mnist-data

September 3, 2024

[105]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal as mvn
```

[106]:
```
mnist_data = pd.read_csv("/content/MNIST_train.csv")
mnist_test_data = pd.read_csv("/content/MNIST_test.csv")
mnist_data
```

[106]:

	Unnamed: 0	index	labels	0	1	2	3	4	5	6	…	774	775	776	\
0	0	0	5	0	0	0	0	0	0	0	…	0	0	0	
1	1	1	0	0	0	0	0	0	0	0	…	0	0	0	
2	2	2	4	0	0	0	0	0	0	0	…	0	0	0	
3	3	3	1	0	0	0	0	0	0	0	…	0	0	0	
4	4	4	9	0	0	0	0	0	0	0	…	0	0	0	
…	…	…	…	..	..	..	..	..	..	..	…	…	…	…	
59995	59995	59995	8	0	0	0	0	0	0	0	…	0	0	0	
59996	59996	59996	3	0	0	0	0	0	0	0	…	0	0	0	
59997	59997	59997	5	0	0	0	0	0	0	0	…	0	0	0	
59998	59998	59998	6	0	0	0	0	0	0	0	…	0	0	0	
59999	59999	59999	8	0	0	0	0	0	0	0	…	0	0	0	

	777	778	779	780	781	782	783
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…
59995	0	0	0	0	0	0	0
59996	0	0	0	0	0	0	0
59997	0	0	0	0	0	0	0
59998	0	0	0	0	0	0	0
59999	0	0	0	0	0	0	0

[60000 rows x 787 columns]

[107]:
```
unique_labels = mnist_data['labels'].unique()
print(unique_labels)
```

```
[5 0 4 1 9 2 3 6 7 8]
```

[108]:
```
for column in mnist_data.columns:
  print(column)
```

```
Unnamed: 0
index
labels
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
```

85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132

```
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
```

[109]:
```
mnist_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60000 entries, 0 to 59999
Columns: 787 entries, Unnamed: 0 to 783
dtypes: int64(787)
memory usage: 360.3 MB
```

[110]:
```
mnist_data.describe()
```

[110]:

	Unnamed: 0	index	labels	0	1	2	\
count	60000.000000	60000.000000	60000.000000	60000.0	60000.0	60000.0	
mean	29999.500000	29999.500000	4.453933	0.0	0.0	0.0	
std	17320.652413	17320.652413	2.889270	0.0	0.0	0.0	
min	0.000000	0.000000	0.000000	0.0	0.0	0.0	
25%	14999.750000	14999.750000	2.000000	0.0	0.0	0.0	
50%	29999.500000	29999.500000	4.000000	0.0	0.0	0.0	
75%	44999.250000	44999.250000	7.000000	0.0	0.0	0.0	
max	59999.000000	59999.000000	9.000000	0.0	0.0	0.0	

```
              3        4        5        6  …           774           775  \
count  60000.0  60000.0  60000.0  60000.0  …  60000.000000  60000.000000
mean       0.0      0.0      0.0      0.0  …      0.200433      0.088867
std        0.0      0.0      0.0      0.0  …      6.042472      3.956189
min        0.0      0.0      0.0      0.0  …      0.000000      0.000000
25%        0.0      0.0      0.0      0.0  …      0.000000      0.000000
50%        0.0      0.0      0.0      0.0  …      0.000000      0.000000
75%        0.0      0.0      0.0      0.0  …      0.000000      0.000000
max        0.0      0.0      0.0      0.0  …    254.000000    254.000000

                776           777           778         779      780      781  \
count  60000.000000  60000.000000  60000.000000  60000.0000  60000.0  60000.0
mean       0.045633      0.019283      0.015117      0.0020      0.0      0.0
std        2.839845      1.686770      1.678283      0.3466      0.0      0.0
min        0.000000      0.000000      0.000000      0.0000      0.0      0.0
25%        0.000000      0.000000      0.000000      0.0000      0.0      0.0
50%        0.000000      0.000000      0.000000      0.0000      0.0      0.0
75%        0.000000      0.000000      0.000000      0.0000      0.0      0.0
max      253.000000    253.000000    254.000000     62.0000      0.0      0.0

