



**BURSA TEKNİK ÜNİVERSİTESİ  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**VERİ MADENCİLİĞİNE GİRİŞ  
PROJE ÖDEVİ**

K-NN Algoritması ile Banka Müşterilerinin Vadeli Mevduat Açma  
Eğilimlerinin Tahmini: Uygulama ve Analiz

20392380066

Tuğçe GÜZEL

2025

## **İÇİNDEKİLER**

<b>1. PROJENİN TANIMI.....</b>	<b>3</b>
<b>1.1. Veri Seti ve Özellikleri.....</b>	<b>4</b>
<b>2. PROJENİN KODLARI.....</b>	<b>5</b>
<b>3. KODLARDAN ELDE EDİLEN ÇIKTI .....</b>	<b>13</b>
<b>4. LİTERATÜRDEKİ AKADEMİK ÇALIŞMA İLE KIYASLAMA .....</b>	<b>18</b>
<b>5. KAYNAKÇA.....</b>	<b>20</b>

## **1. PROJENİN TANIMI**

Günümüzde bankacılık sektörüne bakıldığından müşteri ilişkileri yönetimi ve hedefli pazarlama stratejileri, kurumların rekabet üstünlüğünü elde etmesinde bazı dinamikler ortaya konmaktadır. Özellikle vadeli mevduat gibi ürünlerin pazarlanması, potansiyel müşterilerin doğru tespit edilmesi kaynakların da etkin kullanımını sağlamaktadır. Bu doğrultuda makine öğrenmesi algoritmaları, büyük veri setlerinden anlamlı örüntülerin çıkarılmasını ve müşteri davranışlarının tahmin edilmesini sağlamaktadır. Bu sayede de kurumların zaman ve işgücü açısından karar verme süreçlerine önemli katkılar sunmaktadır.

Pazarlama satış kampanyaları, işletmelerin büyümeye hedeflerine ulaşmasında temel stratejilerden biridir. Şirketler, pazarlama yöntemleriyle belirli müşteri profillerini hedef alarak onlarla iletişime geçer. Uzak erişim temelli pazarlama yöntemi, "tele-pazarlama" (telemarketing) olarak adlandırılır.

Projede, bankanın uzun vadeli mevduatlarının satışına yönelik tele-pazarlama aramalarının başarısını tahmin etmeye yönelik bir sınıflandırma modelinin geliştirilmesi ve değerlendirilmesi amaçlanmıştır.

Proje kapsamında, müşteri davranışlarını öngörmek amacıyla K-Nearest-Neighbor (Yakın Komşu) (K-NN) algoritması incelenmiştir. Modelin sonuçlarının yorumlanabilirliğinin yüksek olması ve uygulanmasının kolay olması avantajı ile bu sınıflandırma yöntemi seçilmiştir. Çalışmada Portekiz perakende bankasından Mayıs 2008'den Haziran 2013'e kadar toplanan 45211 adet müşteri kaydından oluşan bir veri seti kullanılmıştır. K-NN algoritması farklı mesafe metrikleri (Euclidean, Manhattan, Chebyshev) ile denenmiştir.

## **1.1. Veri Seti ve Özellikleri**

Veri setinde her müşteri için toplam 16 adet özellik bulunmaktadır.

1. Demografik özellikler: yaşı, meslek, medeni durum, eğitim düzeyi
2. Finansal özellikler: kredi durumu, ortalama yıllık bakiye, konut kredisi, bireysel kredi
3. İletişim bilgileri: iletişim türü, iletişim günü ve ayı, görüşme süresi
4. Kampanya verileri: iletişim sayısı, önceki kampanyalardan sonra geçen gün sayısı, önceki iletişim sayısı, önceki kampanya sonucu

Hedef değişken, müşterinin vadeli mevduat hesabı açıp açmadığını (“evet” veya “hayır”) belirtmektedir.

## 2. PROJENİN KODLARI

Verileri indirmek için kullanılan kodlar aşağıdaki gibidir.

```
3. import tabula
4. import pandas as pd
5. import os
6. import requests
7. url="https://archive.ics.uci.edu/static/public/222/data.csv"
8.
9. try:
10.     # CSV'yi oku
11.     response = requests.get(url)
12.     response.raise_for_status()
13.     df = pd.read_csv(url)
14.
15.     # NaN değerlerini "NaN" olarak göster
16.     print("Tablo Örneği (ilk 5 satır):")
17.     print(df.head().to_string(na_rep="NaN"))
18.
19.     # İstatistikler (NaN'lar "NaN" olarak)
20.     print("\nSayısal Verilerin İstatistikleri:")
21.     print(df.describe().to_string(na_rep="NaN"))
22.
23.     # CSV'yi kaydet (NaN'lar "NaN" olarak)
24.     df.to_csv("verilerim.csv", index=False, na_rep="NaN")
25.     print("\nTablo 'verilerim.csv' olarak kaydedildi")
26.
27. except requests.exceptions.RequestException as e:
28.     print(f"Bağlantı hatası: {e}")
29. except pd.errors.ParserError as e:
30.     print(f"CSV ayrıştırma hatası: {e}")
31. except Exception as e:
32.     print(f"Beklenmeyen hata: {e}")
```

Modeli Eğitmek ve grafik haline getirmek için kullanılan kodlar aşağıdaki gibidir.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import joblib

