Content

Hazırlayanlar

- #### Grup Numarası: 3
- #### Anıl Dursun İpek - 031890131
- #### Batuhan Arslandaş - 032190097

Veri Setinin Yüklenmesi

In [1]:
```
# Gerekli Kütüphaneler
import numpy as np # lineer cebir kütüphanesi
import matplotlib.pyplot as plt # görselleştirme kütüphanesi
import pandas as pd
import seaborn as sns
from sklearn.metrics import confusion_matrix
sns.set_style('white')
```

In [2]:
```
letters = pd.read_csv('letter_recognition.data')
```

Verinin İncelenmesi

1. Data içerisinde null değer bulunmamaktadır.
2. Toplam veri sayısı: 20000
3. Toplam feature sayısı: 17
4. Veri Türleri:
    - Letter: object
    - F1 - F16: int64

In [3]:
```
letters.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Letter  20000 non-null  object
```

```
 1   F1      20000 non-null  int64
 2   F2      20000 non-null  int64
 3   F3      20000 non-null  int64
 4   F4      20000 non-null  int64
 5   F5      20000 non-null  int64
 6   F6      20000 non-null  int64
 7   F7      20000 non-null  int64
 8   F8      20000 non-null  int64
 9   F9      20000 non-null  int64
 10  F10     20000 non-null  int64
 11  F11     20000 non-null  int64
 12  F12     20000 non-null  int64
 13  F13     20000 non-null  int64
 14  F14     20000 non-null  int64
 15  F15     20000 non-null  int64
 16  F16     20000 non-null  int64
dtypes: int64(16), object(1)
memory usage: 2.6+ MB
```

In [4]:
```
letters.describe()
```

Out[4]:

	F1	F2	F3	F4	F5	F6	F7	
count	20000.000000	20000.000000	20000.000000	20000.00000	20000.000000	20000.000000	20000.000000	20000.00
mean	4.023550	7.035500	5.121850	5.37245	3.505850	6.897600	7.500450	4.62
std	1.913212	3.304555	2.014573	2.26139	2.190458	2.026035	2.325354	2.69
min	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
25%	3.000000	5.000000	4.000000	4.00000	2.000000	6.000000	6.000000	3.00
50%	4.000000	7.000000	5.000000	6.00000	3.000000	7.000000	7.000000	4.00
75%	5.000000	9.000000	6.000000	7.00000	5.000000	8.000000	9.000000	6.00
max	15.000000	15.000000	15.000000	15.00000	15.000000	15.000000	15.000000	15.00

In [5]:
```
# Veri eğitim ve test olarak ayrıldı
train = letters[:16000]
test = letters[16000:]
print("Train length -> ", len(train), " Test length -> ", len(test))
```

```
Train length ->  16000  Test length ->  4000
```

In [6]:
```
train.head()
```

Out[6]:

	Letter	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
0	T	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8
1	I	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10
2	D	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9
3	N	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8
4	G	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10

In [7]:
```
test.head()
```

Out[7]:

	Letter	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
16000	U	4	10	6	7	9	9	6	4	3	6	7	7	9	8	5	6
16001	N	6	9	8	4	3	8	7	3	4	13	5	8	6	8	0	8

16002	V	6	9	8	8	10	7	7	5	4	7	6	8	7	9	7	10
16003	I	5	6	6	4	3	7	6	2	7	7	6	9	0	9	4	8
16004	N	5	9	7	6	4	9	7	3	5	10	4	6	5	8	1	7

In [8]:
```
data = train.values
data
```

Out[8]:
```
array([['T', 2, 8, ..., 8, 0, 8],
       ['I', 5, 12, ..., 8, 4, 10],
       ['D', 4, 11, ..., 7, 3, 9],
       ...,
       ['G', 8, 14, ..., 7, 5, 8],
       ['E', 4, 7, ..., 8, 5, 8],
       ['C', 2, 1, ..., 9, 4, 10]], dtype=object)
```

In [9]:
```
# Veri seti içerisindeki harfler diziye aktarıldı
letter_unique = np.unique(train["Letter"])
letter_unique
```

