[Add your name here] assignment #3 Classification Model

September 21, 2023

[ ]:
```
# import libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

# nothing will go to output
```

[ ]:
```
# load the dataset
iris = load_iris()

# make sure everything loaded ok
print(iris.target_names)
```

[ ]:
```
# split the data
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

#
# Print the sizes (number of samples) of the training and testing
# sets to verify that the split proportions are as expected
#

print(f"Training set size: {len(X_train)} samples")
print(f"Testing set size: {len(X_test)} samples")
```

[ ]:
```
# Create the K-Nearest Neighbors classifier

k = 3  # Number of neighbors (you can choose another value)

knn_classifier = KNeighborsClassifier(n_neighbors=k)
knn_classifier.fit(X_train, y_train)

# Print the KNN classifier object itself to verify that it was instantiated␣
 ↪correctly
```

```
print(knn_classifier)
```

[ ]:
```
# make predictions
y_pred = knn_classifier.predict(X_test)

# Print the predicted labels (y_pred) to see the model's predictions for the␣
 ↪testing data.
print(y_pred)
```

[ ]:
```
# Use the classification_report function to print more detailed classification␣
 ↪metrics


from sklearn.metrics import classification_report

class_report = classification_report(y_test, y_pred, target_names=iris.
 ↪target_names)
print("Classification Report:\n", class_report)
```

[ ]:
```
# evaluate the model
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

[ ]:
```
# Example of tuning the number of neighbors
k_values = [1, 3, 5, 7]

for k in k_values:
    knn_classifier = KNeighborsClassifier(n_neighbors=k)
    knn_classifier.fit(X_train, y_train)
    y_pred = knn_classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"k = {k}: Accuracy = {accuracy * 100:.2f}%")
```

[ ]:
```
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

from sklearn.model_selection import KFold
import numpy as np

# Create a dictionary of classifiers
classifiers = {
    'KNeighborsClassifier': KNeighborsClassifier(n_neighbors=3),
    'SVC': SVC(kernel='linear'),
    'GaussianNB': GaussianNB()
}
```

```
# Initialize KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Iterate through classifiers
for name, classifier in classifiers.items():
    accuracies = []  # Store accuracies for each fold

    # Iterate through each fold
    for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        # Train the classifier
        classifier.fit(X_train, y_train)

        # Make predictions
        predictions = classifier.predict(X_test)

        # Calculate accuracy for this fold
        accuracy = accuracy_score(y_test, predictions)
        accuracies.append(accuracy)

    # Calculate mean and standard deviation of accuracies
    mean_accuracy = np.mean(accuracies)
    std_accuracy = np.std(accuracies)

    # Print results for the classifier
    print(f"{name} Mean Accuracy: {mean_accuracy:.4f}")
    print(f"{name} Standard Deviation: {std_accuracy:.4f}")
```

[ ]:
```
# visualize confusion matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Define class labels
class_labels = iris.target_names

# Create a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted Labels')
```

```
plt.ylabel('Actual Labels')
plt.title('Confusion Matrix')
plt.show()
```

[ ]: