HW8

April 30, 2024

Niall Carbery 22380966

1. What is the best kernel? (Common options include a linear kernel, an RBF kernel)
2. What are the metrics of successful classification?
3. What is the minimum amount of the training data subset should you allocate for a successful classification?

[ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC

import warnings
warnings.filterwarnings("ignore")
```

[ ]:
```
df1 = pd.read_csv("./Homework_Datasets/o2Saturation.csv")
df = pd.read_csv('./Homework_Datasets/heart.csv')
```

[ ]:
```
# Set a variable X to the features of the dataframe and y to the target column.
X = df.drop('output', axis=1)
y = df['output']
```

[ ]:
```
# Split the data into training and testing sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,␣
 ↪random_state=42)
```

[ ]:
```
# Create a StandardScaler object
scaler = StandardScaler()

# Fit the scaler to the training data and transform it
X_train = scaler.fit_transform(X_train)

# Transform the test data using the scaler fitted on the training data
X_test = scaler.transform(X_test)
```

[ ]:
```
# Create and fit a SVC model
svc = SVC()
```

```
svc.fit(X_train, y_train.values.ravel())

# Make predictions on the test data
svc_pred = svc.predict(X_test)

# Evaluate the model
print(confusion_matrix(y_test, svc_pred))
print(classification_report(y_test, svc_pred))
```

```
[[33  8]
 [ 8 42]]
              precision    recall  f1-score   support

           0       0.80      0.80      0.80        41
           1       0.84      0.84      0.84        50

    accuracy                           0.82        91
   macro avg       0.82      0.82      0.82        91
weighted avg       0.82      0.82      0.82        91
```

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

# Define the grid of parameters to search
params_grid = {
    'C': [0.05, 0.1, 0.5, 1, 10],
    'kernel': ['poly','linear', 'rbf', 'sigmoid']
}

# Create an SVC model
model = SVC()

# Define a GridSearchCV to search the best parameters
grid_search = GridSearchCV(estimator = model,
                           param_grid = params_grid,
                           scoring='f1',
                           cv = 5, verbose = 1)

# Search the best parameters with training data
grid_search.fit(X_train, y_train.values.ravel())

# Get the best parameters found by GridSearchCV
best_params = grid_search.best_params_
best_params
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

[ ]:
```
{'C': 0.1, 'kernel': 'linear'}
```

Testing over our datatsets we find that the linear SVC provides us with the best results, this would suggest that the data is linearly seperable.

[ ]:
```
# Retraining an SVC model with the best parameters
best_svc = SVC(C=0.1, kernel='linear')
best_svc.fit(X_train, y_train.values.ravel())

# Make predictions on the test data
svc_pred = best_svc.predict(X_test)

# Evaluate the model
print(confusion_matrix(y_test, svc_pred))
print(classification_report(y_test, svc_pred))
```

```
[[32  9]
 [ 8 42]]
              precision    recall  f1-score   support

           0       0.80      0.78      0.79        41
           1       0.82      0.84      0.83        50

    accuracy                           0.81        91
   macro avg       0.81      0.81      0.81        91
weighted avg       0.81      0.81      0.81        91
```

The metrics of a succesful classification are accruacy, precision, recall, F1-score and support. - Accuracy measures the proportion of correctly classified instances out of the total predictions.

- Precision focuses on the true positive predictions among all positive predictions. It's useful when false positives are costly.
- Recall measures the proportion of actual positive instances correctly predicted by the model. It's essential when false negatives are costly.
- The F1-score balances precision and recall. It's the harmonic mean of precision and recall
- Support refers to the number of instances (data points) in each class. It provides insights into the distribution of data across different classes. Support for the positive class represents the number of positive instances. Support for the negative class represents the number of negative instances.

The minimum amount of training data required for successful classification depends on several factors, including the complexity of the problem, the quality of the data, and the chosen machine learning algorithm. - A common rule of thumb is to allocate at least 70% of your data for training and the remaining 30% for testing (validation or holdout set). This split helps ensure that the model learns from a substantial portion of the data while having unseen examples for evaluation. - If you have a small dataset, use cross-validation. Techniques like k-fold cross-validation (where the data is split into k subsets) allow you to train and evaluate the model on different subsets iteratively. In such cases, the training data subset can be even smaller (e.g., 50% or less). - For complex models (e.g., deep neural networks), more data is generally beneficial. Deep learning models often require large amounts of data to learn intricate features.

- Consider the domain-specific requirements. Some domains (such as medical diagnosis) may demand larger datasets due to the rarity of certain conditions.
- Machine learning is an iterative process. Its standard to start with a reasonable subset, evaluate the model, and gradually increase the data size.