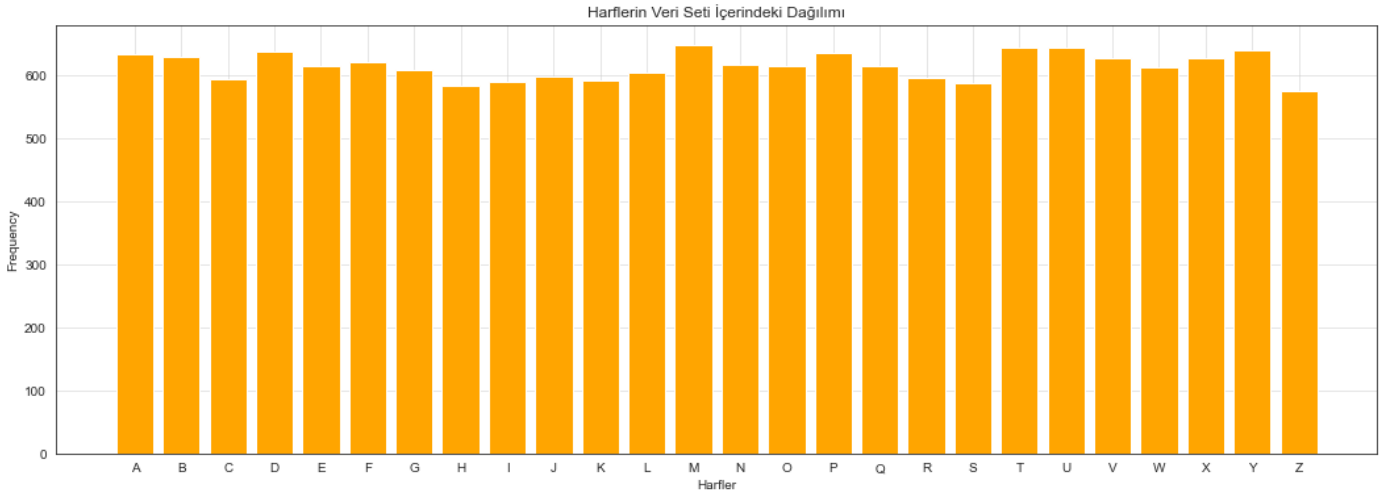
Out[9]:
```
array(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
       'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'],
      dtype=object)
```

Verinin Görselleştirilmesi

In [10]:
```
# Eğitim veri seti içerisindeki her bir harf gruplandırıldı ve toplam sayıları diziye ak
letter_num =  train.groupby('Letter')['Letter'].count()
print(letter_num)
letter_counts =  letter_num.values
letter_sum = sum(letter_counts)
print("Toplam Harf Sayısı(Train datası içerisindeki veri sayısı): ",letter_sum)
```

```
Letter
A    633
B    630
C    594
D    638
E    616
F    622
G    609
H    583
I    590
J    599
K    593
L    604
M    648
N    617
O    614
P    635
Q    615
R    597
S    587
T    645
U    645
V    628
W    613
X    628
Y    641
Z    576
Name: Letter, dtype: int64
Toplam Harf Sayısı(Train datası içerisindeki veri sayısı):  16000
```

In [11]:
```
plt.figure(figsize=(18,6))
plt.bar(letter_num.index,letter_num.values,color="orange")
plt.xlabel("Harfler")
plt.ylabel("Frequency")
plt.title("Harflerin Veri Seti İçerindeki Dağılımı")
plt.grid(True,alpha=0.6)
plt.show()
```



Öncül Olasılıkların Hesabı

In [12]:
```
prior_prob = letter_counts/letter_sum
print("Öncül Olasılıklar \n",prior_prob)
```

```
Öncül Olasılıklar
 [0.0395625 0.039375  0.037125  0.039875  0.0385    0.038875  0.0380625
 0.0364375 0.036875  0.0374375 0.0370625 0.03775   0.0405    0.0385625
 0.038375  0.0396875 0.0384375 0.0373125 0.0366875 0.0403125 0.0403125
 0.03925   0.0383125 0.03925   0.0400625 0.036    ]
```

Toy Bayés Yöntemi Kullanımı Ve Açıklaması

In [13]:
```
# Feature'lar algoritma için değişkene aktarıldı
features = train.columns
num_features = features.shape[0]
print(features)
```

```
Index(['Letter', 'F1', 'F2', 'F3', 'F4', 'F5', 'F6', 'F7', 'F8', 'F9', 'F10',
       'F11', 'F12', 'F13', 'F14', 'F15', 'F16'],
      dtype='object')
```