# Veri setini yükle
df = pd.read_csv("verilerim.csv") # kendi dosyanın adı

# Hedef değişkeni dönüştür (yes/no → 1/0)
le = LabelEncoder()
y = le.fit_transform(df["y"])

# Özellikleri ayıır
X = df.drop("y", axis=1)
categorical_features = X.select_dtypes(include=["object"]).columns.tolist()
numerical_features = X.select_dtypes(include=["int64",
"float64"]).columns.tolist()

# Ön işleme pipeline
categorical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="constant", fill_value="unknown")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])
numerical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])
preprocessor = ColumnTransformer(transformers=[
    ("num", numerical_transformer, numerical_features),
    ("cat", categorical_transformer, categorical_features)
])

# Eğitim ve test seti ayıır
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Mesafe metrikleri
metrics = {
    "euclidean": {"name": "Euclidean (p=2)", "params": {"metric": "minkowski",
"p": 2}},
    "manhattan": {"name": "Manhattan (p=1)", "params": {"metric": "minkowski",
"p": 1}},
    "chebyshev": {"name": "Chebyshev ( $L^\infty$ )", "params": {"metric": "chebyshev"}}
}

# K aralığı belirle
k_range = range(1, 21)

# Tüm metriklerin sonuçlarını saklayacak sözlük
metrics_results = {}
```

```

# Tüm K değerlerinin doğruluk skorlarını saklayacak sözlük
all_k_scores = {}

# Specificity dahil tüm metrikleri hesaplayan fonksiyon
def calculate_specificity(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)

    if cm.shape[0] == 2: # İkili sınıflandırma durumunda
        # Specificity = TN / (TN + FP)
        tn, fp, fn, tp = cm.ravel()
        specificity = tn / (tn + fp) if (tn + fp) > 0 else 0
    else: # Çok sınıflı durumda
        # Her sınıf için specificity hesapla ve ortalamasını al
        specificities = []
        n_classes = len(np.unique(y_true))

        for i in range(n_classes):
            # i sınıfı için true negative'leri hesapla
            # Karmaşıklik matrisinden i. satır ve i. sütunu çıkar
            mask = np.ones(cm.shape, bool)
            mask[i, :] = False
            mask[:, i] = False
            true_neg = np.sum(cm[mask])

            # i sınıfı için false positive'leri hesapla (i. sütunun toplamı - i. sütun i. satırındaki değer)
            false_pos = np.sum(cm[:, i]) - cm[i, i]

            # Sıfıra bölünmeyi önle
            if (true_neg + false_pos) > 0:
                spec_i = true_neg / (true_neg + false_pos)
                specificities.append(spec_i)

        # Tüm sınıfların specificity değerlerinin ortalamasını al
        specificity = np.mean(specificities) if specificities else 0

    return specificity

for key, metric in metrics.items():
    print(f"\n== {metric['name']} ==")

    k_scores = []
    for k in k_range:
        model = Pipeline(steps=[
            ("preprocessor", preprocessor),
            ("classifier", KNeighborsClassifier(n_neighbors=k,
**metric["params"])))
        ])

```

```

model.fit(X_train, y_train)
score = model.score(X_test, y_test)
k_scores.append(score)
print(f"K = {k}: Doğruluk = {score:.4f}")

# K değerlerinin doğruluk skorlarını kaydet
all_k_scores[metric['name']] = k_scores

# En iyi K değerini seç
best_k = k_range[np.argmax(k_scores)]
best_score = max(k_scores)
print(f"\n\☒ En iyi K değeri ({metric['name']}): {best_k}, Doğruluk: {best_score:.4f}")

# En iyi modeli tekrar eğit
best_model = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("classifier", KNeighborsClassifier(n_neighbors=best_k,
**metric["params"])))
])
best_model.fit(X_train, y_train)
y_pred = best_model.predict(X_test)

# Sınıflandırma raporu
print("\nSınıflandırma Raporu:")
print(classification_report(y_test, y_pred, target_names=le.classes_))

# Tüm metrikleri hesapla
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
specificity = calculate_specificity(y_test, y_pred)

# Metrikleri yazdır
print("\nPerformans Metrikleri:")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"Specificity: {specificity:.4f}")

# Sonuçları kaydet
metrics_results[key] = {
    'accuracy': accuracy,
    'precision': precision,
    'recall': recall,
    'f1': f1,
}

```

```

        'specificity': specificity,
        'best_k': best_k
    }

# Karmaşıklik matrisi
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=le.classes_, yticklabels=le.classes_)
plt.title(f"Karmaşıklik Matrisi - {metric['name']}") 
plt.xlabel("Tahmin Edilen")
plt.ylabel("Gerçek Değer")
plt.tight_layout()
cm_path = f"confusion_matrix_{key}.png"
plt.savefig(cm_path)
print(f"Karmaşıklik matrisi kaydedildi: {cm_path}")

# Modeli kaydet
model_path = f"knn_model_{key}.pkl"
joblib.dump(best_model, model_path)
print(f"Model dosyası kaydedildi: {model_path}")

# Örnek tahmin
sample = X_test.iloc[0:1]
pred_class = best_model.predict(sample)
print(f"Örnek Tahmin ({metric['name']}): {le.inverse_transform(pred_class)[0]}\n\n")

# K değerlerinin doğruluk skorlarını CSV olarak kaydet
k_scores_df = pd.DataFrame(all_k_scores, index=list(k_range))
k_scores_df.index.name = 'K Değeri'
k_scores_csv_path = "knn_k_scores.csv"
k_scores_df.to_csv(k_scores_csv_path)
print(f"\nK değerlerinin doğruluk skorları CSV olarak kaydedildi: {k_scores_csv_path}\n\n")

