

TARIFBOT

YAPAY ZEKA DESTEKLİ YEMEK TARIF CHATBOT'U

220601054 -BEGÜM KELEŞ

210601017 -SUEDA AHSEN GÖKALP

VERİSETİ

Bu veri seti, toplam 473 farklı yemek tarifini içermektedir. Her tarif; yemek adı, malzemeler, hazırlanış açıklaması ve kategori bilgisi gibi metin temelli alanlardan oluşmaktadır. Türk mutfağına özgü çeşitli yemekleri barındıran bu veri seti, doğal dil işleme (NLP) ve makine öğrenimi (ML) projeleri için uygundur.

SÜTUNLAR:

- 1 foodname: Tarifin adı
- 2 foodcategory: Ham kategori metinleri (ör. "Ege - Zeytinyağlı", "Ana Yemek - Kebap" gibi)
- 3 materials: Virgülle ayrılmış malzemeler listesi
- 4 description: Tarifin adım adım açıklaması
- 5 toplam_sure_dk: Hazırlık ve pişirme süresinin toplamı (dakika)
- 6 porsiyon_sayisi: Tarifin kaç kişilik olduğu

ETIKETLEME TÜRLERİ VE ÖĞRENME YAKLASIMLARI

PROJEMIZDE KULLANILAN:

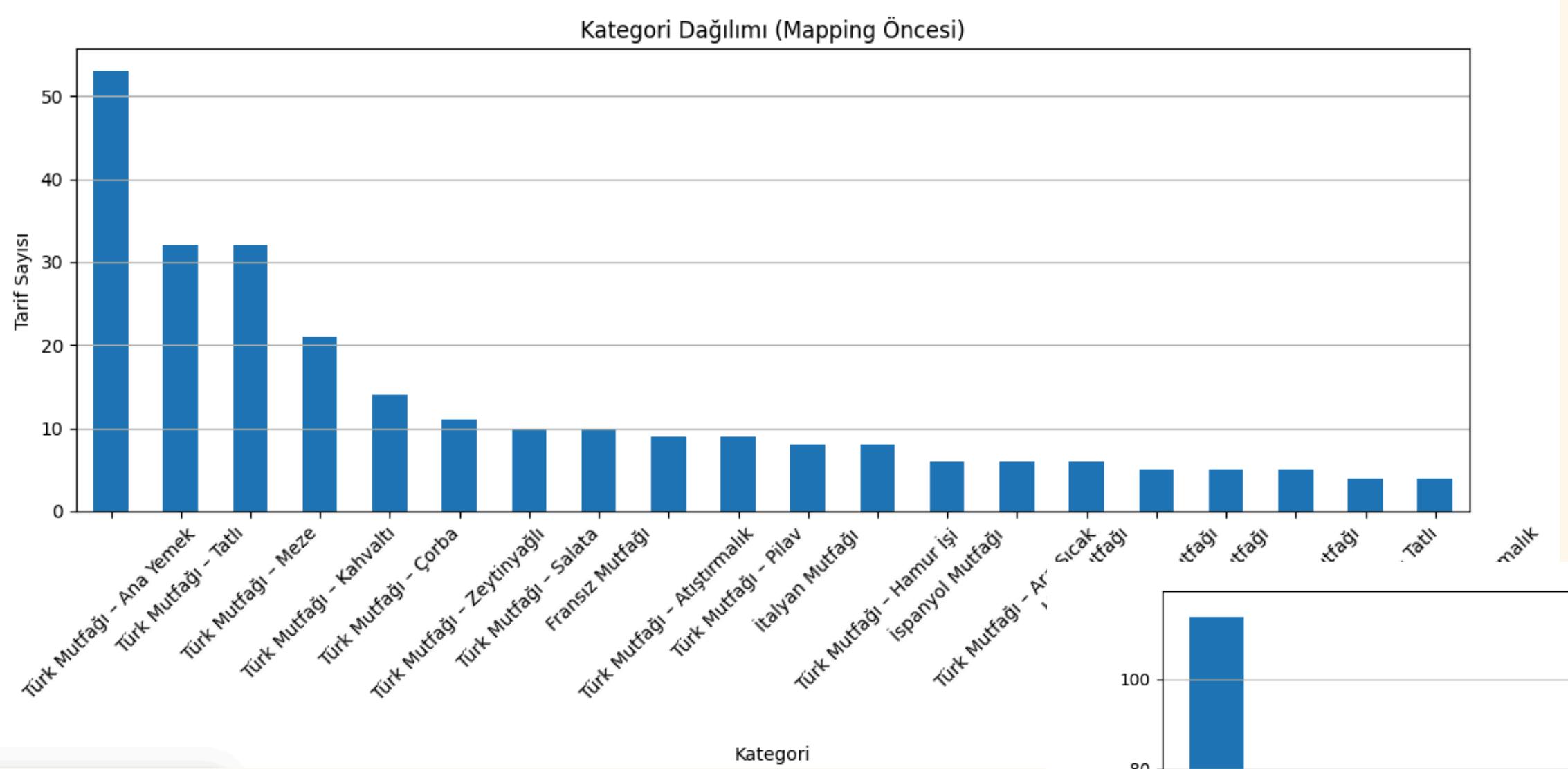
- Etiket: Tariflerin ait olduğu kategori bilgisi (Ana yemek, Tatlı, Corba, vb.)
- Yaklaşım: Denetimli Öğrenme – her tarifin sınıfı etiketli
- Amac: Kullanıcının verdiği malzemelere göre kategori tahmini yapmak

ETIKETLEME TÜRLERİ VE ÖĞRENME YAKLASIMLARI

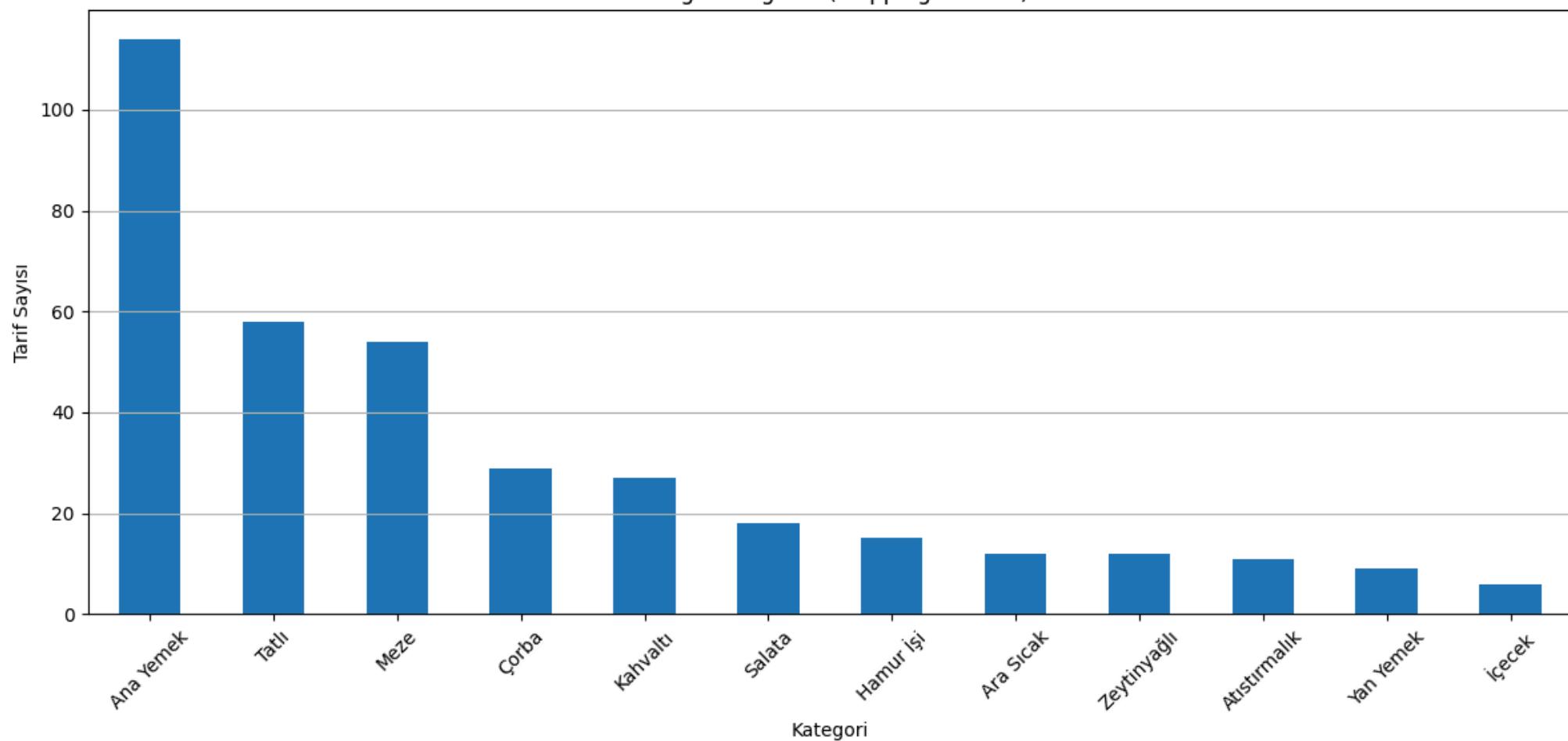


KATEGORİ DAGILIMI

Kategori Dağılımı (Mapping Öncesi)

