmd`. U kobiet można zauważyć, że wraz z wiekiem osiągane są coraz niższe wartości średniej `bmd`.

6. Macierz korelacji

In [ ]:
```
df[numeric_vars].corr()
```

Out[ ]:

	age	weight_kg	bmd
age	1.000000	-0.093117	-0.306576
weight_kg	-0.093117	1.000000	0.555964
bmd	-0.306576	0.555964	1.000000

Najsilniejszą korelację obserwujemy dla zmiennych `bmd` oraz `weight_kg` (wsp. korelacji równy **0.56**). Występuje również umiarkowany związek (liniowy) między parami zmiennych `bmd` oraz `age` (**-0.31**). Natomiast korelacja między zmiennymi niezależnymi (`age` oraz `weight_kg`) jest słaba (**-0.1**)

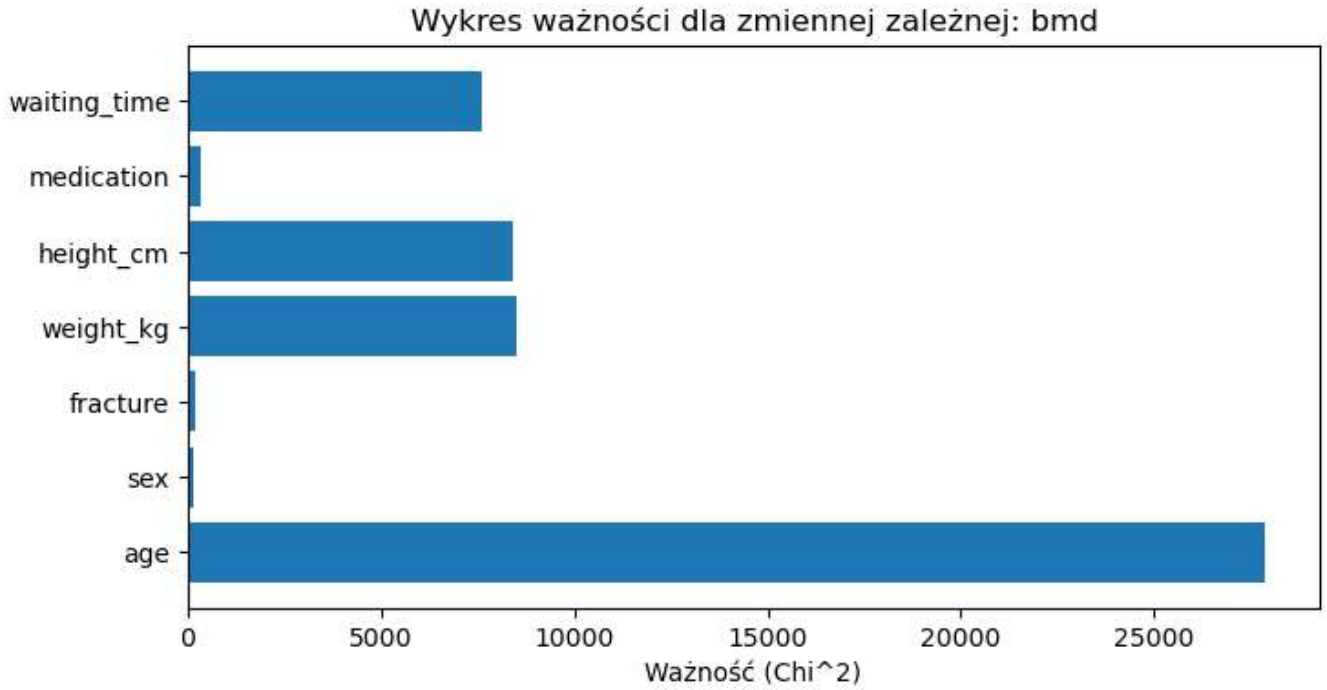
7. Test Chi kwadrat

In [ ]:
```
from scipy.stats import chi2_contingency
df = pd.read_csv('data/bmd_prep.csv')

dependent_var = 'bmd'

X = df
y = X[dependent_var]
X = X.drop(columns=[dependent_var])

chi2_test_df = pd.DataFrame(columns=['variable', 'Chi^2', 'p_value'])
```

```
chi2_test_results = []
# perform chi2 test for each independent variable
for column in X.columns:
    data_cont = pd.crosstab(X[column], y)
    chi2, p_value, _, _ = chi2_contingency(data_cont)
    chi2_test_results.append({'variable': column,'Chi^2': chi2, 'p_value': p_value})
chi2_test_df = chi2_test_df.from_records(chi2_test_results)

s = chi2_test_df.style.set_caption(f'Chi^2 Test Results. Variable: {dependent_var}')
# Plot variable importance diagram
plt.figure(figsize=(8,4))
plt.barh(chi2_test_df['variable'], chi2_test_df['Chi^2'])
plt.xlabel('Ważność (Chi^2)')
plt.title(f'Wykres ważności dla zmiennej zależnej: {dependent_var}')
display(s)
```

Chi^2 Test Results. Variable: bmd

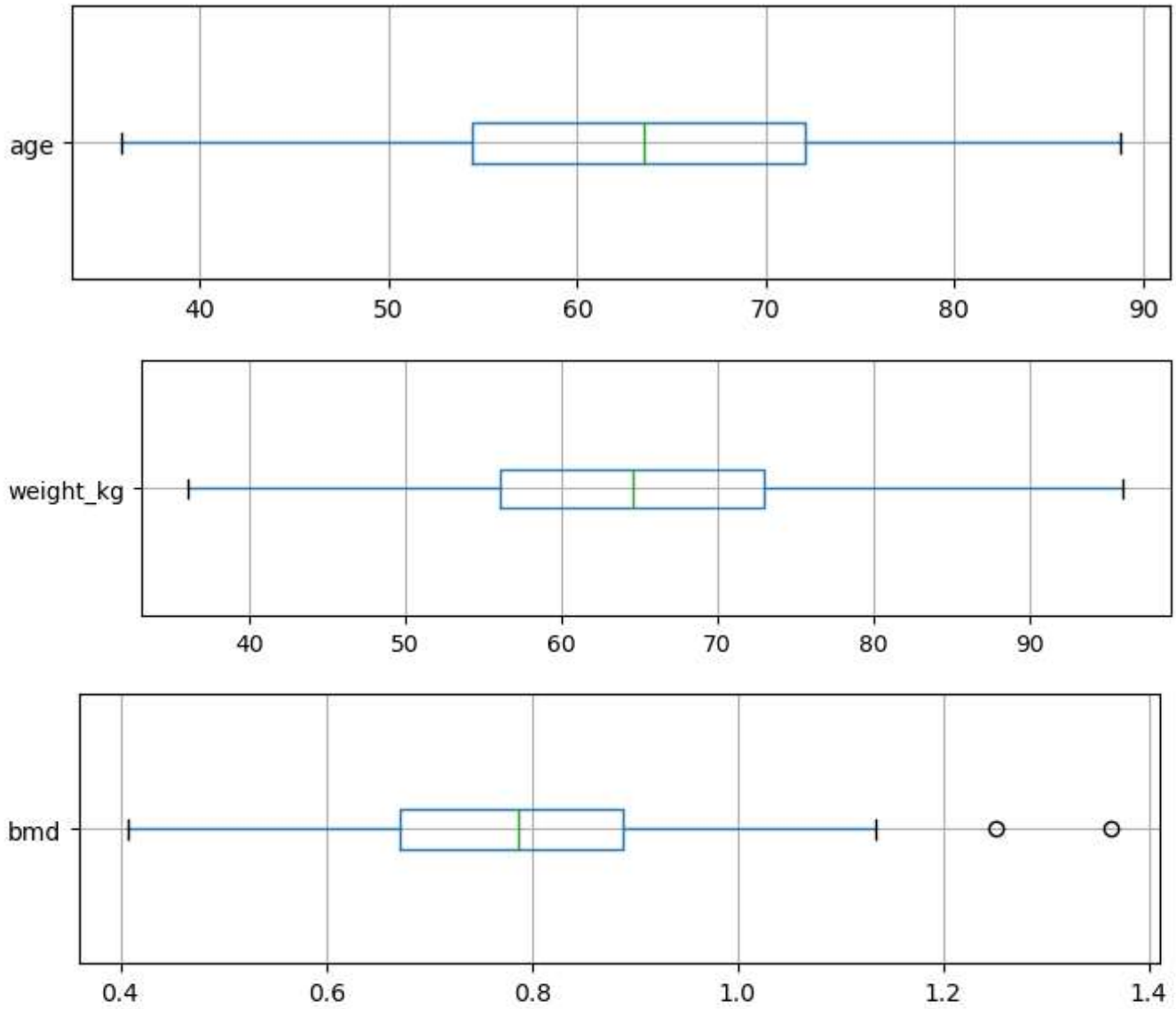
	variable	Chi^2	p_value
0	age	27885.000000	0.241274
1	sex	164.998739	0.485386
2	fracture	169.000000	0.399333
3	weight_kg	8520.316071	0.207943
4	height_cm	8419.016667	0.485603
5	medication	338.000000	0.368880
6	waiting_time	7606.408333	0.444926



Z przeprowadzonego testu Chi^2 wynika, iż najważniejszymi zmiennymi objaśniającymi dla zmiennej `bmd` są `age`, `weight_kg` i `height`. Natomiast wybrana jako predyktor zmienna `sex` okazuje się być nie tak ważna.

8. Wykresy ramka-wąsy dla zmiennych ilościowych

In [ ]:
```
for var in numeric_vars:
    fig = df.boxplot(column=var, vert=False, figsize=(8,2))
    plt.figure(figsize=(8,2))
```

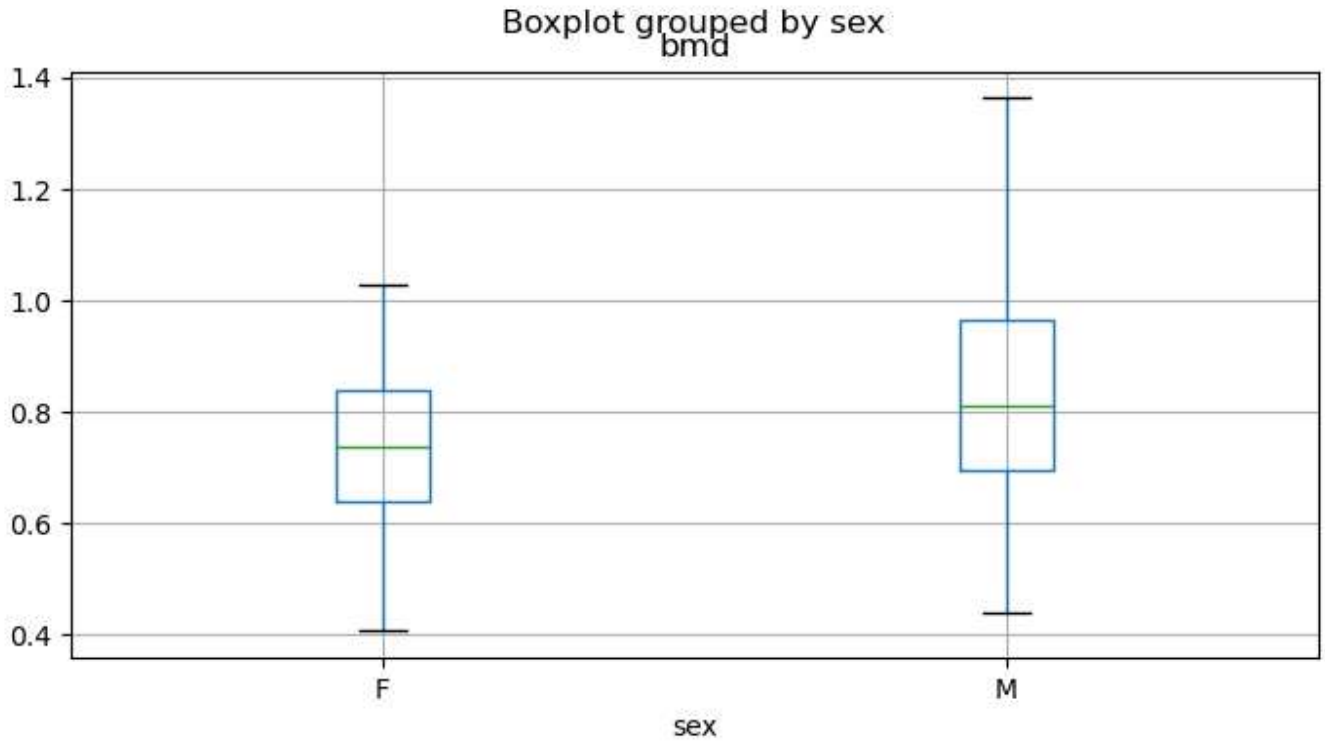



```
<Figure size 800x200 with 0 Axes>
```

Wykresy ramka-wąsy skategoryzowane względem płci

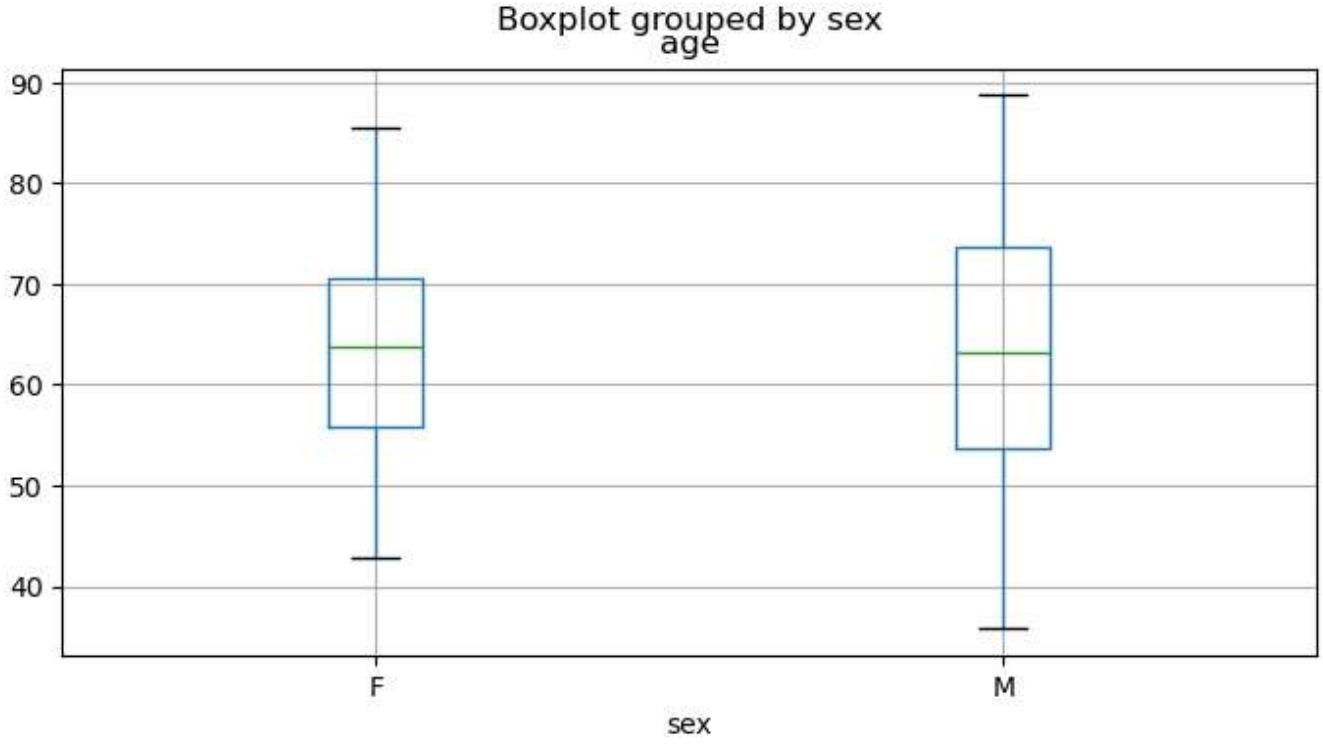
In [ ]:
```
df.boxplot(column='bmd', by='sex', figsize=(8,4))
```

Out[ ]:
```
<Axes: title={'center': 'bmd'}, xlabel='sex'>
```



In [ ]:
```
df.boxplot(column='age', by='sex', figsize=(8,4))
```

Out[ ]: <Axes: title={'center': 'age'}, xlabel='sex'>



Zarówno dla `bmd` jak i dla `age` obserwujemy, że ich wartości cechują się mniejszym rozrzutem w przypadku kobiet niż u mężczyzn.

