```
In [1]: import sklearn
from sklearn import  datasets
from sklearn import model_selection
from sklearn import tree
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn import ensemble
from sklearn.model_selection import KFold
import numpy as np
```

Zadanie 1

```
In [2]: dane = datasets.load_iris()
x = dane["data"]
y = dane["target"]
```

```
In [3]: X_train,X_test, Y_train,Y_test = model_selection.train_test_split(x,y,train_size=0.8)
```

```
In [4]: model = tree.DecisionTreeClassifier()
```

```
In [5]: model.fit(X_train,Y_train)
```

```
Out[5]: DecisionTreeClassifier
DecisionTreeClassifier()
```

```
In [6]: y_pred = model.predict(X_test)
```

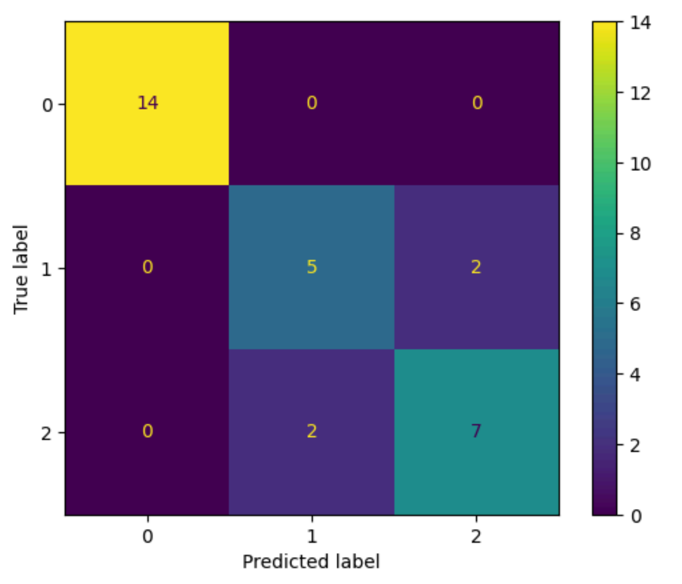
```
In [7]: metrics.accuracy_score(Y_test,y_pred)
```

```
Out[7]: 0.8666666666666667
```

```
In [8]: matrix = metrics.confusion_matrix(Y_test,y_pred)
```

```
In [9]: display = metrics.ConfusionMatrixDisplay(confusion_matrix=matrix,display_labels=model.classes_)
```

```
In [10]: display.plot()
plt.show()
```

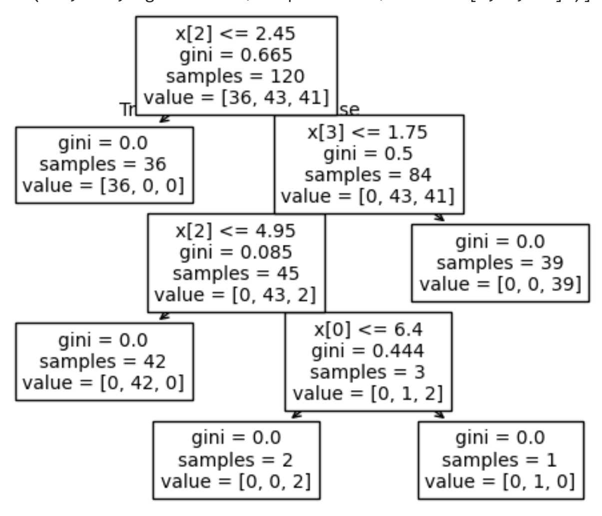



```
In [11]: metrics.classification_report(Y_test,y_pred)
```

```
Out[11]: '              precision    recall  f1-score   support\n\n           0       1.00      1.00      1.00        14\n           1       0.71      0.71      0.71         7\n           2       0.78      0.78      0.78         9\n\n    accuracy                           0.87        30\n   macro avg       0.83      0.83      0.83        30\nweighted avg       0.87      0.87      0.87        30\n'
```

```
In [12]: tree.plot_tree(model)
```

```
Out[12]: [Text(0.4, 0.9, 'x[2] <= 2.45\ngini = 0.665\nsamples = 120\nvalue = [36, 43, 41]'),
 Text(0.2, 0.7, 'gini = 0.0\nsamples = 36\nvalue = [36, 0, 0]'),
 Text(0.30000000000000004, 0.8, 'True  '),
 Text(0.6, 0.7, 'x[3] <= 1.75\ngini = 0.5\nsamples = 84\nvalue = [0, 43, 41]'),
 Text(0.5, 0.8, '  False'),
 Text(0.4, 0.5, 'x[2] <= 4.95\ngini = 0.085\nsamples = 45\nvalue = [0, 43, 2]'),
 Text(0.2, 0.3, 'gini = 0.0\nsamples = 42\nvalue = [0, 42, 0]'),
 Text(0.6, 0.3, 'x[0] <= 6.4\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(0.4, 0.1, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(0.8, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(0.8, 0.5, 'gini = 0.0\nsamples = 39\nvalue = [0, 0, 39]')]
```