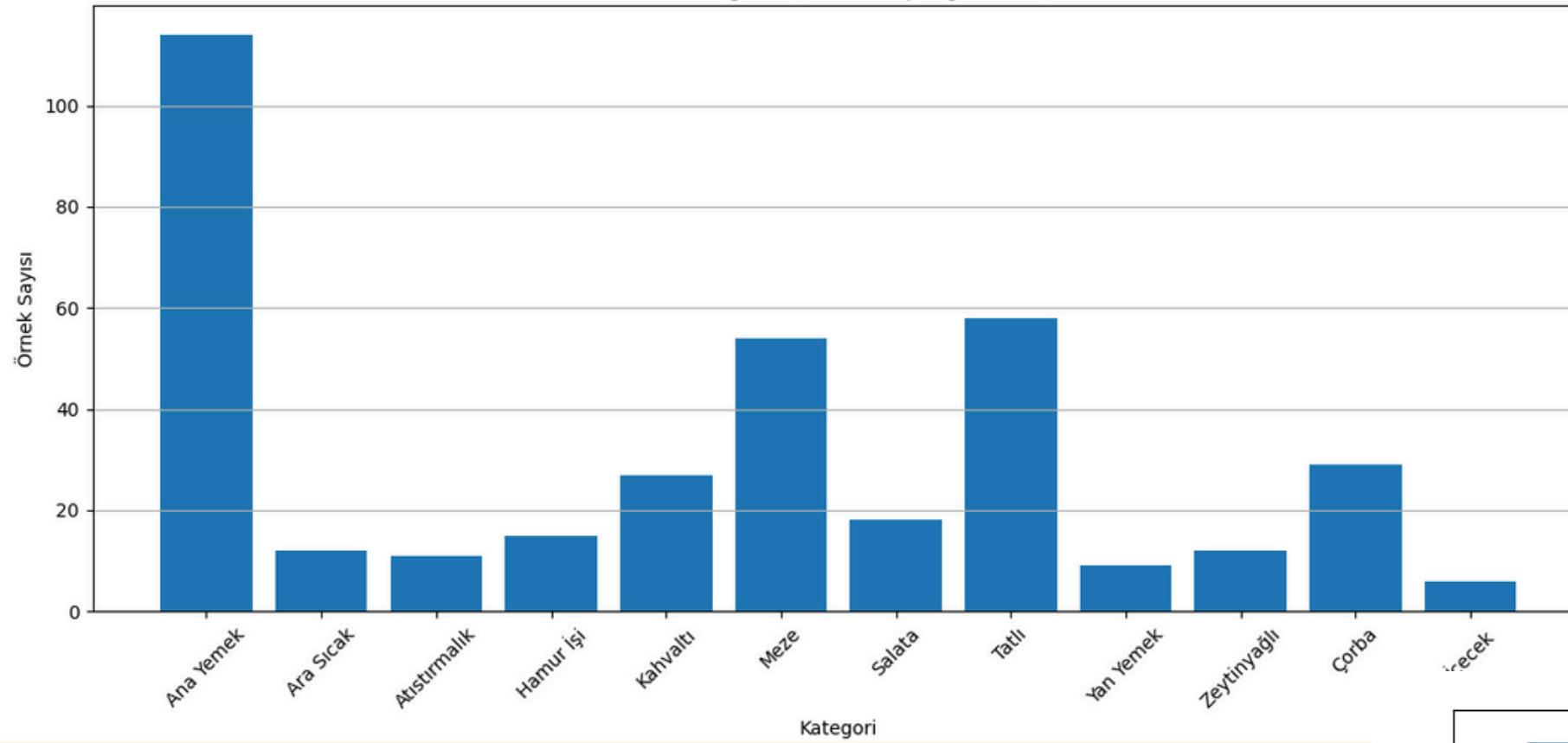


Kategori Dağılımı (Mapping Sonrası)

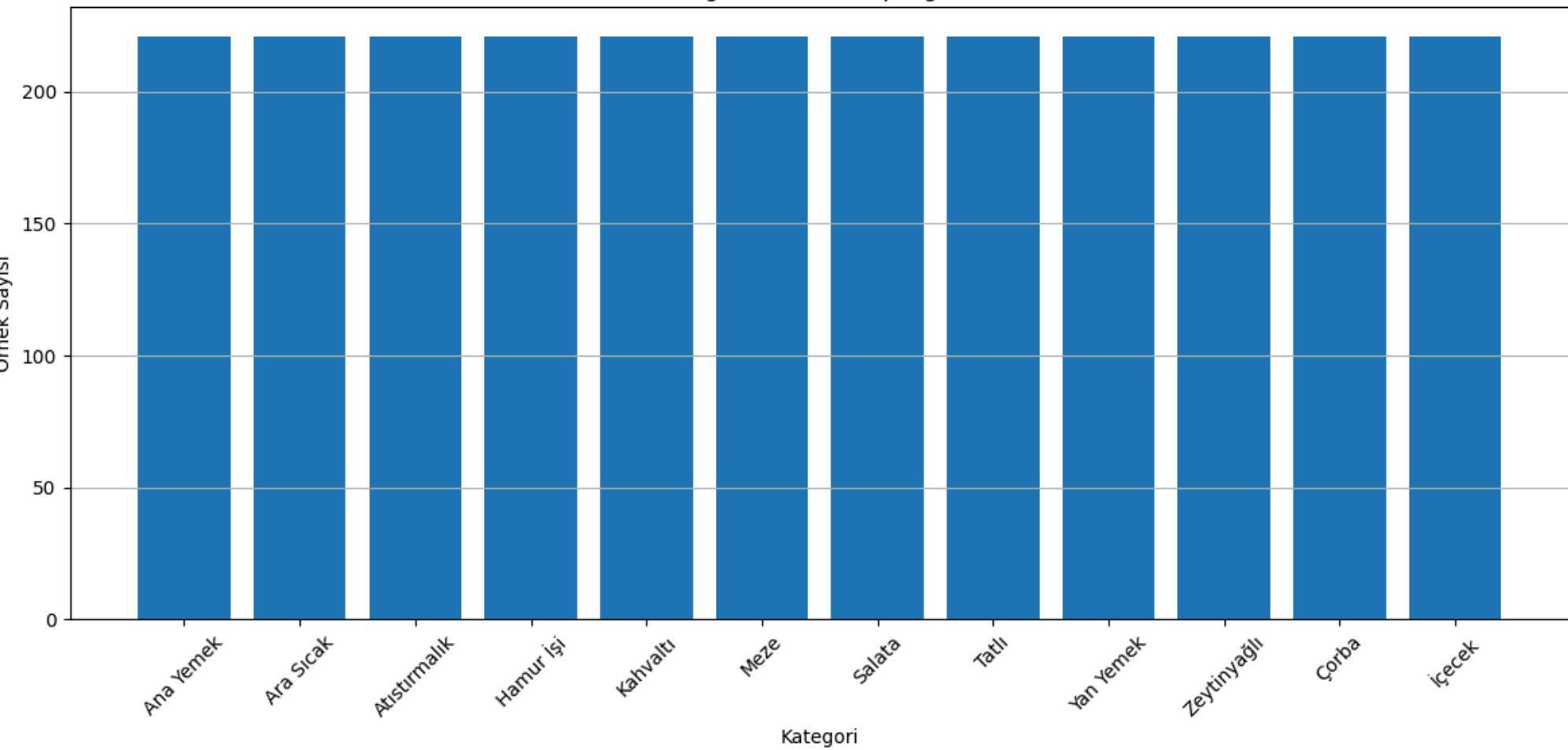


ETIKET DAGILIMI

Etiket Dağılımı (Oversampling Öncesi)