9. Test normalny dla zmiennych

In [ ]:
```
from scipy.stats import normaltest

features = ['age', 'weight_kg', 'bmd']
for feature in features:
    alpha = 0.05
    p_value = normaltest(df[feature])[1]
```

```
    if(p_value < alpha):
        print(f'Zmienna {feature} NIE POCHODZI z rozkładu normalnego. P-value:', p_value)
    else:
        print(f'Zmienna {feature} POCHODZI z rozkładu normalnego. P-value:', p_value)
```

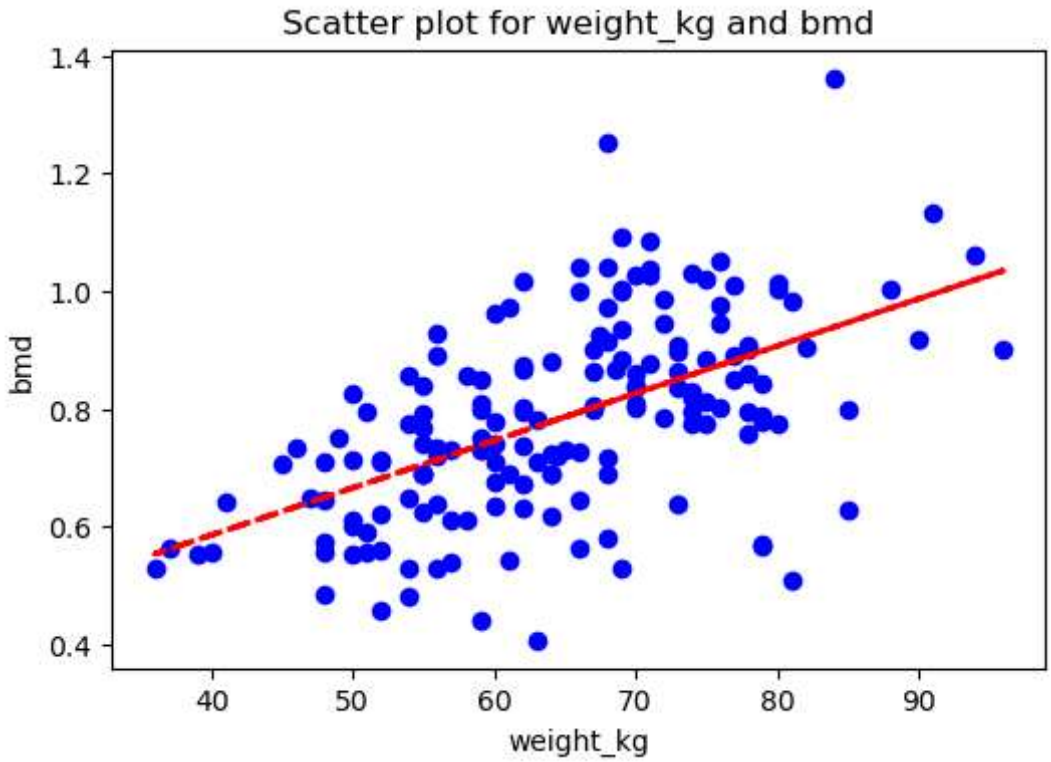
```
Zmienna age NIE POCHODZI z rozkładu normalnego. P-value: 0.03202778072330057
Zmienna weight_kg POCHODZI z rozkładu normalnego. P-value: 0.8899086011365848
Zmienna bmd POCHODZI z rozkładu normalnego. P-value: 0.2986369345960301
```

Zmienne `bmd` oraz `weight_kg` cechują się rozkładem normalnym. Wartośc p dla tych zmiennych jest większa niż założony poziom istotności alfa = 0.05.

10. Wykresy rozrzutu dla skorelowanych zmiennych

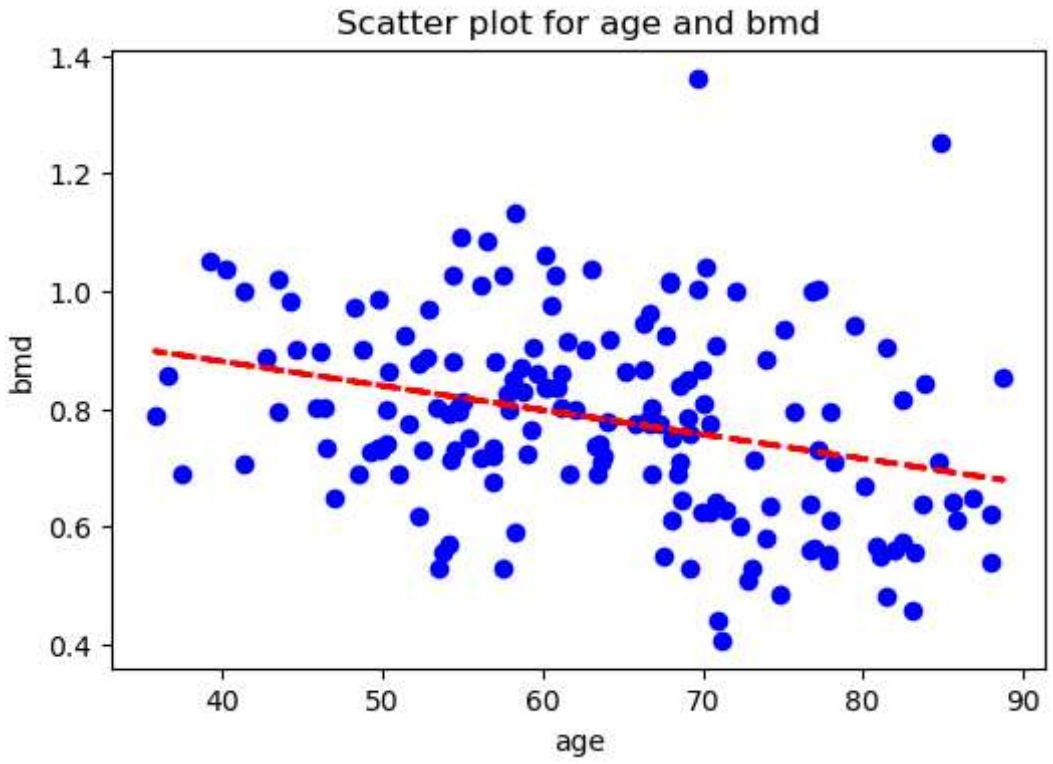
In [ ]:
```
x=df['weight_kg']
y=df['bmd']

def custom_scatter(x, y):
    #find line of best fit
    a, b = np.polyfit(x, y, 1)
    #add points to plot
    plt.figure(figsize=(6,4))
    plt.scatter(x, y, color='blue')
    #add line of best fit to plot
    plt.plot(x, a*x+b, color='red', linestyle='--', linewidth=2)
    plt.xlabel(x.name)
    plt.ylabel(y.name)
    plt.title(f'Scatter plot for {x.name} and {y.name}')
    print('Regression equation:', 'y = ' + '{:.2f}'.format(b) + ' + {:.2f}'.format(a) + 'x')

custom_scatter(x, y)
```

```
Regression equation: y = 0.26 + 0.01x
```



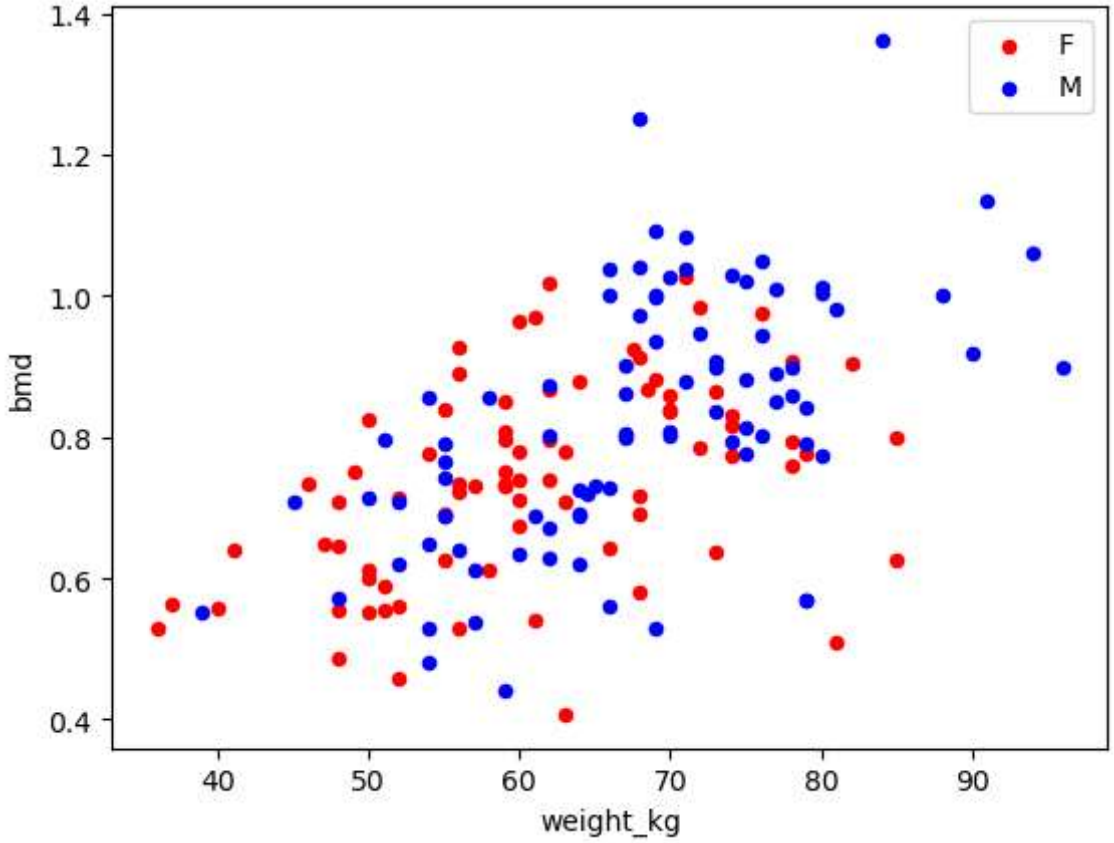
In [ ]:
```
x=df['age']
y=df['bmd']

custom_scatter(x, y)
```

```
Regression equation: y = 1.05 + -0.00x
```

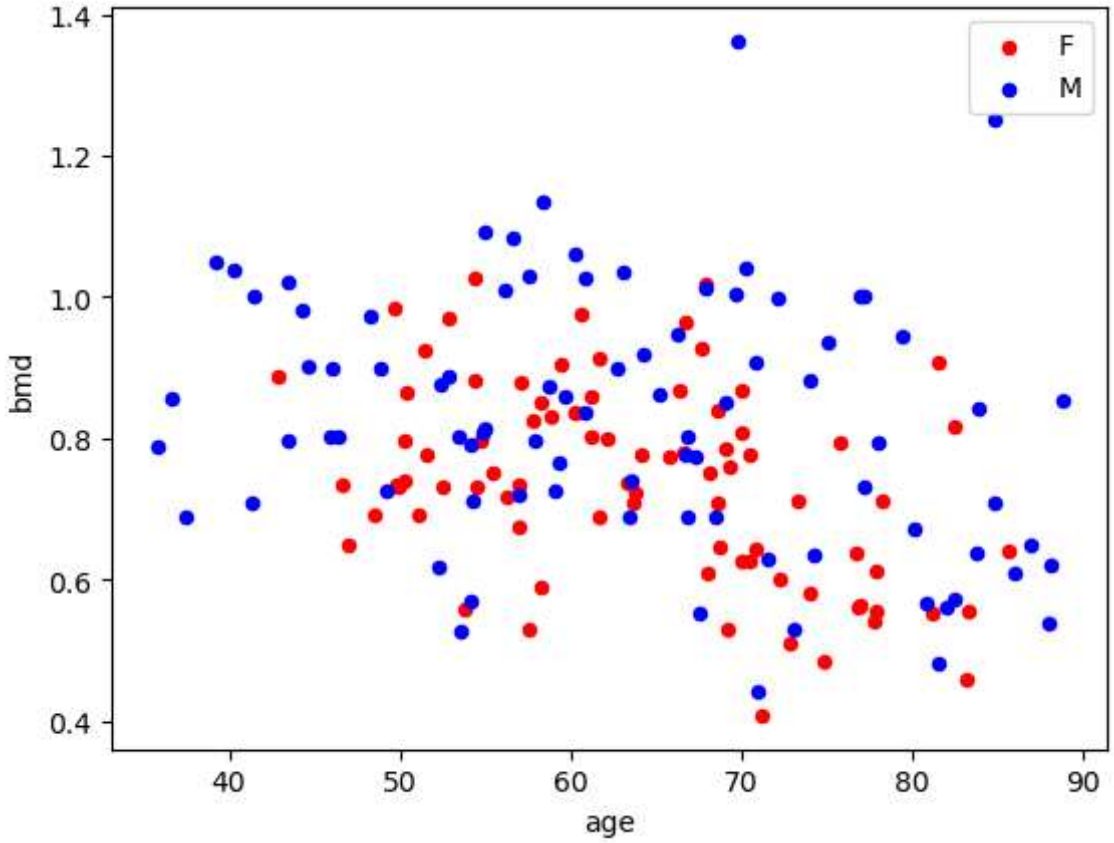


Wykresy rozrzutu skategoryzowane względem płci

In [ ]:
```
x='weight_kg'
y='bmd'

def scatter_by_sex(x, y):
    fig, ax = plt.subplots()
    colors = {'F':'red', 'M':'blue'}

    grouped = df.groupby('sex')
    for key, group in grouped:
        group.plot(ax=ax, kind='scatter', x=x, y=y, label=key, color=colors[key], marker='o')

    plt.show()

scatter_by_sex(x, y)
```



In [ ]:
```
x='age'
y='bmd'

scatter_by_sex(x, y)
```



Obserwujemy więcej wartości odstających w przypadku mężczyzn.

