TUGAS 3 DIKI FERNANDI

datalab

```
# Import scikit-learn dataset library
from sklearn import datasets

# Load dataset
cancer = datasets.load_breast_cancer()
```

```
# Print the names of the 30 features
print("Features:", cancer.feature_names)

# Print the label types of cancer ('malignant' or 'benign')
print("Labels:", cancer.target_names)
```

```
Features: ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels: ['malignant' 'benign']
```

```
# Import train_test_split function from scikit-learn
from sklearn.model_selection import train_test_split

# Split dataset into training set (70%) and test set (30%)
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data,            # Feature data
    cancer.target,          # Target labels
    test_size=0.3,          # 30% for test set
    random_state=109        # Seed for reproducibility
)
```

```
# Import Support Vector Machine (SVM) model from scikit-learn
from sklearn import svm

# Create an SVM classifier with linear kernel
clf = svm.SVC(kernel='linear')  # Linear Kernel for linear classification

# Train the model using the training data
clf.fit(X_train, y_train)  # X_train: training features, y_train: training labels

# Predict the target values for the test set
y_pred = clf.predict(X_test)  # X_test: test features
```

```python
# Import metrics module from scikit-learn for model evaluation
from sklearn import metrics

# Evaluate model performance
print("Model Evaluation Metrics:")
print("-" * 25)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))  # Overall correctness of the model
print("Precision:", metrics.precision_score(y_test, y_pred))  # True positives / (True positives + False positives)
print("Recall:", metrics.recall_score(y_test, y_pred))  # True positives / (True positives + False negatives)
```

```
Model Evaluation Metrics:
-------------------------
Accuracy: 0.9649122807017544
Precision: 0.9811320754716981
Recall: 0.9629629629629629
```