Etiket Dağılımı (Oversampling Sonrası)



VERİ ÖN İSLEME

```
mlb = MultiLabelBinarizer()
etiket_encoded = mlb.fit_transform(df["etiketler"])
df = pd.concat([df, pd.DataFrame(etiket_encoded, columns=mlb.classes_)], axis=1)

X_raw = df["materials"].fillna("")
y = df[mlb.classes_].values
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(X_raw)
```

Etiketler çoklu sınıfılı olduğu için MultiLabelBinarizer ile one-hot encoding yapılır.
materials (malzeme) metni boş olanlar boş string ile doldurulur.
TF-IDF yöntemiyle metin verisi sayısal vektöre çevrilir.

```
ros = RandomOverSampler(random_state=42)
X_ros, y_ros = ros.fit_resample(X, y)
X_train_ros, X_test_ros, y_train_ros, y_test_ros = train_test_split(X_ros, y_ros, test_size=0.2, random_state=42)
```

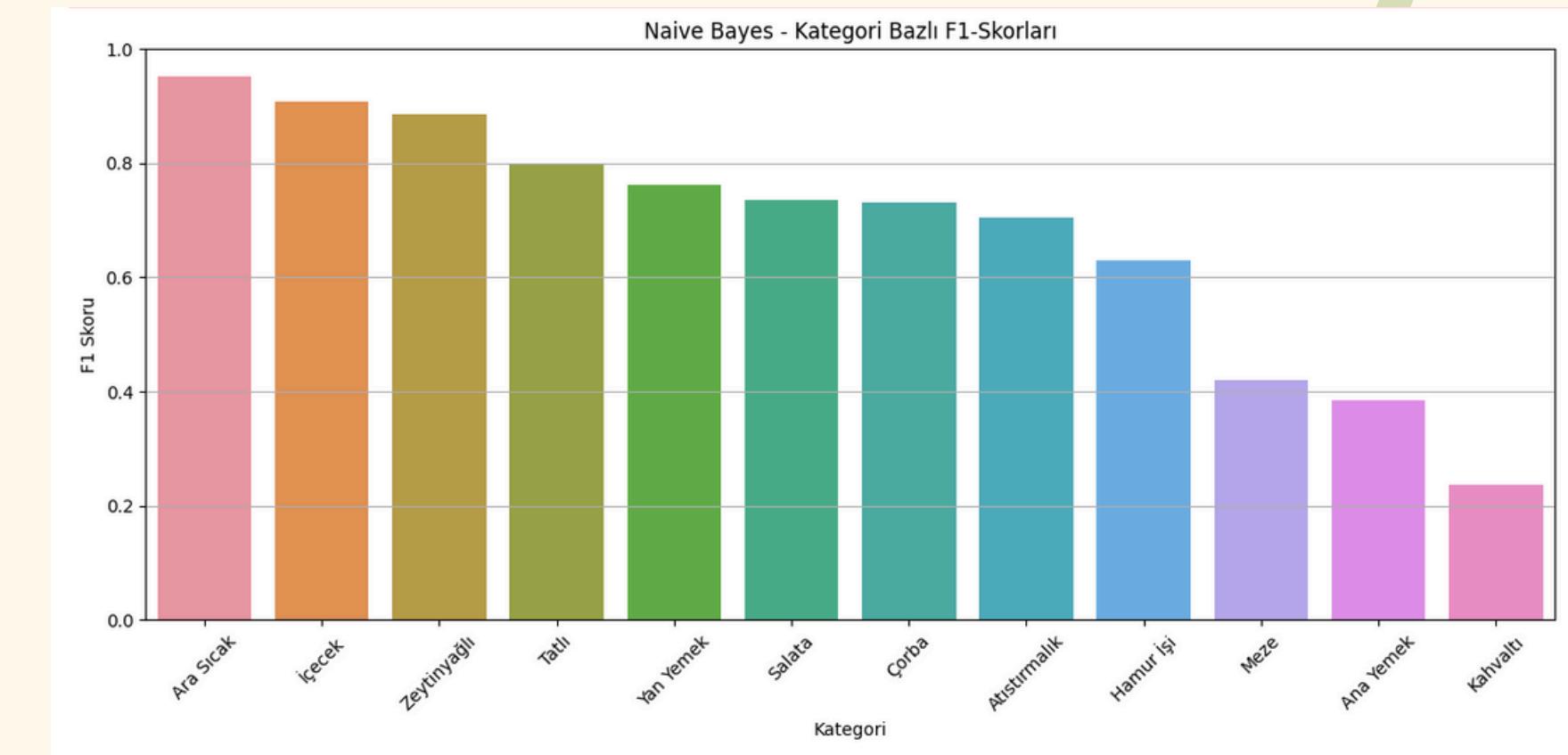
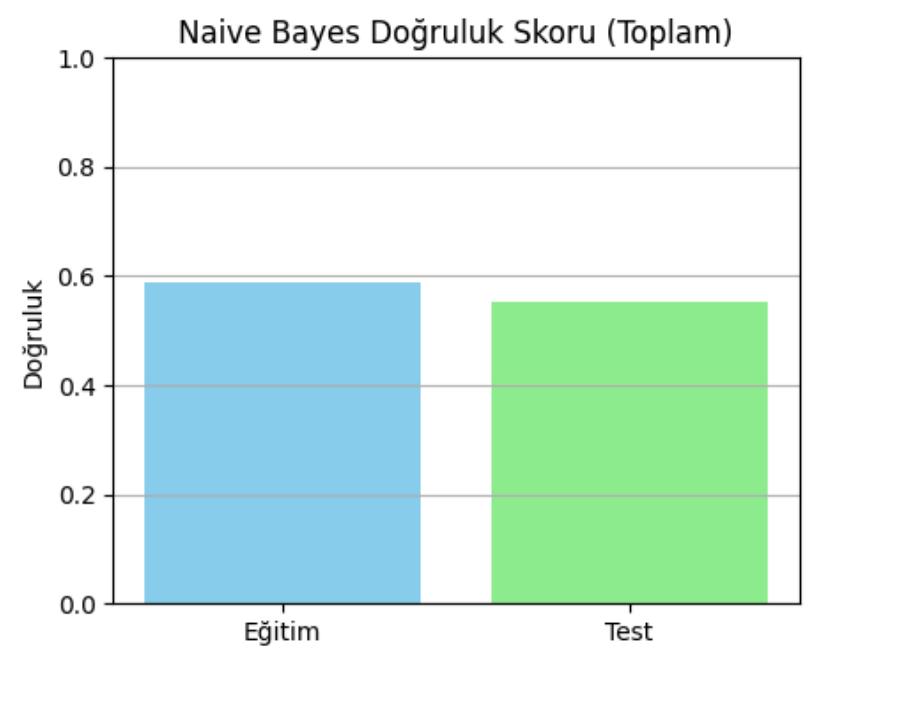
Veri dengesizliğini azaltmak için RandomOverSampler kullanılır. Eğitim ve test verisi bu dengelenmiş veriden ayrıılır.

NAIVE BAYES

Naive Bayes Sonuçları:

	precision	recall	f1-score	support
Ana Yemek	0.91	0.24	0.38	41
Ara Sıcak	0.98	0.93	0.95	43
Atıştırmalık	0.92	0.57	0.71	42
Hamur İşi	0.88	0.49	0.63	47
Kahvaltı	0.64	0.15	0.24	48
Meze	0.86	0.28	0.42	43
Salata	0.96	0.60	0.74	42
Tatlı	0.88	0.73	0.80	49
Yan Yemek	0.91	0.66	0.76	44
Zeytinyağlı	0.90	0.88	0.89	40
Çorba	0.92	0.61	0.73	56
İçecek	1.00	0.83	0.91	36
micro avg	0.91	0.57	0.71	531
macro avg	0.90	0.58	0.68	531
weighted avg	0.89	0.57	0.67	531
samples avg	0.56	0.57	0.57	531

Naive Bayes sınıflandırıcısı eğitilir ve test edilir. Çoklu etiket tahmini için OneVsRest yaklaşımı kullanılır.



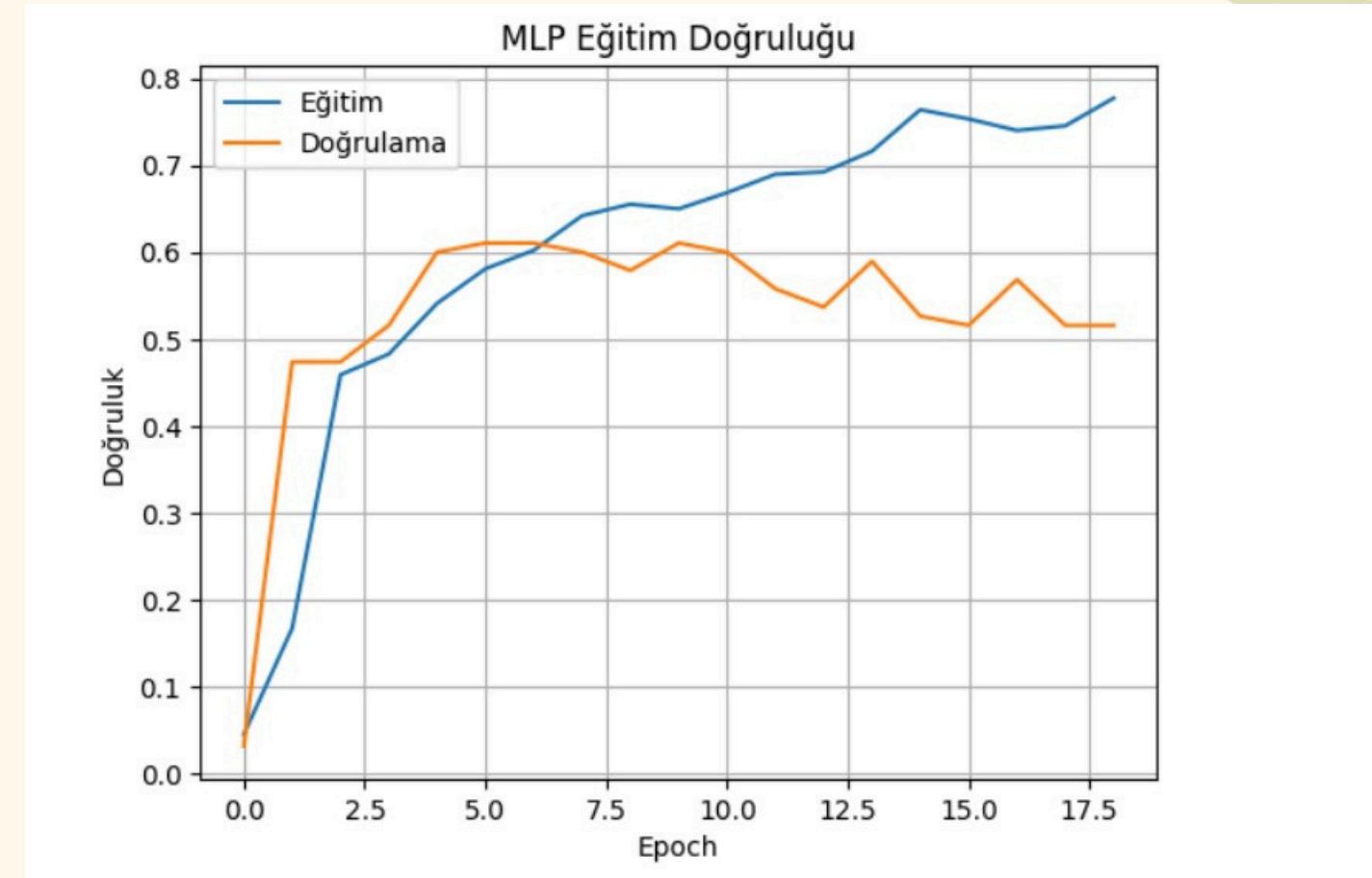
MLP MODELİ (OVERFITTING)

```
mlp_model = Sequential([
    Dense(256, activation='relu', input_shape=(X.shape[1],)),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(y.shape[1], activation='sigmoid')
])

mlp_model.compile(optimizer=Adam(learning_rate=0.001),
                   loss='binary_crossentropy',
                   metrics=['accuracy'])

early_stop = EarlyStopping(patience=5, restore_best_weights=True)

history = mlp_model.fit(
    X_train.toarray(), y_train,
    validation_data=(X_test.toarray(), y_test),
    epochs=30,
    batch_size=16,
    callbacks=[early_stop],
    verbose=1
)
```



Dropout + L2 Regularization ekleyerek
overfitting'i engelleyelim ->

MLP MODELİ (DROPOUT + L2 REGULARIZATION)

```
model = Sequential([
    Dense(256, activation='relu', input_shape=(X.shape[1],), kernel_regularizer=l2(0.001)),
    Dropout(0.5),
    Dense(128, activation='relu', kernel_regularizer=l2(0.001)),
    Dropout(0.5),
    Dense(y.shape[1], activation='sigmoid')
])

model.compile(optimizer=Adam(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics=['accuracy'])

early_stop = EarlyStopping(patience=5, restore_best_weights=True)

history_ros = model.fit(
    X_train_ros.toarray(), y_train_ros,
    validation_data=(X_test_ros.toarray(), y_test_ros),
    epochs=30,
    batch_size=16,
    callbacks=[early_stop],
    verbose=1
```

MLP (Multilayer

Perceptron) mimarisi
tanımlanıyor.

İki katmanlı relu + dropout
ile overfitting önleniyor.
Sigmoid çıkış çoklu sınıflar
için.
Model EarlyStopping ile
eğitiliyor.

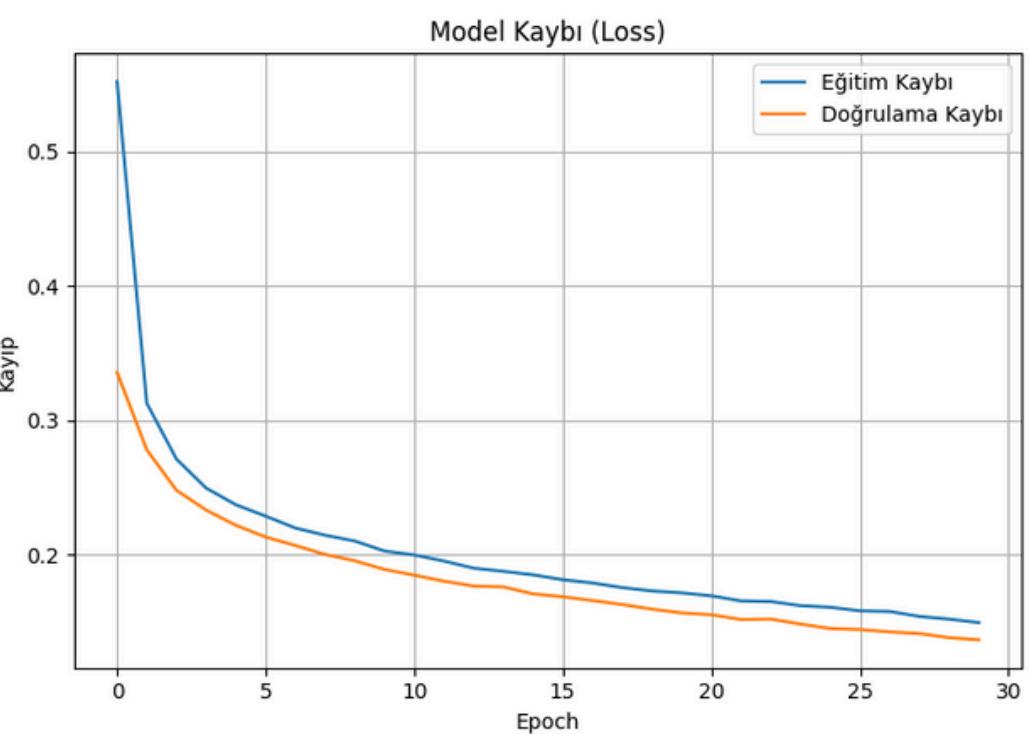
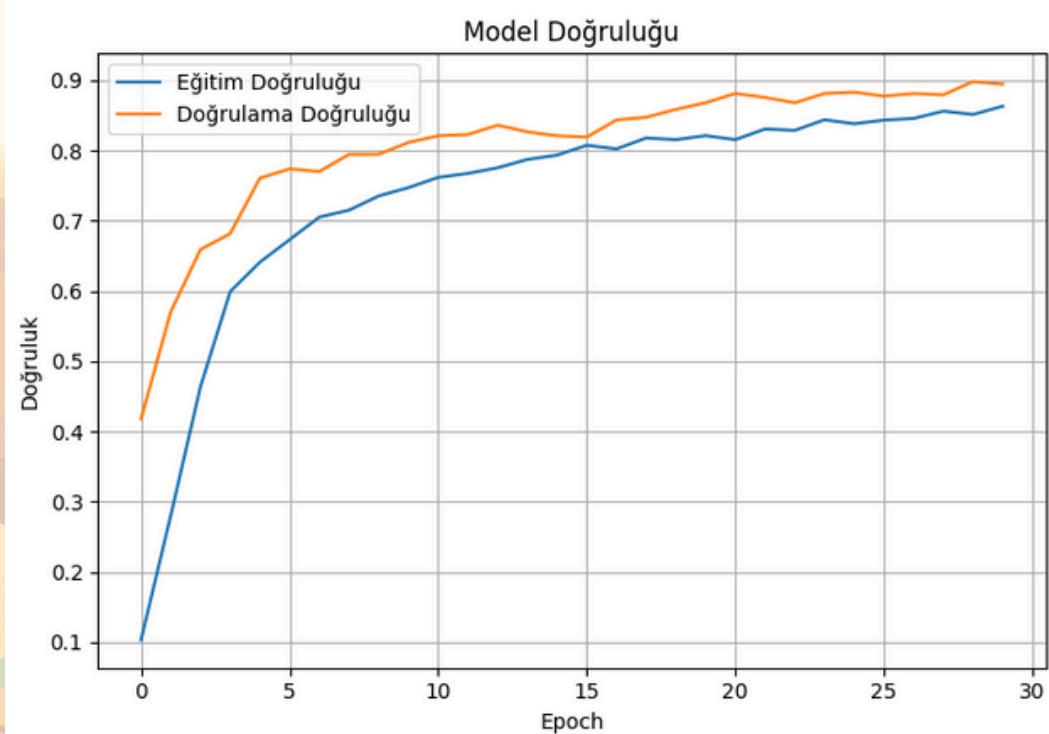
Dense (256 nöron, ReLU aktivasyon, L2
Reg = 0.001)

↓
Dropout (oran = 0.5)

↓
Dense (128 nöron, ReLU aktivasyon, L2
Reg = 0.001)

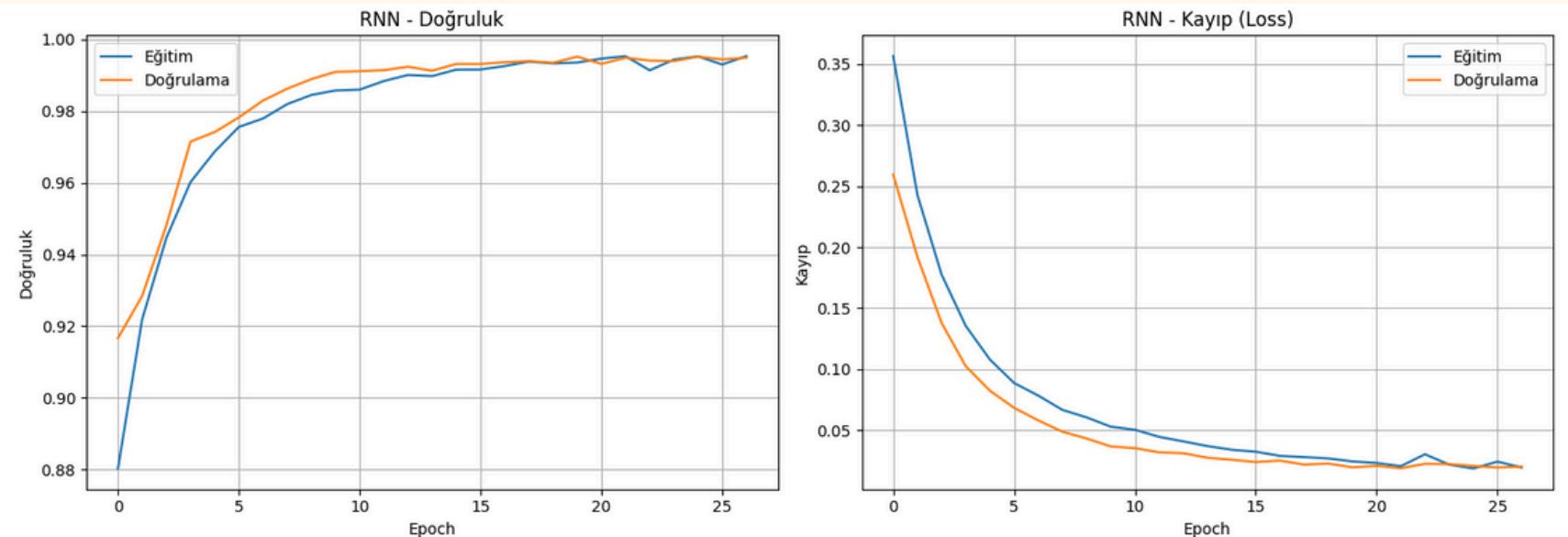
↓
Dropout (oran = 0.5)

Dense (Çıkış: y.shape[1] nöron, Sigmoid
aktivasyon)



RNN MODELİ

```
model2 = Sequential([
    Embedding(input_dim=max_words, output_dim=128, input_length=max_len),
    SimpleRNN(64),
    Dropout(0.5),
    Dense(y.shape[1], activation='sigmoid')
])
model2.compile(
    optimizer=Adam(1e-3),
    loss='binary_crossentropy',
    metrics=[BinaryAccuracy(name='accuracy')])
print("▣ Model-2 (RNN) eğitimi başladı...")
hist2 = model2.fit(
    X_tr_seq, y_tr_seq,
    validation_data=(X_te_seq, y_te_seq),
    epochs=30, batch_size=16,
    callbacks=[early],
    verbose=1)
# test doğruluğu
loss2, acc2 = model2.evaluate(X_te_seq, y_te_seq, verbose=0)
print(f"✓ Model-2 (RNN) test doğruluğu: {acc2:.4f}")
```



Bu model, gömülü
(embedding) katman
sonrası SimpleRNN
katmanı içeriyor.
Dropout overfitting'i
azaltır.

Çok etiketli sınıflama için
çıkışta sigmoid aktivasyon
var.

Eğitim sonrası test seti
Üzerinde doğruluk
hesaplanıyor.

Embedding
input_dim = 5000, output_dim = 128,
input_length = 100

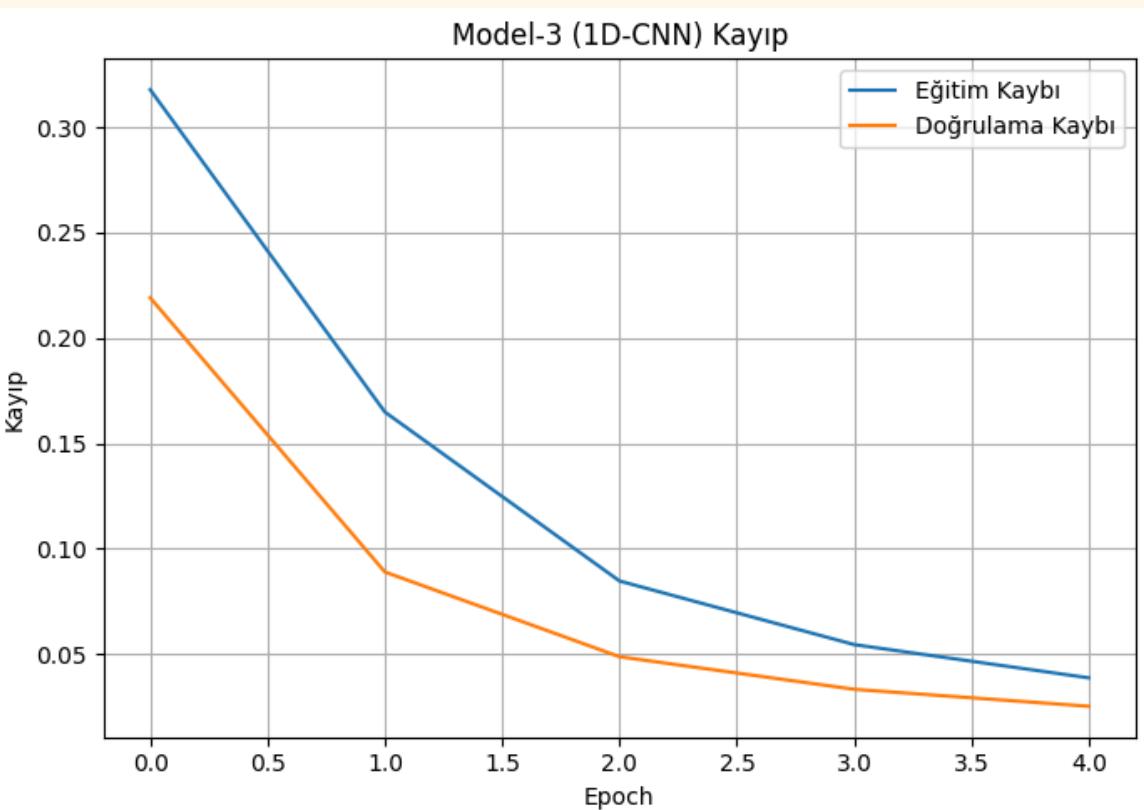
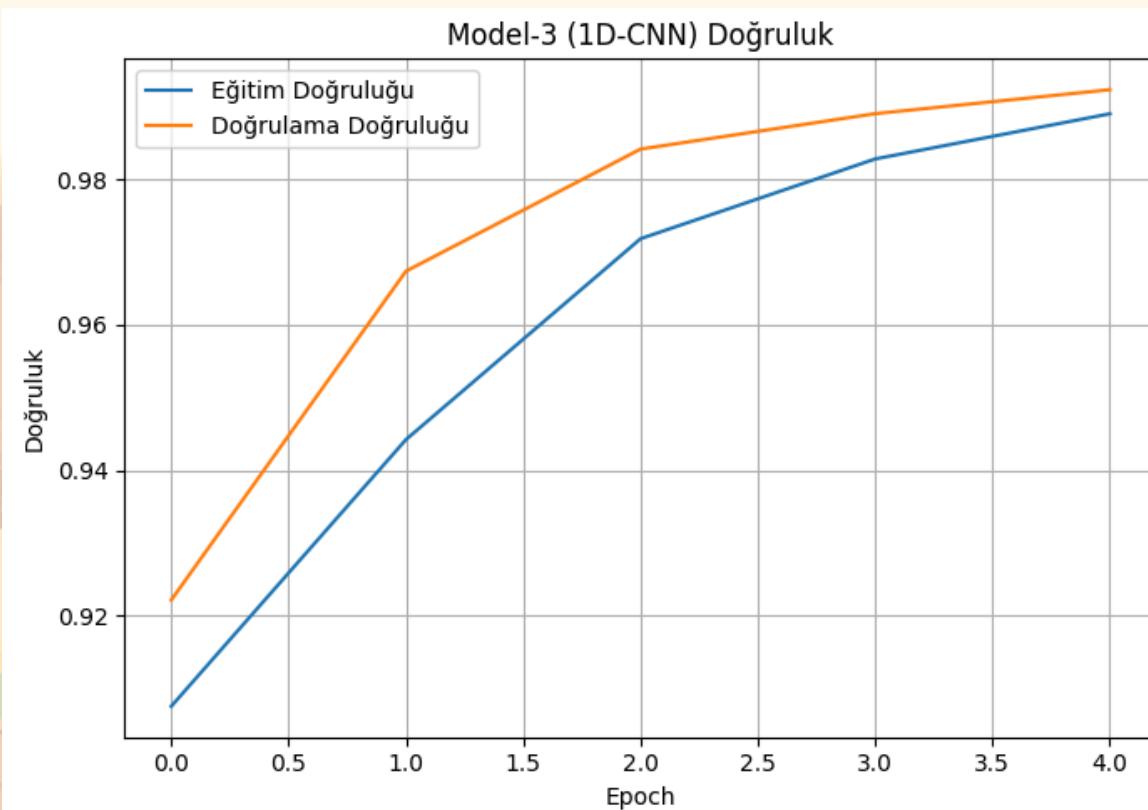
SimpleRNN
64 nöron

Dropout
oran - 0.5

Dense
Sigmoid

1D-CNN MODELİ

```
model3 = Sequential([
    Embedding(input_dim=max_words, output_dim=128, input_length=max_len),
    Conv1D(128, kernel_size=5, activation='relu'),
    GlobalMaxPooling1D(),
    Dropout(0.5),
    Dense(y.shape[1], activation='sigmoid')
])
model3.compile(
    optimizer=Adam(1e-3),
    loss='binary_crossentropy',
    metrics=[BinaryAccuracy(name='accuracy')])
)
print("\n▣ Model-3 (1D-CNN) eğitimi başladı...")
hist3 = model3.fit(
    X_tr_seq, y_tr_seq,
    validation_data=(X_te_seq, y_te_seq),
    epochs=30, batch_size=16,
    callbacks=[early],
    verbose=1
)
# test doğruluğu
loss3, acc3 = model3.evaluate(X_te_seq, y_te_seq, verbose=0)
print(f"▣ Model-3 (1D-CNN) test doğruluğu: {acc3:.4f}")
```



Bu model ise bir 1D
Konvolüsyonel Sinir Ağı.
Metin içindeki kalıpları bulmak
için Conv1D kullanılır.
Ardından en güçlü özellikleri
çekken GlobalMaxPooling1D var.
Diğer yapı RNN'e benzer
(Dropout, Dense, sigmoid çıkış).
Modelin test doğruluğu
hesaplanır

Embedding
input_dim = 5000, output_dim = 128,
input_length = 100

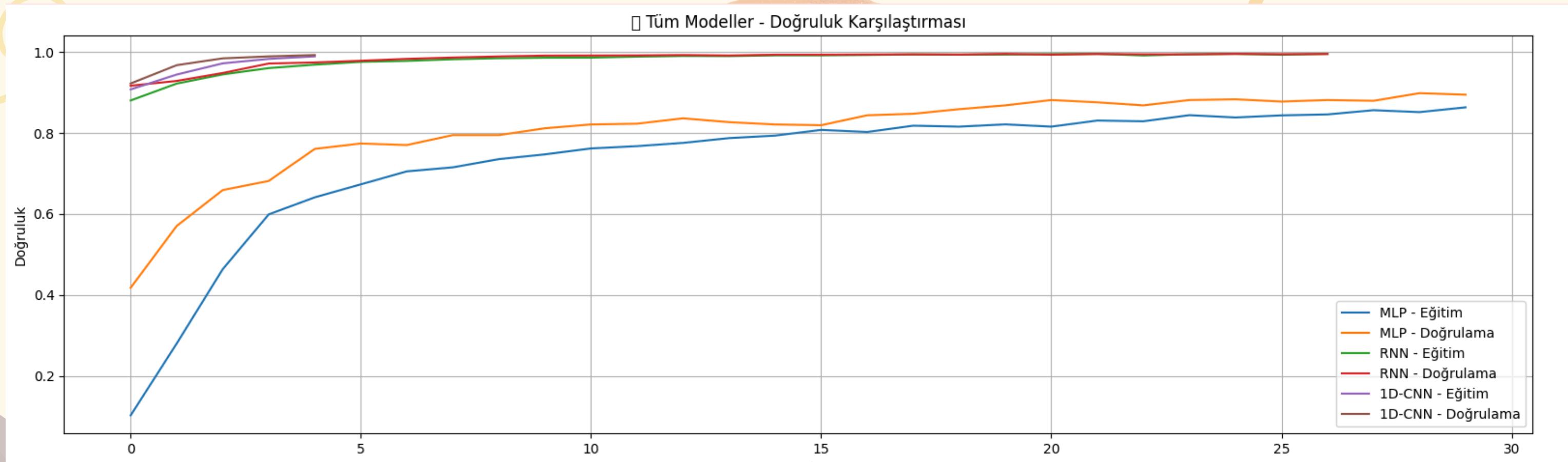
Conv1D
128 filtre, kernel_size = 5, aktivasyon =
ReLU