11. Opracowanie wyników i wnioski

- Zmienne `age` oraz `weight_kg` są skorelowane ze zmienną zależną `bmd`. Współczynnik korelacji liniowej wynosi dla `age`: **-0.31** (umiarkowana korelacja) oraz dla `weight_kg`: **0.56** (silna korelacja).
- Najsilniejszy wpływ ma zmienna `weight_kg`. Otrzymana na wykresie rozrzutu prosta regresji cechuje się największym nachyleniem prostej względem osi poziomej. Możemy zaobserwować pewną zależność liniową między tymi zmiennymi.
- Zmienne objaśniające (`age` oraz `weight_kg`) nie są wzajemnie skorelowane. Korelacja między tymi predyktorami jest słaba (wsp. kor. wynosi ok. 0.1).

Indukcja drzew decyzyjnych - drzewo regresyjne dla zmiennej `bmd`

In [ ]:
```
# Potrzebne biblioteki do indukcji drzew regresyjnych
import pandas as pd
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
```

Przygotowanie zbioru danych - podział na uczący i testowy

In [ ]:
```
df = pd.read_csv('data/bmd_prep.csv')

target_var = 'bmd'
vars_to_exclude = ['medication', 'fracture']

X = df.drop(columns=[target_var]+vars_to_exclude)
y = df[target_var]

# Encode categorical data
X = pd.get_dummies(X)

# Split data to training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
X_train.head()
```

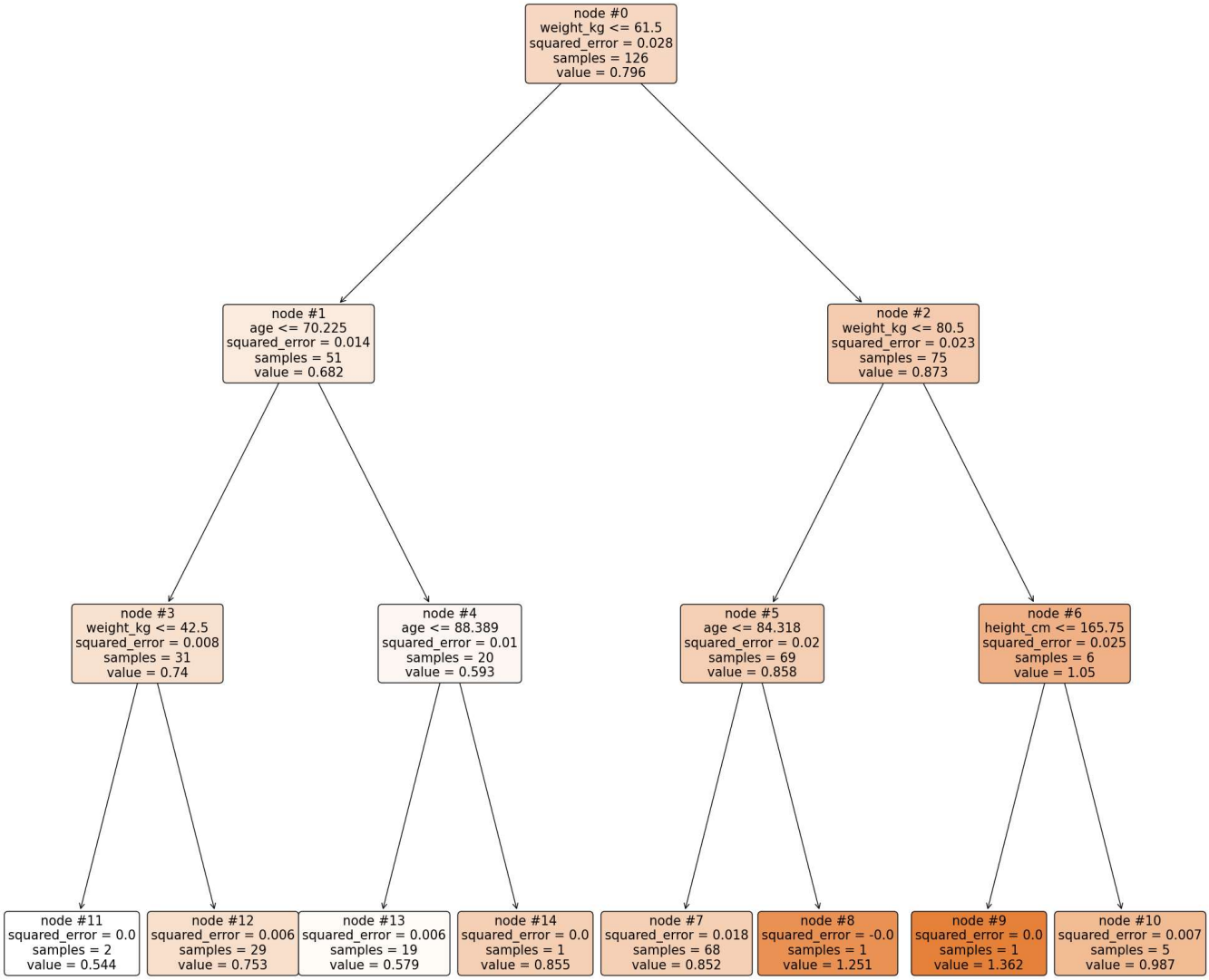
Out[ ]:

	age	weight_kg	height_cm	waiting_time	sex_F	sex_M
16	57.934593	67.0	160.0	5	0	1
66	84.787394	68.0	165.0	11	0	1
136	68.017748	58.0	153.0	8	1	0
40	73.230427	52.0	153.0	89	1	0
160	84.721764	52.0	161.0	32	0	1

1. Utworzenie drzewa regresyjnego

In [ ]:
```
params = {
    'max_depth': 3,
    'min_samples_split': 5,
    'max_leaf_nodes': 10
}

tree = DecisionTreeRegressor(**params)
```

```
# Fits the decision tree to training data
tree.fit(X_train, y_train)

fig = plt.figure(figsize=(25, 25))

# Creates a visual display of the model.
# Keep max_depth small for better visualization
t = plot_tree(
    tree,
    node_ids=True,
    feature_names=X.columns,
    rounded=True,
    filled=True)
```

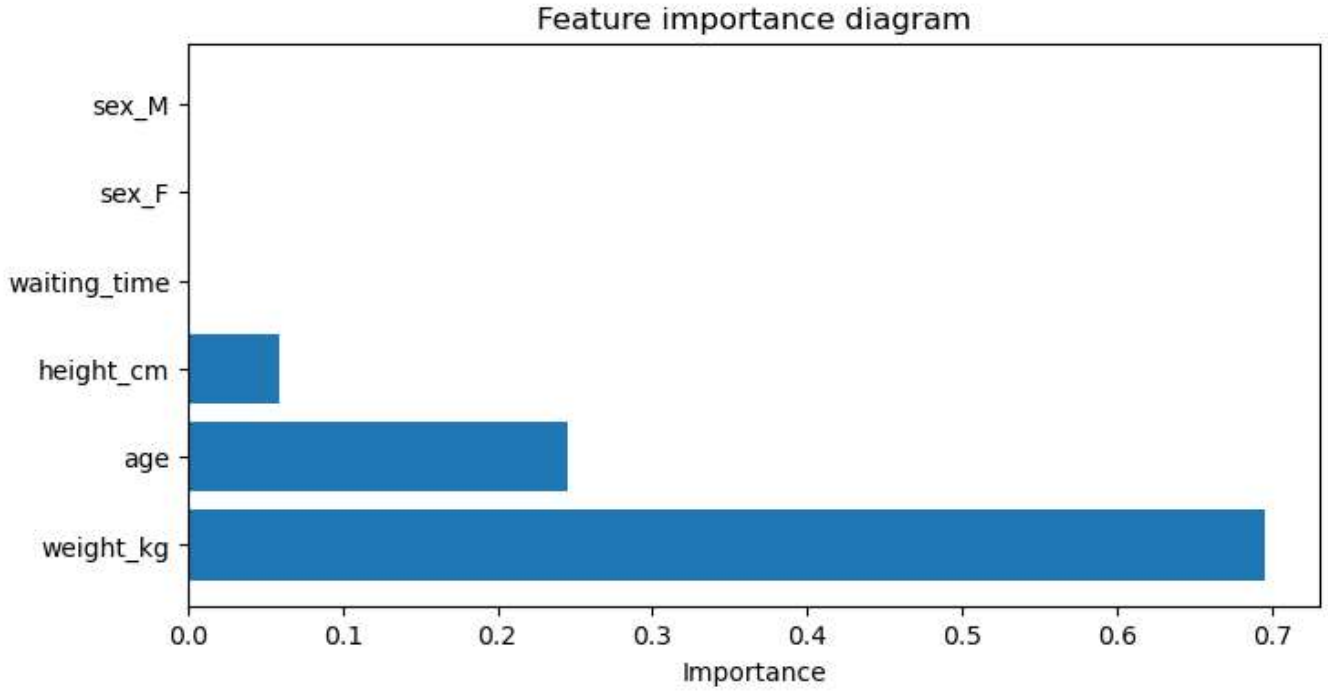


2. Ważność predyktorów

In [ ]:
```
features = X.columns.tolist()

importances_sk = tree.feature_importances_

feature_importance = []
for i, feature in enumerate(features):
```

```
    feature_importance.append({'feature': feature,'importance': round(importances_sk[i], 3)})
# sort features by their importance
feature_importance = sorted(feature_importance, key=lambda d: d['importance'], reverse=True)

feature_importance_df = pd.DataFrame(columns=['feature', 'importance'])
feature_importance_df = feature_importance_df.from_records(feature_importance)

plt.figure(figsize=(8,4))
plt.barh(feature_importance_df['feature'], feature_importance_df['importance'])
plt.xlabel('Importance')
plt.title(f'Feature importance diagram')
```

Out[ ]: Text(0.5, 1.0, 'Feature importance diagram')



Najważniejszymi predyktorami w otrzymanym drzewie są `weight_kg`, `age` oraz `height_cm`. Użyta w hipotezie zmienna `sex` wg. otrzymanego drzewa jest nieważna.

3. Reguły dla 3 najbardziej wyrazistych klas

1. Reguła dla węzła #13:

- Jeśli osoba waży mniej niż 61.5 kg, a jej wiek znajduje się w przedziale od 70 do 89, wskaźnik `bmd` takiej osoby będzie niski i średnio wynosił 0.579
- Wariancja: 0.006
- Średnia: 0.579

2. Reguła dla węzła #12:

- Jeśli waga osoby znajduje się w zakresie od 42.5 do 61.5 kg oraz jej wiek jest mniejszy niż 70 lat, wskaźnik `bmd` takiej osoby będzie średnio wynosił 0.759
- Wariancja: 0.006
- Średnia: 0.759

3. Reguła dla węzła #7:

- Jeśli waga osoby wynosi od 61.5 do 80 kg, a jej wiek jest mniejszy niż 84 lata, wskaźnik `bmd` takiej osoby będzie średnio wynosił 0.852 (dobra wartość)
- Wariancja: 0.018

- Średnia: 0.852

4. Ocena wariancji w liściach

Wariancja całej próby wynosi 0.026. Spośród wszystkich liści, w których liczba obserwacji jest większa niż 1, liście 12 oraz 13 (reguły 1 i 2) cechują się najmniejszą wariancją (0.006). Oznacza to, że stanowią one najbardziej reprezentatywne grupy ze wszystkich otrzymanych liści (najbardziej jednorodny podział). Natomiast w przypadku reguły 3, gdzie wariancja w liściu wynosi 0.018 jest stosunkowo duża, biorąc pod uwagę wariancję pozostałych liści

5. Koszt resubstytucji, współczynnik determinacji drzewa

In [ ]:
```
# Calculate the R-squared value
y_pred = tree.predict(X)
r2 = r2_score(y, y_pred)
RSS = sum((y - y_pred)**2)
cv_score = cross_val_score(tree, X, y, cv=5, scoring='neg_mean_squared_error')
print(f"Koszt resubstytucji (RSS): {RSS:.3f}")
print(f"Współczynnik determinacji (R^2): {r2:.3f}")
```

```
Koszt resubstytucji (RSS): 4.065
Współczynnik determinacji (R^2): 0.127
```

Otrzymana wartośc współczynnika determinacji jest niska (Rmax = 1). Oznacza, to że jakość modelu (drzewa) nie jest dobra

6. Weryfikacja hipotezy na podstawie drzewa

Na podstawie otrzymanych wyników, można stwierdzić, że hipoteza jest słuszna. Stwierdzenie postawione w ramach hipotezy: "Wraz ze wzrostem wagi, rośnie `bmd` " jest słuszne, gdyż dla reguły 2, gdzie zakres wagowy osób wynosi (42.5, 61.5) kg średnia wartość `bmd` wynosiła 0.759, a dla reguły 3, przy zakresie wagowym powyżej 61.5 kg obserwowaliśmy wyższą wartość `bmd` (średnio 0.852). Również stwierdzenie, że wraz z wiekiem, `bmd` maleje jest słuszne - dla reguły 1, gdzie wiek > 70 `bmd` wynosi średnio 0.579, a dla reguły 2, gdzie wiek < 70, `bmd` było wyższe (0.759). Płeć wg. wyników otrzymanych przy pomocy drzewa nie ma znaczenia.
Wniosek: Im cięższa i młodsza osoba, tym średnio więcej minerałów zawierają kości tej osoby, niezależnie od płci.