In [14]:
```
# Data set içerisindeki hesaplanan koşullu olasılıkların aktarılması için dataframe oluş
df = pd.DataFrame(columns=["Letter","Feature","Value","Prob"])

for i in range(1,num_features): # Her bir feature'u hesaplaması için oluşturulan döngü
    fi =  train.groupby(['Letter', features[i]])[features[i]].count() # Veri seti her bi
    print(f"FEATURE {features[i]}")
    for x in letter_unique: # Her bir harfi hesaplaması için oluşturulan döngü
        values = np.sort(train[train.Letter == x][features[i]].unique()) # Harf içerisin
        print("Harfin feature içerisindeki değer aralığı \n",values)
        for y in range(16): # Her bir harfin feature içerisindeki her bir değer aralığın
            print(f"Harf: {x} Deger {y}")
            if(np.isin(values,y).sum()):
                # Eğer değer aralığı içinde var ise toplam kaç adet olduğuna göre hesapl
                # Örneğin A için F1=15 yok. Bu nedenle harfte F1 için 15'in oluşma sayı
```

```
                # Harf içerisinde oluşma sayısı 0 dan farklı olan feature'lar için oluşm
                result = (fi[fi.index == (x,y)].values[0] + 1) / (letter_num[x] + 16)
            else:
                result =  (0 + 1) / (letter_num[x] + 16)
            print("Sonuç -> ", result)
            df.loc[len(df.index)] = [x,features[i],y,result]
        print("----------------------------")
```

```
FEATURE F1
Harfin feature içerisindeki değer aralığı
 [1 2 3 4 5 6 7 8 9]
Harf: A Deger 0
Sonuç ->  0.0015408320493066256
Harf: A Deger 1
Sonuç ->  0.07087827426810478
Harf: A Deger 2
Sonuç ->  0.2357473035439137
Harf: A Deger 3
Sonuç ->  0.325115562403698
Harf: A Deger 4
Sonuç ->  0.17257318952234207
Harf: A Deger 5
Sonuç ->  0.0847457627118644
Harf: A Deger 6
Sonuç ->  0.05238828967642527
Harf: A Deger 7
Sonuç ->  0.02773497688751926
Harf: A Deger 8
Sonuç ->  0.012326656394453005
Harf: A Deger 9
Sonuç ->  0.007704160246533128
Harf: A Deger 10
Sonuç ->  0.0015408320493066256
Harf: A Deger 11
Sonuç ->  0.0015408320493066256
Harf: A Deger 12
Sonuç ->  0.0015408320493066256
Harf: A Deger 13
Sonuç ->  0.0015408320493066256
Harf: A Deger 14
Sonuç ->  0.0015408320493066256
Harf: A Deger 15
Sonuç ->  0.0015408320493066256
----------------------------
Harfin feature içerisindeki değer aralığı
 [ 0  1  2  3  4  5  6  7  8  9 10 11]
Harf: B Deger 0
Sonuç ->  0.0030959752321981426
Harf: B Deger 1
Sonuç ->  0.06037151702786378
Harf: B Deger 2
Sonuç ->  0.13312693498452013
Harf: B Deger 3
Sonuç ->  0.21517027863777088
Harf: B Deger 4
Sonuç ->  0.24303405572755418
Harf: B Deger 5
Sonuç ->  0.1609907120743034
Harf: B Deger 6
Sonuç ->  0.08204334365325078
Harf: B Deger 7
Sonuç ->  0.04179566563467492
Harf: B Deger 8
Sonuç ->  0.030959752321981424
Harf: B Deger 9
Sonuç ->  0.01393188854489164
```

```
Harf: B Deger 10
Sonuç ->  0.0030959752321981426
Harf: B Deger 11
Sonuç ->  0.006191950464396285
Harf: B Deger 12
Sonuç ->  0.0015479876160990713
Harf: B Deger 13
Sonuç ->  0.0015479876160990713
Harf: B Deger 14
Sonuç ->  0.0015479876160990713
Harf: B Deger 15
Sonuç ->  0.0015479876160990713
----------------------------
Harfin feature içerisindeki değer aralığı
 [ 0  1  2  3  4  5  6  7  8  9 10]
Harf: C Deger 0
Sonuç ->  0.003278688524590164
Harf: C Deger 1
Sonuç ->  0.054098360655737705
Harf: C Deger 2
Sonuç ->  0.16229508196721312
Harf: C Deger 3
Sonuç ->  0.1819672131147541
Harf: C Deger 4
Sonuç ->  0.22131147540983606
Harf: C Deger 5
Sonuç ->  0.16229508196721312
Harf: C Deger 6
Sonuç ->  0.10655737704918032
Harf: C Deger 7
Sonuç ->  0.054098360655737705
Harf: C Deger 8
Sonuç ->  0.02459016393442623
Harf: C Deger 9
Sonuç ->  0.018032786885245903
Harf: C Deger 10
Sonuç ->  0.003278688524590164
Harf: C Deger 11
Sonuç ->  0.001639344262295082
Harf: C Deger 12
Sonuç ->  0.001639344262295082
Harf: C Deger 13
Sonuç ->  0.001639344262295082
Harf: C Deger 14
Sonuç ->  0.001639344262295082
Harf: C Deger 15
Sonuç ->  0.001639344262295082
----------------------------
Harfin feature içerisindeki değer aralığı
 [ 1  2  3  4  5  6  7  8  9 10]
Harf: D Deger 0
Sonuç ->  0.0015290519877675841
Harf: D Deger 1
Sonuç ->  0.03669724770642202
Harf: D Deger 2
Sonuç ->  0.13914373088685014
Harf: D Deger 3
Sonuç ->  0.1926605504587156
Harf: D Deger 4
Sonuç ->  0.25229357798165136
Harf: D Deger 5
Sonuç ->  0.18501529051987767
Harf: D Deger 6
Sonuç ->  0.10856269113149847
Harf: D Deger 7
Sonuç ->  0.03363914373088685
```

```
Harf: Y Deger 2
Sonuç ->  0.0015220700152207
Harf: Y Deger 3
Sonuç ->  0.060882800608828
Harf: Y Deger 4
Sonuç ->  0.1659056316590563
Harf: Y Deger 5
Sonuç ->  0.1689497716894977
Harf: Y Deger 6
Sonuç ->  0.0730593607305936
Harf: Y Deger 7
Sonuç ->  0.0715372907153729
Harf: Y Deger 8
Sonuç ->  0.410958904109589
Harf: Y Deger 9
Sonuç ->  0.0289193302891933
Harf: Y Deger 10
Sonuç ->  0.0060882800608828
Harf: Y Deger 11
Sonuç ->  0.0015220700152207
Harf: Y Deger 12
Sonuç ->  0.0030441400304414
Harf: Y Deger 13
Sonuç ->  0.0015220700152207
Harf: Y Deger 14
Sonuç ->  0.0015220700152207
Harf: Y Deger 15
Sonuç ->  0.0015220700152207
----------------------------
Harfin feature içerisindeki değer aralığı
 [ 3  4  5  6  7  8  9 10 11 12]
Harf: Z Deger 0
Sonuç ->  0.0016891891891891893
Harf: Z Deger 1
Sonuç ->  0.0016891891891891893
Harf: Z Deger 2
Sonuç ->  0.0016891891891891893
Harf: Z Deger 3
Sonuç ->  0.006756756756756757
Harf: Z Deger 4
Sonuç ->  0.018581081081081082
Harf: Z Deger 5
Sonuç ->  0.05067567567567568
Harf: Z Deger 6
Sonuç ->  0.06418918918918919
Harf: Z Deger 7
Sonuç ->  0.18074324324324326
Harf: Z Deger 8
Sonuç ->  0.5405405405405406
Harf: Z Deger 9
Sonuç ->  0.0777027027027027
Harf: Z Deger 10
Sonuç ->  0.02533783783783784
Harf: Z Deger 11
Sonuç ->  0.016891891891891893
Harf: Z Deger 12
Sonuç ->  0.008445945945945946
Harf: Z Deger 13
Sonuç ->  0.0016891891891891893
Harf: Z Deger 14
Sonuç ->  0.0016891891891891893
Harf: Z Deger 15
Sonuç ->  0.0016891891891891893
----------------------------
```