# K Değerlerinin karşılaştırma grafiği
plt.figure(figsize=(12, 6))
for key, metric in metrics.items():
    plt.plot(k_range, all_k_scores[metric['name']], marker='o',
label=metric["name"])

plt.title('K Değerine Göre Doğruluk Skorları')
plt.xlabel('K Değeri')
plt.ylabel('Doğruluk')
plt.xticks(k_range)
plt.grid(True)
plt.legend()

```

```

plt.tight_layout()
plt.savefig("knn_k_values_comparison.png")
plt.show()

# Metrik karşılaştırma tablosu
metrics_df = pd.DataFrame({
    key: {
        'Accuracy': results['accuracy'],
        'Precision': results['precision'],
        'Recall': results['recall'],
        'F1 Score': results['f1'],
        'Specificity': results['specificity'],
        'Best K': results['best_k']
    }
    for key, results in metrics_results.items()
})

print("\n==== Tüm Mesafe Metriklerinin Karşılaştırması ===")
print(metrics_df.round(4).T)

# Metrik karşılaştırma grafiği (Best K hariç)
metrics_plot_df = metrics_df.drop('Best K', axis=0)
plt.figure(figsize=(14, 8))
metrics_plot_df.plot(kind='bar', figsize=(14, 8))
plt.title('Mesafe Metriklerine Göre Performans Karşılaştırması')
plt.xlabel('Performans Metriği')
plt.ylabel('Skor')
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend(title='Mesafe Metriği')
plt.tight_layout()
plt.savefig("knn_metrics_comparison.png")
plt.show()

# Metrik karşılaştırma tablosunu CSV olarak kaydet
csv_path = "knn_metrics_comparison.csv"
metrics_df.round(4).T.to_csv(csv_path)
print(f"\nMetrik karşılaştırma tablosu CSV olarak kaydedildi: {csv_path}")

# Daha detaylı bir tablo oluştur
detailed_results = []
for key, metric_info in metrics.items():
    result = metrics_results[key]
    detailed_results.append({
        'Mesafe Metriği': metric_info['name'],
        'En İyi K': result['best_k'],
        'Accuracy': result['accuracy'],
        'Precision': result['precision'],
        'Recall': result['recall'],
        'F1 Score': result['f1'],
        'Specificity': result['specificity']
    })

```

```

        'Recall': result['recall'],
        'F1 Score': result['f1'],
        'Specificity': result['specificity']
    })

# DataFrame oluştur ve CSV olarak kaydet
detailed_df = pd.DataFrame(detailed_results)
detailed_csv_path = "knn_detailed_comparison.csv"
detailed_df.to_csv(detailed_csv_path, index=False)
print(f"\nDetaylı metrik karşılaştırma tablosu CSV olarak kaydedildi:
{detailed_csv_path}")

```

Eğitilen modeli test etmek için kullanılan kodlar aşağıdaki gibidir.

```

import pandas as pd
import joblib

# Kullanılacak modeller ve etiketleri
modeller = {
    "Euclidean (Öklid)": 'knn_model_euclidean.pkl',
    "Manhattan (L1)": 'knn_model_manhattan.pkl',
    "Chebyshev (L∞)": 'knn_model_chebyshev.pkl'
}

# Müşteri örnekleri
potansiyel_musteriler = pd.DataFrame([
    {'age': 42, 'job': 'management', 'marital': 'married', 'education':
'tertiary',
     'default': 'no', 'balance': 8500, 'housing': 'yes', 'loan': 'no',
     'contact': 'cellular', 'day_of_week': 15, 'month': 'may', 'duration':
450,
     'campaign': 1, 'pdays': -1, 'previous': 0, 'poutcome': 'unknown'},

    {'age': 65, 'job': 'retired', 'marital': 'married', 'education':
'secondary',
     'default': 'no', 'balance': 12000, 'housing': 'no', 'loan': 'no',
     'contact': 'cellular', 'day_of_week': 10, 'month': 'jun', 'duration':
320,
     'campaign': 1, 'pdays': -1, 'previous': 0, 'poutcome': 'unknown'},

    {'age': 35, 'job': 'technician', 'marital': 'single', 'education':
'tertiary',
     'default': 'no', 'balance': 5600, 'housing': 'yes', 'loan': 'no',
     'contact': 'cellular', 'day_of_week': 3, 'month': 'apr', 'duration': 280,
     'campaign': 2, 'pdays': -1, 'previous': 0, 'poutcome': 'unknown'}
])

```

```

{'age': 52, 'job': 'self-employed', 'marital': 'married', 'education': 'secondary',
 'default': 'no', 'balance': 7200, 'housing': 'yes', 'loan': 'no',
 'contact': 'cellular', 'day_of_week': 20, 'month': 'jul', 'duration': 380,
 'campaign': 1, 'pdays': 180, 'previous': 1, 'poutcome': 'success'},

 {'age': 29, 'job': 'management', 'marital': 'single', 'education': 'tertiary',
 'default': 'no', 'balance': 6800, 'housing': 'no', 'loan': 'no',
 'contact': 'cellular', 'day_of_week': 12, 'month': 'feb', 'duration': 310,
 'campaign': 1, 'pdays': -1, 'previous': 0, 'poutcome': 'unknown'}
])

# Tahminleri tüm modeller için yap
print("== Vadeli Mevduat Tahmin Karşılaştırması ==\n")

for i, (indeks, musteri) in enumerate(potansiyel_musteriler.iterrows()):
    print(f"\n--- Müşteri {i+1} ---")
    print(f"Profil ► Yaş: {musteri['age']}, Meslek: {musteri['job']}, Eğitim: {musteri['education']}, Bakiye: {musteri['balance']} €")

    for ad, dosya in modeller.items():
        try:
            model = joblib.load(dosya)
            tahmin = model.predict(musteri.to_frame().T)[0]
            olasiliklar = model.predict_proba(musteri.to_frame().T)[0]

            etik = "EVET - Açıbilir" if tahmin == 1 else "HAYIR - Açmayabilir"
            guven = olasiliklar[1] * 100 # 'yes' sınıfı
            print(f"{ad} ► Tahmin: {etik} | Güven: %{guven:.2f}")
        except Exception as e:
            print(f"{ad} ► Tahmin yapılamadı: {e}")

# Genel istatistik vermek ister misin? Ekleyebilirim.
print("\n\n== GENEL İSTATİSTİKLER ==")
for ad, dosya in modeller.items():
    try:
        model = joblib.load(dosya)
        tahminler = model.predict(potansiyel_musteriler)
        toplam = len(tahminler)
        evet_sayisi = sum(1 for t in tahminler if t == 1)
        oran = (evet_sayisi / toplam) * 100
        print(f"{ad} ► {toplam} müşteriden {evet_sayisi} tanesi için 'Vadeli Mevduat Açıbilir' dedi. (%{oran:.2f})")
    except Exception as e:
        print(f"{ad} ► İstatistik hesaplanamadı: {e}")

```

Projede kullanılan kodların dosyaları;



### 3. KODLARDAN ELDE EDİLEN ÇIKTI

Proje sonucunda, bankaların pazarlama stratejilerini en iyilemek için makine öğrenmesi modellerinden yararlanabileceği ve bu sayede kaynaklarını daha etkin kullanabileceği görülmüştür. Gerçek pozitif örnekler TP, gerçek negatif örnekler TN, toplam gerçek negatif örnekler (Gerçek Negatifler + Yanlış Pozitifler) FP ile ifade edildiğinde model performansının değerlendirilmesi için çeşitli metrikler kullanılmaktadır.

Doğruluk (Accuracy), modelin tüm tahminleri içinde doğru yaptığı tahminlerin oranıdır. Modelin genel olarak ne kadar başarılı olduğunu ifade eder.  $[(TP + TN) / (TP + TN + FP + FN)]$  formülü ile hesaplanır.

Kesinlik (Precision), modeldeki pozitif tahminlerin ne kadar güvenilir olduğunu ölçer.  $[TP / (TP + FP)]$  formülü ile hesaplanır.

Duyarlılık (Sensitivity/Recall),其实 pozitif olan örnekler içinde modelin doğru bir şekilde pozitif olarak tahmin ettiği örneklerin oranıdır.  $[TP / (TP + FN)]$  formülü ile hesaplanır. Yüksek duyarlılık, modelin gerçek pozitifleri kaçırma olasılığının düşük olduğu anlamına gelir.

F1 Skoru (F1 Score), kesinlik ve duyarlılık metriklerinin harmonik ortalamasıdır. Bu iki metrik arasında bir denge kurar. Özellikle sınıf dağılımının dengesiz olduğu veri setlerinde doğruluk metriğine göre daha bilgilendirici olabilir.  $[2 * (Precision * Sensitivity) / (Precision + Sensitivity)]$  veya  $[2 * TP / (2 * TP + FP + FN)]$  formülü ile hesaplanır.

Özgüllük (Specificity), bir sınıflandırma modelinin gerçekde negatif olan örnekleri doğru bir şekilde negatif olarak tanımlama yeteneğini ifade eder.  $[TN / (TN + FP)]$  formülü ile hesaplanır.

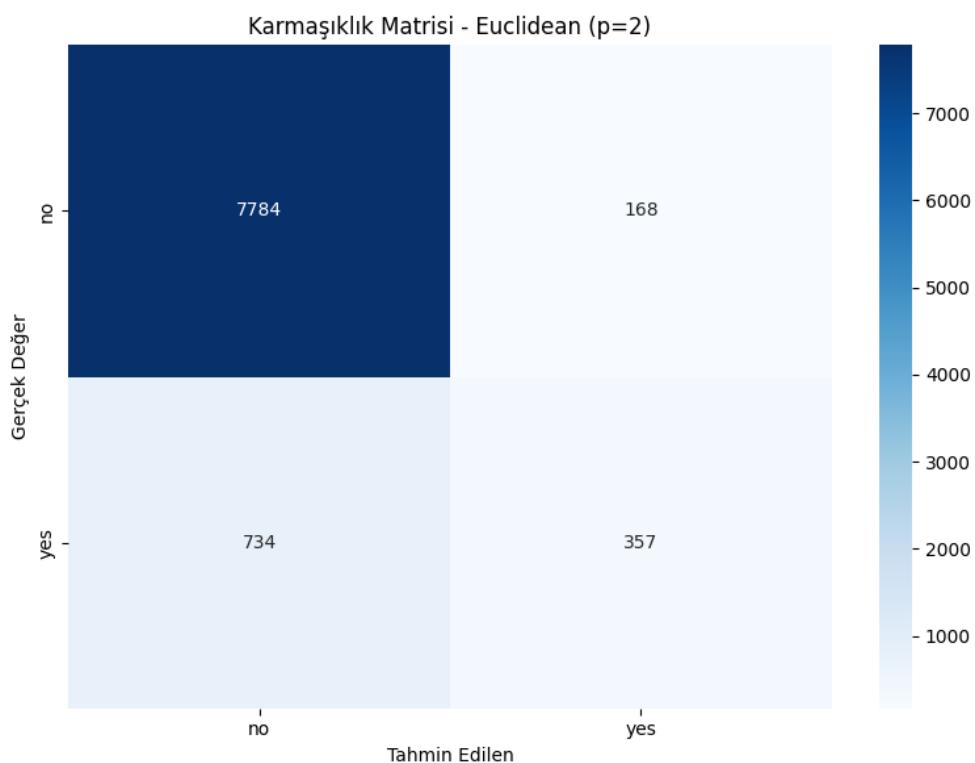
K-NN modeli analizinde Euclidean, Manhattan ve Chebyshev ( $L^\infty$ ) mesafe metrikleri denenmiştir. Euclidean metriği ile en yüksek performansa  $K=13$  değerinde ulaşılmış ve %90.03 doğruluk oranı elde edilmiştir. Manhattan metriği ile en yüksek performansa yine  $K=13$  değerinde ulaşılmış ve %89.93 doğruluk oranı elde edilmiştir. Chebyshev metriği ile en yüksek performansa  $K=6$  değerinde ulaşılmış ve %88.59 doğruluk oranı elde edilmiştir.

Metriklerden elde edilen tüm sonuçlar aşağıdaki tabloda yer almaktadır.

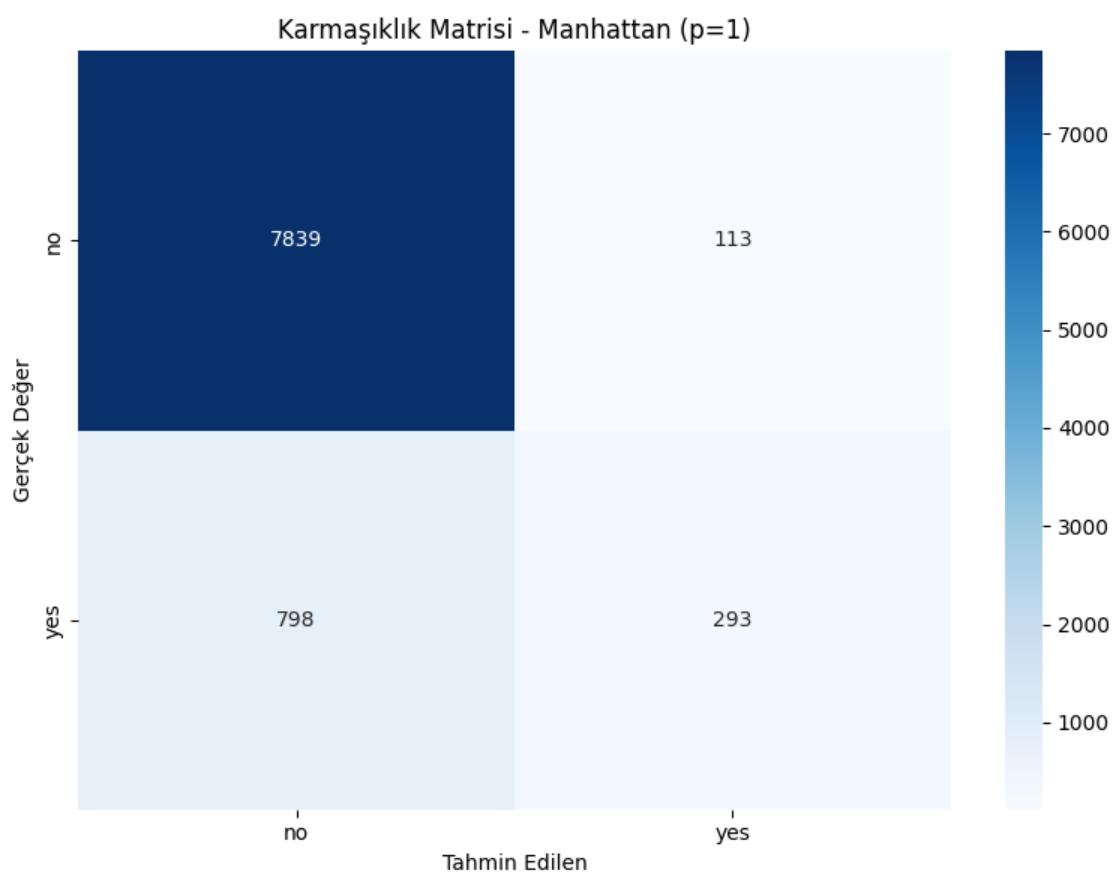
	Accuracy	Precision	Recall	F1 Score	Specificity	Best K
euclidean	0.9003	0.8856	0.9003	0.8845	0.9789	13
manhattan	0.8993	0.8852	0.8993	0.8783	0.9858	13
chebyshev	0.8859	0.8599	0.8859	0.8555	0.9855	6

En iyi model olan Euclidean ( $K=13$ ) için elde edilen karmaşıklık matrisi (confusion matrix) ve sınıflandırma raporu incelendiğinde, modelin “hayır” sınıfını (vadeli hesap açmayanlar) tahmin etmede “evet” sınıfına (vadeli hesap açanlar) kıyasla çok daha başarılı olduğu görülmektedir.

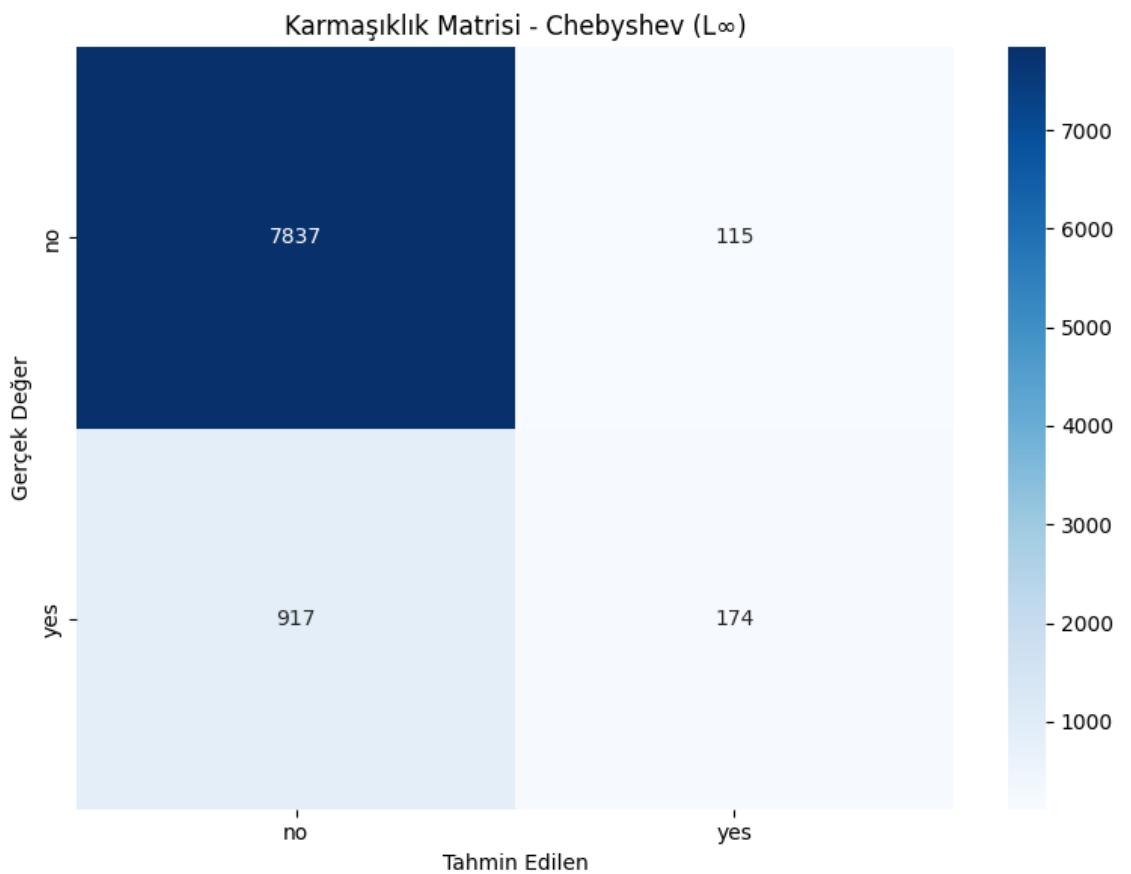
Aşağıdaki şekil K-NN modelinin Euclidean mesafesi ve  $K=13$  değeri ile en iyi versiyonuna ait karmaşıklık matrisini (confusion matrix) göstermektedir. Matris, modelin sınıflandırma performansını detaylı bir şekilde özetlemektedir. Dikey eksen gerçek değerleri (“hayır” veya “evet”) yatay eksen ise modelin tahminlerini (“hayır” veya “evet”) temsil eder. Matrisin içindeki değerler şunları ifade eder: 7784 örnek, gerçekten “hayır” iken doğru bir şekilde “hayır” olarak tahmin edilmiştir (Gerçek Negatif-TN); 168 örnek, gerçekten “hayır” iken yanlışlıkla “evet” olarak tahmin edilmiştir (Yanlış Pozitif-FP); 734 örnek, gerçekten “evet” iken yanlışlıkla “hayır” olarak tahmin edilmiştir (Yanlış Negatif-FN); ve 357 örnek, gerçekten “evet” iken doğru bir şekilde “evet” olarak tahmin edilmiştir (Gerçek Pozitif-TP). Bu matris, modelin ‘hayır’ sınıfını (%97.89 özgüllük ile) tahmin etmede oldukça başarılı olduğunu, ancak “evet” sınıfını yakalamada daha çok zorlandığını açıkça göstermektedir. Özellikle “evet” olması gereklirken “hayır” olarak tahmin edilen örnek sayısı (FN=734), doğru tahmin edilen “evet” sayılarından (TP=357) fazladır. Renk yoğunluğu da en yüksek değerin (TN) olduğu sol üst köşede en koyu mavi olarak bu bulguyu görsel olarak desteklemektedir. Bu durum, veri setindeki sınıf dengesizliğini ve modelin azınlık sınıfını (“evet”) tespit etmedeki zorluklarını yansıtmaktadır.



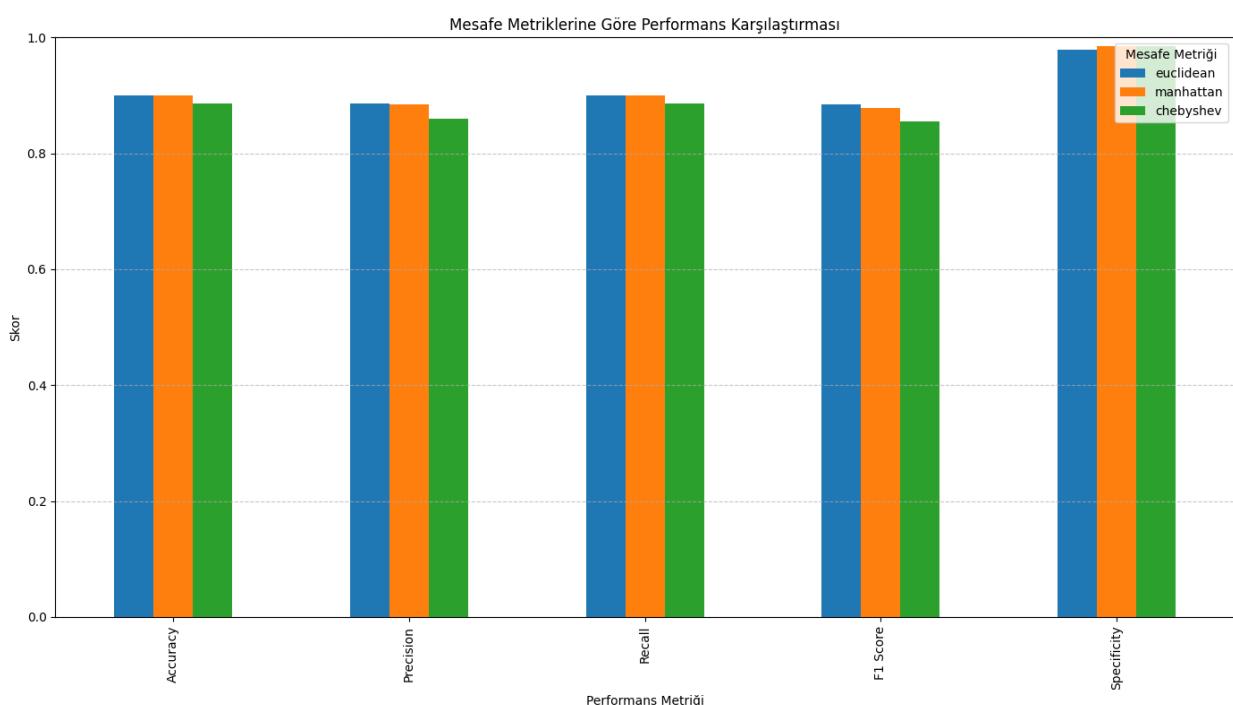
Aşağıdaki şekil K-NN modelinin Manhattan mesafe metriği ve bu metrik için en iyi bulunan  $K=13$  değeri ile yaptığı sınıflandırmanın sonuçlarını gösteren karmaşıklık matrisidir (confusion matrix). Matris, modelin performansını gerçek ve tahmin edilen değerler üzerinden detaylandırır; dikey eksen gerçek sınıfı (“hayır” veya “evet”), yatay eksen ise modelin tahminini belirtir. Matris içindeki sayılar şöyledir: 7839 örnek, gerçekten “hayır” iken doğru bir şekilde “hayır” olarak sınıflandırılmıştır (Gerçek Negatif-TN). 113 örnek, gerçekten “hayır” iken hatalı olarak “evet” diye tahmin edilmiştir (Yanlış Pozitif-FP). 798 örnek, gerçekten “evet” iken hatalı olarak “hayır” diye tahmin edilmiştir (Yanlış Negatif-FN). Son olarak, 293 örnek, gerçekten “evet” iken doğru bir şekilde “evet” olarak sınıflandırılmıştır (Gerçek Pozitif-TP).



Aşağıdaki şekil K-NN modelinin Chebyshev ( $L^\infty$ ) mesafe metriği ve bu metrik için en iyi sonuç veren  $K=6$  değeriyle yaptığı sınıflandırmaya ait karmaşıklık matrisini göstermektedir. Önceki matrisler gibi, dikey eksen gerçek değerleri, yatay eksen ise modelin tahminlerini ifade etmektedir. Matrisin hücrelerindeki değerler şöyledir: 7837 örnek doğru bir şekilde “hayır” olarak sınıflandırılmıştır (Gerçek Negatif-TN); 115 örnek hatalı olarak “evet” diye tahmin edilmiştir (Yanlış Pozitif-FP); 917 örnek hatalı olarak “hayır” diye tahmin edilmiştir (Yanlış Negatif-FN); ve sadece 174 örnek doğru bir şekilde “evet” olarak sınıflandırılmıştır (Gerçek Pozitif-TP).



Aşağıdaki grafik, K-NN sınıflandırma modeli için test edilen 3 farklı mesafe metriğinin (Euclidean, Manhattan ve Chebyshev) optimize edilmiş K değerlerindeki performansını karşılaştırmaktadır. Sütunlar, her bir mesafe metriği için hesaplanan temel performans skorlarını [Doğruluk (Accuracy), Kesinlik (Precision), Duyarlılık (Recall), F1 Skoru ve Özgüllük (Specificity)] göstermektedir.



Grafikten elde edilen bulgulara göre, Euclidean metriği ( $K=13$ ), %90.03 ile en yüksek genel doğruluğa ve %88.45 ile en yüksek F1 Skoruna ulaşarak genel anlamda en başarılı performansı sergilemektedir. Manhattan metriği ( $K=13$ ) de %89.93 doğruluk ile Euclidean'a çok yakın bir performans göstermiş ve %98.58 ile en yüksek özgüllük değerine ulaşmıştır. Chebyshev ( $L^\infty$ ) metriği ( $K=6$ ) ise, diğer iki metriğe kıyasla daha düşük doğruluk (%88.59) ve F1 skoru (%85.55) sunarken özgüllük açısından (%98.55) Manhattan'a yakın bir değer göstermiştir. Bu görselleştirme, mesafe metriği seçiminin modelin farklı performans ölçütleri üzerindeki etkisini açıkça göstermektedir. Grafik, modelin hangi yönün önceliklendirildiğine göre en uygun mesafe metriğini seçmeye yardımcı olur.

## **4. LİTERATÜRDEKİ AKADEMİK ÇALIŞMA İLE KIYASLAMA**

Karşılaştırma yapılacak makalenin ismi: Comparative analysis of machine learning algorithms on a Bank Marketing Dataset (Bir Banka Pazarlama Veri Kümesi Üzerinde Makine Öğrenimi Algoritmalarının Karşılaştırılmış Analizi)'tir.

Bu makalede amaç, müşterinin vadeli mevduata abone olup olmayacağı tahmin etmektir.

Projede sonucunda en iyi olduğu kabul edilen K-NN modeli (Euclidean metriği, K=13) ile başka bir akademik çalışmadaki farklı modellerin performansları karşılaştırıldığında önemli farklılıklar gözlemlenmektedir.

Decision Tree yöntemi ile elde edilen sonuçlar:

```
Accuracy : 0.8706181576910318
Precision Score : 0.45201793721973094
Recall Score : 0.47412982126058323
F1 Score : 0.46280991735537186
```

Neural Networks yöntemi ile elde edilen sonuçlar:

```
Accuracy : 0.8982638504920933
Precision Score : 0.5877300613496933
Recall Score : 0.4506114769520226
F1 Score : 0.5101171458998935
```

Doğruluk (Accuracy): Genel sınıflandırma başarısına bakıldığından K-NN modeli %90.03 doğruluk oranıyla en yüksek performansı sergilemektedir. Neural Networks modeli %89.83 ile K-NN modeline çok yakın bir doğruluk sunarken Decision Tree modeli %87.06 ile bu metrikte diğer iki modelin gerisinde kalmaktadır. Bu, K-NN modelinin genel olarak doğru tahmin yapma oranının en yüksek olduğu anlamına gelir.

Kesinlik (Precision): Modelin “evet” olarak tahmin ettiklerinin ne kadarının gerçekten “evet” olduğunu gösteren kesinlik metriğinde K-NN modeli (~%68.05) en iyi performansı göstermektedir. NN modeli (%58.77) ikinci sırada yer alırken, DT modeli (%45.20) bu metrikte en düşük performansa sahiptir. Yani, K-NN “evet” dediğinde bunun doğru olma olasılığı diğerlerine göre daha yüksektir.

Duyarlılık (Recall): “Evet” diyen müşterileri tespit etme (kaçırılmama) konusunda DT modeli (%47.41) en başarılıdır. NN modeli (%45.06) onu yakından takip ederken, K-NN modelinin “evet” sınıfı için duyarlılığı (~%33.00) belirgin şekilde daha düşüktür. Bu, DT ve NN

modelinin potansiyel müşterileri bulmada K-NN modelinden daha iyi olduğu, ancak belki daha fazla yanlış pozitif üretebileceği anlamına gelir.

F1 Skoru (F1 Score): Kesinlik ve duyarlılık arasında bir denge kuran F1 skoruna bakıldığında NN modeli (%51.01) “evet” sınıfı için en dengeli performansı sunmaktadır. DT modeli (%46.28) ikinci sırada yer alırken, K-NN modelinin F1 skoru (~%44.45) bu sınıf için en düşüktür. Bu, NN modelinin “evet” sınıfını hem makul bir oranda yakalama (recall) hem de bu tahminleri yaparken kabul edilebilir bir kesinliğe (precision) sahip olma arasında en iyi dengeyi kurduğunu gösterir.

Accuracy, Precision, Recall ve F1 Score 1'e ne kadar yakınsa sonuç o kadar iyi kabul edilir. Ancak her bir metrik için kabul edilebilir bir minimum değer yoktur ve veri seti ve problem bağlamına göre değişebilir. Örneğin sınıf dengesi çok farklı olan bir veri setinde sadece çoğunluk sınıfına tahmin yaparak yüksek bir Accuracy elde edilebilir. Ancak bu sonuç diğer metriklerde düşük olur. Bu sebeple doğru metriklerin seçilmesi ve sonuçların problem bağlamına göre yorumlanması önemlidir.

Sonuç olarak K-NN modeli doğruluk açısından en başarılı modeldir. Bunun sebebi, çoğunluktaki “hayır” yanıtını (%97.89 başarıyla) çok iyi tahmin etmesidir. Ancak K-NN, asıl hedef olan “evet” müşterilerini bulmakta zayıf kalıyor (düşük recall). NN, genel doğruluğu K-NN modeline yakın olmakla birlikte “evet” sınıfını tahmin etmede en dengeli performansı (en yüksek F1 skoru) sunmaktadır. Decision Tree ise “evet” sınıfını en yüksek oranda yakalayabilen (en yüksek recall) modeldir ancak bunu düşük bir kesinlik pahasına yapmaktadır. Dolayısıyla, iş amacına göre model seçimi değişebilir. Doğruluğu maksimize etmek için K-NN, potansiyel müşterileri dengeli bir şekilde belirlemek için NN, mümkün olan en fazla sayıda potansiyel müşteriyi yakalamak için DT modeli tercih edilmelidir.

## **5. KAYNAKÇA**

İncelenen Makale: <https://archive.ics.uci.edu/dataset/222/bank+marketing>

Kıyaslama Yapılan Akademik Çalışma:

<https://medium.com/%40shobhit.chauhan98/comparative-analysis-of-machine-learning-algorithms-on-a-bank-marketing-dataset-e48f9512a15>