Zadanie 2

```
In [13]: dane_wina = datasets.load_wine()
x2 = dane_wina['data']
y2 = dane_wina['target']
```

```
In [14]: X_train,X_test, Y_train,Y_test = model_selection.train_test_split(x2,y2,train_size=0.7)
```

```
In [15]: model_linear = sklearn.svm.SVC(kernel='linear')
model_rbf = sklearn.svm.SVC(kernel="rbf")
model_poly = sklearn.svm.SVC(kernel='poly')
```

```
In [16]: model_linear.fit(X_train,Y_train)
model_rbf.fit(X_train,Y_train)
model_poly.fit(X_train,Y_train)
```

```
Out[16]: SVC
SVC(kernel='poly')
```

```
In [17]: y_pred_lin = model_linear.predict(X_test)
y_pred_rbf = model_rbf.predict(X_test)
y_pred_poly = model_poly.predict(X_test)
```

```
In [18]: metrics.accuracy_score(Y_test,y_pred_lin)
```

```
Out[18]: 0.9259259259259259
```

```
In [19]: metrics.accuracy_score(Y_test,y_pred_rbf)
```

```
Out[19]: 0.7222222222222222
```

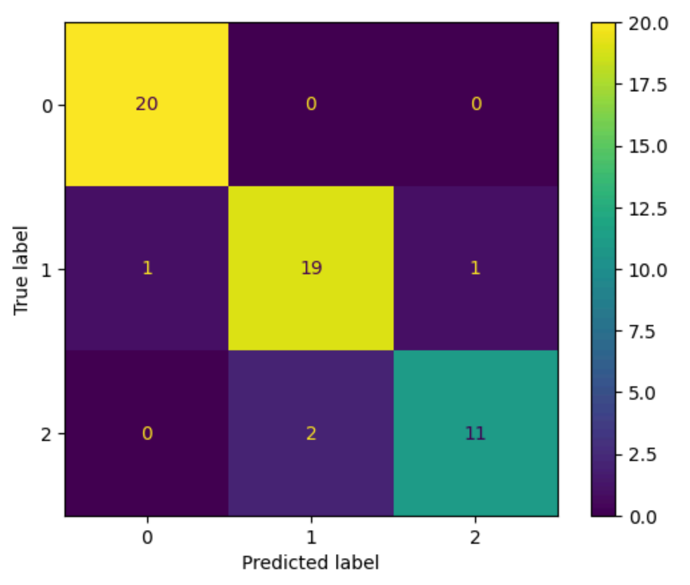
```
In [20]: metrics.accuracy_score(Y_test,y_pred_poly)
```