In [15]:
```
df.head()
```

Out[15]:

	Letter	Feature	Value	Prob
0	A	F1	0	0.001541
1	A	F1	1	0.070878
2	A	F1	2	0.235747
3	A	F1	3	0.325116
4	A	F1	4	0.172573

In [16]:
```
df.tail()
```

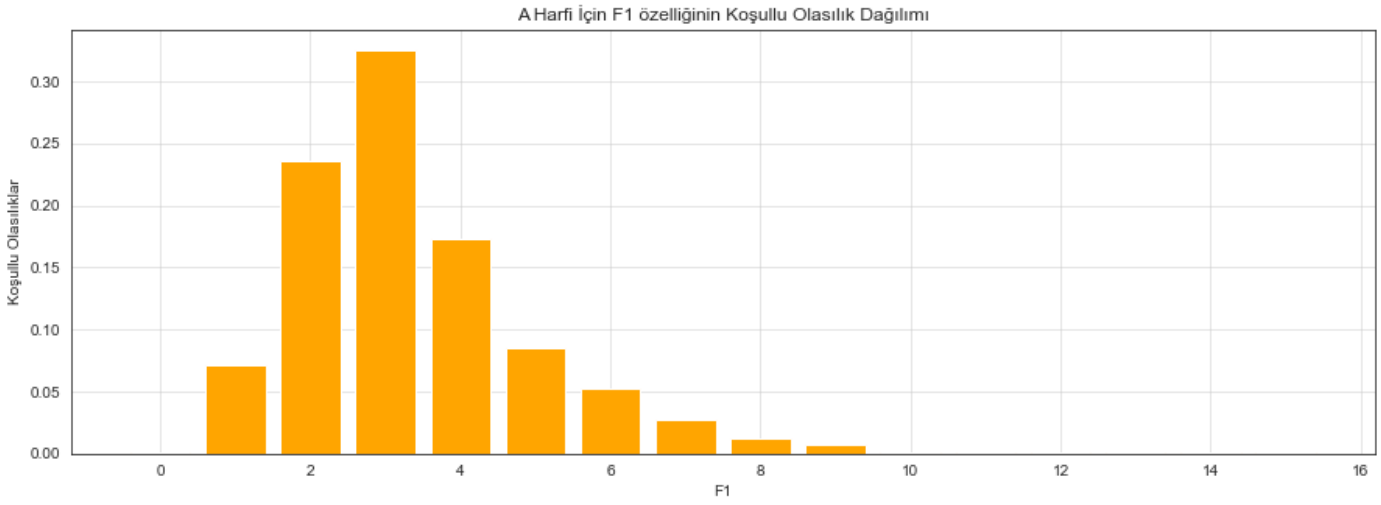
Out[16]:

	Letter	Feature	Value	Prob
6651	Z	F16	11	0.016892
6652	Z	F16	12	0.008446
6653	Z	F16	13	0.001689
6654	Z	F16	14	0.001689
6655	Z	F16	15	0.001689

