YAPAY SİNİR AĞLARI PROJE ÖDEVİ

- **MILK QUALITY PREDICTION**

FATMA SENA YÜKSEL

MILK QUALITY PREDICTION

- Süt, insan beslenmesinde önemli bir role sahiptir ve sütün kalitesi, insanların sağlıklı ve dengeli beslenmesi açısından kritik önem taşır. pH, sıcaklık, tat, koku, yağ oranı, bulanıklık ve renk gibi parametreler süt kalitesini belirlemektedir. Geleneksel yöntemlerle süt kalitesinin değerlendirilmesi zaman alıcı ve maliyetli olabilmektedir.

- Yapay sinir ağları ve derin öğrenme modelleri kullanılarak daha hızlı ve doğru tahminler yapılabilmektedir.Süt kalitesinin bu yöntemlerle etkin bir şekilde değerlendirilmesi, kalite standartlarının sağlanması, gıda güvenliğinin artırılması ve tüketici memnuniyetinin yükseltilmesi gibi çok sayıda avantaj sağlar.

MILK QUALITY PREDICTION PARAMETERS

- Bu veri seti, süt kalitesini tahmin etmek için kullanılan 7 bağımsız değişkenden oluşmaktadır: pH, Sıcaklık, Tat, Koku, Yağ, Bulanıklık ve Renk. Hedef değişken ise sütün kalite sınıfıdır: Düşük (Kötü), Orta (Orta) veya Yüksek (İyi).
- Veri seti, gözlemler yoluyla elle toplanmıştır. Tat, Koku, Yağ ve Bulanıklık değişkenleri ikili olup, 1 optimal koşulları, 0 ise optimal koşulları karşılamadığını göstermektedir. Sıcaklık ve pH değişkenleri ise gerçek değerleriyle verilmiştir.

pH	Tempratu	Taste	Odor	Fat	Turbidity	Colour	Grade
6,6	35	1	0	1	0	254	high
6,6	36	0	1	0	1	253	high
8,5	70	1	1	1	1	246	low
9,5	34	1	1	0	1	255	low
6,6	37	0	0	0	0	255	medium
6,6	37	1	1	1	1	255	high
5,5	45	1	0	1	1	250	low
4,5	60	0	1	1	1	250	low
8,1	66	1	0	1	1	255	low
6,7	45	1	1	0	0	247	medium

Bu veri seti toplam 7sütun 1060 satırdan oluşmaktadır

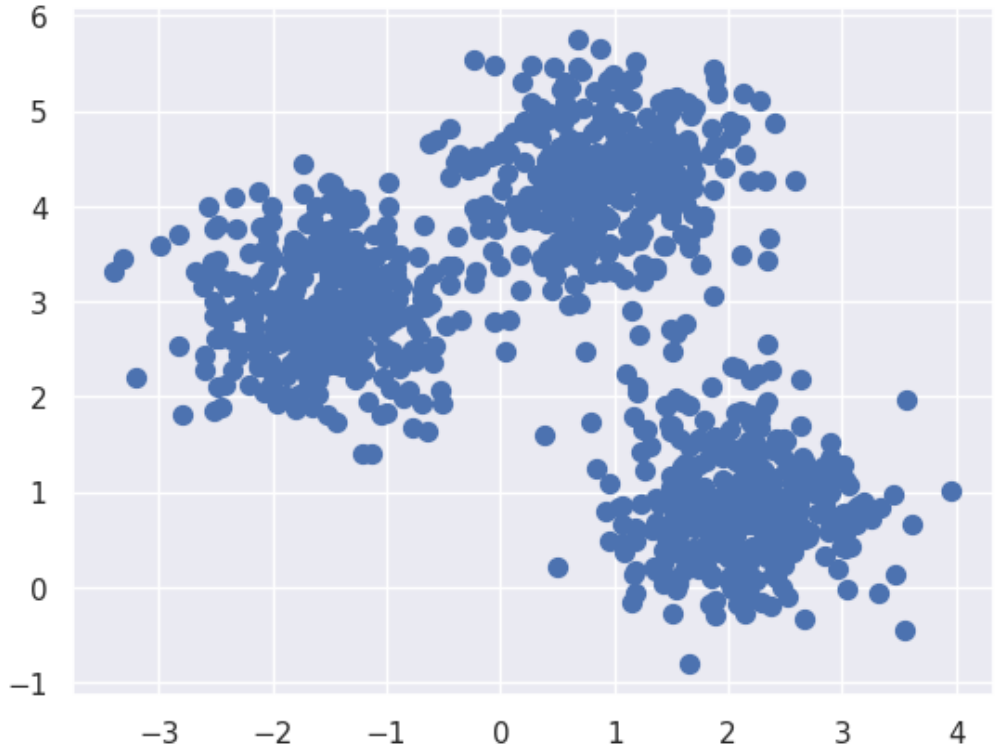
UYGULANAN YÖNTEMLER

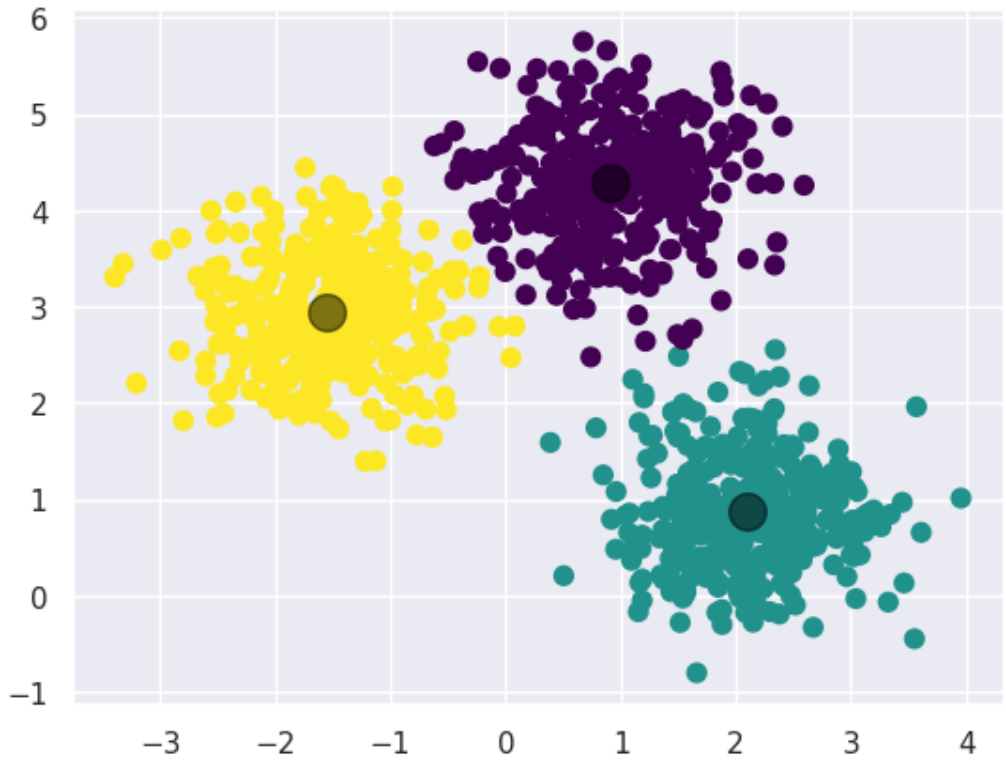
- K-Means
- LVQ
- ANFIS
- ANN
- Logistic Regression

K-Means

0.2 holdout

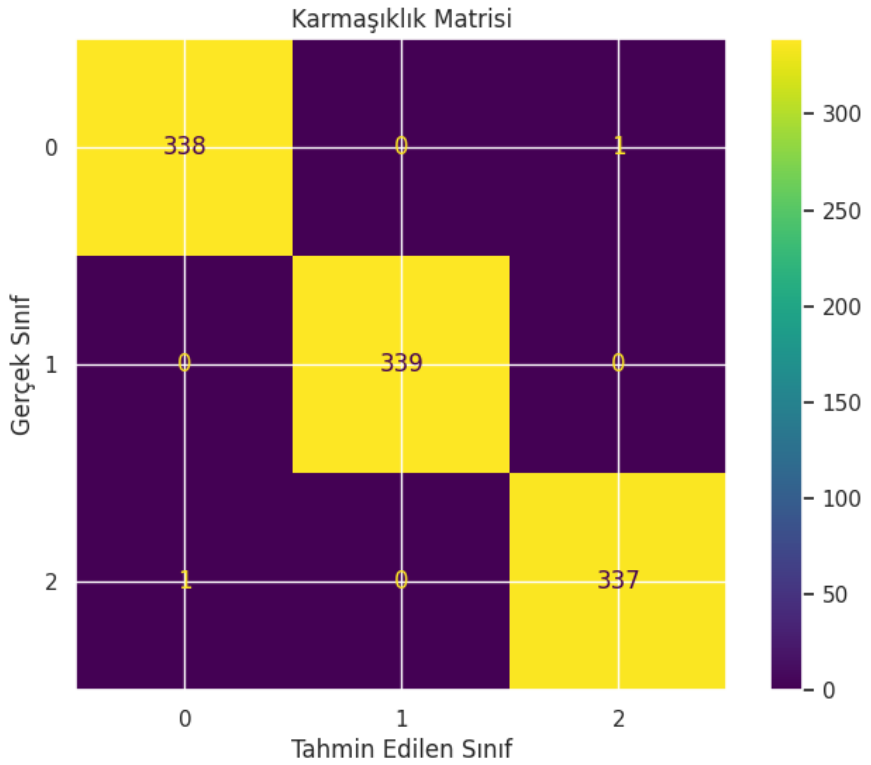
- Silhouette Score: 0.4719170614058787

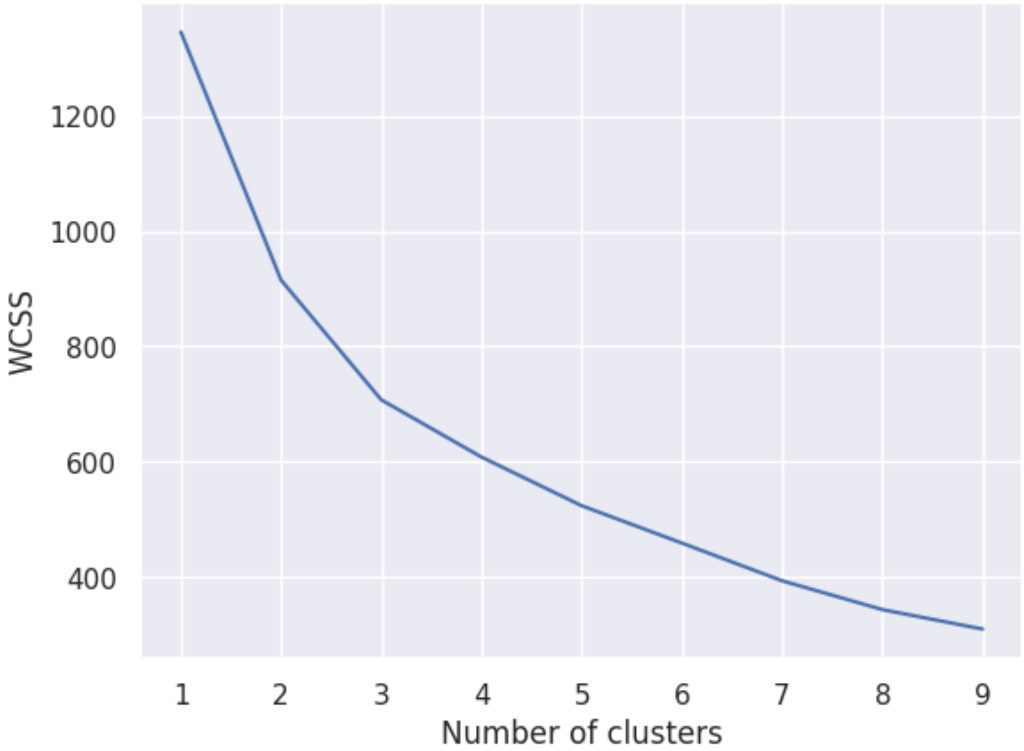




K-Means

- Precision: 0.9980
- Recall: 0.9980
- Accuracy: 0.9980
- F1-score: 0.9980





LVQ

0.2 holdout

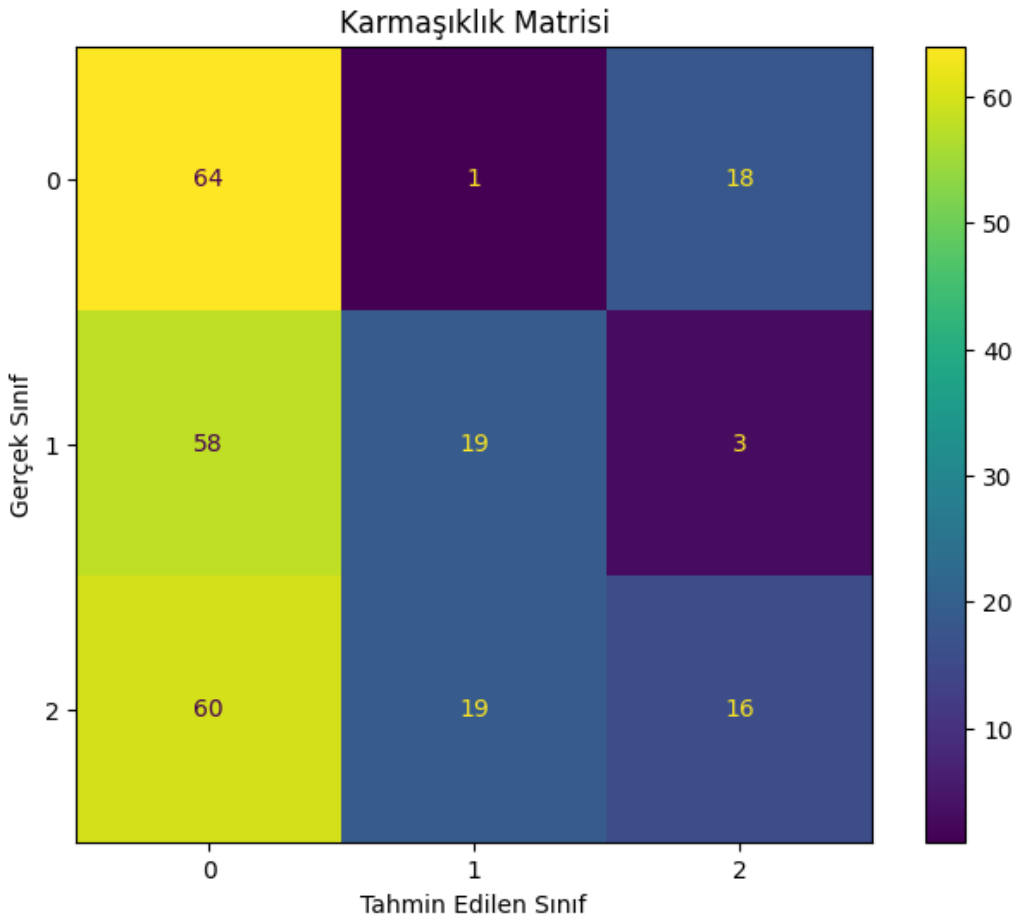
```python
class LVQModel:
    def __init__(self, prototypes, prototype_labels):
        self.prototypes = prototypes
        self.prototype_labels = prototype_labels
    def predict(self, X):
        predictions = []
        for x in X:
            distances = np.sum((self.prototypes - x)**2, axis=1)
            nearest_prototype_idx = np.argmin(distances)
            predictions.append(self.prototype_labels[nearest_prototype_idx])
        return predictions
```

```python
# Initialization (example with 3 prototypes per class)
num_classes = len(np.unique(y_train))
prototypes_per_class = 3
prototypes = []
prototype_labels = []
for label in range(num_classes):
    class_samples = X_train[y_train == label]
    indices = np.random.choice(len(class_samples), prototypes_per_class, replace=False)
    for index in indices:
        prototypes.append(class_samples[index])
        prototype_labels.append(label)

prototypes = np.array(prototypes)
prototype_labels = np.array(prototype_labels)

# Training
num_epochs = 30
learning_rate = 0.01
```

- Accuracy: 0.38
- Precision: 0.42
- Recall: 0.38
- F1-score: 0.34



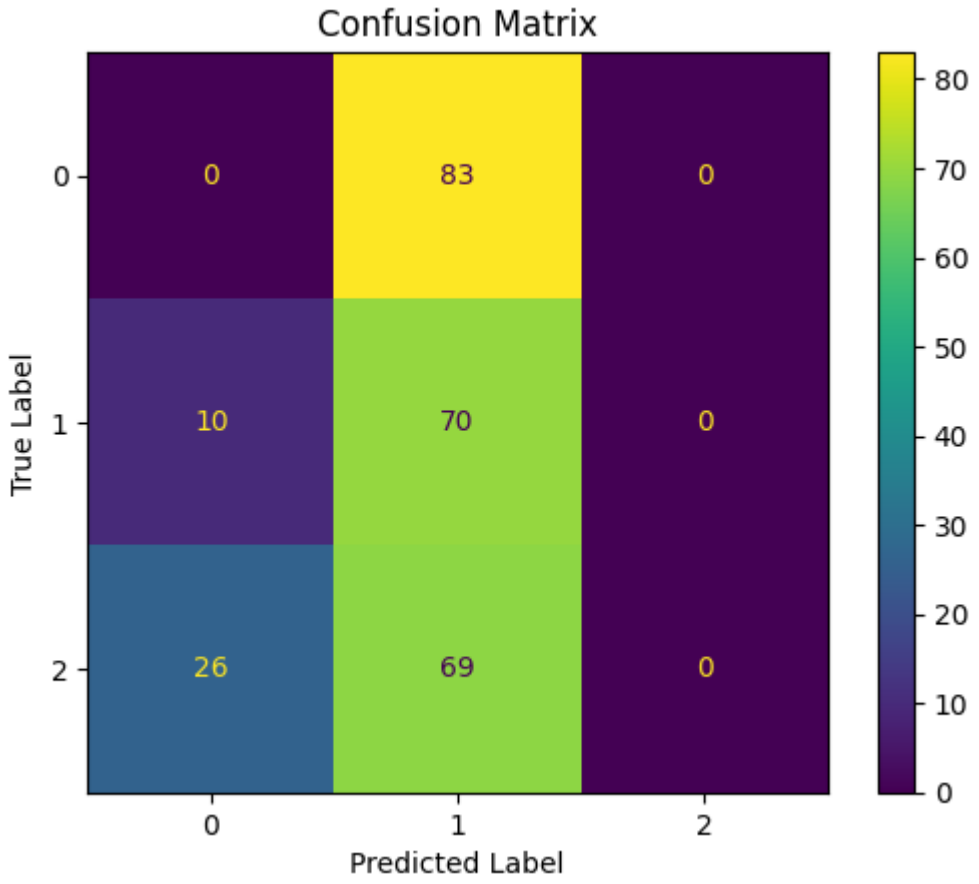
ANFIS

0.2 holdout

```python
# Train the ANFIS model
for i in range(len(X_train)):
    grade_sim.input['pH'] = X_train.iloc[i, 0]
    grade_sim.input['Temprature'] = X_train.iloc[i, 1]
    grade_sim.input['Taste'] = X_train.iloc[i, 2]
    grade_sim.input['Odor'] = X_train.iloc[i, 3]
    grade_sim.input['Fat '] = X_train.iloc[i, 4]
    grade_sim.input['Turbidity'] = X_train.iloc[i, 5]
    grade_sim.input['Colour'] = X_train.iloc[i, 6]
    grade_sim.compute()
    print(f"Iterasyon {i} için çıktı: {grade_sim.output['Grade']}")

# Test the model
predictions = []
for i in range(len(X_test)):
    grade_sim.input['pH'] = X_test.iloc[i, 0]
    grade_sim.input['Temprature'] = X_test.iloc[i, 1]
    grade_sim.input['Taste'] = X_test.iloc[i, 2]
    grade_sim.input['Odor'] = X_test.iloc[i, 3]
    grade_sim.input['Fat '] = X_test.iloc[i, 4]
    grade_sim.input['Turbidity'] = X_test.iloc[i, 5]
    grade_sim.input['Colour'] = X_test.iloc[i, 6]
    grade_sim.compute()
    predictions.append(grade_sim.output['Grade'])
    print(f"Predicted: {grade_sim.output['Grade']}")
```

- Accuracy: 0.2713
- Precision: 0.4660
- Recall: 0.2713
- F1 Score: 0.1437



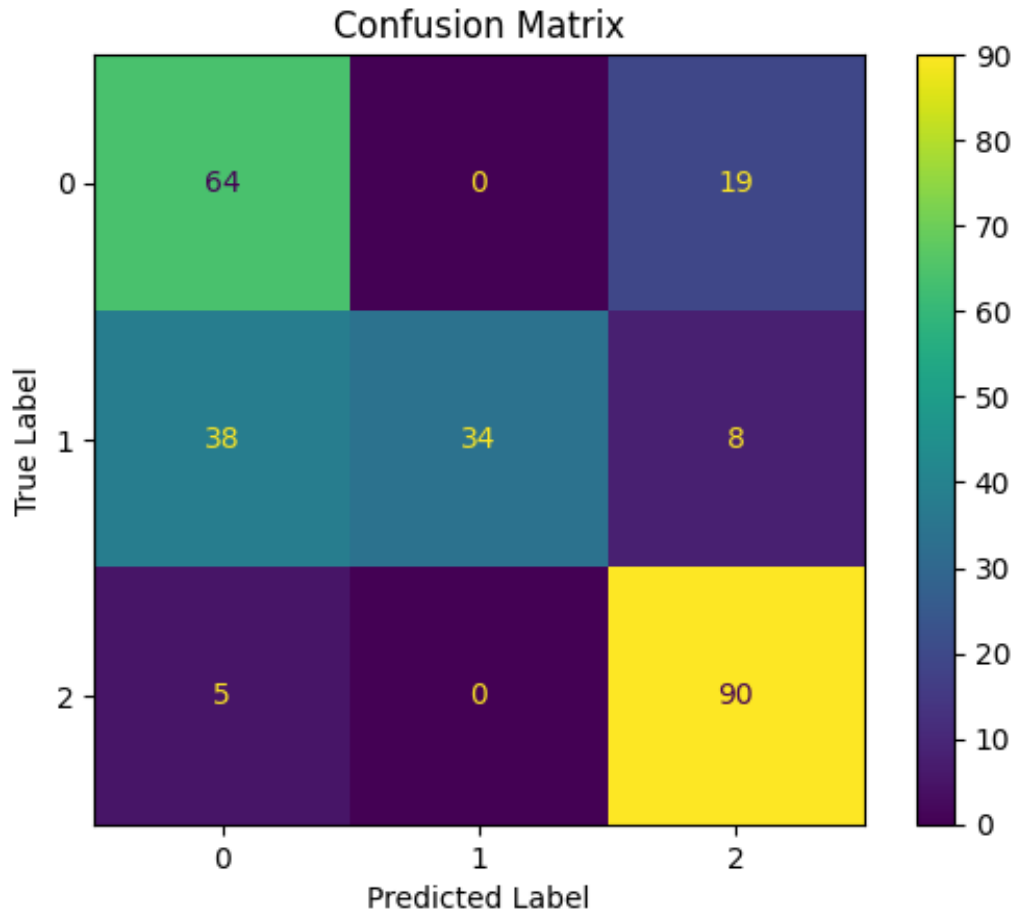
ANN

0.2 holdout

- Accuracy: 0.7287
- Loss: 0.5504
- Precision: 0.7857
- Recall: 0.7287
- F1-Score: 0.7143

```
model = Sequential()
model.add(Dense(64, input_dim=7, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(3, activation='softmax'))

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



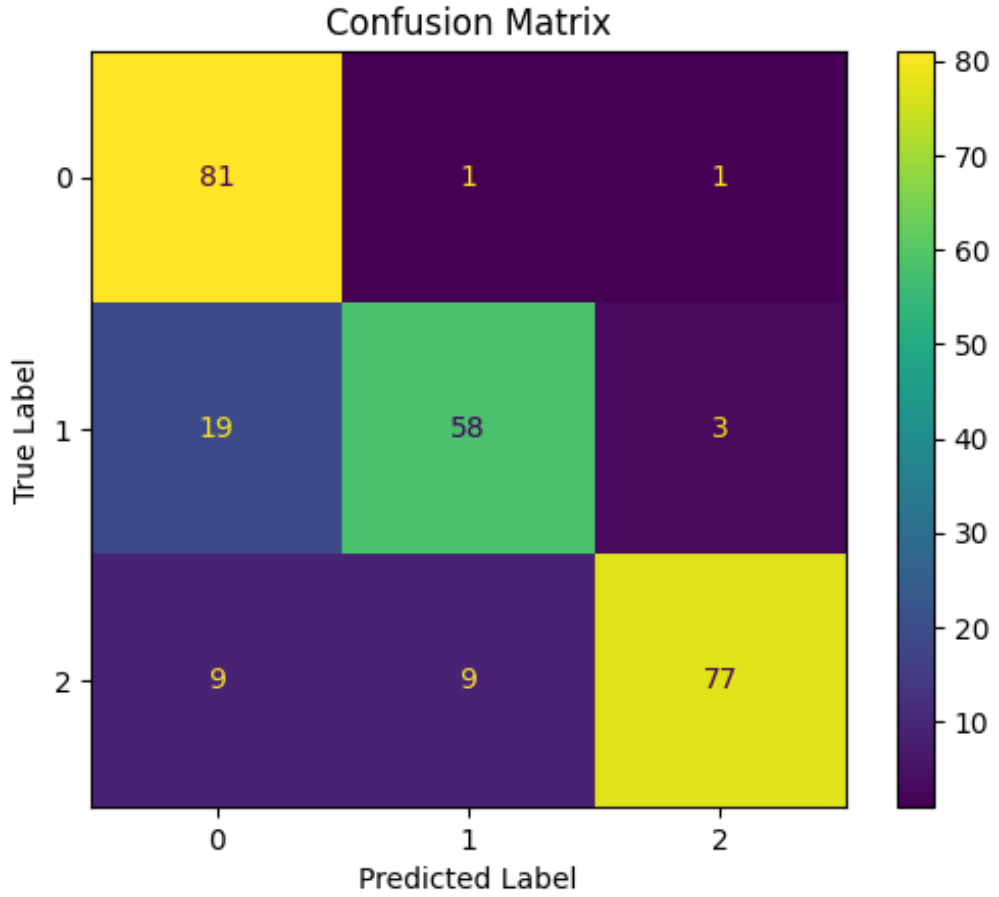
LOGISTIC REGRESSION

0.2 holdout

```
# Create a logistic regression model
model = LogisticRegression()


# Train the model
model.fit(X_train, y_train)
```

- Accuracy: 0.84
- Precision: 0.85
- Recall: 0.84
- F1-score: 0.84



SONUÇLAR

	K-MEANS	LVQ	ANFIS	ANN	LOGISTIC REGRESSION
ACCURACY	0.99	0.32	0.27	0.72	0.84
PRECISON	0.99	0.42	0.46	0.78	0.85
RECALL	0.99	0.38	0.27	0.72	0.84
F1 - SCORE	0.99	0.34	0.14	0.71	0.84