           782      783
count  60000.0  60000.0
mean       0.0      0.0
std        0.0      0.0
min        0.0      0.0
25%        0.0      0.0
50%        0.0      0.0
75%        0.0      0.0
max        0.0      0.0

[8 rows x 787 columns]
```

[111]:
```
mnist_data.isnull().sum()
```

[111]:
```
Unnamed: 0    0
index         0
labels        0
0             0
1             0
             ..
779           0
780           0
781           0
782           0
783           0
Length: 787, dtype: int64
```

[112]:
```
mnist_data.shape
```

[112]:
```
(60000, 787)
```

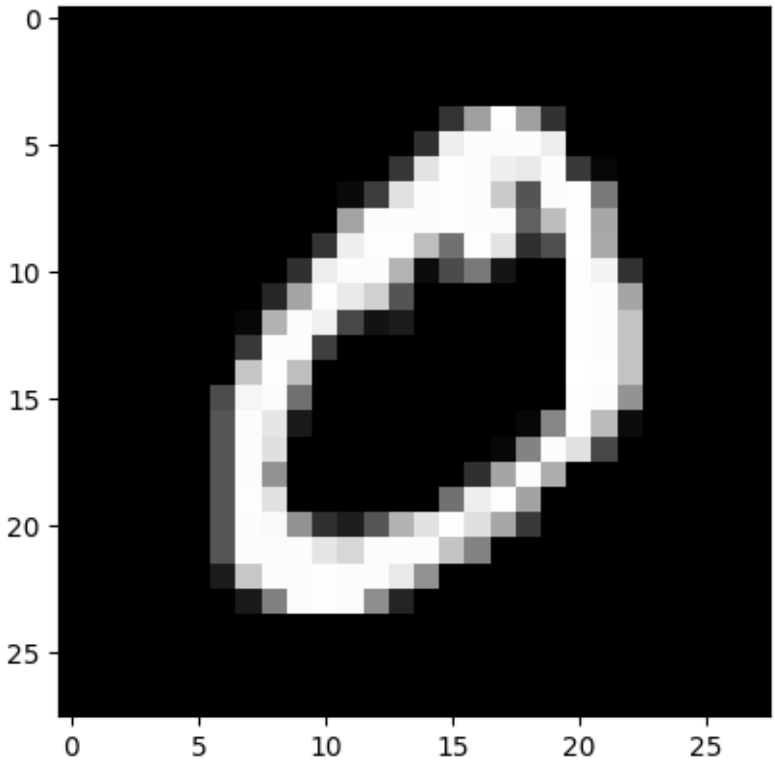
[113]:
```
mnist_data= mnist_data.drop_duplicates()
mnist_data.shape
```

[113]:
```
(60000, 787)
```

[114]:
```
columns_to_drop = ['Unnamed: 0', 'index', 'labels']
X= mnist_data.drop(columns=columns_to_drop).to_numpy()
X_test = mnist_test_data.drop(columns=columns_to_drop).to_numpy()
y = mnist_data["labels"].to_numpy()
y_test = mnist_test_data["labels"].to_numpy()
X
```

[114]:
```
array([[0, 0, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0],
       …,
       [0, 0, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0],
       [0, 0, 0, …, 0, 0, 0]])
```

[115]:
```
obs_1=X[1,:]
obs_1 = obs_1.reshape(28, 28)
plt.imshow(obs_1, cmap='gray')
plt.show()
```

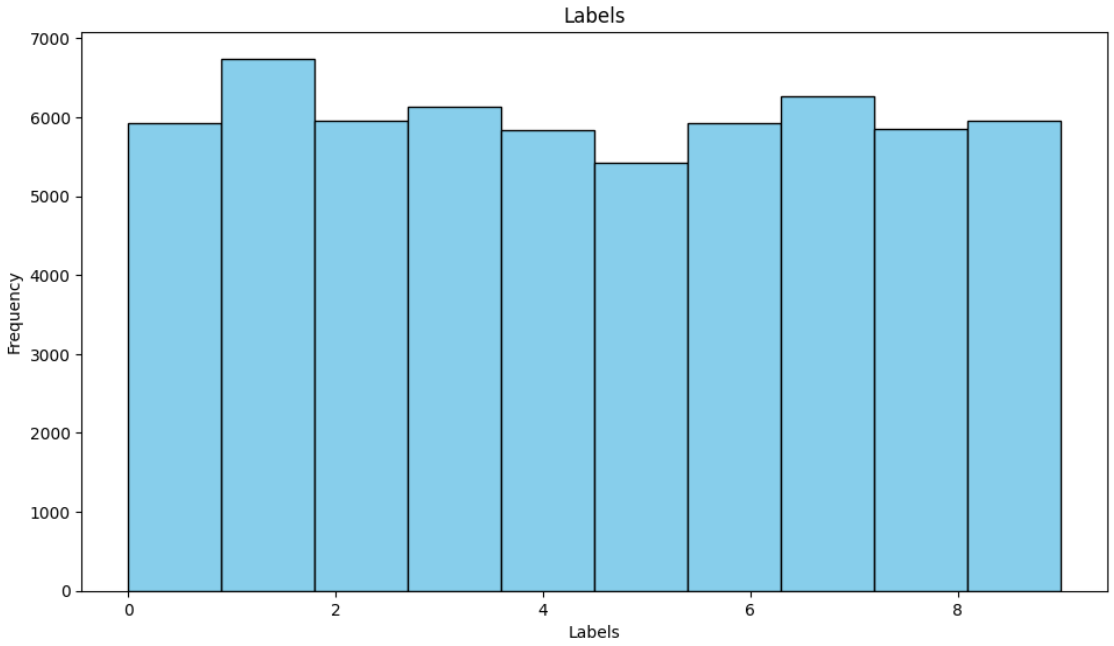


[116]:
```
# Function to plot variable distributions
def plot_variable_distributions(df):
    features = ['labels']
    plt.figure(figsize=(10, 6))