In [17]:
```
df[(df["Letter"] == "A") & (df["Feature"] == "F1")].Prob.values
```

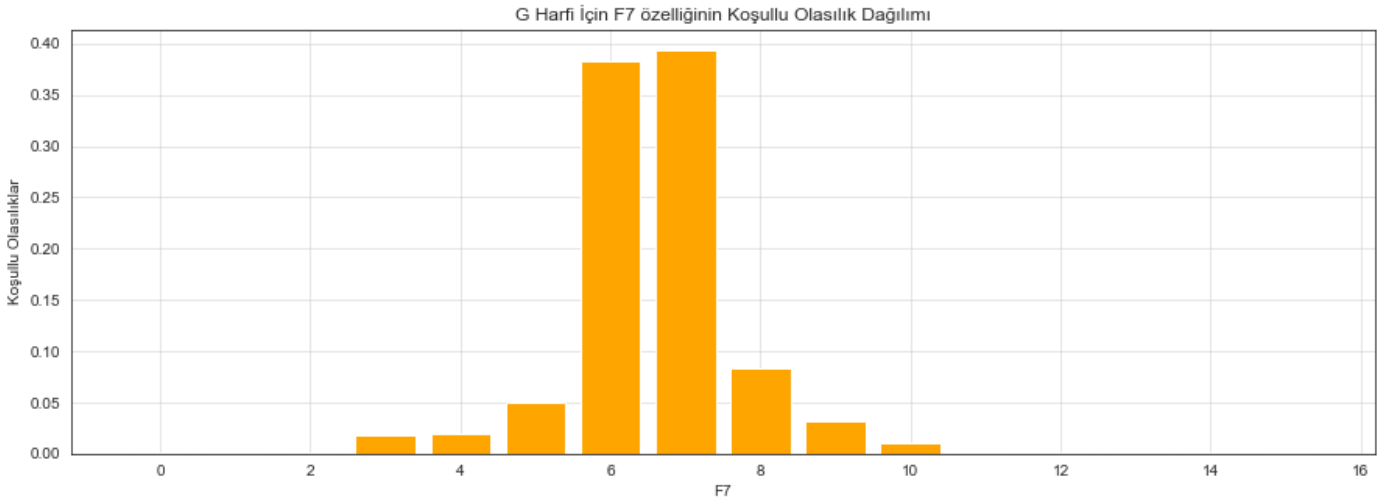
Out[17]:
```
array([0.00154083, 0.07087827, 0.2357473 , 0.32511556, 0.17257319,
       0.08474576, 0.05238829, 0.02773498, 0.01232666, 0.00770416,
       0.00154083, 0.00154083, 0.00154083, 0.00154083, 0.00154083,
       0.00154083])
```

In [18]:
```
# Örnek görselleştirme
plt.figure(figsize=(15,5 ))
plt.bar(range(16), df[(df["Letter"] == "A") & (df["Feature"] == "F1")].Prob.values, colo
plt.xlabel("F1")
plt.ylabel("Koşullu Olasılıklar")
plt.title("A Harfi İçin F1 özelliğinin Koşullu Olasılık Dağılımı")
plt.grid(True,alpha=0.6)
plt.show()
```



In [19]:
```
# Örnek görselleştirme - 2
plt.figure(figsize=(15,5 ))
plt.bar(range(16), df[(df["Letter"] == "G") & (df["Feature"] == "F7")].Prob.values, colo
plt.xlabel("F7")
plt.ylabel("Koşullu Olasılıklar")
```

```
plt.title("G Harfi İçin F7 özelliğinin Koşullu Olasılık Dağılımı")
plt.grid(True,alpha=0.6)
plt.show()
```



Test Verisi Üzerinden Tahminde Bulunma

In [20]:
```
predictions = pd.DataFrame(columns=["Real_Val","Prediction","Accuracy"]) # Tahminlerin s
for i in range(len(test)): # Test verisi üzerindeki her bir veri döngü ile test edildi
    total = 0
    maximum = 0
    probs = [] # Argmax almak için her bir harfin olasılık sonuçları diziye aktarıldı
    print("------------------------------------------------------------")
    print(test.values[i])
    for j in letter_unique: # Her bir test verisi için 26 harf test edildi ve olasılıksa
        total = 0
        for k in range(1,num_features):
            total += df[(df["Letter"] == j) & (df["Feature"] == features[k]) & (df["Valu
        total += np.log10(letter_num[test.values[i][0]] / sum(letter_num.values))
        probs.append(total)
    #print("probs ->",probs)
    predictions.loc[len(predictions.index)] = [test.values[i][0],letter_unique[np.argmax
    # Diziye aktarılan değerler arasındaki maksimum orana sahip olan değer tahmin olarak
    print("İşlenen Veri -> ", (i+1), " || Max -> ", probs[np.argmax(probs)] , "Harf ->",
```

```
------------------------------------------------------------
['U' 4 10 6 7 9 9 6 4 3 6 7 7 9 8 5 6]
İşlenen Veri ->  1  || Max ->  1.4165852428152563 Harf -> B
------------------------------------------------------------
['N' 6 9 8 4 3 8 7 3 4 13 5 8 6 8 0 8]
İşlenen Veri ->  2  || Max ->  2.823788943753555 Harf -> I
------------------------------------------------------------
['V' 6 9 8 8 10 7 7 5 4 7 6 8 7 9 7 10]
İşlenen Veri ->  3  || Max ->  2.649945271642327 Harf -> I
------------------------------------------------------------
['I' 5 6 6 4 3 7 6 2 7 7 6 9 0 9 4 8]
İşlenen Veri ->  4  || Max ->  3.1789432501083326 Harf -> I
------------------------------------------------------------
['N' 5 9 7 6 4 9 7 3 5 10 4 6 5 8 1 7]
İşlenen Veri ->  5  || Max ->  2.31602300128253 Harf -> N
------------------------------------------------------------
['H' 5 8 8 6 6 5 8 3 6 10 8 8 4 8 4 6]
İşlenen Veri ->  6  || Max ->  2.0506971530713702 Harf -> H
------------------------------------------------------------
['E' 0 0 1 0 0 5 7 5 6 7 6 12 0 8 6 10]
İşlenen Veri ->  7  || Max ->  2.9006422476603153 Harf -> I
------------------------------------------------------------
```

['Y' 1 0 2 0 0 7 10 3 1 7 12 8 1 11 0 8]
İşlenen Veri -> 8 || Max -> 2.5053255658887674 Harf -> Y

['G' 5 9 7 7 5 6 7 7 5 5 6 11 4 8 4 8]
İşlenen Veri -> 9 || Max -> 2.6344655639452044 Harf -> O

['E' 2 4 4 3 2 7 8 2 8 11 7 9 2 8 4 8]
İşlenen Veri -> 10 || Max -> 2.867106299128754 Harf -> E

['N' 5 4 5 6 2 7 7 15 2 4 6 8 6 8 0 8]
İşlenen Veri -> 11 || Max -> 3.647221286987878 Harf -> I

['B' 1 3 2 1 1 8 7 3 4 10 5 7 1 8 3 9]
İşlenen Veri -> 12 || Max -> 2.0286570993053967 Harf -> B

['G' 4 7 5 5 3 7 6 7 7 7 5 11 2 9 4 8]
İşlenen Veri -> 13 || Max -> 2.505297309976951 Harf -> G

['L' 4 9 4 6 2 0 2 4 6 1 0 7 0 8 0 8]
İşlenen Veri -> 14 || Max -> 2.283187583027914 Harf -> I

['E' 4 7 6 5 5 8 9 7 3 6 6 11 4 7 7 10]
İşlenen Veri -> 15 || Max -> 1.5838759434546175 Harf -> Q

['G' 6 8 8 7 9 7 6 6 4 7 7 9 10 8 9 9]
İşlenen Veri -> 16 || Max -> 1.7972694618756844 Harf -> E

['M' 4 10 6 8 6 7 6 6 5 7 7 9 8 5 2 8]
İşlenen Veri -> 17 || Max -> 2.842394782250813 Harf -> M

['D' 2 4 4 3 3 9 7 4 6 10 4 6 2 8 3 8]
İşlenen Veri -> 18 || Max -> 3.297948402487256 Harf -> D

['Y' 6 7 6 5 2 3 12 5 5 13 12 6 2 11 2 6]
İşlenen Veri -> 19 || Max -> 1.5839292694334248 Harf -> F

['R' 5 5 5 6 3 5 12 8 3 7 2 9 3 7 6 11]
İşlenen Veri -> 20 || Max -> 2.0335182962711604 Harf -> R

['P' 4 7 6 10 9 8 11 5 0 9 7 6 4 10 5 8]
İşlenen Veri -> 21 || Max -> 1.3728472910231484 Harf -> P

['D' 3 9 5 7 5 8 7 5 7 7 6 5 3 8 3 7]
İşlenen Veri -> 22 || Max -> 3.340761858144748 Harf -> D

['E' 4 9 4 7 3 3 6 6 12 7 7 15 0 8 7 7]
İşlenen Veri -> 23 || Max -> 2.276916425711032 Harf -> E

['W' 4 5 6 4 3 7 11 2 3 7 9 8 7 11 1 8]
İşlenen Veri -> 24 || Max -> 3.103213305852951 Harf -> W

['D' 4 2 5 4 3 7 7 7 7 6 6 5 2 8 3 7]
İşlenen Veri -> 25 || Max -> 3.10528785202854 Harf -> D

['Q' 8 13 7 7 4 7 5 4 8 10 5 9 3 7 9 9]
İşlenen Veri -> 26 || Max -> 1.2013946110966536 Harf -> R

['R' 4 7 5 5 5 7 8 5 6 6 4 8 3 6 5 9]
İşlenen Veri -> 27 || Max -> 2.2112533468039954 Harf -> H

['G' 4 6 4 4 3 6 7 5 5 9 7 10 2 9 4 10]
İşlenen Veri -> 28 || Max -> 2.89889730997695 Harf -> G

['Y' 3 7 4 5 2 8 10 2 2 6 12 8 2 11 0 8]
İşlenen Veri -> 29 || Max -> 2.7366802082023147 Harf -> Y

['Q' 3 5 4 6 4 8 9 4 2 5 8 11 2 10 5 8]
İşlenen Veri -> 3968 || Max -> 1.9543203505107967 Harf -> V

['C' 6 9 6 7 4 4 7 5 7 10 9 14 4 9 5 5]
İşlenen Veri -> 3969 || Max -> 1.2580174456838165 Harf -> U

['S' 4 9 6 6 7 9 4 4 4 9 6 9 4 7 10 9]
İşlenen Veri -> 3970 || Max -> 1.1942081675313305 Harf -> R

['F' 7 10 9 8 7 9 7 2 6 12 4 6 5 9 4 9]
İşlenen Veri -> 3971 || Max -> 1.257866120689328 Harf -> D

['C' 5 10 7 9 8 5 6 4 4 7 6 11 5 11 8 10]
İşlenen Veri -> 3972 || Max -> 0.9686812406503924 Harf -> K