Wybrany algorytm data mining - KNN

In [ ]:
```
# Niezbędne biblioteki
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
```

1. Przygotowanie danych i wybór zmiennych

In [ ]:
```
df = pd.read_csv('data/bmd_prep.csv')
```

```
all_num_predictors = df.select_dtypes(include=np.number).columns.tolist()
# normalization of numeric variables
df[all_num_predictors] = MinMaxScaler().fit_transform(df[all_num_predictors])

target_var = 'bmd'
num_predictors = ['age', 'weight_kg', 'waiting_time']
cat_predictors =  ['sex']

X = df[num_predictors + cat_predictors]
y = df[target_var]

# one-hot encode categorical variables
X = pd.get_dummies(X)

# Split data to training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
X_train.head()
```

	age	weight_kg	waiting_time	sex_F	sex_M
16	0.417844	0.516667	0.000000	0	1
66	0.925077	0.533333	0.065934	0	1
136	0.608308	0.366667	0.032967	1	0
40	0.706773	0.266667	0.923077	1	0
160	0.923837	0.266667	0.296703	0	1

2. Optymalizacja wartości k - V-krotny sprawdzian krzyżowy

In [ ]:
```
# Sets up the kNN regressor object
knn = KNeighborsRegressor()

# Search parameters
param = range(3, 15, 2)

# Sets up GridSearchCV object and stores it in grid variable
grid = GridSearchCV(
    knn,
    {'n_neighbors': param})

# Fits the grid object and gets the best model
best_knn = grid \
    .fit(X, y) \
    .best_estimator_

# Displays the optimum model
best_knn.get_params()
```

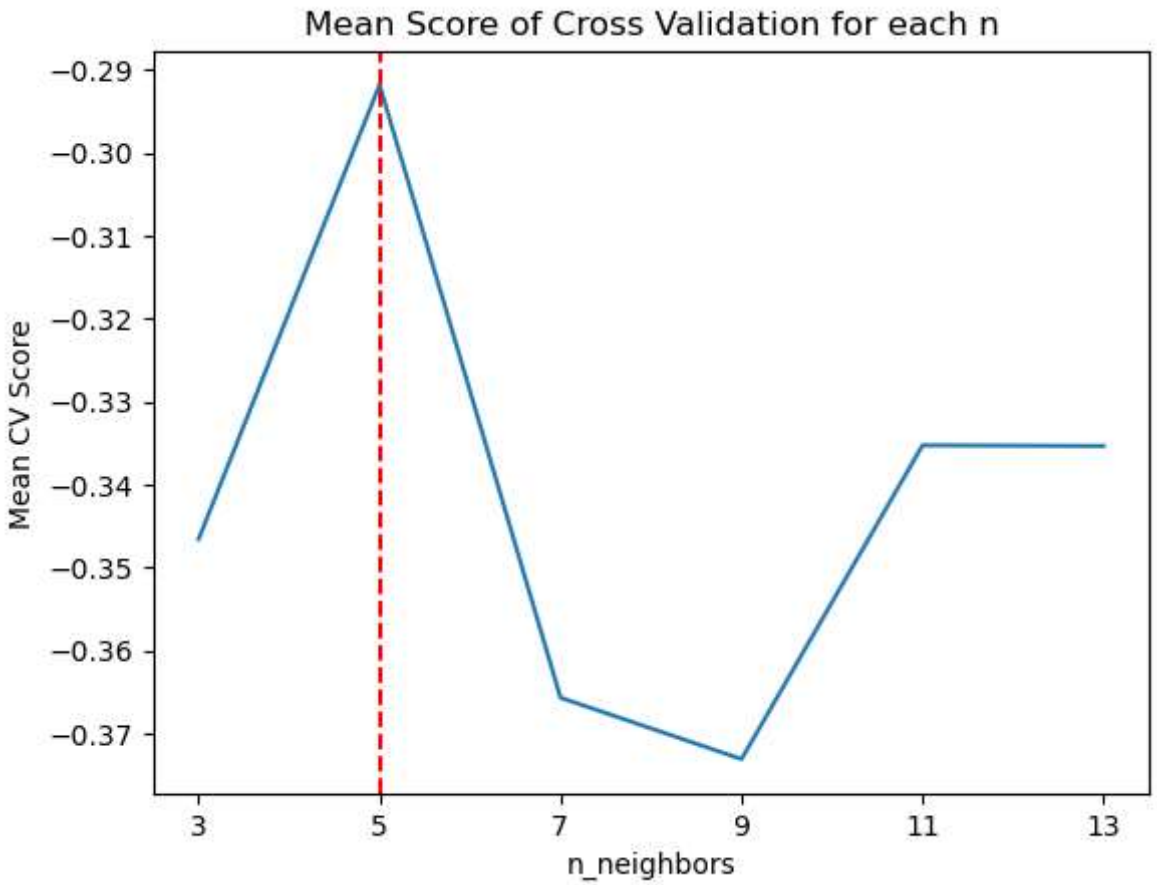
```
{'algorithm': 'auto',
 'leaf_size': 30,
 'metric': 'minkowski',
 'metric_params': None,
 'n_jobs': None,
 'n_neighbors': 5,
 'p': 2,
 'weights': 'uniform'}
```

Wykres wyniku sprawdzianu krzyżowego w zależności od ilości sąsiadów

In [ ]:
```
# Plots mean_test_scores vs. total neighbors
plt.plot(
```

```
    param,
    grid.cv_results_['mean_test_score'])

# Adds labels to the plot
plt.xticks(param)
plt.ylabel('Mean CV Score')
plt.xlabel('n_neighbors')
plt.title('Mean Score of Cross Validation for each n')

# Draws a vertical line where the best model is
plt.axvline(
    x=best_knn.n_neighbors,
    color='red',
    ls='--')
```

```
<matplotlib.lines.Line2D at 0x24159dd15e0>
```



Przeprowadzony sprawdzian krzyżowy wykazał, że najbardziej optymalną ilością sąsiadów jest `5`

3. Algorytm KNN dla optymalnych parametrów

In [ ]:
```
params = best_knn.get_params()
knn = KNeighborsRegressor(**params)
knn.fit(X, y)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

Dokładność otrzymanego modelu

In [ ]:
```
print(
    'Accuracy on training data: '
    + str("{:.2%}".format(knn.score(X_train, y_train))))
```

```
print(
    'Accuracy on testing data: '
    + str("{:.2%}".format(knn.score(X_test, y_test))))
```

```
Accuracy on training data: 57.72%
Accuracy on testing data: 55.54%
```

4. Interpretacja wyników

Otrzymana dokładność modelu nie jest ogólnie zadowalająca. Biorąc pod uwagę, że rozkład punktów w zbiorze danych stanowi tak naprawdę jedną wielką chmurę punktów (co jest widoczne na wykresach rozrzutu), KNN wbrew pozorom poradził sobie nienajgorzej. Na podstawie wybranych zmiennych objaśniających, model jest w stanie poprawnie przewidzieć wartość wskaźnika `bmd` z dokładnością nieco ponad `50%`.

Hipoteza 2 - badanie zależności rodzaju leczenia, wieku oraz wartości wskaźnika `bmd` na wystąpienie złamania

Badanie struktury populacji

In [ ]:
```
# Potrzebne biblioteki
import numpy as np
import matplotlib.pyplot as plt
```

1. Statystyki opisowe - zmienne ilościowe

In [ ]:
```
numeric_vars = ['age', 'bmd']
np.round(df[numeric_vars].describe(),3)
```

Out[ ]:

	age	bmd
count	169.000	169.000
mean	63.632	0.783
std	12.357	0.167
min	35.814	0.408
25%	54.424	0.671
50%	63.488	0.786
75%	72.081	0.889
max	88.754	1.362

Najstarsza osoba w zbiorze ma niemal 89 lat natomiast najmłodsza niecałe 36. Waga pacjentów mieści się w zakresie od 36 do 96 kg. Średnia wartość wskaźnika `bmd` wynosi 0.783 a mediana 0.786.

wariancja

In [ ]:
```
np.round(df[numeric_vars].var(), 3)
```

Out[ ]:
```
age    152.694
bmd      0.028
dtype: float64
```

moda

In [ ]:
```
np.round(df[numeric_vars].mode(), 3)
```

Out[ ]:

	age	bmd
0	35.814	0.690
1	36.648	0.713
2	37.462	0.889
3	39.255	NaN
4	40.232	NaN
...	...	...
164	85.902	NaN
165	86.860	NaN
166	87.935	NaN
167	88.024	NaN
168	88.754	NaN

169 rows × 2 columns

Najczęsciej występujące w zbiorze wartości `bmd` to 0.69, 0.713 oraz 0.889.

2. Tabele liczności - zmienne jakościowe

In [ ]:
```
categorical_vars = ['medication', 'fracture']
df[categorical_vars].value_counts()
```

Out[ ]:
```
medication       fracture
No medication    no fracture    95
                 fracture       41
Glucocorticoids  no fracture    19
Anticonvulsant   no fracture     5
Glucocorticoids  fracture        5
Anticonvulsant   fracture        4
dtype: int64
```

Większośc osób w badanej próbie stanowią osoby zdrowe - bez złamania oraz nieleczone. Drugą najliczniejszą grupę stanowią osoby, które doznały złamania i nie były leczone. Pozostałe osoby są to osoby leczone.

In [ ]:
```
for var in categorical_vars:
    ft = df[[var]].value_counts()
    display(ft)
```

```
medication
No medication      136
Glucocorticoids     24
Anticonvulsant       9
dtype: int64
fracture
no fracture    119
fracture        50
dtype: int64
```

Pod względem zmiennej `fracture` mamy dość nierównomierny rozkład - ponad 2/3 wszystkich osób stanowią osoby, które nie doznały złamania. Niewielka część całej próby stanowią osoby leczone - ok 20% (33/169 * 100%)

3. Tabela wielodzielcza

Tabela składać się będzie z dwóch części - pierwsza z nich dotyczy osób, które nie doznały złamania (`no fracture`), a druga, które doznały złamania (`fracture`)

In [ ]:
```
# group values in numeric features to equal ranges
df = pd.read_csv('data/bmd_prep.csv')
df['age'] = pd.cut(df['age'], 6, precision=0)

# create pivot tables for unique values in 'fracture' column
fracture_distinct_values = df['fracture'].unique().tolist()
for value in fracture_distinct_values:
    pivot = df[df.fracture == value].pivot_table(values='fracture',
                    index=['medication', ],
                    columns=['age'],
                    aggfunc=['count'],
                    margins=True,
                    margins_name='Checksum',
                    fill_value='-')
    print(f'Pivot table for: "{value}"')
    display(pivot)
```

```
Pivot table for: "no fracture"
```

							count
age	**(36.0, 45.0]**	**(45.0, 53.0]**	**(53.0, 62.0]**	**(62.0, 71.0]**	**(71.0, 80.0]**	**(80.0, 89.0]**	**Checksum**
medication							
Anticonvulsant	0	0	4	1	0	0	5
Glucocorticoids	1	3	6	5	3	1	19
No medication	8	17	27	28	11	4	95
Checksum	9	20	37	34	14	5	119

```
Pivot table for: "fracture"
```

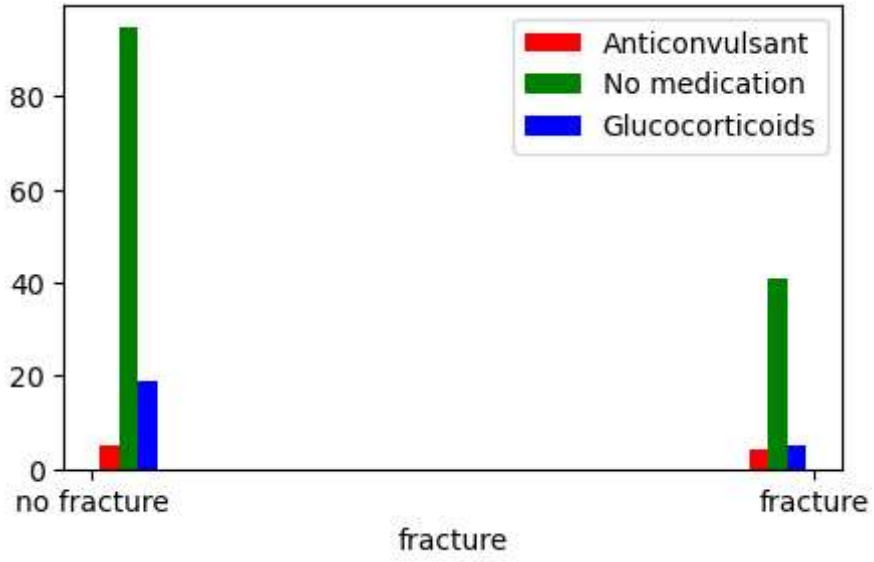
							count
age	**(36.0, 45.0]**	**(45.0, 53.0]**	**(53.0, 62.0]**	**(62.0, 71.0]**	**(71.0, 80.0]**	**(80.0, 89.0]**	**Checksum**
medication							
Anticonvulsant	0	1	0	0	0	3	4
Glucocorticoids	0	0	1	0	2	2	5
No medication	2	4	6	10	9	10	41
Checksum	2	5	7	10	11	15	50

Na podstawie otrzymanych tabel dla zmiennej `fracture` możemy zaobserwować, że najwięcej przypadków osób ze złamaniem w badanej próbie to osoby w przedziale wiekowymi od 60 do 90 lat (osoby starsze). Zdecydowana większość osób w tej próbie nie miała leczenia. Większośc osób, które nie doznały złamania to osoby w wieku od 45 do 70 lat. Niewielka częśc osób miała podany lek

przeciwdrgawkowy (anticonvulsant) lub glikokortykoidy, przy czym tylko kilka osób, które miały podane te leki, doznały złamania (4+5 = 9).

4. Histogram dla zmiennej `fracture` skategoryzowany względem rodzaju leczenia (`medication`)

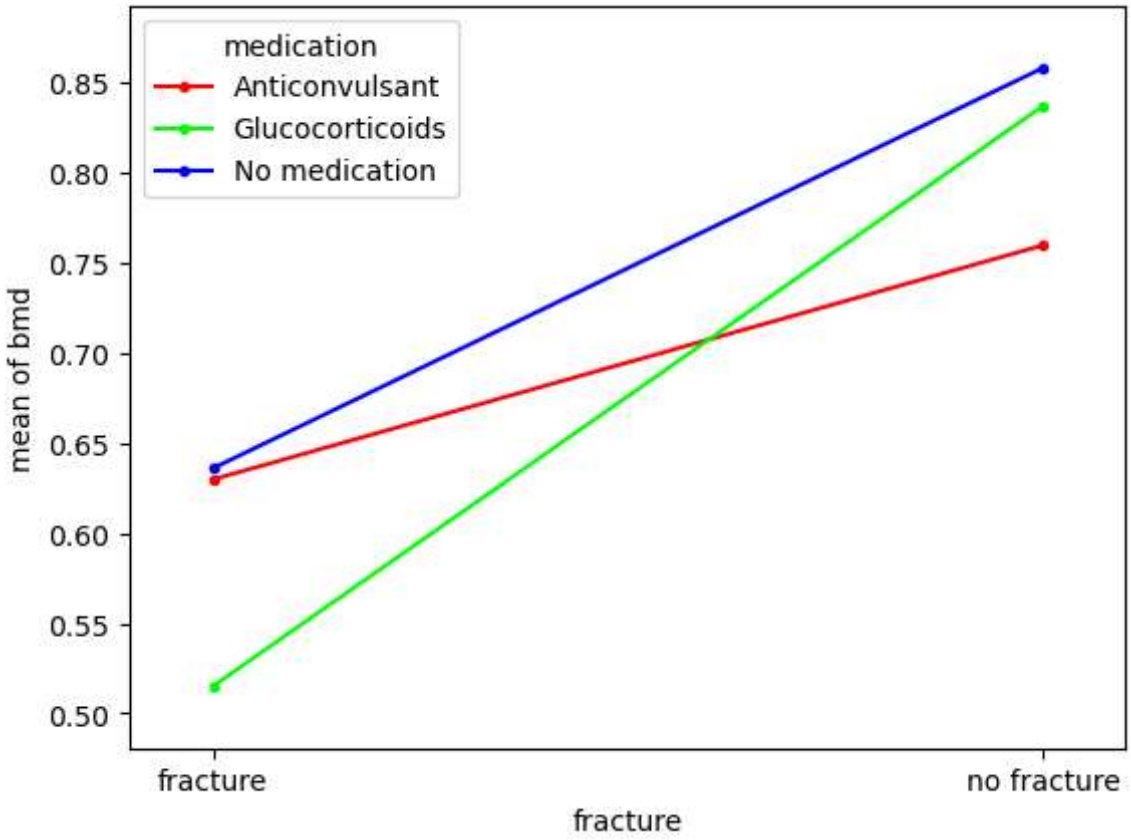
In [ ]:
```
df = pd.read_csv('data/bmd_prep.csv')
var = 'fracture'
grouping_var = 'medication'