    for i, feature in enumerate(features, 1):
        plt.subplot(1, 1, i)
        df[feature].hist(grid=False, color='skyblue', edgecolor='black')
        plt.title(feature.capitalize(), fontsize=12)
        plt.xlabel("Labels", fontsize=10)
        plt.ylabel("Frequency", fontsize=10)
        plt.xticks(rotation=0)

    plt.tight_layout(pad=2.0)
    plt.suptitle("Variable Distributions", fontsize=16, y=1.05)
    plt.show()

plot_variable_distributions(mnist_data)
```



[117]:
```
X= X/255
X_test = X_test/255
X
```

[117]:
```
array([[0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       …,
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.]])
```

[118]:
```
class KNNClassifier():
    def fit(self, X, y):
        self.X = X
        self.y = y

    def predict(self, X, K, epsilon=1e-3):
        N = len(X)
        y_hat = np.zeros(N)

        for i in range(N):
            dist2 = np.sum((self.X - X[i])**2, axis=1)
            idxt = np.argsort(dist2)[:K]
```

```
            gamma_k = 1 / (np.sqrt(dist2[idxt] + epsilon))

            y_hat[i] = np.bincount(self.y[idxt], weights=gamma_k).argmax()
        return y_hat
```

[119]:
```
def accuracy(y, y_hat):
  return np.mean(y==y_hat)
```

[120]:
```
MNIST_KNN = KNNClassifier()
MNIST_KNN.fit(X, y)
y_pred = MNIST_KNN.predict(X_test, K=5)
accuracy(y_pred, y_test)
```

[120]: 0.9691

[122]:
```
mnist_data_sampled= mnist_data.copy()
min_class_size = mnist_data_sampled['labels'].value_counts().min()
sampled_df = mnist_data_sampled.groupby('labels').apply(lambda x: x.
 ↪sample(n=min_class_size)).reset_index(drop=True)
X_sampled = sampled_df.drop(columns=['labels', 'Unnamed: 0', 'index']).
 ↪to_numpy()
y_sampled = sampled_df['labels'].to_numpy()
X_sampled = X_sampled/255.0

MNIST_KNN_resampled = KNNClassifier()
MNIST_KNN_resampled.fit(X_sampled, y_sampled)
y_pred_resampled = MNIST_KNN_resampled.predict(X_test, K=5)
accuracy(y_pred_resampled, y_test)
```

[122]: 0.9686