image-classification

May 21, 2024

0.1 Image Classification (Numbers)

[ ]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import StandardScaler
```

[ ]:

```
# Fetch MNIST data (might take some time)
mnist = fetch_openml('mnist_784')

X = mnist.data.astype('float32')
y = mnist.target.astype('int64')

# Standardize the data
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

[ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split, GridSearchCV,␣
  ↪cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report,␣
  ↪accuracy_score
```

[ ]:

```
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.2)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5)
```

[ ]:

```
knn_params = {'n_neighbors': [3, 5, 7, 9]}
knn = GridSearchCV(KNeighborsClassifier(), knn_params, cv=5)
knn.fit(X_train, y_train)
print(f'Best K-NN Params: {knn.best_params_}')
```

```
knn_pred = knn.predict(X_val)
print(confusion_matrix(y_val, knn_pred))
print(classification_report(y_val, knn_pred))
print(f'Validation Accuracy: {accuracy_score(y_val, knn_pred)}')
```

[ ]:

```
log_reg_params = {'C': [0.01, 0.1, 1, 10]}
log_reg = GridSearchCV(LogisticRegression(max_iter=1000), log_reg_params, cv=5)
log_reg.fit(X_train, y_train)
print(f'Best Logistic Regression Params: {log_reg.best_params_}')

log_reg_pred = log_reg.predict(X_val)
print(confusion_matrix(y_val, log_reg_pred))
print(classification_report(y_val, log_reg_pred))
print(f'Validation Accuracy: {accuracy_score(y_val, log_reg_pred)}')
```

[ ]:

```
tree_params = {'max_depth': [10, 20, 30, 40, None]}
tree = GridSearchCV(DecisionTreeClassifier(), tree_params, cv=5)
tree.fit(X_train, y_train)
print(f'Best Decision Tree Params: {tree.best_params_}')

tree_pred = tree.predict(X_val)
print(confusion_matrix(y_val, tree_pred))
print(classification_report(y_val, tree_pred))
print(f'Validation Accuracy: {accuracy_score(y_val, tree_pred)}')
```

[ ]:

```
svm_params = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
svm = GridSearchCV(SVC(), svm_params, cv=5)
svm.fit(X_train, y_train)
print(f'Best SVM Params: {svm.best_params_}')

svm_pred = svm.predict(X_val)
print(confusion_matrix(y_val, svm_pred))
print(classification_report(y_val, svm_pred))
print(f'Validation Accuracy: {accuracy_score(y_val, svm_pred)}')
```

[ ]:

```
knn_test_pred = knn.predict(X_test)
print('K-NN Test Accuracy:', accuracy_score(y_test, knn_test_pred))

log_reg_test_pred = log_reg.predict(X_test)
print('Logistic Regression Test Accuracy:', accuracy_score(y_test,␣
 ↪log_reg_test_pred))

tree_test_pred = tree.predict(X_test)
print('Decision Tree Test Accuracy:', accuracy_score(y_test, tree_test_pred))

svm_test_pred = svm.predict(X_test)
print('SVM Test Accuracy:', accuracy_score(y_test, svm_test_pred))
```

[ ]:

```
print(f"Best K-NN Params: {knn.best_params_}, Test Accuracy:␣
 ↪{accuracy_score(y_test, knn_test_pred)}")
print(f"Best Logistic Regression Params: {log_reg.best_params_}, Test Accuracy:␣
 ↪{accuracy_score(y_test, log_reg_test_pred)}")
print(f"Best Decision Tree Params: {tree.best_params_}, Test Accuracy:␣
 ↪{accuracy_score(y_test, tree_test_pred)}")
print(f"Best SVM Params: {svm.best_params_}, Test Accuracy:␣
 ↪{accuracy_score(y_test, svm_test_pred)}")
```