medication_labels = df['medication'].unique().tolist()

anticonv = df.loc[df[grouping_var]==medication_labels[0], var]
gluco = df.loc[df[grouping_var]==medication_labels[1], var]
no_med = df.loc[df[grouping_var]==medication_labels[2], var]

plt.figure(figsize=(5,3))
plt.hist([anticonv, gluco, no_med], label=medication_labels, color=['red', 'green', 'blue'])
plt.legend(loc='upper right')
plt.xlabel(var)
plt.show()
```



Możemy zauważyć na otrzymanym histogramie, że proporcja między osobami, które nie były leczone a tymi, które przyjęły jako lek glikokortykoidy jest większy dla przypadków, gdzie nie wystąpiło złamanie niż dla tych, gdzie ono wystąpiło. Możnaby wnioskować, że glikokortykoidy pomagają zabopiec złamaniu.

5. Wykres interakcji (wykres średnich `bmd` w zależności od złamania oraz rodzaju leczenia)

In [ ]:
```
from statsmodels.graphics.factorplots import interaction_plot

df = pd.read_csv('data/bmd_prep.csv')
fig = interaction_plot(x=df['fracture'], trace=df['medication'], response=df['bmd'])
plt.show()
```



Spostrzeżenia:

- Średnia wartość `bmd` w przypadku osób nieleczonych jest najwyższa sposród wszystkich metod leczenia.
- Bardzo niska średnia wartość `bmd` jest w przypadku osób leczonych przy pomocy glukokortykozy. Można wnioskować, że leki te stosowane są często przy naprawdę słabym stanie kości.

6. Macierz korelacji

In [ ]:
```
df = pd.read_csv('data/bmd_prep.csv')
numeric_vars = ['bmd', 'age']
df[numeric_vars].corr()
```

Out[ ]:

	bmd	age
bmd	1.000000	-0.306576
age	-0.306576	1.000000

Jedynymi zmiennymi ilościowymi w badanej hipotezie są zmienne opisujące `age` oraz `bmd`. Korelacja między nimi jest umiarkowana. Ponieważ zmienna zależna jest jakościowa, nie da się dla niej policzyć współczynnika korelacji

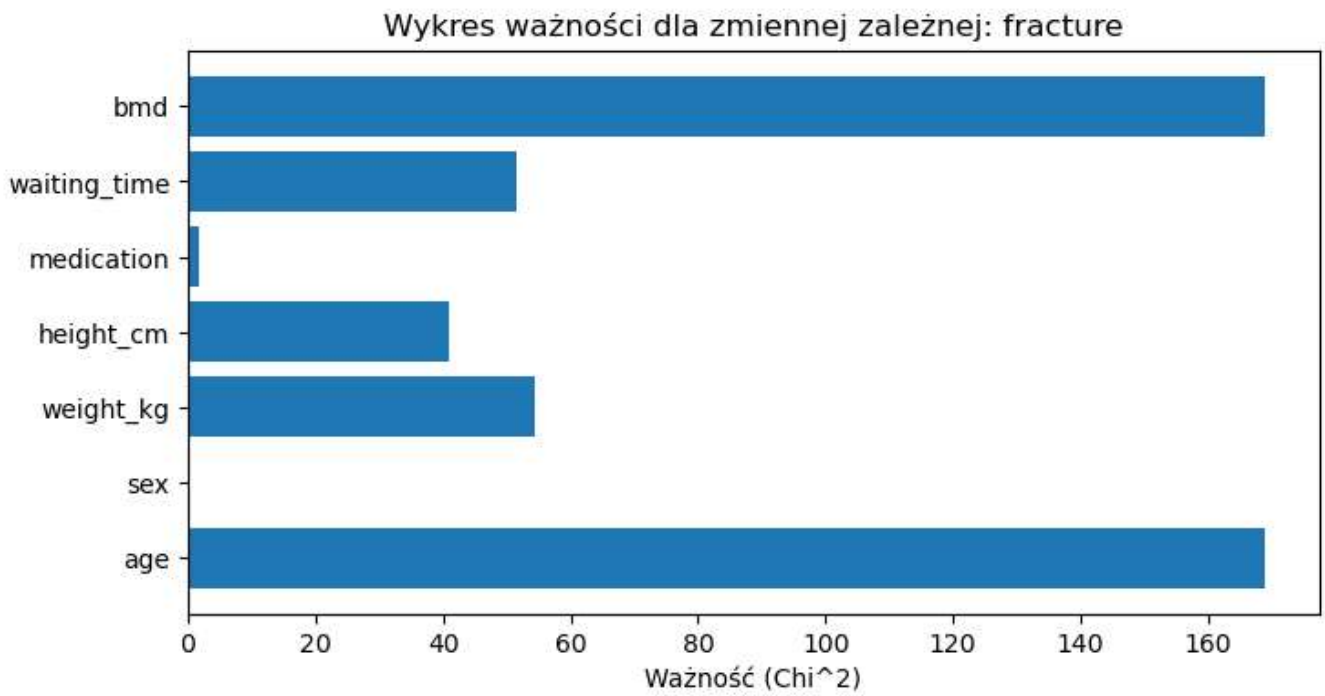
7. Test Chi kwadrat

In [ ]:
```
from scipy.stats import chi2_contingency
df = pd.read_csv('data/bmd_prep.csv')

dependent_var = 'fracture'

X = df
y = X[dependent_var]
```

```
X = X.drop(columns=[dependent_var])

chi2_test_df = pd.DataFrame(columns=['variable', 'Chi^2', 'p_value'])
chi2_test_results = []
# perform chi2 test for each independent variable
for column in X.columns:
    data_cont = pd.crosstab(X[column], y)
    chi2, p_value, _, _ = chi2_contingency(data_cont)
    chi2_test_results.append({'variable': column,'Chi^2': chi2, 'p_value': p_value})
chi2_test_df = chi2_test_df.from_records(chi2_test_results)

s = chi2_test_df.style.set_caption(f'Chi^2 Test Results. Variable: {dependent_var}')
# Plot variable importance diagram
plt.figure(figsize=(8,4))
plt.barh(chi2_test_df['variable'], chi2_test_df['Chi^2'])
plt.xlabel('Ważność (Chi^2)')
plt.title(f'Wykres ważności dla zmiennej zależnej: {dependent_var}')
display(s)
```

Chi^2 Test Results. Variable: fracture

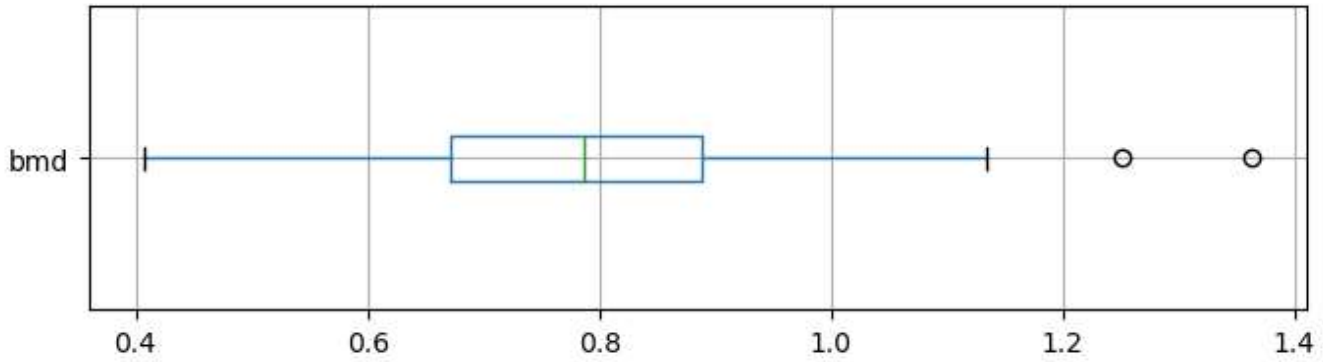
	variable	Chi^2	p_value
0	age	169.000000	0.463822
1	sex	0.000000	1.000000
2	weight_kg	54.281698	0.350510
3	height_cm	40.926944	0.842458
4	medication	1.856893	0.395167
5	waiting_time	51.552251	0.265660
6	bmd	169.000000	0.399333

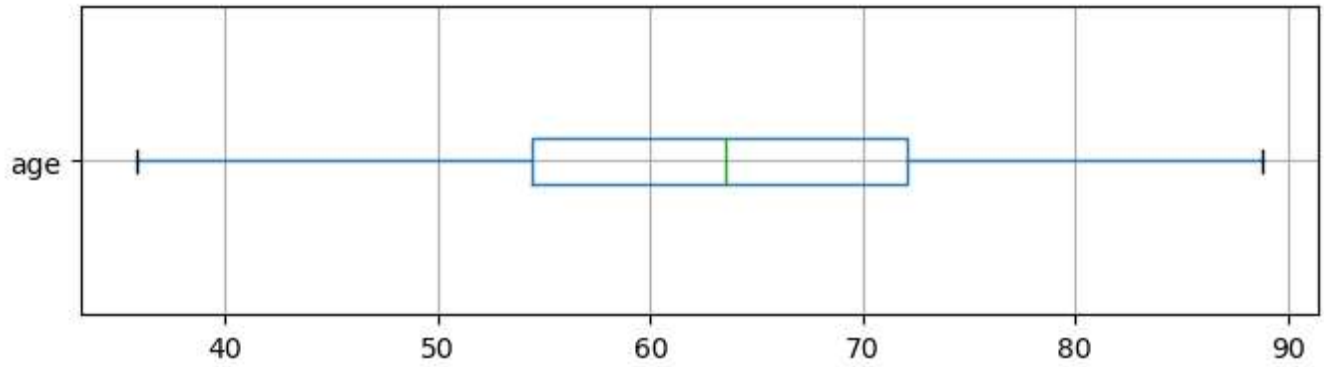


Dla zmiennej `fracture` najważniejszymi predyktorami okazały się `age` oraz `bmd`, które zostały wybrane przy definiowaniu hipotezy. Natomiast zmienna `medication` okazała się bardzo mało ważna.

8. Wykresy ramka-wąsy

In [ ]:
```
for var in numeric_vars:
    fig = df.boxplot(column=var, vert=False, figsize=(8,2))
    plt.figure(figsize=(8,2))
```



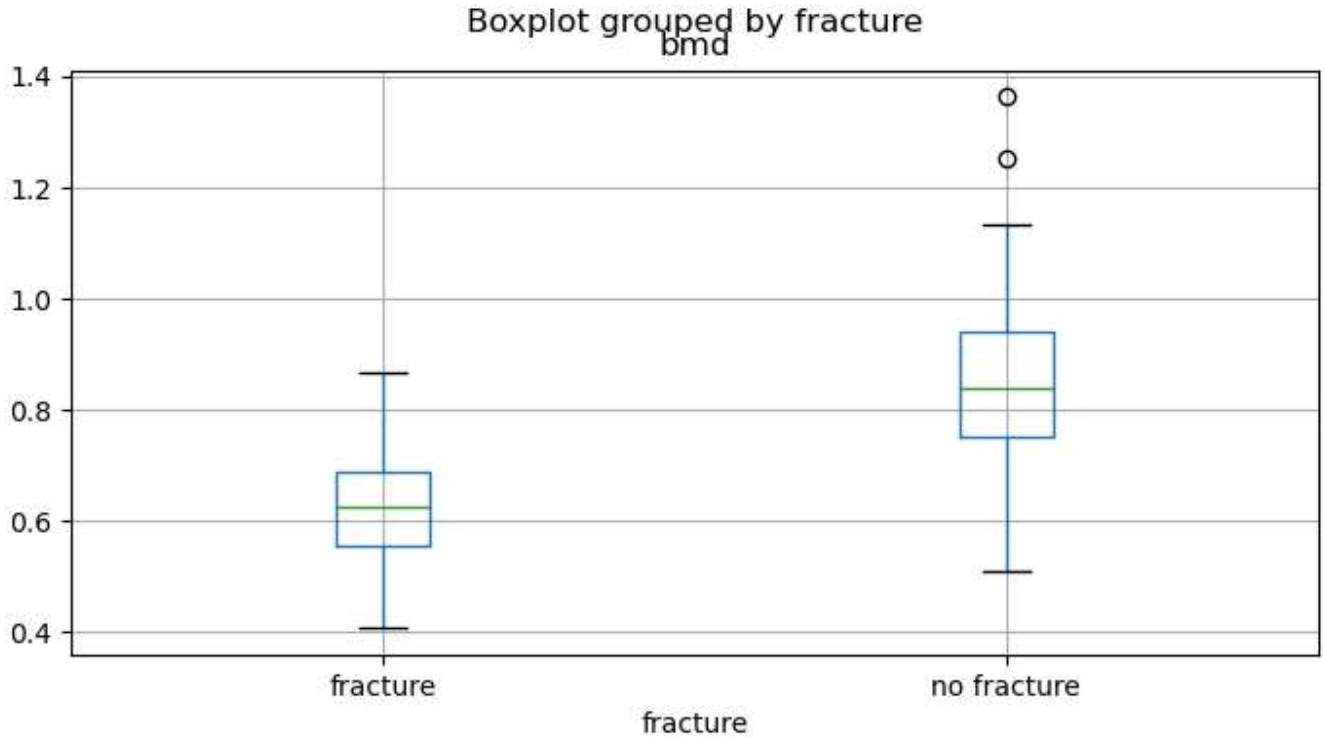



