Challenge W10 - SVM

1. Execute code from DataCamp

Running by Ulivia Embun Tresna Wardani (10322015)

Loading data

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

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

Exploring Data

```
# Features
print("Features: ", cancer.feature_names)

# Targets
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']
```

Splitting Data

```
# import train_test_split function
from sklearn.model_selection import train_test_split

# split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,
random_state=109)
# 70% training and 30% test
```

Generating Model

Let's build support vector machine model. Fisrt, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function. Then, fit your model on train set using fit() and perform prediction on the test set using predict().

```
# import svm model
from sklearn import svm

# create a svm classifier
clf = svm.SVC(kernel='linear')

# train the model using the training sets
clf.fit(X_train, y_train)

# predict the response for test dataset
y_pred = clf.predict(X_test)
```

Evaluating Model

```
# import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# Model precision: what percentage of positive tuples are labeled as such?
print("Precision:", metrics.precision_score(y_test, y_pred))

# Model recall: what percentage of positive tuples are labelled as such?
print("Recall:", metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.9649122807017544
Precision: 0.9811320754716981
Recall: 0.9629629629629629
```