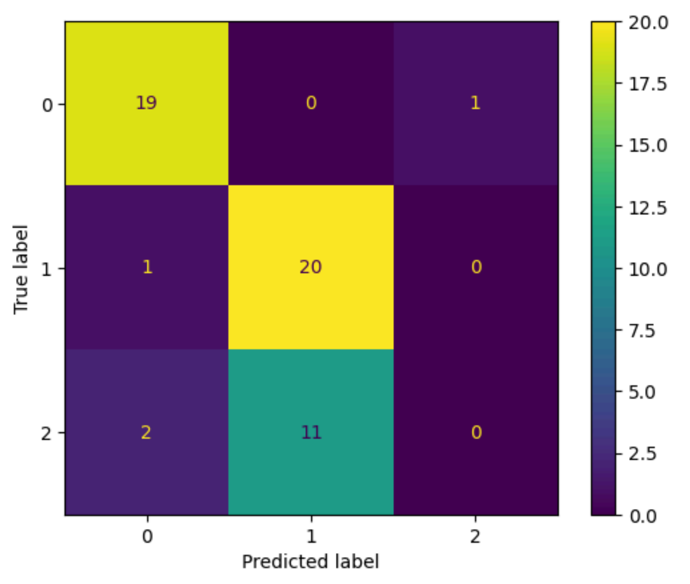
```
Out[20]: 0.7407407407407407
```

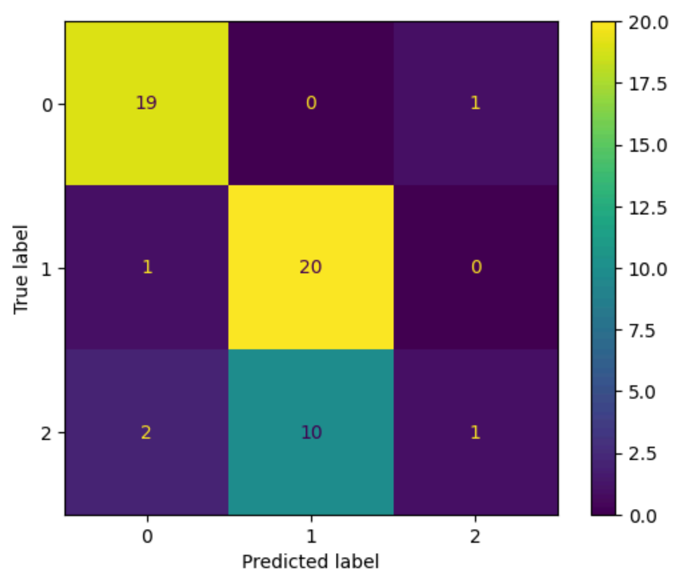
```
In [21]: matrix_lin = metrics.confusion_matrix(Y_test,y_pred_lin)
matrix_rbf = metrics.confusion_matrix(Y_test,y_pred_rbf)
matrix_poly = metrics.confusion_matrix(Y_test,y_pred_poly)
```

```
In [22]: display1 = metrics.ConfusionMatrixDisplay(confusion_matrix=matrix_lin,display_labels=model.classes_)
display2 = metrics.ConfusionMatrixDisplay(confusion_matrix=matrix_rbf,display_labels=model.classes_)
display3 = metrics.ConfusionMatrixDisplay(confusion_matrix=matrix_poly,display_labels=model.classes_)
```

```
In [23]: display1.plot()
display2.plot()
display3.plot()
plt.show()
```








```
In [24]: metrics.classification_report(Y_test,y_pred_lin)
```

```
Out[24]: '              precision    recall  f1-score   support\n\n           0       0.95      1.00      0.98        20\n           1       0.90      0.90      0.90        21\n           2       0.92      0.85      0.88        13\n\n    accuracy                           0.93        54\n   macro avg       0.92      0.92      0.92        54\nweighted avg       0.93      0.93      0.93        54\n'
```

```
In [25]: metrics.classification_report(Y_test,y_pred_rbf,zero_division=True)
```

```
Out[25]: '              precision    recall  f1-score   support\n\n           0       0.86      0.95      0.90        20\n           1       0.65      0.95      0.77        21\n           2       0.00      0.00      0.00        13\n\n    accuracy                           0.72        54\n   macro avg       0.50      0.63      0.56        54\nweighted avg       0.57      0.72      0.63        54\n'
```

```
In [26]: metrics.classification_report(Y_test,y_pred_poly)
```

```
Out[26]: '              precision    recall  f1-score   support\n\n           0       0.86      0.95      0.90        20\n           1       0.67      0.95      0.78        21\n           2       0.50      0.08      0.13        13\n\n    accuracy                           0.74        54\n   macro avg       0.68      0.66      0.61        54\nweighted avg       0.70      0.74      0.67        54\n'
```