GlobalMaxPooling1D

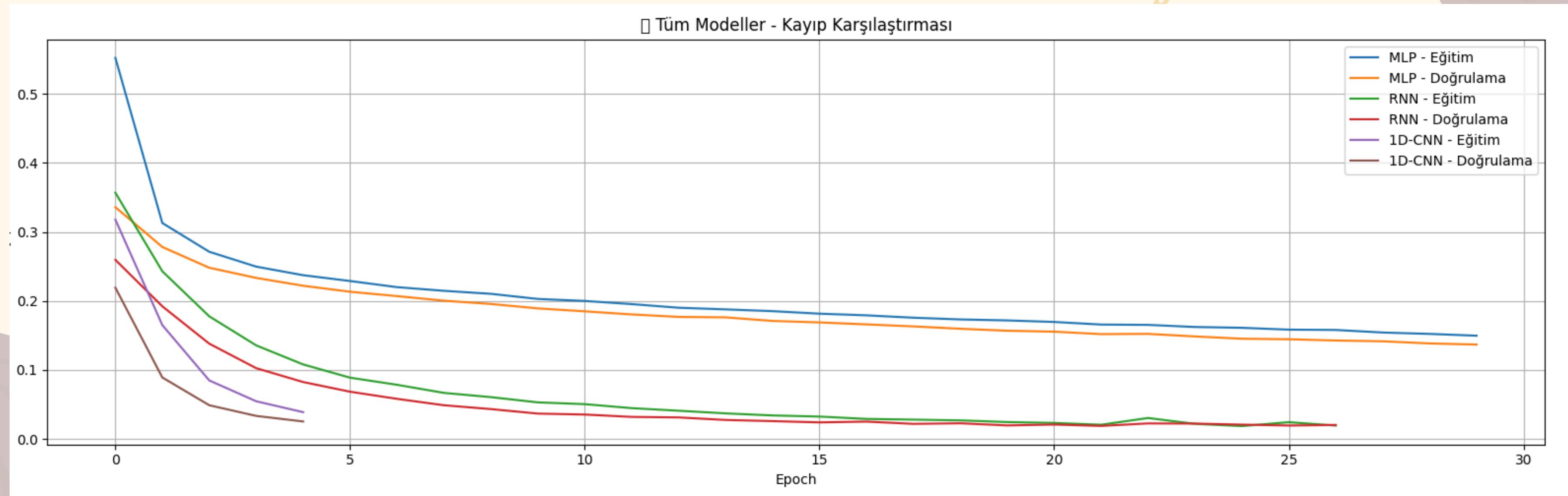
Dropout

Dense (Çıkış katmanı), nöron sayısı =
y.shape[1], aktivasyon = sigmoid

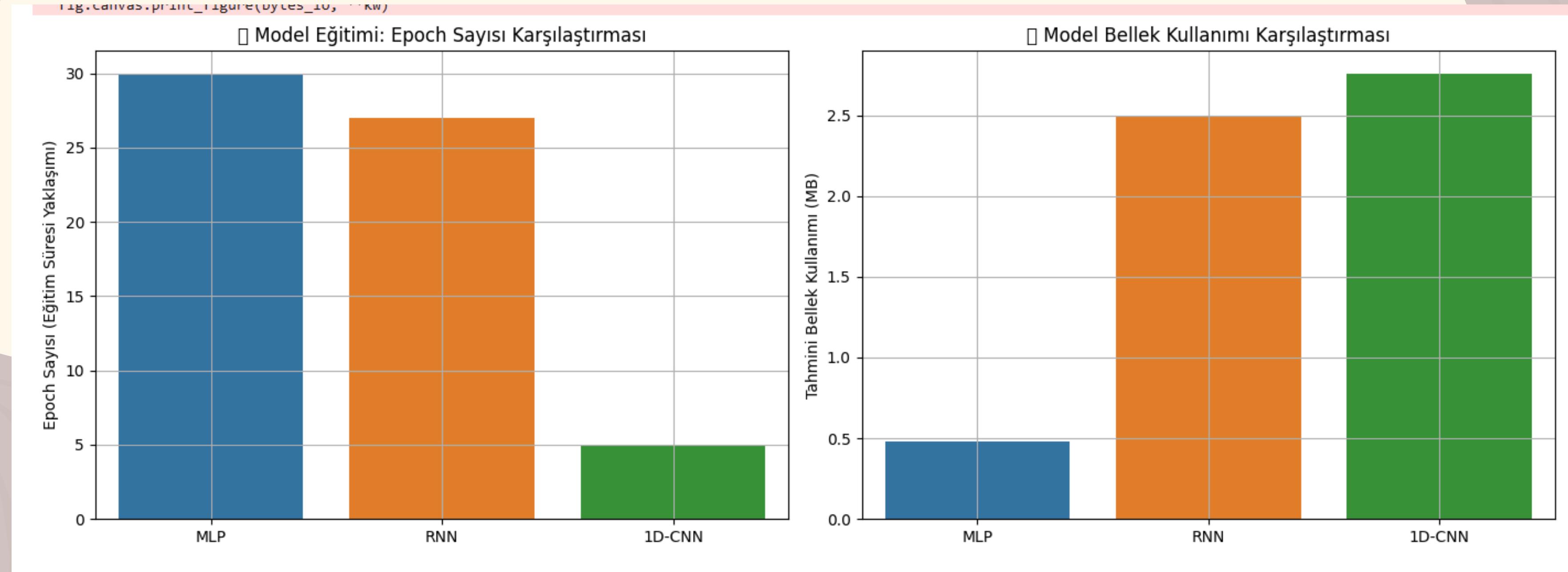
MODELLERİN KARSILASTIRILMASI



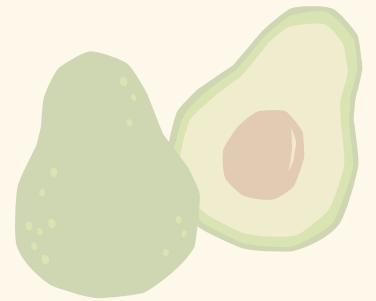
MODELLERİN KARSILASTIRILMASI



MODELLERİN KARSILAŞTIRILMASI



CHATBOT



AKIS SEMASI

Kullanıcı Girdisi (Örn: "Sadece elma ve şeker var")



parse_user_input
(doğal dil → malzeme listesi)



malzeme_temizle
(temizleme, normalize etme)



param_to_mb (TF-IDF / Binary)
(vektörle çevirme)



predict_categories_from_materials
(Model ile kategori tahmini)



match_recipes
(uygun tarifleri filtrele ve sırala)



kategori_ipuclari_bul + sonuç gösterimi
(kategori ipucu + tarif listesi)

👤 TarifBot'a hoş geldin! Ne yapmak istediğini yaz (örnek: elma tarçınlı tatlı):
👤 Sen: elma

- 📌 Tarif Adına Göre Eşleşme Bulundu!
- 📌 Tarif: Elmalı Kereviz Salatası
- 📌 Malzemeler: kereviz, elma, ceviz
- 🕒 Süre: 16 dk | 🍽️ Porsiyon: 2.0
- 👨‍🍳 Açıklama:

Kereviz ve elma rendelenir, yoğurtla karıştırılır. Ceviz serpilerek servis edilir.

101 TarifBot'a hoş geldin! Ne yapmak istediğini yaz (örnek: elma tarçınlı tatlı):

Sen: bana elmalı tatlı tarifi

1/1 0s 23ms/step

⚠ MLP kararsız kaldı, tahmin yapılmayacak.

🔍 Elde edilen malzemeler: ['elma', 'tatlı']

📁 Tahmin edilen kategoriler: []

💬 Cümleden anlaşılan kategori ipuçları: ['Tatlı']

⚠ Bu tarif için eksik malzeme olabilir!

