T1 - Pipeline

February 28, 2021

1 Pipelines

1.1 Setting up

[ ]:
```
import pandas as pd
import numpy as np

#Loading the Breast Cancer Wisconsin dataset
from sklearn.datasets import load_breast_cancer

# Load data
dataObj = load_breast_cancer()
X = dataObj.data
y = dataObj.target

# Create DataFrame with features
df = pd.DataFrame(X)
df.columns = dataObj.feature_names

# Add class column
df.insert(loc=0, column="Class", value=y)

# Explore data
display(df.head())
print(df.shape)
display(df.describe())
print(df['Class'].value_counts())
```

[ ]:
```
from sklearn.model_selection import train_test_split

# Splitting data
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.20,
    stratify=y,
    random_state=1)
```

1.2 Method 1: Without using a pipeline

[ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression

# Standardize
sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)

# PCA
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_std)
X_test_pca = pca.transform(X_test_std)

# Logistic Regression
lr = LogisticRegression(random_state=1)
lr.fit(X_train_pca, y_train)

# Making prediction from testing data
y_pred = lr.predict(X_test_pca)
print(y_pred)

# Training accuracy
training_accuracy = lr.score(X_train_pca, y_train)
print(f"Training Accuracy:{training_accuracy:6.3f}")

# Testing accuracy
testing_accuracy = lr.score(X_test_pca, y_test)
print(f"Testing Accuracy:{testing_accuracy:6.3f}")
```

1.3 Method 2: Using a pipeline

[ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

# Construct pipeline object
pipe_lr = Pipeline([('scl', StandardScaler()),
                    ('pca', PCA(n_components=2)),
                    ('clf', LogisticRegression(random_state=1))])

# Training
pipe_lr.fit(X_train, y_train)
```

```
# Making prediction from testing data
y_pred = pipe_lr.predict(X_test)
print(y_pred)

# Training accuracy
training_accuracy = pipe_lr.score(X_train, y_train)
print(f"Training Accuracy:{training_accuracy:6.3f}")

# Testing accuracy
testing_accuracy = pipe_lr.score(X_test, y_test)
print(f"Testing Accuracy:{testing_accuracy:6.3f}")
```

[ ]:
```
# Get parameter names
for k, v in pipe_lr.get_params().items():
    print(f"{k:25.25s}: {str(v)}")
```