```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

```
# Provided dataset
data = np.array([
    [2, 5, 'speaker'],
    [2, 6, 'speaker'],
    [7, 6, 'leader'],
    [7, 2.5, 'intel'],
    [8, 6, 'leader'],
    [4, 7, 'speaker'],
    [5, 3, 'intel'],
    [3, 5.5, 'speaker'],
    [8, 3, 'intel'],
    [6, 5.5, 'leader'],
    [6, 4, 'intel'],
    [6, 7, 'leader'],
    [6, 2, 'intel'],
    [9, 7, 'leader']
])
```

```
data
```

```
array([['2', '5', 'speaker'],
       ['2', '6', 'speaker'],
       ['7', '6', 'leader'],
       ['7', '2.5', 'intel'],
       ['8', '6', 'leader'],
       ['4', '7', 'speaker'],
       ['5', '3', 'intel'],
       ['3', '5.5', 'speaker'],
       ['8', '3', 'intel'],
       ['6', '5.5', 'leader'],
       ['6', '4', 'intel'],
       ['6', '7', 'leader'],
       ['6', '2', 'intel'],
       ['9', '7', 'leader']], dtype='<U32')
```

```
# Splitting features and labels
X = data[:, :-1].astype(float)
y = data[:, -1]
```

```
# Split the dataset into training and testing sets (70% training, 30% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Create a K-NN classifier with a specified number of neighbors (k)
k = 3
knn_classifier = KNeighborsClassifier(n_neighbors=k)
```

```
# Train the classifier on the training data
knn_classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
# Predict the labels for the test data
y_pred = knn_classifier.predict(X_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction functions (e.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 1.00
```

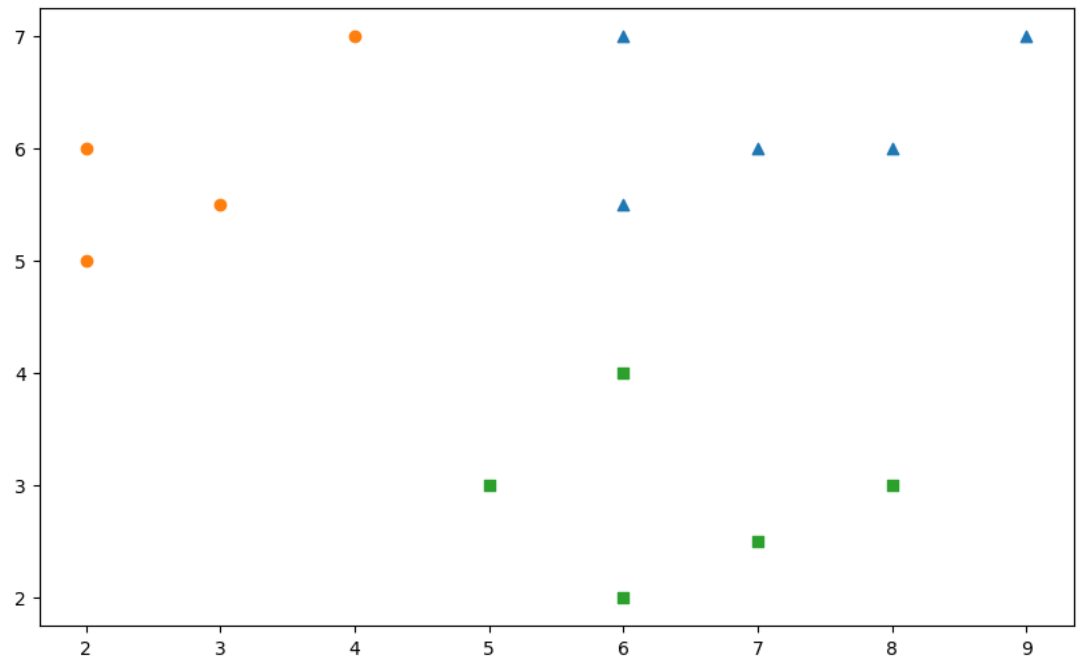
```
# Generate and display the classification report
classification_rep = classification_report(y_test, y_pred)
print("Classification Report:")
print(classification_rep)
```

```
Classification Report:
              precision    recall  f1-score   support

       intel       1.00      1.00      1.00         1
      leader       1.00      1.00      1.00         2
     speaker       1.00      1.00      1.00         2

    accuracy                           1.00         5
   macro avg       1.00      1.00      1.00         5
weighted avg       1.00      1.00      1.00         5
```

```
# Visualize the data points on a graph
plt.figure(figsize=(10, 6))
markers = {'speaker': 'o', 'intel': 's', 'leader': '^'}
for label in set(y):
    indices = np.where(y == label)
    plt.scatter(X[indices, 0], X[indices, 1], label=label, marker=markers[label])
```




```
# New data point for prediction
new_data_point = np.array([[5, 4.5]])
```

```
# Predict the label for the new data point
predicted_label = knn_classifier.predict(new_data_point)
```

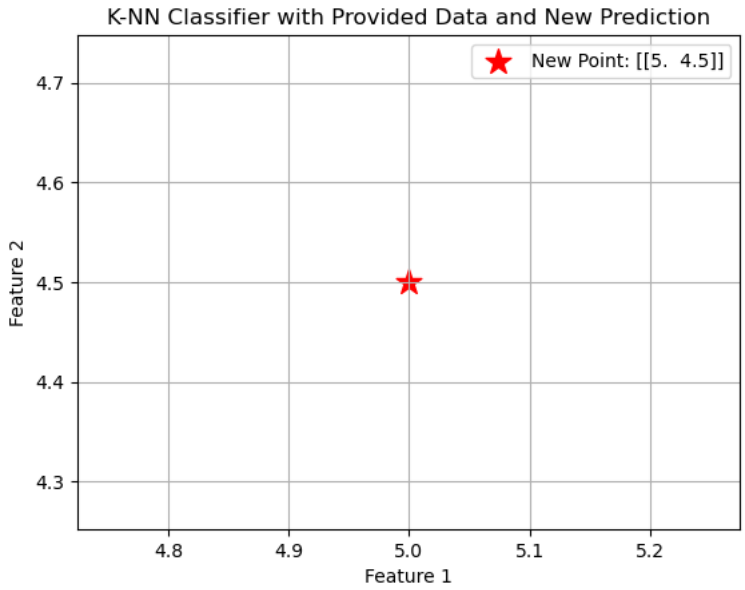
```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction functions (e.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

```
# Set plot labels and legend
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')

# Add a custom legend entry for the new data point
plt.scatter(new_data_point[:, 0], new_data_point[:, 1], marker='*', s=200, c='red', label=f'New Point: {new_data_point}')

plt.legend()
plt.title('K-NN Classifier with Provided Data and New Prediction')
plt.grid(True)
plt.show()

# Display the predicted label for the new data point
print(f"New Data Point: {new_data_point} -> Predicted Label: {predicted_label[0]}")
```




```
New Data Point: [[5.  4.5]] -> Predicted Label: intel
```