MAKİNE ÖĞRENİMİ İLE FOTOĞRAFLARDAKİ EL YAZISI HARFLERİ VE RAKAMLARI TANIMA

Ders: Bilgisayar Zeki Sistem Uygulamaları

Hazırlayanlar: Mustafa Sedat Şenli - 460121058 | Yağmur Nisa Erdoğan - 460121020

Hazırlama Süreci: Konumuz python ve kütüphanelerini kullanarak, makine öğrenimi ile fotoğrafları alınmış el yazısı harfleri ve rakamları tanıması, araştırmalarımız ve izlediğimiz kaynaklar doğrultusunda farklı yazılış biçimlerine sahip, harfleri ve rakamları içeren bir datasete(veri kümesine) ihtiyacımız vardı, harfler ile sayıları birlikte içeren Scikit-Learn gibi interaktif kütüphanelerde hazır bir dataset bulamadık. Sorunu Kaggle'dan temin ettiğimiz iki ayrı harf ve rakam datasetini birleştirerek çözdük, makinenin öğrenimi için verileri kütüphane kullanmadan bilgisayarımızdan çektik. Projemizi çalıştırmak için gerekli kodları iki ayrı video kaynağını harmanlayarak deneyerek ve öğrenerek yaptık, aldığımız hataları ise Stack Overflow gibi websiteleri yardımıyla çözdük. Her hücredeki kodların amaçlarını ise her kod bloklarının altında açıkladık, sayfayı aşağı sürükleyerek inceleyebilirsiniz.

Kütüphane Yüklemeleri

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Gerekli Python Kütüphaneler

NumPy: Python programlama dili için büyük, çok boyutlu dizileri ve matrisleri destekleyen, bu diziler üzerinde çalışacak üst düzey matematiksel işlevler ekleyen bir kitaplıktır.

Pandas: Veri işlemesi ve analizi için Python programlama dilinde yazılmış olan bir yazılım kütüphanesidir.

Matplotlip: Python programlama dili ve sayısal matematik uzantısı NumPy için bir çizim kitaplığıdır.

Scikit-Learn: Python programlama dili için ücretsiz bir yazılım makine öğrenimi kitaplığıdır. Logistic Regression (Lojistik Regresyon) sınıflandırma işlemi yapmaya yarayan bir regresyon yöntemidir.Train Test Split, verileri tek satırlık bir satıra bölmek için tek bir çağrıya veri girmek için uygulamayı sarmalar.

OpenCV: Gerçek zamanlı bilgisayar görüsü uygulamalarında kullanılan açık kaynaklı kütüphane.

Dataset Sınıfının Oluşturulması

In [11]:

```
import os

path = "C:/Users/90531/Desktop/Bilgisayar Zeki Sistem Uygulamaları Projesi/dataset/train/"
files = os.listdir(path)[:36]
print(files)

classes={'0':0,'1':1,'2':2,'3':3,'4':4,'5':5,'6':6,'7':7,'8':8,'9':9,'A':'A','B':'B','C':'C','D':'D','E':'E',
        'F':'F','G':'G','H':'H','I':'I','J':'J','K':'K','L':'L','M':'M', 'N':'N','O':'O','P':'P','Q':'Q','R':'R',
        'S':'S','T':'T','U':'U','V':'V','W':'W','Y':'Y','Z':'Z'}
```

```
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

Bu Python kodu ile bir klasörün içindeki dosyaların isimlerini okuduk ve bunları bir liste içinde sakladık. Bu kodla bir sözlük oluşturduk ve bu sözlükte her bir harf veya sayı için bir değer atandı.

İlk satırda,"os" modülünü içe aktardık. Bu modül, Python'da işletim sistemiyle etkileşim kurmak için fonksiyonlar sağlar.

İkinci satırda, "path" adlı bir değişkene datasetin bulunduğu dosya yolunu atadık.

Üçüncü satırda, "os.listdir()" fonksiyonu kullanarak, "path" değişkeninde belirtilen klasörün içindeki harfler ve rakamların bulunduğu klasör isimlerini bir liste olarak aldık.Bu listeyi, sonraki satırda "files" adlı değişkene atadık.

Dördüncü satırda, "classes" adlı bir sınıflandırma sözlüğü oluşturduk. Bu sözlükte, her bir harf veya sayı için bir anahtar ve değer atandı.

In [12]:

```
import cv2

X=[]
Y=[]

for cl in classes:
    pth = path+cl
    for img_name in os.listdir(pth):
        img = cv2.imread(pth+"/"+img_name,0)
        X.append(img)
        Y.append(classes[cl])
print("Dataset başarıyla oluşturuldu!")
```

```
Dataset başarıyla oluşturuldu!
```

Bu Python kodu ile bir klasörün içindeki görüntüleri okuduk ve bu görüntüleri X ve Y adlı iki listeye ekledik. X listesi, görüntülerin pixel değerlerini saklıyor, Y listesi ise görüntülerin hangi sınıfa ait olduğunu gösteriyor.

İlk satırda, "cv2" modülünü içe aktardık. Bu modül, OpenCV adlı bir görüntü işleme kütüphanesini kullanarak görüntülerle çalışmak için fonksiyonlar sağladık.İkinci ve üçüncü satırda, X ve Y adlı iki liste oluşturduk.

Dördüncü satırda, "classes" sözlüğündeki her bir harf veya sayı için bir döngü oluşturduk. Bu döngü, sözlükteki her bir harf veya sayıyı bir değişkene atar ve bu değişken "cl" olarak adlandırır.

Beşinci satırda, her bir harf veya sayı için bir dosya yolu oluşturduk. Bu dosya yolu, "path" değişkeninde belirtilen yolu ve "cl" değişkeninde belirtilen harf veya sayıyı içerir. Örneğin, "path" değişkeninde "C:/Users/90531/Desktop/Bilgisayar Zeki Sistem Uygulamaları Projesi/dataset/train/" olarak belirtilmiş ve "cl" değişkeni "0" olarak atanmışsa, dosya yolu "C:/Users/90531/Desktop/Bilgisayar Zeki Sistem Uygulamaları Projesi/dataset/train/0" olarak oluşturulur.

*Altıncı satırda, "pth" değişkeninde belirtilen klasörün içindeki dosyaların isimleri bir liste olarak aldık ve bu liste içinde bir döngü oluşturduk. Bu döngüde, her bir dosya için **aşağıdaki işlemleri** gerçekleştiriyoruz:*

• Görüntü dosyası okundu ve "img" değişkenine atandı. "cv2.imread()" fonksiyonu, görüntü dosyasını okuyor ve görüntüyü numpy dizisi olarak döndürüyor. Bu fonksiyonun ikinci argümanını "0" olarak belirledik, bu da görüntünün siyah beyaz olarak okunmasını sağladı.

• Görüntüyü, "X" listesine ekledik.

• Görüntünün hangi sınıfa ait olduğu bilgisini, "Y" listesine ekledik. Bu bilgiyi, "classes" sözlüğünden aldık ve "cl" değişkeninde belirtilen harf veya sayıya göre sözlükten değer aldık. Örneğin, "cl" değişkeni "0" olarak atanmışsa, sözlükten "0" değeri aldık ve "Y" listesine ekledik.

In [13]:

```
pd.Series(Y).value_counts()
```

Out[13]:

```
6    1444
1    1443
5    1388
0    1301
3    1301
4    1277
2    1267
8    1198
9    1195
A    1105
I    1078
S    1016
G    1011
B     988
Z     965
N     931
R     910
P     838
F     836
Q     805
T     784
L     728
H     727
D     690
K     672
O     667
M     651
V     638
U     610
Y     589
W     581
E     548
J     526
C     470
7     285
dtype: int64
```

Bu kod satırıyla, "Y" listesi içindeki elemanların sayısını hesapladık ve bu elemanların sayılarını bir sözlük şeklinde döndürdük. "Y" listesi içindeki elemanların sayısını gösterdik. Örnek verecek olursak, "6" elemanı 1444 tane olduğu, "1" elemanı 1443 tane olduğu vb. görülmekte. Bu çıktı, veri setinin dengeli olup olmadığı hakkında bize bilgi veriyor. Eğer "Y" listesi içinde bazı elemanların çok daha fazla sayıda olduğunu görürsek, veri setinin dengesiz olduğunu anlarız ve bu durum modelin doğruluk performansını etkileyeceği için düzenlememiz gerekir.

In [14]:

```
X[0].shape
```

Out[14]:

```
(32, 32)
```

Bu kod satırı, "X" listemizdeki ilk elemanının boyutunu gösteriyor. "X" listemiz, görüntülerin numpy dizisi olarak sakladığımız bir listedir.

In [15]:

```
img_src = cv2.imread('C:/Users/90531/Desktop/Bilgisayar Zeki Sistem Uygulamaları Projesi/dataset/train/0/1.jpg',0)
print (img_src)
```

```
None
```

Bu kod satırıyla, "C:/Users/90531/Desktop/Bilgisayar Zeki Sistem Uygulamaları Projesi/dataset/train/0/1.jpg" dizininde bulunan bir görüntü dosyasını "cv2.imread()" fonksiyonuyla numpy dizisine döndürdük, ikinci argümanı "0" olarak belirttik ki bu da görüntünün siyah-beyaz olarak okunmasını sağladı. Görüntüyü "img_src" değişkenine atadık.Sonra, "print (img_src)" kod satırı ile görüntünün bilgisi ekrana yazdırdık çalıştığını görmüş olduk. Bu bilgi, görüntünün satır sayısını, sütun sayısını ve kanal sayısını içeren bilgileri içeriyor.

In [16]:

```
print(type(X))
X = np.array(X)
Y = np.array(Y)
print(type(X))
```

```
<class 'list'>
<class 'numpy.ndarray'>
```

İlk olarak, "print(type(X))" kod satırı ile "X" listesinin tipi ekrana yazdırdık. Bu sayede, "X" listesinin ne tür bir veri yapısında olduğunu gösterdik.

Sonra, "X = np.array(X)" kod satırı ile "X" listesini numpy dizisi olarak dönüştürdük. Numpy, Python'da sayısal verileri işlemek için kullanılan bir kütüphanedir.

Aynı şekilde, "Y = np.array(Y)" kod satırı ile de "Y" listesini numpy dizisi olarak dönüştürdük.

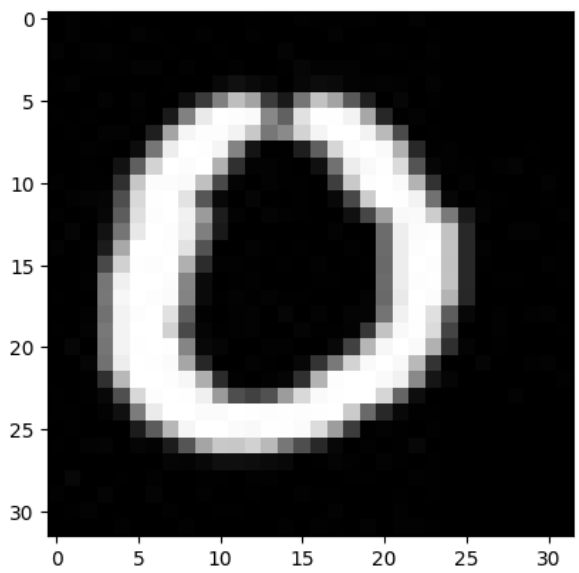
Son olarak, "print(type(X))" kod satırı ile "X" listesinin tipini tekrar ekrana yazdırdık. Bu sayede, "X" listesinin numpy dizisi olarak dönüştürülmüş olup olmadığı kontrol ettik.

Veri Analizi

In [17]:

```
plt.imshow(X[1000],cmap="gray")
print(Y[1000])
```

```
0
```



Bu kod satırıyla "X" listesinin 1000. elemanı olan bir görüntüyü ekranda gösterdik ve bu görüntünün etiketini de ekranda yazdırdık.

İlk olarak, "plt.imshow(X[1000],cmap="gray")" kod satırı ile "X" listesinin 1000. elemanı olan görüntüyü ekranda gösterdik. "plt.imshow()" fonksiyonu, bir görüntüyü ekranda göstermek için kullanılır ve "cmap="gray"" argümanı ile görüntü siyah-beyaz olarak gösterilir. Eğer "cmap" argümanını belirtilmeseydik, görüntü renkli olarak gösterilecekti.

Son olarak, "print(Y[1000])" kod satırı ile görüntünün etiketini ekrana yazdırdık. "Y" listesi, görüntülerin etiketlerini içeren bir liste olduğu için "Y[1000]" ile "X" listesinin 1000. elemanına karşılık gelen etiketi almış olduk.

In [18]:

```
X.shape
```

Out[18]:

```
(31463, 32, 32)
```

"X.shape" kod satırı, "X" değişkeninin boyut bilgisini verir. "X" değişkeni, bir numpy olduğu için ve numpy dizilerinin boyut bilgisi, dizinin satır sayısı ve sütun sayısı bilgilerini öğrenmek göstermek için ekrana yazdırdık.

Verileri Hazırlama

In [19]:

```
X_new = X.reshape(len(X),-1)
print(X_new.shape)
print(Y.shape)
```

```
(31463, 1024)
(31463,)
```

Bu kod satırı ile "X" dizisini yeniden şekillendirdik ve "Y" dizisinin boyut bilgisini verdik.

İlk olarak, "X_new = X.reshape(len(X),-1)" kod satırı ile "X" dizisi yeniden şekillendirdik. "X.reshape()" fonksiyonu, bir numpy dizisinin şeklini değiştirir ve dizinin yeni şeklini belirtir. Bu fonksiyonun ilk argümanıyla dizinin satır sayısını belirttik ve ikinci argümanıyla ise dizinin sütun sayısını belirttik. "len(X)" ile "X" dizisinin satır sayısını bulduk ve "-1" ile sütun sayısını otomatik olarak hesapladık. Bu sayede, "X" dizisi tek bir satıra dönüştürüldü ve tüm elemanları tek bir sütunda toplandı.

Sonra, "print(X_new.shape)" kod satırı ile "X_new" dizisinin boyut bilgisini verdik. Bu bilgi, "X_new" dizisinin satır sayısı ve sütun sayısı bilgilerini içeriyor.

Son olarak, "print(Y.shape)" kod satırı ile de "Y" dizisinin boyut bilgisini verdik. Bu bilgi, "Y" dizisinin satır sayısı bilgisini içeriyor.

In [20]:

```
32*32
```

Out[20]:

```
1024
```

In [21]:

```
print(X.shape)
print(X.ndim)
print(X_new.ndim)
```

```
(31463, 32, 32)
3
2
```

Bu kod satırı, "X" ve "X_new" dizilerinin boyut bilgisi ve boyut sayısını verir.

İlk olarak, "print(X.shape)" kod satırı ile "X" dizisinin boyut bilgisi verdik. Bu bilgi, "X" dizisinin satır sayısı, sütun sayısı ve kanal sayısı bilgilerini içeriyor.

Sonra, "print(X.ndim)" kod satırı ile "X" dizisinin boyut sayısını verdik. "ndim" özelliği, bir numpy dizisinin boyut sayısını verir. Örneğin, "X" dizisinin boyut sayısı 3 ise, bu dizinin 3 boyutlu olduğu anlamına geliyor.

Son olarak, "print(X_new.ndim)" kod satırı ile "X_new" dizisinin boyut sayısını verdik. "X_new" dizisi, "X" dizisinin tek satıra dönüştürülmüş hali olduğundan, "X_new" dizisinin boyut sayısı 2 oluyor.

Verilerin Bölünmesi

In [22]:

```
xtrain, xtest, ytrain, ytest = train_test_split(X_new, Y,
                                                test_size=.20, random_state=10)
```

Bu kod satırıyla "X_new" ve "Y" dizilerini eğitim ve test kümelerine ayırdık.

"train_test_split()" fonksiyonu, veri setini eğitim ve test kümelerine ayırmak için kullanılır. Bu fonksiyonun ilk argümanı eğitim verilerini, ikinci argümanı ise hedef değişkenlerini (etiketlerini) içeren bir numpy dizisidir. Fonksiyonun üçüncü argümanı olarak "test_size" belirttik ve bu argümanla, test veri kümesinin veri setinin toplamının yüzde kaç olacağını belirttik. Örneğin, "test_size=.20" ifadesi ile test veri kümesinin veri setinin toplamının yüzde 20'si kullanılır. Dördüncü argüman olarak "random_state" bu argüman, eğitim ve test veri kümelerinin rastgele nasıl ayırılacağını belirtiyor.

In [23]:

```
print(xtrain.shape, ytrain.shape)
print(xtest.shape, ytest.shape)
```

```
(25170, 1024) (25170,)
(6293, 1024) (6293,)
```

İlk olarak, "print(xtrain.shape, ytrain.shape)" kod satırı ile eğitim veri kümelerinin boyut bilgisini verdik. "xtrain.shape" ile eğitim veri kümesinin satır sayısı ve sütun sayısı bilgisi verdik, "ytrain.shape" ise hedef değişkenlerinin (etiketlerin) satır sayısı bilgisini verdik.

Sonra, "print(xtest.shape, ytest.shape)" kod satırı ile test veri kümelerinin boyut bilgisini verdik. "xtest.shape" ile test veri kümesinin satır sayısı ve sütun sayısı bilgisini verdik, "ytest.shape" ise hedef değişkenlerin (etiketlerin) satır sayısı bilgisini verdik.

Ölçekleme Özelliği

In [24]:

```
print(xtrain.max())
print(xtest.max())
x_train = xtrain/255
x_test = xtest/255
print(x_train.max())
print(x_test.max())
```

```
255
255
1.0
1.0
```

Bu kod satırı, eğitim ve test veri kümelerinin piksel değerlerini normalize ediyor.

İlk olarak, "print(xtrain.max())" ve "print(xtest.max())" kod satırları ile eğitim ve test veri kümelerinin en yüksek piksel değerlerini verdik. Bu değerler, görüntülerin en parlak piksellerini temsil ediyor.

Sonra, "x_train = xtrain/255" ve "x_test = xtest/255" kod satırları ile eğitim ve test veri kümelerinin piksel değerlerini normalize ettik. Normalizasyon, veri setindeki değerlerin bir aralıkta (örneğin, 0 ile 1 arasında) değiştirilmesidir. Bu sayede, veri setindeki değerler arasındaki farklılığı azaltmayı amaçladık ve modelin eğitimini daha kolay hale getirdik. Bu kod satırlarında, eğitim ve test veri kümelerinin piksel değerleri 255'e bölündü ve bu sayede, piksel değerleri 0 ile 1 arasında normalize edilmiş oldu.

Son olarak, "print(x_train.max())" ve "print(x_test.max())" kod satırları ile normalize edilen eğitim ve test veri kümelerinin en yüksek piksel değerlerini verdik. Bu değerler, normalizasyon işlemi sonrasında eğitim ve test veri kümelerinin en yüksek piksel değerlerini gösteriyor.

Özellik Seçimi: PCA(Temel Bileşenler Analizi)

In [25]:

```
from sklearn.decomposition import PCA
```

"PCA" (Principal Component Analysis), veri setindeki değişkenler arasındaki ilişkileri açıklamaya yönelik bir boyut indirgeme yöntemidir. Bu yöntem, veri setindeki değişkenler arasındaki ilişkileri inceler ve veri setini daha az sayıda değişkene indirger. Bu sayede, veri setinin anlaşılırlığını arttırıyor ve veri setini işleme ve modelleme işlemlerini daha kolay hale getiriyor.

"from sklearn.decomposition import PCA" kod satırı, "sklearn" (Scikit-learn) kütüphanesinden "PCA" sınıfını içe aktardı. Bu sayede, veri setinde "PCA" yöntemini kullanarak boyut indirgeme işlemlerini gerçekleştirebileceğiz.

In [26]:

```
print(x_train.shape, x_test.shape)
pca = PCA(.98)
xtrain = pca.fit_transform(x_train)
xtest = pca.transform(x_test)
print(xtrain.shape, xtest.shape)
print(pca.n_components)
print(pca.n_features_in_)
```

```
(25170, 1024) (6293, 1024)
(25170, 413) (6293, 413)
0.98
1024
```

Bu kod satırı ile eğitim ve test veri kümelerinde "PCA" yöntemini kullanarak boyut indirgeme işlemini gerçekleştirdik.

İlk olarak, "print(x_train.shape, x_test.shape)" kod satırı ile eğitim ve test veri kümelerinin boyut bilgisini verdik. Bu bilgi, eğitim ve test veri kümelerinin satır sayısı ve sütun sayısı bilgisini içeriyor.

"pca = PCA(.98)" kod satırı ile "PCA" sınıfının bir nesnesini oluşturduk. Bu nesnenin parametresi ".98" olarak belirttik. Bu parametre, veri setinin hangi yüzdesini açıklamaya yönelik değişkenler oluşturulacağını belirtiyor. Örneğin, ".98" parametresi kullanılırsa, veri setindeki değişkenlerin yüzde 98'ini açıklamaya yönelik değişkenler oluşturulur.

"xtrain = pca.fit_transform(x_train)" kod satırı ile eğitim veri kümesinde "PCA" yöntemini kullanarak boyut indirgeme işlemi gerçekleştirdik. Bu işlem sonucu, eğitim veri kümesi "pca.n_components" sayıda değişkene indirgenmiş oldu.

Sonra, "xtest = pca.transform(x_test)" kod satırı ile test veri kümesinde "PCA" yöntemini kullanarak boyut indirgeme işlemi gerçekleştirdik. Bu işlem sonucunda test veri kümesi "pca.n_components" sayıda değişkene indirgendi.

Son olarak, "print(xtrain.shape, xtest.shape)" kod satırı ile indirgenmiş eğitim ve test veri kümelerinin boyut bilgisi verildi. Bu bilgi, indirgenmiş eğitim ve test veri kümelerinin satır sayısı ve sütun sayısı bilgisini veriyor. "print(pca.n_components)" kod satırı ile indirgenmiş veri kümelerinde kullanılan değişken sayısını yazdırdık. "print(pca.n_features_in_)" kod satırı ile başlangıç veri kümelerinde bulunan değişken sayısını yazdık.

In [27]:

```
ytest[:1000]
```

Out[27]:

