**Categorical and Numeric Relations Dataset**

Ref: <https://phys-techsciences.datastations.nl/dataset.xhtml?persistentId=doi:10.17026/dans-zxp-t7tf>

Collaboratively constructed knowledge bases play an important role in information systems, but are essentially always incomplete. Thus, a large number of models has been developed for Knowledge Base Completion, the task of predicting new attributes of entities given partial descriptions of these entities. Virtually all of these models either concentrate on numeric attributes (what is Italy’s GDP?) or they concentrate on categorical attributes (Tim Cook is the chairman of Apple). This dataset was created as a part of a research experiment to develop a model for the joint prediction of numeric and categorical attributes based on embeddings learned from textual occurrences of the entities in question. This dataset consists of numeric and categorical relation tuples spanning from 7 different domains such as 'animal', 'country', 'people', etc. The tuples presented in this dataset have been used to train and test a neural network framework to perform the above mentioned task. All data presented in this dataset has been scraped from FreeBase. **Categorical and Numeric Files**: These files are divided into 'test', 'train', and 'validation' sets for different domains like 'animal', 'book', etc. Each set contains both 'categorical' and 'numeric' data files:

* **Test Files**: Used to test the model's performance after training.
* **Train Files**: Used to train our model.
* **Validation Files**: Used to validate the model during the training process.

Dataset in txt, converted into csv files.

The reason for categorizing the Numeric Value column ("0", "1") in our dataset is that classification algorithms need to work with categorical values rather than continuous values. Classification algorithms make predictions between a certain number of defined classes, so continuous values need to be assigned to these classes.

Since our Numeric Value column contains continuous numeric values, it cannot be used directly with a classification algorithm. I have converted this column into a form that classification algorithms can use by dividing it into specific ranges and marking each range as a separate category.

**Code:**

1. from sklearn.preprocessing import OneHotEncoder, StandardScaler

2. from sklearn.neighbors import KNeighborsClassifier

3. from sklearn.neural\_network import MLPClassifier

4. from sklearn.naive\_bayes import MultinomialNB

5. from sklearn.metrics import classification\_report

6. from scipy import sparse

7. import pandas as pd

8. import numpy as np

9.

10. def prepare\_data(file\_path, encoder, scaler):

11.     df = pd.read\_csv(file\_path)

12.     df.fillna(0, inplace=True)

13.     df[categorical\_columns] = df[categorical\_columns].astype(str)

14.

15.     X\_categorical = encoder.transform(df[categorical\_columns])

16.     X\_numeric = scaler.transform(df[['Numeric Value']])

17.

18.     # Negatif değerler ayarlanmalıydı çünkü NB modeli negatif değerleri kabul etmiyor

19.     X\_numeric\_adjusted = X\_numeric + abs(X\_numeric.min()) + 1

20.

21.     X = sparse.hstack((X\_categorical, sparse.csr\_matrix(X\_numeric\_adjusted)))

22.     y = df['Numeric Value'].astype(int)

23.

24.     return X, y

25.

26. # OneHotEncoder ve StandardScaler nesneleri

27. encoder = OneHotEncoder(sparse\_output=True, handle\_unknown='ignore')

28. scaler = StandardScaler()

29.

30. # Kategorik sütunlar

31. categorical\_columns = ['Relation', 'Target', 'Relatum']

32.

33. # Eğitim seti

34. df\_train = pd.read\_csv('./processed\_train\_combined.csv')

35. df\_train.fillna(0, inplace=True)

36. df\_train[categorical\_columns] = df\_train[categorical\_columns].astype(str)

37. X\_train\_categorical = encoder.fit\_transform(df\_train[categorical\_columns])

38. X\_train\_numeric = scaler.fit\_transform(df\_train[['Numeric Value']])

39.

40. # Negatif değerleri ayarlayın

41. X\_train\_numeric += abs(X\_train\_numeric.min()) + 1

42.

43. X\_train = sparse.hstack((X\_train\_categorical, sparse.csr\_matrix(X\_train\_numeric)))

44. y\_train = df\_train['Numeric Value'].astype(int)

45.

46. # Test ve doğrulama setleri

47. X\_test, y\_test = prepare\_data('./processed\_test\_combined.csv', encoder, scaler)

48. X\_val, y\_val = prepare\_data('./processed\_validation\_combined.csv', encoder, scaler)

49.

50. # Model değerlendirme fonksiyonu

51. def evaluate\_model(model, X\_train, y\_train, X\_test, y\_test, model\_name):

52.     model.fit(X\_train, y\_train)

53.     y\_pred = model.predict(X\_test)

54.     print(f"{model\_name} Classification Report:")

55.     print(classification\_report(y\_test, y\_pred))

56.

57. # Model değerlendirme

58. for k in [3, 7, 11]:

59.     evaluate\_model(KNeighborsClassifier(n\_neighbors=k), X\_train, y\_train, X\_val, y\_val, f"KNN with K={k}")

60.

61. for layers in [(32,), (32, 32), (32, 32, 32)]:

62.     evaluate\_model(MLPClassifier(hidden\_layer\_sizes=layers, max\_iter=1000), X\_train, y\_train, X\_val, y\_val, f"MLP with layers {layers}")

63.

64. evaluate\_model(MultinomialNB(), X\_train, y\_train, X\_val, y\_val, "Naive Bayes")

65.

**Report:**

KNN with K=3 Classification Report:

precision recall f1-score support

0 1.00 0.97 0.98 45431

1 0.91 0.99 0.95 15926

accuracy 0.97 61357

macro avg 0.95 0.98 0.97 61357

weighted avg 0.97 0.97 0.97 61357

KNN with K=7 Classification Report:

precision recall f1-score support

0 0.99 0.97 0.98 45431

1 0.91 0.97 0.94 15926

accuracy 0.97 61357

macro avg 0.95 0.97 0.96 61357

weighted avg 0.97 0.97 0.97 61357

KNN with K=11 Classification Report:

precision recall f1-score support

0 0.99 0.96 0.98 45431

1 0.90 0.96 0.93 15926

accuracy 0.96 61357

macro avg 0.94 0.96 0.95 61357

weighted avg 0.96 0.96 0.96 61357

MLP with layers (32,) Classification Report:

precision recall f1-score support

0 0.97 0.98 0.97 45431

1 0.94 0.91 0.93 15926

accuracy 0.97 61357

macro avg 0.95 0.97 0.96 61357

weighted avg 0.97 0.97 0.97 61357

MLP with layers (32, 32, 32) Classification Report:

precision recall f1-score support

0 0.99 0.96 0.98 45431

1 0.90 0.97 0.94 15926

accuracy 0.97 61357

macro avg 0.95 0.97 0.96 61357

weighted avg 0.97 0.97 0.97 61357

Naive Bayes Classification Report:

precision recall f1-score support

0 0.77 0.92 0.84 45431

1 0.50 0.23 0.31 15926

accuracy 0.74 61357

macro avg 0.63 0.57 0.58 61357

weighted avg 0.70 0.74 0.70 61357