Zadanie 3

```
In [27]: dane_domy = datasets.fetch_california_housing()
x3 = dane_domy['data']
y3 = dane_domy['target']
```

```
In [28]: X_train,X_test, Y_train,Y_test = model_selection.train_test_split(x3,y3,train_size=0.75)
```

```
In [29]: model_boost = ensemble.GradientBoostingRegressor()
```

```
In [30]: kf = KFold(n_splits=10, shuffle=True, random_state=50)
```

```
In [34]: print(model_selection.cross_val_score(model_boost,x3,y3,cv=kf))
```

```
[0.77881172 0.7867148  0.78881941 0.78398672 0.78651334 0.78666609
 0.8068168  0.78909195 0.79188371 0.7869703 ]
```

```
In [35]: model_boost.fit(X_train,Y_train)
```

```
Out[35]: GradientBoostingRegressor
GradientBoostingRegressor()
```

```
In [37]: y_pred = model_boost.predict(X_test)
```

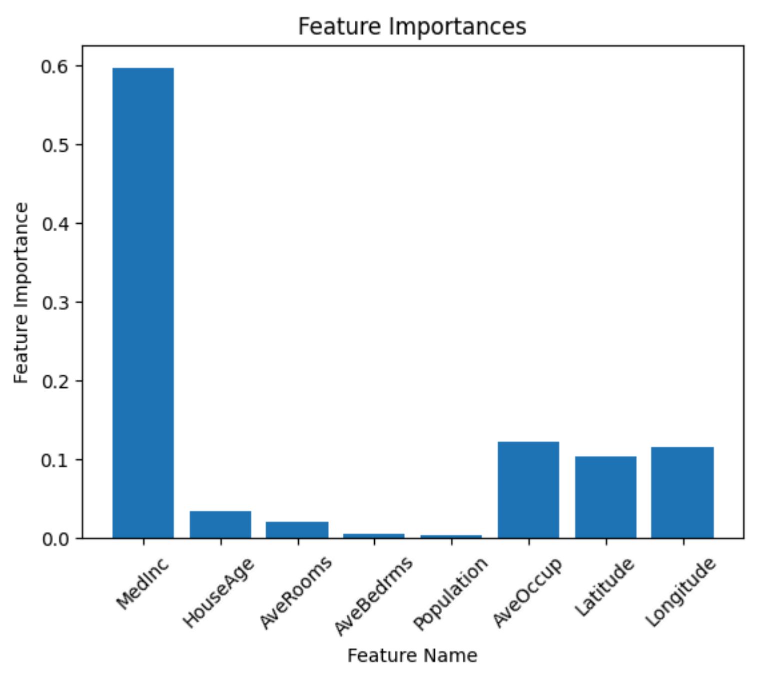
```
In [38]: print(metrics.r2_score(Y_test,y_pred))
print(metrics.mean_absolute_error(Y_test,y_pred))
```

```
0.7929867314675171
0.36510129848669515
```

```
In [42]: x5 = model_boost.feature_importances_
feature_names = dane_domy['feature_names']
```

```
In [43]: plt.bar(feature_names, x5)
plt.xticks(rotation=45)
plt.ylabel('Feature Importance')
plt.xlabel('Feature Name')
plt.title('Feature Importances')
```

```
Out[43]: Text(0.5, 1.0, 'Feature Importances')
```




```
In [44]: plt.show()
```

```
In [ ]:
```

```
In [ ]:
```