```
<Figure size 800x200 with 0 Axes>
```

Skategoryzowane wykresy ramka wąsy względem zmiennej `fracture`

In [ ]:
```
df.boxplot(column='bmd', by='fracture', figsize=(8,4))
```

Out[ ]:
```
<Axes: title={'center': 'bmd'}, xlabel='fracture'>
```

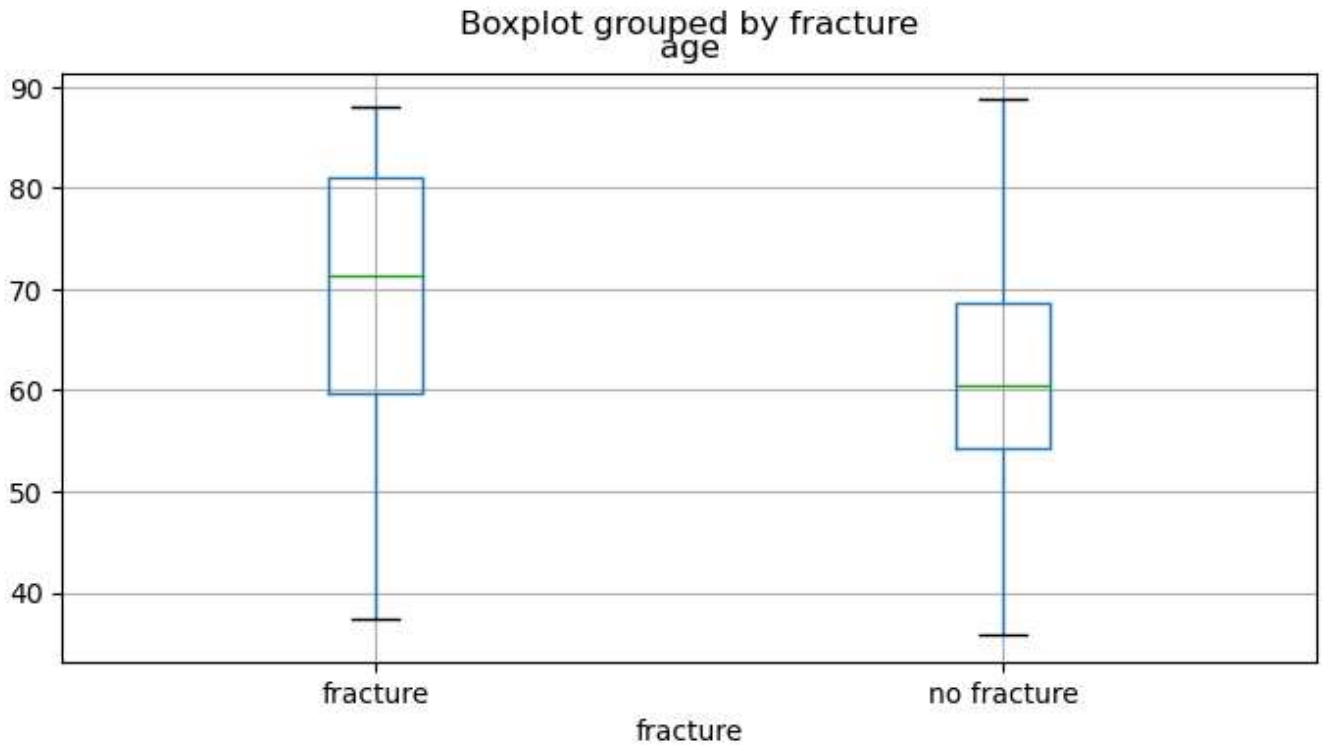


Osoby, które doznały złamania cechują się na ogół niższą wartością `bmd` oraz mniejszym rozstępem tych wartości. W przypadku osób które nie doznały złamania, cechuje ich większy rozstęp wartości `bmd`,

wartość mediany oraz średniej. Zdarzyły się nawet 2 przypadki odstające, u których kości mają wyjątkowo dużo minerałów (wysokie `bmd`).

In [ ]:
```
df.boxplot(column='age', by='fracture', figsize=(8,4))
```

Out[ ]:
```
<Axes: title={'center': 'age'}, xlabel='fracture'>
```



Na ogół osoby, które doznały złamania są średnio starsze niż osoby bez złamania. Rozkład osób, które nie doznały złamania bardziej rozłożony i przypomina rozkład normalny.

9. Test normalny

In [ ]:
```
from scipy.stats import normaltest

features = ['age', 'bmd']
for feature in features:
    alpha = 0.05
    p_value = normaltest(df[feature])[1]
    if(p_value < alpha):
        print(f'Zmienna {feature} NIE POCHODZI z rozkładu normalnego. P-value:', p_value)
    else:
        print(f'Zmienna {feature} POCHODZI z rozkładu normalnego. P-value:', p_value)
```

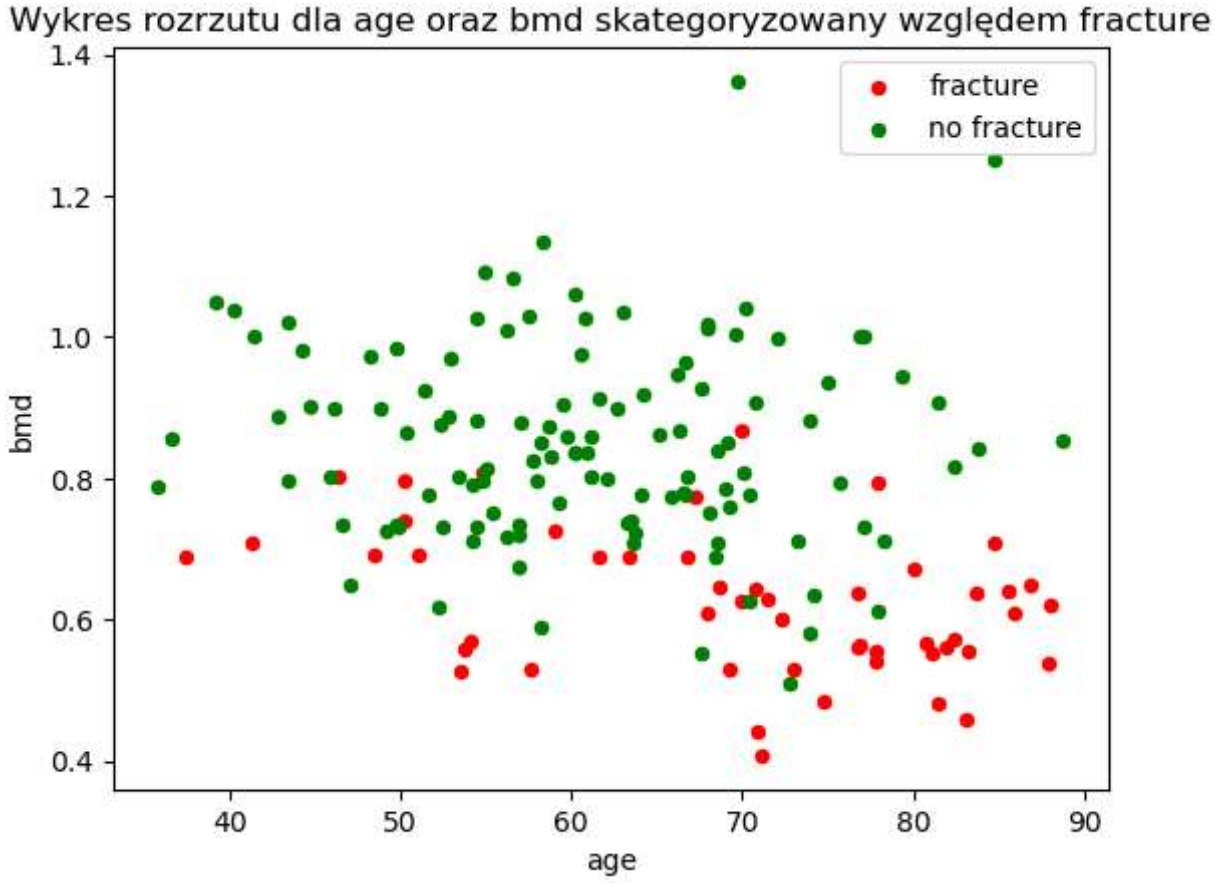
```
Zmienna age NIE POCHODZI z rozkładu normalnego. P-value: 0.03202778072330057
Zmienna bmd POCHODZI z rozkładu normalnego. P-value: 0.2986369345960301
```

Spośród zmiennych użytych do badania słuszności hipotezy 2, tylko zmienna `bmd` cechuje się rozkładem normalnym, zakładając poziom istotności alfa = `0.05`.

10. Wykresy rozrzutu

Ponieważ zmienna zależna nie jest zmienną ilościową, nie da się dla niej wygenerować wykresu rozrzutu. Jednak można wykonać wykres rozrzutu dla zmiennych opisujących `bmd` oraz `age` skategoryzowany względem zmiennej zależnej `fracture`.

In [ ]:
```
x='age'
y='bmd'

fig, ax = plt.subplots()
colors = {'fracture':'red', 'no fracture':'green'}

grouped = df.groupby('fracture')
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x=x, y=y, label=key, color=colors[key], marker='o')

plt.title(f'Wykres rozrzutu dla {x} oraz {y} skategoryzowany względem fracture')
plt.show()
```



Obserwacje:

- Znaczna część przypadków, gdzie doszło do złamania występuje dla wartości wskaźnika `bmd` mniejszego niż ok. 0.7
- Przesuwając się na prawo wzdłuż osi poziomej - im większy wiek tym więcej przypadków złamania.

11. Opracowanie wyników i wnioski

- Dla tej hipotezy nie można określić korelacji między predyktorami a zmienną zależną, gdyż jest ona jakościowa.
- Występuje umiarkowana korelacja między predyktorami `age` oraz `bmd` (wsp. kor. = -0.31.).
- Widzimy wyraźny związek między zmienną `fracture` a predyktorami `bmd` oraz `age`. Można pokusić się o stwierdzenie, że im starsza dana osoba i im mniej minerałów zawierają kości tej osoby, tym większa szansa, że osoba dozna złamania.
- W przypadku zmiennej `medication` relacja ta nie jest tak oczywista. Stosunek osób, które doznały złamania do osób bez złamania, gdzie każda z nich była leczona przy pomocy glikokortykoidów wynosi 5:19. Zatem ok. 80% osób, które były leczone przy pomocy glukokortykozy nie doznały

złamania - możliwe że ten lek jest skuteczny w leczeniu niedoboru minerału w kościach i pozwala zapobiec złamaniom

Indukcja drzew decyzyjnych - drzewo klasyfikacyjne dla zmiennej `fracture`

In [ ]:
```
# Niezbędne biblioteki do drzew klasyfikacyjnych
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Przygotowanie danych do modelu

In [ ]:
```
df = pd.read_csv('data/bmd_prep.csv')

target_var = 'fracture'
# features to exclude from model
vars_to_exclude = ['sex', 'waiting_time', 'weight_kg']

y = df[target_var]
X = df.drop(columns=[target_var]+vars_to_exclude)

fracture_labels = ['no fracture', 'fracture']
medication_labels = ['No medication', 'Glucocorticoids', 'Anticonvulsant']
# Treat two medication methods as one
X['medication'] = X['medication'].replace(medication_labels, [0, 1, 1])
y = y.replace(fracture_labels, [0, 1])

# Split data to training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
X_train.head()
```

Out[ ]:

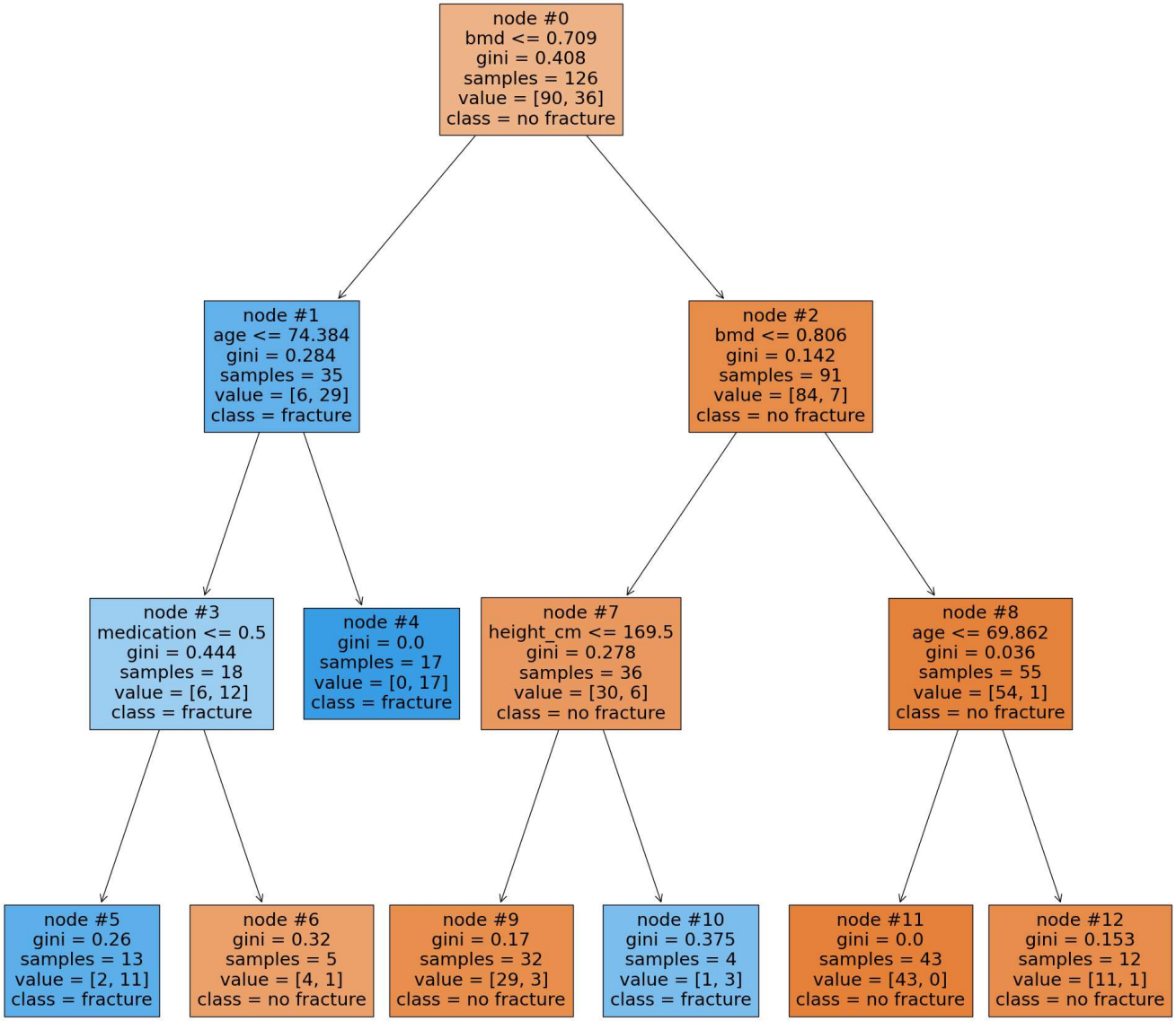
	age	height_cm	medication	bmd
16	57.934593	160.0	1	0.7978
66	84.787394	165.0	0	1.2508
136	68.017748	153.0	0	0.6103
40	73.230427	153.0	0	0.7128
160	84.721764	161.0	0	0.7085