Yine de deneyebilirsin:

📌 Tarif: Tarte Tatin

📅 Malzemeler: elma, şeker, tereyağı, hamur

🕒 Süre: 43 dk | 101 Porsiyon: 4.0

📝 Açıklama:

Karamelize elmalar fırın kabına yerleştirilir, üzerine hamur serilip pişirilir ve ters çevrilerek servis edilir.

 Sen: elmalı tarçınlı tarif

 1/1 —  0s 21ms/step

 MLP Tahmin Skorları:

Tatlı: 0.47200000286102295

Meze: 0.0460000008940697

Ana Yemek: 0.03700000047683716

Çorba: 0.03299999821186066

Kahvaltı: 0.017000000923871994

İçecek: 0.014000000432133675

Zeytinyağlı: 0.009999999776482582

Yan Yemek: 0.00800000037997961

Ara Sıcak: 0.001000000474974513

Hamur İşi: 0.001000000474974513

Salata: 0.001000000474974513

Atıştırmalık: 0.0

 Elde edilen malzemeler: ['elma', 'tarçın']

 Tahmin edilen kategoriler: ['Tatlı']

 Cümleden anlaşılan kategori ipuçları: []

 Verdiğin malzemeleri içeren bir tarif buldum! İşte önerim:

 Tarif: Apple Pie

 Malzemeler: elma, tarçın, şeker, hamur, tereyağı

 Süre: 60 dk |  Porsiyon: 4.0

 Açıklama:

Elmalı iç harç hamura sarılır ve fırında pişirilerek servis edilir.

 Sen: sadece elma

 1/1  0s 23ms/step

 MLP Tahmin Skorları:

Tatlı: 0.4350000023841858

Meze: 0.09200000017881393

Ana Yemek: 0.04500000178813934

Çorba: 0.02999999329447746

Kahvaltı: 0.02500000037252903

İçecek: 0.008999999612569809

Ara Sıcak: 0.00600000052154064

Zeytinyağlı: 0.00600000052154064

Hamur İşi: 0.001000000474974513

Salata: 0.001000000474974513

Yan Yemek: 0.001000000474974513

Atıştırmalık: 0.0

 Elde edilen malzemeler: ['elma']

 Tahmin edilen kategoriler: ['Tatlı']

 Cümleden anlaşılan kategori ipuçları: []

 Uyarı: Elinizde sadece 1 malzeme var. Tarifler sınırlı olabilir.

 Verdiğin malzemeleri içeren bir tarif buldum! İşte önerim:

 Tarif: Elma Dilim

 Malzemeler: elma

 Süre: 59 dk |  Porsiyon: 4.0

 Açıklama:

Elma dilimlenerek taze meyve olarak ya da hafif tatlı niyetine servis edilir.

 Sen: sadece domates

 1/1  0s 22ms/step

 MLP Tahmin Skorları:

Salata: 0.1930000369548798

Çorba: 0.185000023841858

Ana Yemek: 0.1019999809265137

Meze: 0.02899999165534973

İçecek: 0.0160000075995922

Atıştırmalık: 0.0080000037997961

Kahvaltı: 0.0080000037997961

Yan Yemek: 0.00499999888241291

Zeytinyağlı: 0.00499999888241291

Ara Sıcak: 0.0

Hamur İşi: 0.0

Tatlı: 0.0

 Elde edilen malzemeler: ['domates']

 Tahmin edilen kategoriler: ['Ana Yemek', 'Salata', 'Çorba']

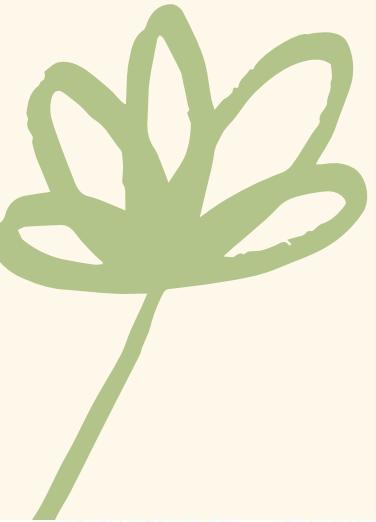
 Cümleden anlaşılan kategori ipuçları: []

 Uyarı: Elinizde sadece 1 malzeme var. Tarifler sınırlı olabilir.

 Uygun tarif bulunamadı.

 Malzeme: ['domates']

 Kategori: ['Çorba', 'Salata', 'Ana Yemek']



Sen: fellah

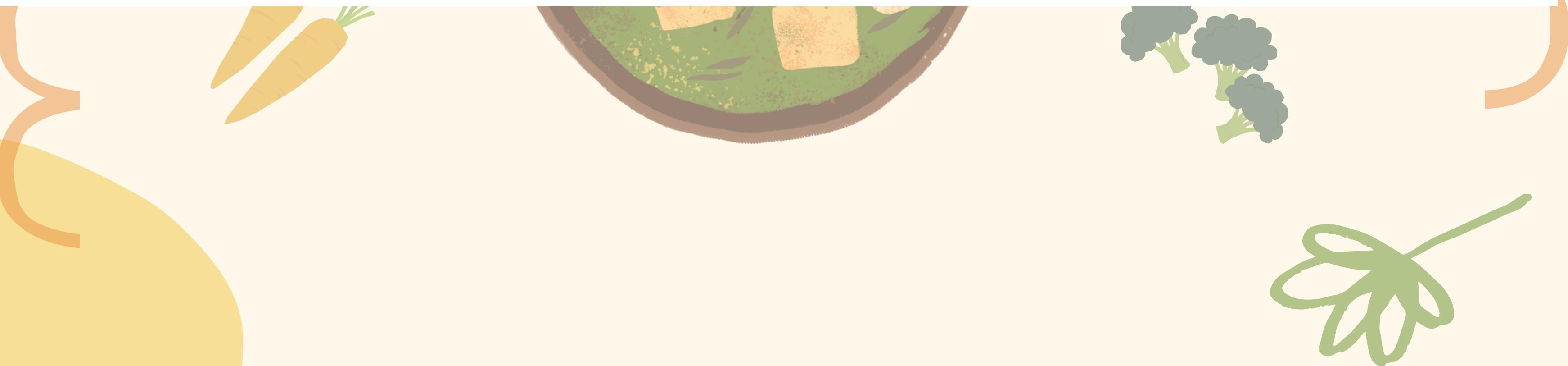
📌 Tarif Adına Göre Eşleşme Bulundu!

📌 Tarif: Fellah Köftesi

📝 Malzemeler: ince bulgur, irmik, salça, un, sarımsak, maydanoz

⌚ Süre: 50 dk | 🍽️ Porsiyon: 4.0

💡 Açıklama:
Bulgur, irmik, salça ve un yoğrularak küçük köfteler yapılır. Haşlanıp sarımsaklı domates sosuyla karıştırılır. Üzerine maydanoz serpilerek servis edilir.



 Sen: mercimekli yemek tarifi

 1/1  0s 22ms/step

 MLP Tahmin Skorları:

Meze: 0.2540000081062317
Çorba: 0.2529999911785126
Ana Yemek: 0.12999999523162842
Kahvaltı: 0.02500000037252903
Tatlı: 0.013000000268220901
Salata: 0.008999999612569809
Ara Sıcak: 0.007000000216066837
Atıştırmalık: 0.00499999888241291
İçecek: 0.00499999888241291
Yan Yemek: 0.001000000474974513
Zeytinyağlı: 0.001000000474974513
Hamur İşi: 0.0

 Elde edilen malzemeler: ['mercimek']

 Tahmin edilen kategoriler: ['Ana Yemek', 'Meze', 'Çorba']

 Cümleden anlaşılan kategori ipuçları: []

 Verdiğin malzemeleri içeren bir tarif buldum! İşte önerim:

 Tarif: Koshari

 Malzemeler: makarna, pirinç, mercimek, domates sosu, soğan

 Süre: 58 dk |  Porsiyon: 6.0

 Açıklama:

Malzemeler ayrı ayrı haşlanır. Üzerine domates sosu ve kızarmış soğan eklenerek servis edilir.



PROJE KAGGLE LINKI:

<https://www.kaggle.com/code/begumkeles/chatbot>