['V' 4 7 6 5 6 8 6 4 2 7 8 8 7 9 4 6]
İşlenen Veri -> 3973 || Max -> 2.153458103052654 Harf -> W

['T' 4 4 5 3 2 5 12 2 8 11 9 4 0 10 2 4]
İşlenen Veri -> 3974 || Max -> 1.5146681737342598 Harf -> T

['N' 5 9 5 4 2 9 11 5 3 5 6 9 5 11 2 6]
İşlenen Veri -> 3975 || Max -> 1.246743567983101 Harf -> Y

['E' 1 0 1 0 0 5 8 5 7 7 6 12 0 8 6 10]
İşlenen Veri -> 3976 || Max -> 2.8395861420497543 Harf -> I

['L' 3 8 3 6 2 0 2 4 6 1 0 8 0 8 0 8]
İşlenen Veri -> 3977 || Max -> 2.824441708440455 Harf -> I

['A' 3 9 5 6 2 6 5 3 1 6 1 8 2 7 2 7]
İşlenen Veri -> 3978 || Max -> 2.8530618475463303 Harf -> A

['K' 5 11 5 8 5 3 8 7 3 6 4 11 3 8 2 11]
İşlenen Veri -> 3979 || Max -> 1.8743534299201599 Harf -> K

['M' 6 9 10 7 12 7 5 3 2 7 5 8 15 7 4 6]
İşlenen Veri -> 3980 || Max -> 1.4839157911612462 Harf -> H

['R' 2 3 3 2 2 7 7 5 5 7 5 6 2 7 4 8]
İşlenen Veri -> 3981 || Max -> 2.53693371355281 Harf -> O

['S' 6 12 6 7 3 6 8 3 6 13 7 7 2 9 3 7]
İşlenen Veri -> 3982 || Max -> 1.5136984464605423 Harf -> C

['Y' 3 9 5 6 3 7 9 1 6 6 11 8 2 11 2 7]
İşlenen Veri -> 3983 || Max -> 2.1902570727380826 Harf -> Y

['V' 7 10 5 5 2 6 11 5 4 11 9 4 4 11 3 10]
İşlenen Veri -> 3984 || Max -> 1.160659937578967 Harf -> P

['S' 2 0 2 1 1 8 7 4 6 5 6 8 0 8 7 8]
İşlenen Veri -> 3985 || Max -> 3.9539570624860794 Harf -> I

['M' 5 6 8 4 5 9 6 2 4 9 5 7 8 6 2 8]
İşlenen Veri -> 3986 || Max -> 2.3288405653833433 Harf -> M

['O' 9 15 6 8 5 5 7 7 4 10 7 10 5 9 5 8]
İşlenen Veri -> 3987 || Max -> 1.6903975948344494 Harf -> O

['L' 3 7 3 5 1 0 1 6 6 0 0 6 0 8 0 8]
İşlenen Veri -> 3988 || Max -> 2.368852439836175 Harf -> L

['D' 6 9 8 8 8 7 6 5 7 7 5 9 6 5 10 3]
İşlenen Veri -> 3989 || Max -> 1.060038045462828 Harf -> M

```
['P' 2 1 3 2 1 4 10 3 5 10 8 5 0 9 3 7]
İşlenen Veri ->  3990  || Max ->  1.6503778805670852 Harf -> F
------------------------------------------------------------
['W' 3 8 5 6 5 11 11 2 2 5 8 7 7 12 1 7]
İşlenen Veri ->  3991  || Max ->  1.721655277236099 Harf -> W
------------------------------------------------------------
['O' 4 3 5 4 2 7 6 8 8 6 5 7 3 8 4 8]
İşlenen Veri ->  3992  || Max ->  3.0142071186439736 Harf -> O
------------------------------------------------------------
['E' 4 9 5 6 3 5 9 2 10 10 8 9 2 8 5 5]
İşlenen Veri ->  3993  || Max ->  1.8164733877363481 Harf -> E
------------------------------------------------------------
['J' 2 11 3 8 2 15 4 4 5 13 1 8 0 7 0 8]
İşlenen Veri ->  3994  || Max ->  2.3554850575551685 Harf -> I
------------------------------------------------------------
['T' 5 8 7 7 7 7 9 4 8 7 7 8 3 10 8 6]
İşlenen Veri ->  3995  || Max ->  1.8675432378224146 Harf -> H
------------------------------------------------------------
['D' 2 2 3 3 2 7 7 7 6 6 6 4 2 8 3 7]
İşlenen Veri ->  3996  || Max ->  2.6229229182874594 Harf -> O
------------------------------------------------------------
['C' 7 10 8 8 4 4 8 6 9 12 9 13 2 9 3 7]
İşlenen Veri ->  3997  || Max ->  1.6532730197023178 Harf -> C
------------------------------------------------------------
['T' 6 9 6 7 5 6 11 3 7 11 9 5 2 12 2 4]
İşlenen Veri ->  3998  || Max ->  1.6614155262304782 Harf -> T
------------------------------------------------------------
['S' 2 3 4 2 1 8 7 2 6 10 6 8 1 9 5 8]
İşlenen Veri ->  3999  || Max ->  2.5864775780511495 Harf -> Z
------------------------------------------------------------
['A' 4 9 6 6 2 9 5 3 1 8 1 8 2 7 2 8]
İşlenen Veri ->  4000  || Max ->  2.8176227104122775 Harf -> A
```