Do modelu została uwzględniona zmienna `height_cm` ze względu na większą dokładność otrzymywanego modelu

1. Utworzenie drzewa klasyfikacyjnego

In [ ]:
```
params = {
    'max_depth': 3,
    'min_samples_split': 5,
    'max_leaf_nodes': 10
}
tree = DecisionTreeClassifier(**params)

# Fits the decision tree to training data
```

```
tree.fit(X_train, y_train)

fig = plt.figure(figsize=(25, 25))

# Creates a visual display of the model.
# Keep max_depth small for better visualization
t = plot_tree(
    tree,
    feature_names=X.columns,
    class_names=fracture_labels,
    node_ids=True,
    filled=True)
```

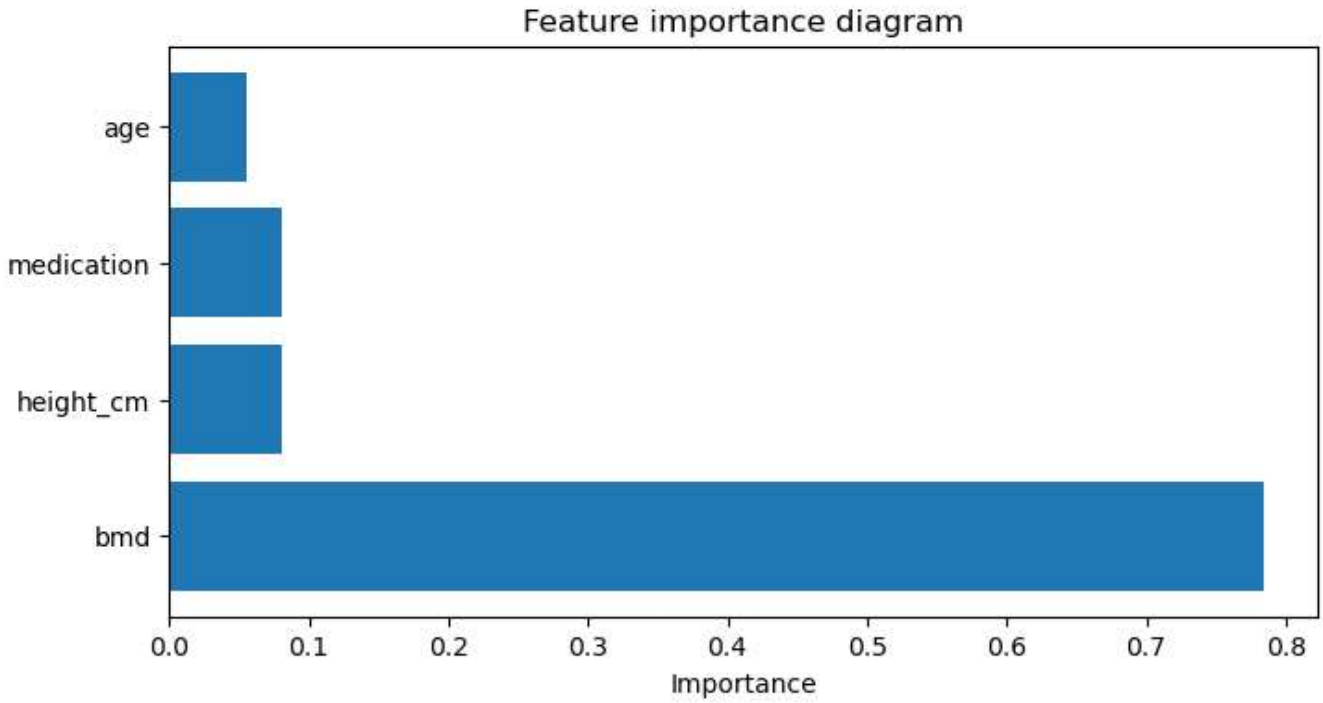


2. Ważność predyktorów

In [ ]:
```
features = X.columns.tolist()

importances_sk = tree.feature_importances_

feature_importance = []
for i, feature in enumerate(features):
    feature_importance.append({'feature': feature,'importance': round(importances_sk[i], 3)})
```

```python
# sort features by their importance
feature_importance = sorted(feature_importance, key=lambda d: d['importance'], reverse=True)

feature_importance_df = pd.DataFrame(columns=['feature', 'importance'])
feature_importance_df = feature_importance_df.from_records(feature_importance)

plt.figure(figsize=(8,4))
plt.barh(feature_importance_df['feature'], feature_importance_df['importance'])
plt.xlabel('Importance')
plt.title(f'Feature importance diagram')
```

Out[ ]: Text(0.5, 1.0, 'Feature importance diagram')



Najważniejszą zmienną objaśniającą w otrzymanym modelu jest wskaźnik `bmd`. Pozostałe zmienne wykorzystane w modelu są mniej ważne

3. Reguły dla najbardziej wyrazistych klas

1. Reguła dla węzła #5 - Jeśli wskaźnik `bmd` jest mniejszy bądź równy 0.709, wiek mniejszy niż 74 lata oraz pacjent nie był leczony, istnieje niemal 85% szansy, że osoba ta dozna złamania

- Pewność reguły: 11/13 * 100% = `84.5%`
- Wsparcie reguły: 13/126 * 100% = `10.3%`

2. Reguła dla węzła #4 - Jeśli wskaźnik `bmd` danej osoby jest mniejszy bądź równy 0.709 oraz jej wiek (`age`) jest większy niż 75 lat, to osoba ta doznała złamania

- Pewność reguły: 17/17 * 100% = `100%`
- Wsparcie reguły: 17/126 * 100% = `13.5%`

3. Reguła dla węzła #9 - Jeśli wskaźnik `bmd` osoby jest w przedziale (0.709, 0.806], a jej wzrost jest mniejszy bądź równy 169.5 cm, jest 90% szansy, że osoba ta nie doznała złamania

- Pewność reguły: 29/32 * 100% = `90.5%`
- Wsparcie reguły: 32/126 * 100% = `25.4%`

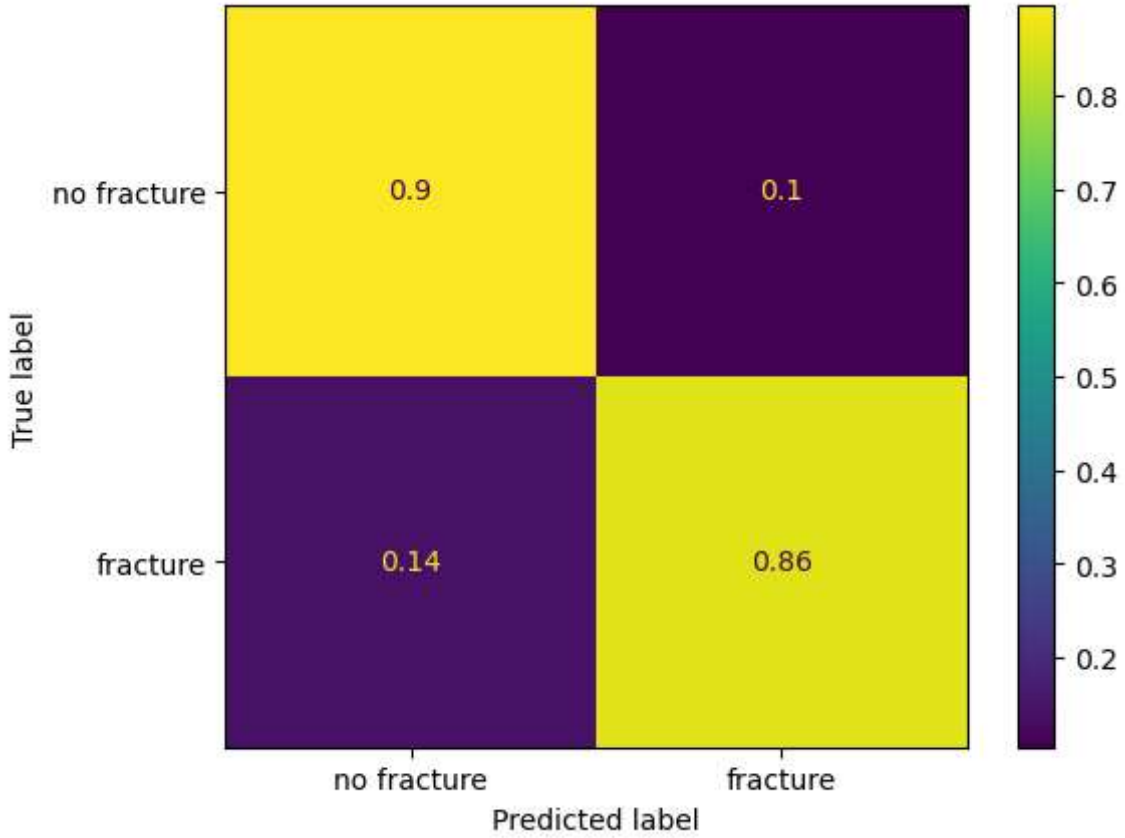
4. Reguła dla węzła #8 - Jeśli wskaźnik `bmd` osoby jest większy niż 0.806, to jest ponad 98% szansy, że osoba ta nie doznała złamania.

- Pewność reguły: 54/55 * 100% = `98.2%`
- Wsparcie reguły: 55/126 * 100% = `43.6%`

4. Macierz klasyfikacji

In [ ]:
```
predictions = tree.predict(X_test)
cm = confusion_matrix(y_test, predictions, labels=tree.classes_, normalize='true')
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=fracture_labels)
disp.plot()
```

Out[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x17f59a8d9a0>
```



Wygenerowane drzewo przewiduje poprawnie brak złamania z `90%` dokładnością (True Negative) oraz jego obecność z dokładnością `86%` (True Positive)

Całkowity błąd drzewa

In [ ]:
```
print(
    'Accuracy on training data: '
    + str("{:.2%}".format(tree.score(X_train, y_train))))

print(
    'Accuracy on testing data: '
    + str("{:.2%}".format(tree.score(X_test, y_test))))
```

```
Accuracy on training data: 93.65%
Accuracy on testing data: 88.37%
```

Błąd drzewa w przypadku danych uczących wyniósł 100%-93.65% = `6.35%`, a dla zbioru testowego `11.63%`. Zatem jest to dość dokładny model

5. Weryfikacja hipotezy na podstawie drzewa

Na podstawie reguły 1 można stwierdzić, że osoby, które miały słabe kości (`bmd` niskie) oraz nie były leczone, istniała duża szansa, że osoba ta miała złamanie. Gdyby regułę zdefiniować dla sąsiedniego liścia, gdzie pacjenci byli leczeni istnieje duża szansa (4/5), że osoba ta nie doznała złamania. Zatem stwierdzenie, że leczenie wpływa na wystąpienie złamania zdefiniowane w ramach hipotezy jest słuszne.

Porównując ze sobą reguły 2, 3 i 4 widzimy, że im wyższy wskaźnik `bmd` tym większa szansa, że osoba nie doznała złamania. Zatem stwierdzenie, że `bmd` wpływa na złamanie również jest uzasadnione.

Wiek również ma (co prawda najmniejszy spośród wszystkich zmiennych) wpływ. Wg. reguły 2, osoby, które miały niski wskaźnik `bmd` oraz były w wieku powyżej 74 lat, doznały złamania ze 100% pewnością. Osoby młodsze niż 74 lata również doznały złamań ale szansa na to była mniejsza, gdy dodatkowo zostało zastosowane leczenie.

Podsumowując, na podstawie wyników otrzymanych przy pomocy drzewa, hipotezę 2 można przyjąć. Każda ze zmiennych objaśniających ma większy lub mniejszy ale istotny wpływ na poprawne zaklasyfikowanie pacjenta pod względem wystąpienia złamania bądź nie.

Wybrany algorytm data mining - Naiwny klasyfikator Bayesa

In [ ]:
```
# Niezbędne biblioteki
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
```

1. Przygotowanie zbioru danych i wybór zmiennych

In [ ]:
```
df = pd.read_csv('data/bmd_prep.csv')

target_var = 'fracture'

numeric_predictors = df.select_dtypes(include=np.number).columns.tolist()
categorical_predictors =  df.select_dtypes(exclude=np.number).columns.tolist()
# # exclude less important predictors
numeric_predictors.remove('height_cm')
numeric_predictors.remove('waiting_time')
# exclude target variable from training set
categorical_predictors.remove(target_var)

fracture_labels = ['no fracture', 'fracture']

X = df[numeric_predictors + categorical_predictors]
y = df[target_var]

