

Pattern Recognition HW5

Train verilerininin bütün featureları için mutual information hesapladım.Sonuçları azalan sıralamasında sıraladım en yüksek 2 feature seçtim.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import mutual_info_classif
from sklearn.neighbors import KNeighborsClassifier
import numpy as np

# Training data
d1_train = ["Free", "free", "free", "buy", "discount", "combo", "pleasure"]
d2_train = ["Free", "free", "free", "discount", "pleasure", "smile", "smile", "smile"]
d3_train = ["cat", "mouse"]
d4_train = ["cat", "cat", "dog", "dog", "dog", "dog"]
d5_train = ["mouse"]

# Class labels
d1_class = ["S", "S", "S", "S", "S", "S", "S"]
d2_class = ["S", "S", "S", "S", "S", "S", "S"]
d3_class = ["N", "N"]
d4_class = ["N", "N", "N", "N", "N", "N"]
d5_class = ["N"]

# Test data
d6_test = ["dog", "cat", "mouse", "cat"]
d7_test = ["free", "free", "smile"]

# Combine all training data
all_train = d1_train + d2_train + d3_train + d4_train + d5_train
all_classes = d1_class + d2_class + d3_class + d4_class + d5_class

# Combine training and test data
all_data = all_train + d6_test + d7_test

# Convert word features to TF-IDF scores
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(all_data)

# Compute Mutual Information
mi_scores = mutual_info_classif(X[:len(all_train)], all_classes)

# Create a dictionary of word-feature pairs
word_features = {word: mi for word, mi in zip(vectorizer.get_feature_names_out(), mi_scores)}

# Sort the word features by MI score in descending order
sorted_features = sorted(word_features.items(), key=lambda x: x[1], reverse=True)
```

Seçilen top 2 featureların tf*idf değerlerini hesapladım.

```
[19]

Feature: dog, MI Score: 0.1929506176423086
Feature: free, MI Score: 0.1417028527380234

# STEP 2

# Print TF-IDF scores
for feature, score in top_features:
    print("TF-IDF scores:")
    for i, doc in enumerate(all_train):
        tfidf_score = X[i, vectorizer.vocabulary_[feature]]
        print(f"Document {i + 1}: {tfidf_score}")

TF-IDF scores:
Document 1: 0.0
Document 2: 0.0
Document 3: 0.0
Document 4: 0.0
Document 5: 0.0
Document 6: 0.0
Document 7: 0.0
Document 8: 0.0
Document 9: 0.0
Document 10: 0.0
Document 11: 0.0
Document 12: 0.0
Document 13: 0.0
Document 14: 0.0
Document 15: 0.0
Document 16: 0.0
Document 17: 0.0
Document 18: 0.0
Document 19: 0.0
Document 20: 1.0
Document 21: 1.0
Document 22: 1.0
Document 23: 1.0
Document 24: 0.0
TF-IDF scores:
Document 1: 1.0
Document 2: 1.0
Document 3: 1.0
Document 4: 0.0
```

Test verilerinin $tf \cdot idf$ değerlerini hesapladım.

The screenshot shows a Google Colab notebook with the following code and output:

```
# STEP 3
# Create a matrix to store the TF-IDF values of the selected features for each document
tfidf_matrix = np.zeros((5, 2))

# Fill the matrix with the TF-IDF values
for i, doc in enumerate([d1_train, d2_train, d3_train, d4_train, d5_train]):
    for j, (feature, _) in enumerate(top_features):
        tfidf_matrix[i, j] = X[i, vectorizer.vocabulary_[feature]]

# Print the TF-IDF matrix
print("TF-IDF matrix:")
print(tfidf_matrix)
```

Output:

```
TF-IDF Matrix:
[[0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 0.]
 [0. 0.]]
```

```
[24] # STEP 4

# Calculate the TF-IDF values for d6_test
tfidf_vector_d6 = np.zeros((1, 2))
for j, (feature, _) in enumerate(top_features):
    tfidf_vector_d6[0, j] = X[len(all_train) + 0, vectorizer.vocabulary_[feature]]

# Print the TF-IDF vector for d6_test
print("TF-IDF vector for d6_test:")
print(tfidf_vector_d6)
```

Output:

```
TF-IDF vector for d6_test:
[[1. 0.]]
```

```
[23] # STEP 5

# Calculate the TF-IDF values for d7_test
tfidf_vector_d7 = np.zeros((1, 2))
for j, (feature, _) in enumerate(top_features):
    tfidf_vector_d7[0, j] = X[len(all_train) + 1, vectorizer.vocabulary_[feature]]

# Print the TF-IDF vector for d7_test
print("TF-IDF vector for d7_test:")
print(tfidf_vector_d7)
```

Output:

```
TF-IDF vector for d7_test:
[[1. 0.]]
```

The notebook is titled "Mithat_Ucar_152120161061.ipynb" and shows a progress bar at the bottom indicating it is completed at 2:55 PM.

Test verilerinin ait olduğu classları bulmak için knn algoritmasını kullandım.

The screenshot shows a Google Colab notebook with the following code and output:

```
[41] # STEP 6

# Train the KNN classifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(tfidf_matrix, ["S", "S", "N", "N", "N"])

# Convert the TF-IDF values for d6_test
tfidf_vector_d6 = np.zeros((len(d6_test), 2))
for i, doc in enumerate(d6_test):
    for j, (feature, _) in enumerate(top_features):
        tfidf_vector_d6[i, j] = vectorizer.transform([doc]).toarray()[i, vectorizer.vocabulary_[feature]]

# Predict the class label of d6
prediction_d6 = knn.predict(tfidf_vector_d6)

# Print the predicted class label of d6
print("Predicted class label of d6:", prediction_d6[0])
```

Output:

```
Predicted class label of d6: N
```

```
# STEP 7

# Train the KNN classifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(tfidf_matrix, ["S", "S", "N", "N", "N"])

# Convert the TF-IDF values for d7_test
tfidf_vector_d7 = np.zeros((len(d7_test), 2))
for i, doc in enumerate(d7_test):
    for j, (feature, _) in enumerate(top_features):
        tfidf_vector_d7[i, j] = vectorizer.transform([doc]).toarray()[i, vectorizer.vocabulary_[feature]]

# Predict the class label of d7
prediction_d7 = knn.predict(tfidf_vector_d7)

# Print the predicted class label of d7
print("Predicted class label of d7:", prediction_d7[0])
```

Output:

```
Predicted class label of d7: S
```

The notebook is titled "Mithat_Ucar_152120161061.ipynb" and shows a progress bar at the bottom indicating it is completed at 2:55 PM.