```
array(['K', '8', '3', '7', '1', 'N', '6', '5', 'P', 'V', '1', 'K', 'O',
       '2', 'L', 'K', '4', '5', 'P', 'I', 'L', 'Q', '1', '0', 'P', 'B',
       'L', 'Z', '1', '3', '8', 'R', 'M', 'D', 'T', 'H', '9', 'D', '2',
       'C', '0', 'D', 'A', 'G', 'Y', '4', '2', 'Y', 'P', 'Y', '0', '8',
       'R', '1', 'I', 'G', '1', '1', 'A', 'D', 'N', 'S', 'K', 'G', '6',
       'W', 'L', 'I', 'I', 'C', 'K', 'T', '5', '4', 'A', '6', 'O', 'N',
       'H', 'D', 'V', '4', 'A', 'Z', 'F', 'N', '5', '4', '3', '8', '1',
       '6', 'A', 'E', 'G', 'H', '8', 'O', 'K', 'F', 'A', 'G', 'G', 'N',
       '2', 'E', 'F', '1', 'U', '9', 'U', '4', '0', 'F', '9', '7', 'G',
       'T', 'O', '4', '6', 'C', '3', '6', '6', 'T', '8', 'W', '1', 'J',
       '2', 'G', 'T', 'K', 'S', '4', 'H', 'H', '1', 'F', 'Q', 'Z', 'D',
       'O', '7', 'B', '9', 'W', 'E', '0', 'F', 'A', 'Z', 'A', '3', 'W',
       'K', 'T', '8', 'G', '3', '0', 'S', '3', '8', '6', '3', 'D', 'S',
       'A', 'E', 'T', 'Q', '2', '9', 'Y', '0', '5', 'M', '2', '1', '8',
       'J', 'B', 'N', 'P', '2', 'F', 'G', 'V', 'Z', 'M', 'G', '9', 'Y',
       '3', 'W', 'F', '3', 'L', '0', 'Z', 'H', 'V', 'O', 'C', '5', '1',
       'G', 'H', 'M', 'E', 'N', 'P', '4', 'Q', 'R', '2', 'G', 'V', '5',
       '6', 'H', 'Y', '0', 'O', 'J', '5', 'M', '8', 'T', 'D', 'K', 'N',
       '3', '8', '1', 'Q', 'D', 'D', '4', 'D', 'R', '8', 'E', 'C', '3',
       'P', 'B', '3', 'B', '2', 'L', '4', 'R', 'M', '9', 'G', '6', 'S',
       'T', 'Y', 'I', 'W', '3', '3', 'L', 'O', '5', 'G', '5', '1', '4',
       'N', '5', '0', '4', 'A', 'Q', 'R', 'M', 'K', '9', '1', 'Q', 'L',
       'K', 'I', 'A', '5', 'M', '0', '3', '1', 'A', 'A', 'B', 'T', 'Q',
       'G', 'F', '8', '5', '2', '6', 'M', '1', '1', 'W', 'B', '1', '1',
       'P', 'F', '4', 'B', '5', '7', '8', '6', '1', 'N', 'B', '8', 'R',
       'T', 'A', '4', '0', 'V', 'I', '6', 'Q', 'I', 'Q', 'V', 'G', '5',
       '5', 'H', '9', 'I', '4', 'Q', '1', '4', 'I', '5', '5', 'O', 'U',
       'P', 'P', 'Q', 'Y', '0', 'M', 'T', 'P', 'R', 'G', '6', '1', 'F',
       'K', 'J', 'C', 'L', '6', '8', 'L', 'T', '7', '5', 'A', 'S', '9',
       'N', 'O', 'I', '1', 'P', 'G', 'I', 'F', '4', 'A', 'A', 'V', 'I',
       '3', 'A', '3', 'G', 'E', '5', '9', 'G', 'J', 'J', 'Q', 'C', '3',
       'U', 'K', '2', 'S', 'S', '6', 'L', '0', '4', 'B', 'S', 'B', 'Q',
       'V', 'F', 'U', 'Y', '2', 'Z', '6', 'V', '8', '4', 'U', 'F', '3',
       'Q', 'S', 'D', '4', 'G', '8', '2', '2', 'C', '2', 'N', 'Z', '0',
       'N', 'I', 'G', '8', 'L', '1', '5', 'U', 'A', 'K', 'U', '6', 'V',
       'A', 'P', 'F', '0', '6', 'G', 'L', 'K', 'G', '1', 'B', 'Q', '8',
       'I', 'Q', 'F', 'Q', 'C', '9', 'B', 'G', 'A', 'Y', 'B', 'W', 'Y',
       '4', '9', 'O', 'Z', 'D', 'I', 'V', 'E', '5', 'P', 'D', 'Z', 'F',
       'O', '1', '0', 'J', 'K', '6', '1', 'E', '5', 'Z', 'S', 'D', 'G',
       '5', '0', 'O', '5', '1', 'O', '9', 'T', '6', '0', 'V', 'M', '0',
       '4', '9', '6', '2', '1', '9', 'T', 'P', '8', '7', 'S', '4', '0',
       '6', '2', '6', 'W', 'M', '4', '0', 'L', '7', '4', 'E', 'Q', 'D',
       '6', '4', 'A', '8', '5', 'R', 'V', 'U', 'R', '3', '4', '0', '4',
       '0', '9', '1', '8', 'D', 'R', 'M', '1', 'S', '1', '2', 'Z', '0',
       'F', 'V', 'Y', 'H', 'I', '5', 'H', 'F', '1', 'S', 'R', '9', '8',
       '1', '7', 'Z', 'Q', 'A', '2', 'P', '8', 'L', '7', 'Y', 'T', 'U',
       'N', '2', 'T', 'I', 'A', '0', '5', 'G', '1', 'K', 'B', '9', 'V',
       '1', 'Y', 'F', 'Y', '3', 'P', '5', 'G', 'Y', '1', '1', 'F', 'B',
       'I', '0', 'I', 'B', 'F', '9', '6', 'I', 'Y', '0', '3', 'W', '1',
       'N', 'B', 'D', '9', '8', 'C', '4', '5', 'G', '1', '1', 'T', 'Z',
       '1', '5', '4', '1', 'I', 'Z', 'U', 'H', '0', '5', 'A', 'O', 'N',
       'Z', 'K', 'D', 'H', 'U', '4', '0', '2', 'D', '0', '8', 'P', 'T',
       '6', '5', '4', 'W', 'F', '8', '2', '9', 'B', 'B', 'Y', '6', 'O',
       '4', '5', '4', '1', '2', 'F', '3', 'S', 'S', '4', '3', 'Q', '1',
       'P', 'D', '3', 'Z', 'E', 'A', 'S', 'S', '3', 'F', '2', 'V', 'A',
       'Z', 'H', 'Y', 'N', 'A', 'H', '3', 'R', '2', 'A', 'O', 'B', 'P',
       '6', '6', '5', 'H', 'A', '6', '2', 'R', 'G', 'N', '2', 'G', 'G',
       'I', 'C', '0', '6', 'I', 'K', 'F', 'R', 'Y', '6', '8', '1', 'C',
       '3', 'B', '9', 'Y', '9', 'W', 'S', 'I', '8', 'G', 'L', '8', 'G',
       '8', 'G', 'R', '1', 'S', '1', '9', '1', '8', 'D', 'O', '4', '1',
       'O', '2', 'F', 'I', '3', '3', 'M', '2', '1', '6', '4', 'O', '6',
       'B', '2', 'V', 'R', '1', 'Z', 'B', '1', '3', 'E', '9', '8', 'O',
       'Y', 'T', '2', '8', '9', '6', '0', '2', 'H', '3', '5', '4', 'H',
       '6', 'Q', 'S', '3', 'N', 'P', 'F', '0', 'R', 'N', 'B', 'Y', 'T',
       '4', 'T', '3', 'Y', '4', 'Y', 'M', 'J', '3', 'U', 'U', '3', 'W',
       '8', '2', 'H', '1', '3', '9', 'K', 'G', 'F', 'M', '4', 'O', '1',
       '5', 'K', '3', '0', 'N', 'T', '6', 'R', '5', '4', '6', 'I', '8',
       'W', '8', 'W', 'K', '6', '6', 'K', 'R', '4', 'Y', 'I', 'A', 'C',
       'Z', '9', 'M', 'E', '6', '5', '5', '1', 'G', '9', '8', '3', 'T',
       'T', 'D', '7', '1', 'O', 'N', '8', 'K', 'J', '2', '2', 'J', '5',
       'K', 'I', 'L', '5', '4', 'U', 'F', 'W', 'V', 'J', 'T', '6', 'L',
       '3', 'J', '5', 'T', 'W', 'O', 'D', 'G', '9', 'I', 'R', 'A', 'F',
       'M', '8', 'V', 'I', 'M', '2', 'D', 'G', 'C', 'H', '6', '0', 'O',
       'B', 'M', 'C', 'U', '0', '6', '0', 'Z', '3', 'T', 'H', '6', '3',
       '0', 'M', '5', 'A', '4', 'O', '4', 'U', '9', 'F', 'T', 'Q', 'G',
       '3', 'N', '6', 'S', 'D', 'Z', '0', 'A', '1', 'Z', 'R', 'P', 'I',
       '2', '5', 'W', 'A', '1', 'W', 'J', '5', '9', '3', 'V', '8'],
      dtype='<U11')
```

"ytest[:1000]" kod satırı, test veri kümesinin etiketlerini (y labels) içeren liste döndürüyor. Bu kod satırı, "ytest" listesinin ilk 1000 elemanını alıyor. Bu elemanlar, test veri kümesinin ilk 1000 görüntüsünün etiketlerini temsil ediyor.

Etiketler, veri kümelerinde her bir görüntünün hangi sınıfa ait olduğunu belirtir. Örneğin, eğer veri kümesi el yazısı harflerden oluşuyorsa, her bir görüntünün etiketi bir harf olabilir. Bu etiketler, veri kümesini sınıflara ayırmak için kullandık ve modelin eğitimi sırasında sınıflar arasındaki farklılıkları öğrenmesine yardımcı olduk.

Öğrenme Modeli

In [28]:

```
log = LogisticRegression(solver='lbfgs', max_iter=1000)
log.fit(xtrain, ytrain)
```

Out[28]:

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

Bu kod satırıyla "Logistic Regression" (Lojistik Regresyon) modelini eğitiyoruz. Lojistik Regresyon, çok sınıflı sınıflandırma (multiclass classification) problemlerinde kullanılan bir model türüdür. Bu model, veri kümesinde bulunan özellikleri (features) kullanarak, her bir görüntünün hangi sınıfa ait olduğunu tahmin etmeye çalışır.

İlk olarak, "log = LogisticRegression(solver='lbfgs', max_iter=1000)" kod satırı ile "LogisticRegression" sınıfından bir nesne oluşturduk. Bu nesne, "solver='lbfgs'" ve "max_iter=1000" parametreleriyle birlikte oluşturuldu. "solver" parametresi, modelin ağırlıklarını nasıl optimize edeceğini belirtiyor. "lbfgs" değeri, modelin ağırlıklarını "Limited-memory BFGS" (BFGS: Broyden–Fletcher–Goldfarb–Shanno) algoritmasını kullanarak optimize edilmesini belirtiyor. "max_iter" parametresi ise, modelin en fazla kaç iterasyon (adım) yapacağını belirtiyor.

Sonra, "log.fit(xtrain, ytrain)" kod satırı ile model eğitimi gerçekleştirdik. Bu kod satırında, "fit()" fonksiyonu ile model eğitiliyor. Bu fonksiyonun ilk argümanı "xtrain" (indirgenmiş eğitim veri kümesi), ikinci argümanı ise "ytrain" (eğitim veri kümesinin etiketleri) olarak verdik. Bu sayede, model "xtrain" veri kümesindeki özellikleri kullanarak, "ytrain" veri kümesindeki etiketleri tahmin etmeye çalışıyor. Model, bu tahminlerini yaparken, veri kümesindeki özellikler arasındaki ilişkileri öğreniyor. Model öğrendiği ilişkileri kullanarak, yeni görüntüler için etiket tahminleri yapmaya çalışıyor.

Sonra, modelin tahmin performansını ölçmek için "accuracy_score" fonksiyonunu kullandık. Bu fonksiyon, verilen tahminler ile gerçek etiketler arasındaki uyum oranını hesaplıyor ve bu oranı yüzde olarak döndürüyor. Eğer tüm tahminler doğru ise, bu fonksiyon %100 oranını döndürür. Eğer tüm tahminler yanlış ise, %0 oranını döndürür.

"predictions = log.predict(xtest)" kod satırında, model "xtest" veri kümesindeki görüntülerin etiketlerini tahmin etmeye çalışıyor. Bu tahminleri "predictions" değişkenine atadık.

"accuracy = accuracy_score(ytest, predictions)" kod satırında, "accuracy_score()" fonksiyonu ile tahminlerin doğruluk oranını hesaplıyoruz. Bu fonksiyonun ilk argümanı "ytest" (test veri kümesinin etiketleri), ikinci argümanı ise "predictions" (modelin tahminleri) olarak veriliyor. Bu sayede, "accuracy_score()" fonksiyonu "ytest" ve "predictions" değişkenleri arasındaki uyum oranını hesaplayıp ve "accuracy" değişkenine atıyoruz.

Son olarak, "print('Modelin doğruluk oranı:', accuracy)" kod satırı ile modelin doğruluk oranını ekrana yazdırıyoruz.

Tahmin Etme

In [29]:

```
tr_pred = log.predict(xtrain)
ts_pred = log.predict(xtest)
print(ts_pred)
type(ts_pred[1])
```

```
['R' '8' 'J' ... 'S' '0' '6']
```

Out[29]:

```
numpy.str_
```

Bu kod satırlarında, modelin eğitim veri kümesindeki görüntüler için yaptığı tahminleri "tr_pred" değişkenine, test veri kümesindeki görüntüler için yaptığı tahminler ise "ts_pred" değişkenine atıyoruz.

"tr_pred = log.predict(xtrain)" kod satırında, "predict()" fonksiyonu ile model "xtrain" veri kümesindeki görüntülerin etiketlerini tahmin etmeye çalışıyor. Bu tahminleri "tr_pred" değişkenine atadık.

"ts_pred = log.predict(xtest)" kod satırında ise, "predict()" fonksiyonu ile model "xtest" veri kümesindeki görüntülerin etiketlerini tahmin etmeye çalışıyor. Bu tahminleri "ts_pred" değişkenine atadık.

"print(ts_pred)" kod satırında ise, "ts_pred" değişkeninin içeriğini ekrana yazdırdık. Bu sayede, modelin test veri kümesindeki görüntüler için yaptığı tahminleri gösterdik.

Son olarak, "type(ts_pred[1])" kod satırında, "ts_pred[1]" değişkeninin veri türü öğreniliyor. Bu değişkenin veri türünü, ekrana yazdırılarak gösterdik.

Doğru Tahmin Ölçümü

In [30]:

```
print("Öğrenme Skoru:", accuracy_score(ytrain,tr_pred))
print("Test Skoru:", accuracy_score(ytest,ts_pred))
```

```
Öğrenme Skoru: 0.8252681764004768
Test Skoru: 0.7115843000158907
```

Bu kod satırlarında, modelin eğitim ve test veri kümelerinde yaptığı tahminlerin doğruluk oranlarını hesaplıyoruz ve ekrana yazdırıyoruz. Karakterlerin hepsini dahil etmeden belli karaktere kadar denediğimizde skor yüzdemiz daha yüksekti, ancak datasetimiz içerisinde 30 bini aşkın veri bulunmakta ve bu veriler farklı yazım şekillerinde olduğu için doğru tahmin oranımızı düşürdü. Ayrıca genellikle yapılan projeler yalnızca harf veya yalnızca rakamları içeriyor, biz içine rakamları ve Türkçe karakterleri de ekleyince doğruluk oranımız biraz daha düştü. Oranı yükseltmek ve makinenin öğrenimini kolaylaştırmak için Türkçe karakterleri çıkartmak zorunda kaldık.

"Öğrenme Skoru" ve "Test Skoru" ifadeleri, ekrana yazdırdığımız mesajların başlıklarıdır ve modelin eğitim ve test veri kümelerinde yaptığı tahminlerin doğruluk oranlarını gösteriyor.

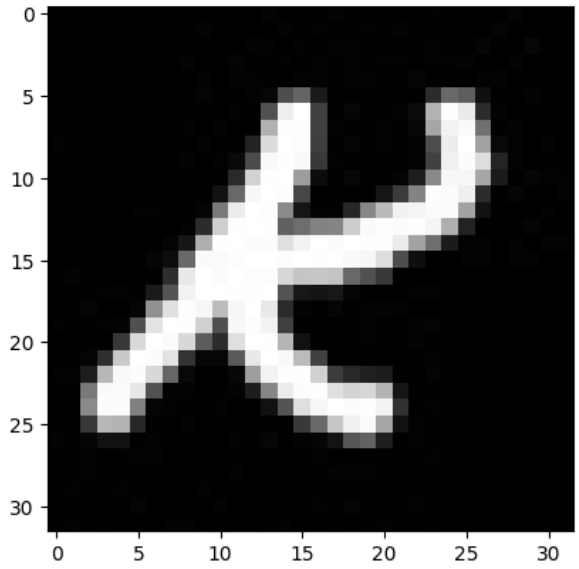
"print("Öğrenme Skoru", accuracy_score(ytrain,tr_pred))" kod satırında, "accuracy_score()" fonksiyonu ile modelin eğitim veri kümesindeki görüntüler için yaptığı tahminlerin doğruluk oranını hesaplıyoruz. Bu fonksiyonun ilk argümanı "ytrain" (eğitim veri kümesinin etiketleri), ikinci argümanı ise "tr_pred" (modelin eğitim veri kümesindeki görüntüler için yaptığı tahminler) olarak verdik. Bunun sonucunda, "accuracy_score()" fonksiyonu "ytrain" ve "tr_pred" değişkenleri arasındaki uyum oranınını hesapladık ve "Öğrenme Skoru" mesajı ile birlikte ekrana yazdırdık.

"print("Test Skoru", accuracy_score(ytest,ts_pred))" kod satırında ise, "accuracy_score()" fonksiyonu ile modelin test veri kümesindeki görüntüler için yaptığı tahminlerin doğruluk oranını hesapladık. Bu fonksiyonun ilk argümanı "ytest" (test veri kümesinin etiketleri), ikinci argümanı ise "ts_pred" (modelin test veri kümesindeki görüntüler için yaptığı tahminler) olarak verdik. Bununla birlikte "accuracy_score()" fonksiyonunun "ytest" ve "ts_pred" değişkenleri arasındaki uyum oranını hesapladık ve "Test Skoru" mesajı ile birlikte ekrana yazdırdık.

In [31]:

```
plt.imshow(x_test[0].reshape(32,32), cmap='gray')
print(ytest[0])
```

```
K
```



Bu kod satırında, "x_test[0]" adresinde saklanan görüntünün resmini ekrana gösterdik ve görüntünün etiketini de yazdırmış olduk.

"plt.imshow()" fonksiyonu, bir görüntüyü ekrana göstermek için kullanılır. Bu fonksiyonun ilk argümanıyla gösterilecek görüntüyü numpy dizisi olarak verdik. "cmap='gray'" argümanı, görüntüyü siyah tonlarında gösterdik.

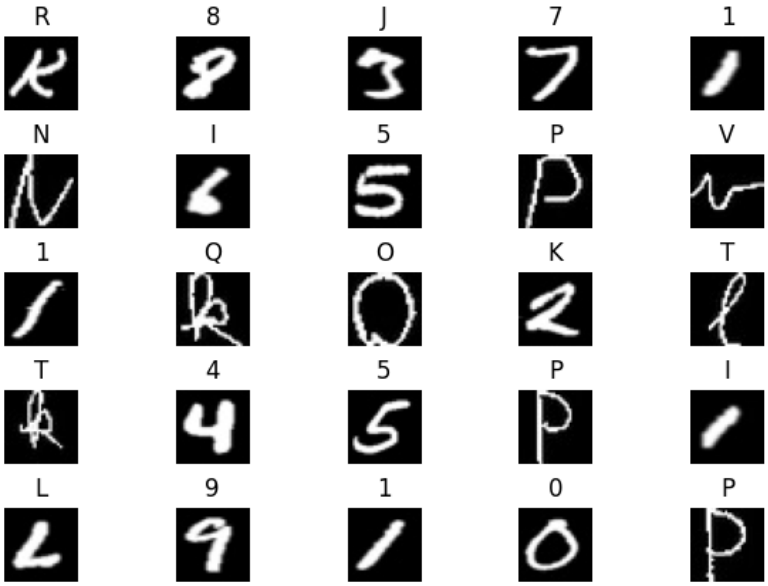
"x_test[0]" ifadesi, test veri kümesinin 0. indisteki görüntüyü gösteriyor. Bu görüntü, "x_test" değişkenine atamış olduğumuz numpy dizisinin bir öğesidir ve 32x32 boyutlarındadır. Ancak, "plt.imshow()" fonksiyonu sadece tek boyutlu dizileri kabul ediyor. Bu nedenle, "x_test[0]" dizisi "reshape()" fonksiyonu ile 32x32 boyutlarına tekrar düzenleyip ve "plt.imshow()" fonksiyonuna argüman olarak verdik.

"print(ytest[0])" kod satırında ise, "ytest" değişkeninin 0. indisteki etiketini yazdırdık. Bu etiket, "ytest" değişkenine atamış olduğumuz numpy dizisinin bir öğesidir ve gösterilen görüntünün hangi sınıfa ait olduğunu belirtiyor.

Test Modeli

In [32]:

```
decode = {'0':0,'1':1,'2':2,'3':3,'4':4,'5':5,'6':6,'7':7,'8':8,'9':9,'A':'A','B':'B','C':'C','D':'D','E':'E',
        'F':'F','G':'G','H':'H','I':'I','J':'J','K':'K','L':'L','M':'M', 'N':'N','O':'O','P':'P','Q':'Q','R':'R',
        'S':'S','T':'T','U':'U','V':'V','W':'W','Y':'Y','Z':'Z'}
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.imshow(x_test[i].reshape(32,32),cmap='gray')
    title = str(decode[str(ts_pred[i])])
    plt.title(title)
    plt.subplots_adjust(left=0.1,
                        bottom=0.1,
                        right=1,
                        top=0.9,
                        wspace=0.3,
                        hspace=0.6)
   # plt.title(decode[ts_pred[i]])
    plt.axis('off')
```



Bu kod satırlarında, test datasetinden 25 adet görüntünün resmini ekrana gösterdik ve görüntülerin tahmin edilen sınıflarını her bir görüntünün üstüne yazdırdık.

Öncelikle, "decode" adında bir sözlük oluşturduk. Bu sözlükte, her bir sınıf için bir anahtar ve değer oluşturduk. Anahtar, sınıfı temsil eden bir harf ve sayıdır, değer ise bu harfin karşılığı olan sayıdır veya harfttir. Bu sözlüğü, daha sonra ekrana yazdıracağımız sınıf etiketlerini harf veya sayılara çevirmek için kullanacağız.

Daha sonra, "for" döngüsü kullanarak 25 adet görüntü için bir ekran parçası oluşturduk. "plt.subplot()" fonksiyonu, birden fazla görüntüyü aynı ekranda göstermek için kullanılır. Bu fonksiyonun ilk argümanı, kaç satır kaç sütundan oluşan bir düzen oluşturulacağını belirtiyor. İkinci argüman ise, oluşturulan düzenin hangi sıradaki ekran parçasına veri gösterileceğini belirtiyor. Bu örnekte, 5 satır ve 5 sütunluk bir düzen oluşturduk ve döngüde kaçıncı görüntünün ekrana gösterileceğini belirttik.

"plt.imshow()" fonksiyonu ile, görüntüyü ekranda gösterdik. Bu fonksiyonun ilk argümanı olarak, düzenlenmiş "x_test[i]" görüntüsünü verdik. "cmap='gray'" argümanı ile de, görüntüyü gri-siyah tonları olarak gösterdik.

Daha sonra, görüntünün öngörülen sınıfını harf karşılığı olarak "title" değişkenine atadık. Bu değişken, "decode" sözlüğü kullanılarak karakterler karşılığı olan karakterlere çevrilir. Son olarak, "plt.title()" fonksiyonu ile ekrana yazdırdık.

In [33]:

```
np.where(ts_pred!=ytest)
d = pd.DataFrame({'Gercek':ytest, 'Tahmin':ts_pred})
d[d['Gercek']!=d['Tahmin']]
```

Out[33]:

	Gercek	Tahmin
0	K	R
2	3	J
6	6	I
11	K	Q
13	2	K
...	...	...
6277	5	F
6280	L	M
6281	Y	3
6286	Z	G
6288	4	F

1815 rows × 2 columns

Bu kod bloğu ile tahmin edilen sınıfların gerçek sınıflarla karşılaştırılmasını yaptık. Öncelikle, gerçek sınıflar ve tahmin edilen sınıfların bir NumPy dizisi olarak saklandığı ytest ve ts_pred değişkenlerinden oluşan bir NumPy dizisi oluşturduk. Bu dizideki değerlerin eşleşip eşleşmediğini, "ts_pred!=ytest" ifadesiyle kontrol ettik. Eşleşmeyen değerlerin indekslerini, "np.where()" fonksiyonuyla bulduk. Daha sonra, bu indeksleri kullanarak gerçek ve tahmin edilen sınıfların bir Pandas veri çerçevesini oluşturduk ve veri çerçevesinde eşleşmeyen değerleri seçilerek gösterdik. Bu kod bloğunu, tahmin edilen sınıfların gerçek sınıflarla ne kadar benzer olduğunu ölçmek için kullanıyoruz.

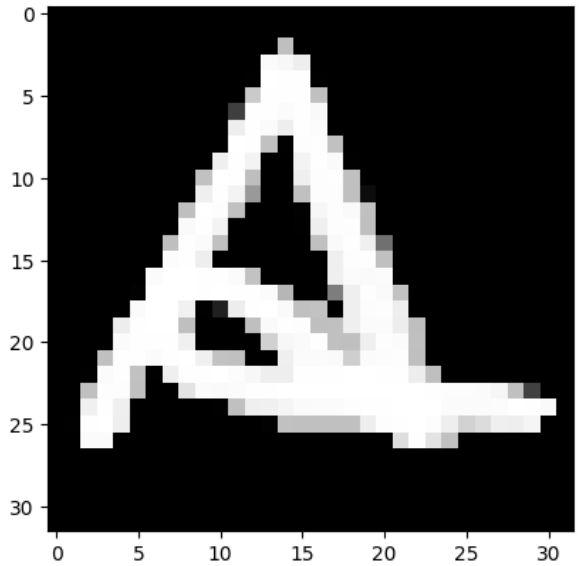
Gerçek Veriler Üzerinde Test Etme

In [34]:

```
img = cv2.resize(cv2.imread('adeneme.jpg',0), (32,32))
plt.imshow(img,cmap='gray')
```

Out[34]:

```
<matplotlib.image.AxesImage at 0x2461c8d4460>
```



Bu kod satırıyla, "cv2.resize()" fonksiyonu kullanılarak örneğin, yolunu verdiğimiz "adeneme.jpg" adlı bir görüntü dosyasını okudu ve önceden belirlemiş olduğumuz (32,32) boyutlarına yeniden ölçeklendirdik. Buradaki amacımız, kendi yazdığımız bir harfi doğru tahmin etmesini test etmekti. Daha sonra, bu yeniden ölçeklendirilmiş görüntüyü "plt.imshow()" fonksiyonu ile ekranda gösterdik. "cmap='gray'" argümanı ise önceden de değindiğimiz gibi görüntünün gri tonlamalı olarak gösterilmesini sağladı.

In [35]:

```
img = pca.transform(img.reshape(1,-1)/255)
```

Bu kod satırında, verilen görüntünün boyutlarını 32x32 piksel olarak değiştirdik ve numpy dizisi olarak okundu. Daha sonra, görüntüyü "img" değişkenine atadık.

Sonra, "pca" değişkenine atanan PCA nesnesinin "transform()" fonksiyonunu kullanarak, görüntünün özelliklerini önceden belirlediğimiz sayıda özellikle özetledik. Bu işlem sırasında, görüntünün orijinal boyutlarındaki özellikleri azalttık ve görüntünün özetlenmiş özelliklerini numpy dizisi olarak döndürdük. Bu özellikleri, daha önceden belirlediğimiz n_components değerine göre seçtik. Bu değer, PCA sırasında kullanılan özellik sayısını belirtiyor. "img" değişkeninin yeni özellikleri, "img" değişkenine atanmış oldu.

In [36]:

```
decode[log.predict(img)[0]]
```

Out[36]:

```
'A'
```

"decode" değişkeninin, görüntülerin sınıfını temsil eden harflerin sayısal karşılıklarını tutan bir sözlük verisi olduğunu söylemiştik. "log.predict(img)[0]" kod satırı, "log" adlı Logistic Regression modelini kullanarak verilen görüntünün sınıfını tahmin ettik ve tahmin sonucunu bir dize olarak döndürmüş olduk. "decode" sözlüğünde bu dizeyle eşleşen sayısal karşılığı arandı ve eşleşen sayısal karşılık döndürüldü. Örneğin, eğer "log.predict(img)[0]" dizesi "A" olarak döndürürse, "decode" sözlüğünde "A" anahtarı ile eşleşen sayısal değer 10 olarak döndürülür. Son olarak, tahmin edilen harf çıktısı ekranda yazdırılır örneğin, biz "A" harfinin testini yaptık ve doğru tahminde bulunup 'A' çıktısını aldık.