# Encode categorical data
X_encoded = pd.get_dummies(X)
y_encoded = y.replace(fracture_labels, [0, 1])

# Split data to training and testing set
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded, test_size=0.25, ran
X_train.head()
```

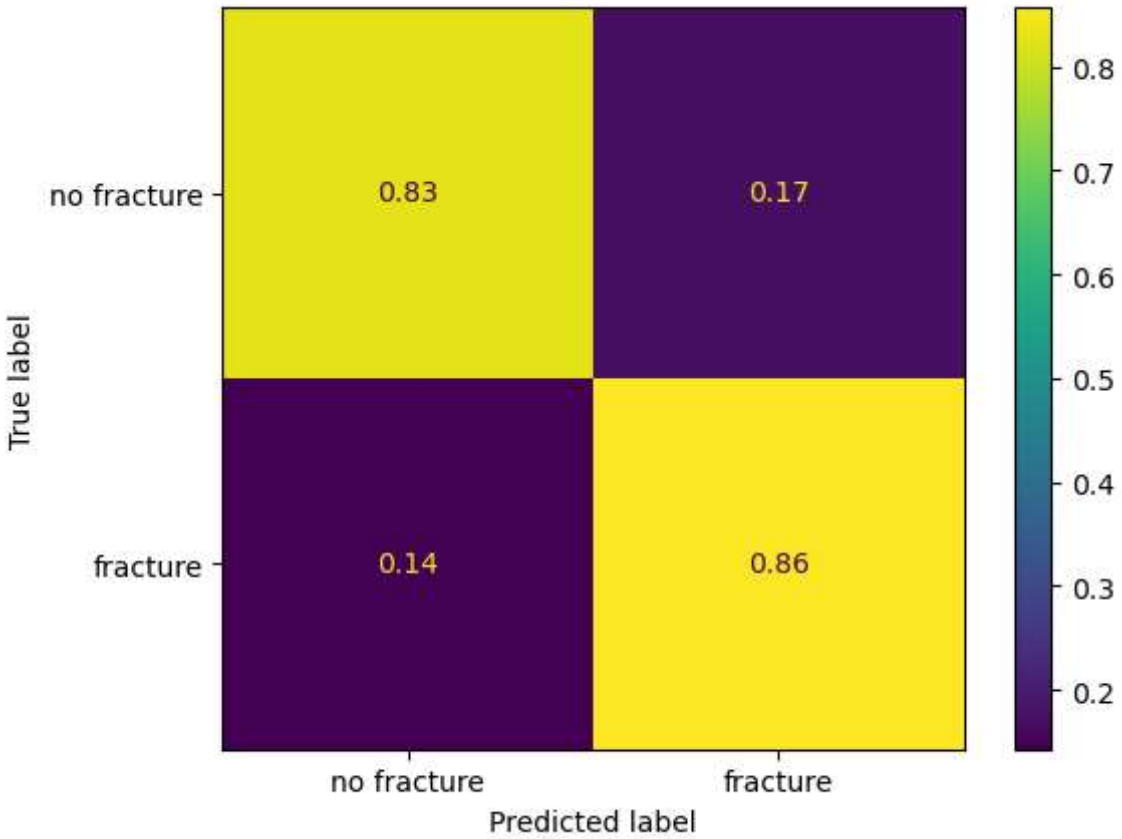
	age	weight_kg	bmd	sex_F	sex_M	medication_Anticonvulsant	medication_Glucocorticoids	medi n
16	57.934593	67.0	0.7978	0	1	0	1	
66	84.787394	68.0	1.2508	0	1	0	0	
136	68.017748	58.0	0.6103	1	0	0	0	
40	73.230427	52.0	0.7128	1	0	0	0	
160	84.721764	52.0	0.7085	0	1	0	0	

Dla wybranych wyżej zmiennych naiwny klasyfikator Bayesa osiągnął najmniejszą wartość błędu

2. NaiveBayes - macierz klasyfikacji

In [ ]:
```
# fit data to NaiveBayes object
nb = GaussianNB() \
    .fit(X_train, y_train)

predictions = nb.predict(X_test)
cm = confusion_matrix(y_test, predictions, labels=nb.classes_, normalize='true')
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=fracture_labels)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x24f0c2b5a60>
```



In [ ]:
```
print(
    'Accuracy on training data: '
    + str("{:.2%}".format(nb.score(X_train, y_train))))
print(
    'Accuracy on testing data: '
    + str("{:.2%}".format(nb.score(X_test, y_test))))
```

```
Accuracy on training data: 84.92%
Accuracy on testing data: 83.72%
```

3. Interpretacja wyników

Naiwny klasyfikator Bayesa stanowi jeden z najprostszych algorytmów uczenia nadzorowanego. Jest na ogół szybki, dokładny i niezawodny. Cechują się wysoką dokładnością na dużych zbiorach danych. Jednak w tym przypadku do analizy wykorzystany zbiór danych nie jest duży, ale mimo to poradził sobie całkiem dobrze.
Na zbiorze treningowym i testowym osiągnął błąd rzędzu `14-15%`. Jest to wynik gorszy niż uzyskany przy pomocy drzewa klasyfikacyjnego, ale również pozwolił na dość dobre przewidzenie wartości zmiennej zależnej (tutaj `fracture`) przy pomocy wybranych zmiennych objaśniających.

Naiwny klasyfikator Bayesa również potwierdza słuszność hipotezy iż wiek, rodzaj leczenia i wskaźnik `bmd` (zmienne objaśniające użyte w modelu) wpływają na wystąpienie złamania.

Analiza skupień

In [ ]:
```python
# Niezbędne biblioteki do analizy skupień
import pandas as pd
import numpy as np
import warnings
from sklearn.cluster import KMeans
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.mixture import GaussianMixture
```

1. Przygotowanie danych i wybór zmiennych

In [ ]:
```python
df = pd.read_csv('data/bmd_prep.csv')

numeric_predictors = df.select_dtypes(include=np.number).columns.tolist()
categorical_predictors =  df.select_dtypes(exclude=np.number).columns.tolist()
# exclude less important predictors
categorical_predictors.remove('sex')
numeric_predictors.remove('height_cm')
numeric_predictors.remove('waiting_time')

X = df[numeric_predictors + categorical_predictors]

# one-hot encode categorical variables
X = pd.get_dummies(X)

# Standarize data
scaler = MinMaxScaler()
X[numeric_predictors] = scaler.fit_transform(X[numeric_predictors])
X.head()
```

Out[ ]:

	age	weight_kg	bmd	fracture_fracture	fracture_no fracture	medication_Anticonvulsant	medication_Glucoc
0	0.401187	0.466667	0.494030	0	1	1	
1	0.754200	0.700000	0.405320	0	1	0	
2	0.660465	0.616667	0.522727	0	1	0	
3	0.801536	0.400000	0.317972	0	1	0	
4	0.347146	0.316667	0.401445	0	1	0	

Do analizy zostały wybrane wszystkie zmienne oprócz `sex`, `height_cm` oraz `waiting_time`.

- Zmienną `sex` można odrzucić, gdyż jej ważność określona przy okazji testu Chi kwadrat oraz indukcji drzew decyzyjnych okazała się bardzo mała. Nie obsewowano dużej różnicy między mężczyznami a kobietami.
- Zmienną `height_cm` oraz `waiting_time` również można odrzucić, ze względu na potrzebę redukcji wymiarowości. Innym powodem jest fakt, iż te zmienne nie zostały wzięte pod uwagę przy definiowaniu hipotez.

2. Analiza skupień - metoda k-średnich

Poniższy kod wyznacza optymalną liczbę skupień przy pomocy 10-krotnego sprawdzianu krzyżowego

In [ ]:

```
warnings.filterwarnings('ignore')

def custom_scorer(estimator,X):
    return silhouette_score(X, estimator.predict(X), metric='euclidean')

# KMeans object
km = KMeans(init='k-means++', n_init=10, max_iter=50)

# List of possible number of clusters
param = range(2,10)

# Sets up GridSearchCV object and stores in grid variable
grid = GridSearchCV(
    km,
    {'n_clusters': param},
    scoring=custom_scorer,
    cv=10) # cross validations

# Fits the grid object to data
grid.fit(X)

# Accesses the optimum model
best_km = grid.best_estimator_

# Displays the optimum model
best_km.get_params()
```

Out[ ]:
```
{'algorithm': 'lloyd',
 'copy_x': True,
 'init': 'k-means++',
 'max_iter': 50,
 'n_clusters': 5,
 'n_init': 10,
 'random_state': None,
 'tol': 0.0001,
 'verbose': 0}
```

Optymalna liczba skupień na podstawie sprawdzianu krzyżowego - `5`

Wykonanie analizy

In [ ]:
```
params = best_km.get_params()
km = KMeans(**params)
# Fits the model to the data
km.fit(X)
```

Out[ ]:
```
▾ KMeans
KMeans(max_iter=50, n_clusters=5, n_init=10)
```

3. Interpretacja wyników - metoda k-średnich

Charakterystyka skupień - statystyki dla zmiennych ilościowych

In [ ]:
```
km_results = df[numeric_predictors]
km_results['Cluster'] = km.labels_

km_results.groupby('Cluster')[numeric_predictors].agg(['min', 'max', 'mean', 'std'])
```

	age				weight_kg						
	min	max	mean	std	min	max	mean	std	min	max	mean
Cluster											
0	50.276059	86.859869	73.803475	13.105046	40.0	79.0	57.333333	10.665365	0.4076	0.7399	0.566000
1	35.814058	84.787394	60.776615	11.216124	39.0	96.0	67.505263	10.608982	0.5514	1.2508	0.857752
2	37.461995	88.024151	68.886213	13.436667	36.0	85.0	58.646341	11.174885	0.4407	0.8664	0.635888
3	43.450712	88.753795	62.691945	10.998231	47.0	90.0	67.947368	11.896422	0.5090	1.3624	0.836605
4	56.931266	70.421873	60.047460	5.823149	51.0	74.0	60.800000	8.871302	0.5899	1.0287	0.759460

Charakterystyka skupień - tabele liczności dla zmiennych jakościowych

In [ ]:
```
km_results = df[categorical_predictors]
km_results['Cluster'] = km.labels_

for predictor in categorical_predictors:
    counter_table = km_results.groupby(['Cluster', predictor])[predictor].agg(['count'])
    s = counter_table.style.set_caption(f'Counter table for predictor: {predictor}')
    display(s)
```

Counter table for categorical predictor: fracture

Cluster	fracture	count
0	**fracture**	9
1	**no fracture**	95
2	**fracture**	41
3	**no fracture**	19
4	**no fracture**	5

Counter table for categorical predictor: medication

Cluster	medication	count
0	**Anticonvulsant**	4
	Glucocorticoids	5
1	**No medication**	95
2	**No medication**	41
3	**Glucocorticoids**	19
4	**Anticonvulsant**	5

Otrzymane skupienia są jednorodne pod względem wystąpienia złamania (`fracture`)

Skupienie 1 (0):

- najstarsza grupa wiekowa spośród uzyskanych, średnia wieku 73 lata
- osoby, które średnio ważą najmniej
- najmniejsza średnia wartość wskaźnika `bmd` (z kośćmi najmniej bogatymi w minerały)
- są to osoby, które pomimo leczenia, doznały złamania

Skupienie 2:

- najliczniejsza i jedna z najmłodszych otrzymanych grup
- osoby cechują się stosunkowo dużą wagą, choć zdarzają się osoby o małej wadze
- mają zdrowe kości - `bmd` na wysokim poziomie
- są to osoby zdrowe, gdyż nie doznały złamania i nie były leczone

Skupienie 3:

- skupienie podobne do skupienia 1 pod względem wiekowym, wagowym i wskaźnika `bmd`
- W przeciwieństwie do skupienia 1, gdzie osoby były leczone, w tym skupieniu osoby nie były leczone
- osoby te doznały złamania - najprawdopodobniej spowodowane słabymi kośćmi (niskie `bmd`)

Skupienie 4:

- grupa osób młodszych w stosunku do pozostałych, o średnio dużej wadzę
- duża ilość minerałów w kościach
- nie doznali złamania, leczeni przy pomocy glikokortykoidów (można tą grupę interpretować jako tych co dzięki lekom nie doznali złamania)

Skupienie 5:

- ludzie, którzy nie doznali złamania
- stosunkowo mała waga
- leczeni przy pomocy leków przeciwdrgawkowych, nie doznali złamań

4. Analiza skupień - metoda EM

In [ ]:
```
gmm = GaussianMixture(n_components=5)

# Fit model to data
gmm.fit(X)

# Predict cluster labels for data
labels = gmm.predict(X)
```

Charakterystyka skupień - statystyki zmiennych ilościowych

In [ ]:
```
em_results = df[numeric_predictors]
em_results['Cluster'] = labels

em_results.groupby('Cluster')[numeric_predictors].agg(['min', 'max', 'mean', 'std'])
```

Out[ ]:

	age				weight_kg						
	min	max	mean	std	min	max	mean	std	min	max	mean
Cluster											
0	50.276059	86.859869	75.072445	16.763269	52.0	62.0	57.000000	4.760952	0.4586	0.7399	0.629700
1	35.814058	84.787394	60.776615	11.216124	39.0	96.0	67.505263	10.608982	0.5514	1.2508	0.857752
2	37.461995	88.024151	69.310353	13.173154	36.0	85.0	58.532609	11.393176	0.4076	0.8664	0.622752
3	43.450712	88.753795	62.691945	10.998231	47.0	90.0	67.947368	11.896422	0.5090	1.3624	0.836605
4	56.931266	70.421873	60.047460	5.823149	51.0	74.0	60.800000	8.871302	0.5899	1.0287	0.759460

Charakterystyka skupień - tabele liczności dla zmiennych jakościowych

In [ ]:
```
em_results = df[categorical_predictors]
em_results['Cluster'] = labels

for predictor in categorical_predictors:
    counter_table = em_results.groupby(['Cluster', predictor])[predictor].agg(['count'])
    s = counter_table.style.set_caption(f'Counter table for predictor: {predictor}')
    display(s)
```

Counter table for predictor: fracture

Cluster	fracture	count
0	**fracture**	4
1	**no fracture**	95
2	**fracture**	46
3	**no fracture**	19
4	**no fracture**	5

Counter table for predictor: medication

Cluster	medication	count
0	**Anticonvulsant**	4
1	**No medication**	95
2	**Glucocorticoids**	5
	No medication	41
3	**Glucocorticoids**	19
4	**Anticonvulsant**	5

5. Porównanie charakterystyk skupień z hipotezami badawczymi

Otrzymane skupienia przy pomocy metod k-średnich oraz EM są bardzo podobne (wręcz identyczne) zatem nie ma sensu opisywać dla metody EM od nowa, de facto tej samej charakterystyki.

Zgodność z hipotezami została poniżej opisana na podstawie wyników z metody k-średnich.

Otrzymane skupienia potwierdzają zgodność hipotez. Widać, że wraz ze wzrostem wieku wartość `bmd` spada, a im większa waga tym większe `bmd`. W skupieniach 2 i 4 znajdują się osoby, które ważą dużo, są stosunkowo młodsze (ok. 60 lat) i rzeczywiście ich wskaźnik `bmd` jest na wysokim poziomie. Jako kontrast, w skupieniach 1 i 3 znajdują się osoby, które są starsze (ok. 70 lat), ważą mniej niż w skupieniach 2 i 4, a ich wskaźnik `bmd` jest niższy. Tym samym słuszność hipotezy 1 jest potwierdzona

Hipoteza 2 jest również słuszna - ludzie z niskim `bmd` i starsi doznają na ogół złamań (jak w skupieniach 1 i 3), natomiast ludzie, u których `bmd` ma wysoką wartość i są młodsze (skupienia 2 i 4) - na ogól nie doznają złamań.

Ciekawą, wyróżniającą się grupą jest skupienie 5, u których waga jak i wskaźnik `bmd` nie są wcale największe a mimo to nie doznali złamań. Należy zaznaczyć, że jest to grupa osób u których podano leki. Można wnioskować, że podane pacjentom leki sprawiły, że uniknęły one złamania. Zatem leki mogą mieć wpływ na wystąpienie złamania (jak założono w hipotezie 2).