In [21]:
```
predictions.head(20)
```

Out[21]:

	Real_Val	Prediction	Accuracy
0	U	B	False
1	N	I	False
2	V	I	False
3	I	I	True
4	N	N	True
5	H	H	True
6	E	I	False
7	Y	Y	True
8	G	O	False
9	E	E	True
10	N	I	False
11	B	B	True
12	G	G	True
13	L	I	False
14	E	Q	False
15	G	E	False
16	M	M	True

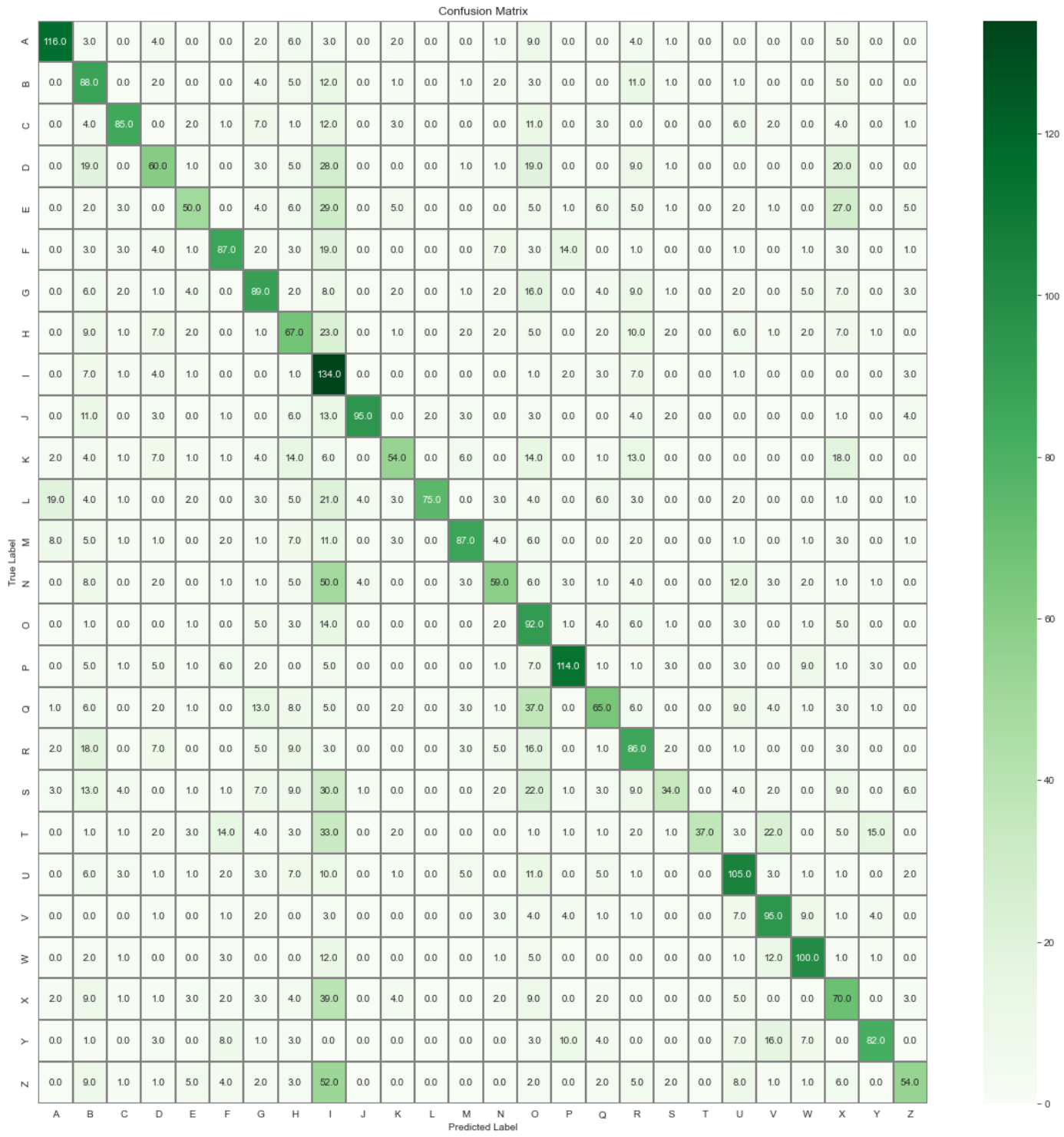
17	D	D	True
18	Y	F	False
19	R	R	True

In [25]:
```
# Toplam tahmin başarı oranı
print("Başarı Yüzdesi -> ", predictions.Accuracy.values.sum() / len(predictions))
```

```
Başarı Yüzdesi ->  0.52
```

Karmaşıklık Matrisi

In [23]:
```
# Confusiın Matrix
confusion_mtx = confusion_matrix(predictions["Real_Val"], predictions["Prediction"])
# plot the confusion matrix
f,ax = plt.subplots(figsize=(20, 20))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01,cmap="Greens",linecolor="gray", f
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```


Confusion Matrix
True Label
Predicted Label
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
0 20 40 60 80 100 120

In [ ]: