Meme Kanseri Teşhisinde Makine Öğrenmesi Modelleri ile Karar Destek Sistemi

Hazırlayan:
Mehmet Emin Palabıyık

Üniversite/Bölüm:

İstanbul Bilgi Üniversitesi

Bilgisayar Mühendisliği

14/06/2025

Table of Contents

Bölüm 1: Özet:

Bu projede, meme kanseri teşhisinde farklı makine öğrenmesi modellerinin başarımlarını karşılaştırarak en yüksek doğruluk oranına sahip modeli belirlemek ve bu modeli kullanıcı dostu bir web uygulaması haline getirmek amaçlanmıştır. Çalışmada Breast Cancer Wisconsin veri seti kullanılmış, veri ön işleme adımlarının ardından Lojistik Regresyon, Karar Ağaçları, Random Forest, Destek Vektör Makineleri (SVM), K-En Yakın Komşu (KNN) gibi algoritmalar eğitilmiştir. Modeller; doğruluk, kesinlik, duyarlılık, F1 skoru gibi metriklerle değerlendirilmiştir. Elde edilen sonuçlara göre en iyi performansı gösteren model, Streamlit kütüphanesi kullanılarak geliştirilen interaktif bir web arayüzüne entegre edilmiştir. Bu uygulama, kullanıcıdan alınan girdilere göre meme kanseri riskini anlık olarak tahmin edebilmektedir.

Bölüm 2: Giriş:

Meme kanseri, dünya genelinde kadınlar arasında en sık görülen kanser türlerinden biri olup, erken teşhis edilmediği takdirde ölümle sonuçlanabilmektedir. Dünya Sağlık Örgütü (WHO) verilerine göre her yıl yüz binlerce kadına meme kanseri tanısı konulmakta ve bu sayı her geçen yıl artış göstermektedir. Bu durum, erken teşhis sistemlerinin geliştirilmesini hem bireysel hem de toplumsal sağlık açısından hayati bir öncelik haline getirmiştir.

Bu bağlamda, yapay zekâ ve makine öğrenmesi temelli karar destek sistemleri, hekimlere doğru, hızlı ve tutarlı teşhis süreçlerinde yardımcı olabilecek potansiyele sahiptir. Bu projede temel amaç, farklı makine öğrenmesi algoritmalarını kullanarak meme kanseri teşhisi üzerine bir sınıflandırma modeli geliştirmek ve bu modelleri performans açısından karşılaştırarak en iyi sonucu veren modeli belirlemektir. Belirlenen bu model, Streamlit kütüphanesi aracılığıyla kullanıcı dostu bir arayüze dönüştürülerek, pratik ve erişilebilir bir karar destek uygulaması haline getirilmiştir.

Literatürde bu alanda çeşitli çalışmalar bulunmaktadır. Örneğin, Mangasarian ve ark. (1995) tarafından geliştirilen yapay sinir ağı modelleri ile meme kanseri verileri üzerinde yüksek doğruluk oranları elde edilmiştir. Benzer şekilde, Polat ve Güneş (2007), destek vektör makineleri kullanarak %98'in üzerinde doğruluk sağlamışlardır. Günümüzde XGBoost ve Random Forest gibi topluluk (ensemble) yöntemlerinin de teşhis başarımını artırdığı çeşitli akademik çalışmalarda vurgulanmaktadır.

Bu proje, literatürde yer alan yöntemleri temel alarak, farklı modellerin uygulamalı karşılaştırmasını yapmayı ve bu modellerin bir web uygulamasına entegre edilmesini hedeflemektedir.

Bölüm 3: Veri Seti Tanımı:

Bu projede, **Breast Cancer Wisconsin (Diagnostic) Dataset** kullanılmıştır. Veri seti, Wisconsin Üniversitesi tarafından sağlanmış olup meme kanseri teşhisinde yaygın olarak kullanılan güvenilir bir kaynak niteliğindedir. Veri seti, toplam 569 örnek (hasta) ve 32 sütundan oluşmaktadır.

Veri setinde yer alan id sütunu herhangi bir sınıflandırma veya modelleme sürecinde anlamlı bir bilgi içermediği için analiz öncesinde veri setinden çıkarılmıştır. Ayrıca veri setinde **Unnamed: 32** adında tamamen boş olan bir sütun bulunduğu gözlemlenmiş, bu sütun da analiz dışında bırakılmıştır.

Denklem 1 Veri Temizleme Örnek Kod

```
1.  # Veri setini temizleme ve hazırlama
def clean_data():
    df = pd.read_csv('dataset/data.csv') # Veri seti yolu
    df = df.drop(['Unnamed: 32', 'id'], axis=1) # Gereksiz sütunları kaldır
    df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0}) # M ve B etiketlerini sayısal
değerlere dönüştür
    return df
```

1.1 Veri setinin hedef sütunu olan diagnosis, hastalığın teşhis durumunu belirtmektedir. Bu sütunda iki farklı sınıf bulunmaktadır:

M = Malignant (Kötü huylu tümör)

B = Benign (İyi huylu tümör)

1.2 Makine öğrenmesi modelleri genellikle sayısal (numerik) veri ile daha etkili şekilde çalıştığı için, diagnosis sütunu şu şekilde dönüştürülmüştür:

B (Benign) → 0

M (Malignant) → 1

Bu dönüşüm sayesinde modellerin sınıflar arasında daha kolay ayrım yapması ve performans metriklerinin doğru ölçülmesi sağlanmıştır. Sayısal kodlama, özellikle lojistik regresyon, destek vektör makineleri (SVM) gibi algoritmalarda daha kararlı karar yüzeyleri oluşturulmasına yardımcı olur.

Veri setindeki kalan 30 özellik, her biri hücre çekirdeğinin çeşitli ölçümlerini içerecek şekilde gruplandırılmıştır. Bu özellikler aşağıdaki 4 kategori altında sınıflandırılmıştır:

1.3 Temel Ölçümler:

(radius_mean, texture_mean, perimeter_mean, area_mean)

1.4 Doku Özellikleri (Ortalama):

(smoothness_mean, compactness_mean, concavity_mean, concave points_mean, symmetry_mean, fractal_dimension_mean)

1.5 Standart Hatalar (SE- Standard Error):

(radius_se, texture_se, perimeter_se, area_se, smoothness_se, compactness_se, concavity_se, concave points_se, symmetry_se, fractal_dimension_se)

1.6 En Kötü Değerler (Worst):

(radius_worst, texture_worst, perimeter_worst, area_worst, smoothness_worst, compactness_worst, concavity_worst, concave points_worst, symmetry_worst, fractal_dimension_worst)

Denklem 2 Hücre Özellikleri Örnek Kod

```
# Hücre özelliklerini mantıklı gruplara ayırma
feature_groups = {
    "Temel Ölçümler": [
        'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean'
    ],
    "Doku Özellikleri (Ortalama)": [
        'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean',
        'symmetry_mean', 'fractal_dimension_mean'
    ],
    "Standart Hatalar": [
        'radius_se', 'texture_se', 'perimeter_se', 'area_se',
        'smoothness_se', 'compactness_se', 'concavity_se', 'concave points_se',
        'symmetry_se', 'fractal_dimension_se'
    ],
    "En Kötü Değerler": [
        'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst',
        'smoothness_worst', 'compactness_worst', 'concavity_worst', 'concave points_worst',
        'symmetry_worst', 'fractal_dimension_worst'
    ]
}
```

Bu gruplandırma, özelliklerin daha anlaşılır hale gelmesini ve modelleme sürecinde görselleştirme ile yorumlamayı kolaylaştırmıştır.

Bölüm 4: Veri Ön İşleme:

Veri analizi ve modelleme sürecinde, doğru ve etkili sonuçlar elde edebilmek için veri seti üzerinde bazı ön işleme adımları uygulanmıştır.

1.7 Eksik Veri ve Gereksiz Sütunların Temizlenmesi:

Veri setinde yer alan "Unnamed: 32" sütunu tamamen boş olduğu için veri analizine katkı sağlamamaktadır. Ayrıca "id" sütunu da sınıflandırma için anlamlı bir değişken olmadığı ve modelin öğrenme sürecine herhangi bir katkı sunmadığı gerekçesiyle veri setinden çıkarılmıştır.

1.8 Hedef Değişkenin Sayısallaştırılması:

"diagnosis" sütunu, iyi huylu (B) ve kötü huylu (M) tümörleri ifade eden kategorik etiketler içermektedir. Makine öğrenmesi algoritmalarının sayısal verilerle daha iyi çalıştığı bilindiğinden, bu sütun sayısal değerlere dönüştürülmüştür:

B (Benign) → 0

M (Malignant) → 1

Bu dönüşüm, modelin sınıflar arasındaki ayrımı daha net bir şekilde öğrenmesine yardımcı olmuş ve algoritmaların doğruluk, kesinlik gibi metriklerde daha başarılı sonuçlar vermesini sağlamıştır.

1.9 Özellik Seçimi:

Veri setindeki tüm sütunlar anlamlı ve yüksek ayırt ediciliğe sahip olduğundan, ayrıca bir özellik seçimi uygulanmamıştır. Yapılan denemelerde, tüm sütunlar kullanıldığında modellerin %97 üzeri doğruluk oranları elde ettiği görülmüştür. Bu nedenle, herhangi bir bilgi kaybı yaşamamak adına tüm sütunlar modellemeye dahil edilmiştir.

1.10 Normalizasyon / Standardizasyon:

Özellikler farklı ölçeklerde değerlere sahip olduğu için modellerin dengesiz öğrenme yapmalarını engellemek amacıyla StandardScaler ile standardizasyon uygulanmıştır. Bu işlem, her bir özelliğin ortalamasını 0, standart sapmasını 1 yaparak değişkenleri aynı ölçeğe getirir. Bu özellikle SVM, KNN ve Lojistik Regresyon gibi mesafe tabanlı algoritmalar için önemlidir.

1.11 Eğitim/Test Bölünmesi:

Veri seti %80 eğitim ve %20 test oranında olacak şekilde ikiye ayrılmıştır. Bu sayede modellerin öğrenme başarısı eğitim verisi üzerinde, genelleme yeteneği ise daha önce görülmemiş test verisi üzerinde değerlendirilmiştir. Eğitim/test bölünmesi, modelin aşırı öğrenmesini (overfitting) kontrol altında tutmak açısından önemlidir.

Bölüm 5: Kullanılan Modeller:

Bu projede, meme kanseri teşhisine yönelik sınıflandırma problemini çözmek amacıyla farklı makine öğrenmesi algoritmaları kullanılmıştır. Modellerin her biri farklı varsayımlara, öğrenme stratejilerine ve güçlü yönlere sahip olup, nihai amaç en yüksek doğrulukla tahmin yapabilmektir.

1.12 Model 1: Lojistik Regresyon:

Lojistik regresyon, doğrusal bir sınıflandırma yöntemidir. Özellikle ikili sınıflandırma problemleri için etkili ve yorumlanabilir bir modeldir. Aşırı karmaşıklık olmadan güçlü performans gösterebilmesi, onu temel bir başlangıç modeli yapmaktadır.

1.13 Model 2: Karar Ağaçları:

Karar ağaçları, veriyi dallara ayırarak karar kuralları üretir. Yorumu kolaydır ancak tek başına kullanıldığında aşırı öğrenmeye (overfitting) eğilimlidir. Bu sebeple ensemble yöntemlerle birlikte daha güçlü hale gelir.

1.14 Model 3: Random Forest:

Random Forest, birçok karar ağacının bir araya gelerek oluşturduğu güçlü bir topluluk (ensemble) yöntemidir. Aşırı öğrenme riskini azaltırken genelleme yeteneğini artırır. Bu model, özellikle karmaşık veri yapılarında güçlü performans sergilemektedir.

1.15 Model 4: Support Vector Machine (SVM):

SVM, sınıflar arasındaki en geniş ayrımı yapacak karar sınırlarını bulmaya çalışır. Özellikle yüksek boyutlu verilerde etkili sonuçlar verir. Çekirdek (kernel) fonksiyonları sayesinde doğrusal olmayan ayrımlar da yapılabilir.

1.16 Model 5: K-Nearest Neighbors (KNN):

KNN algoritması, sınıflandırmayı veri noktalarının komşularına göre yapar. Basit yapısına rağmen doğru mesafe ölçütü ve uygun k değeri ile yüksek performans sağlayabilir.

1.17 Yapay Sinir Ağı (MLPClassifier):

Çok katmanlı yapay sinir ağı (MLP), nöronlardan oluşan gizli katmanlar yardımıyla karmaşık karar sınırları oluşturabilir. Bu model, doğrusal olmayan ilişkileri öğrenmede avantaj sağlar.

Bölüm 6: GridSearchCV Kullanımı:

Bazı modellerde en iyi sonuçları elde edebilmek için hiperparametre optimizasyonu yapılması gerekmektedir. Örneğin, Random Forest, Gradient Boosting ve SVM gibi modellerin başarısı, max_depth, learning_rate veya C gibi parametrelere bağlıdır.

Bu amaçla, GridSearchCV yöntemi kullanılmıştır. Bu yöntem:

Belirlenen parametre kombinasyonlarını sistematik olarak dener,

5 katlı çapraz doğrulama (k-fold CV) ile her kombinasyonu değerlendirir,

En yüksek doğruluk oranını veren modeli otomatik olarak seçer.

GridSearchCV, modelin aşırı öğrenme riskini azaltmak ve genelleme yeteneğini artırmak için etkili bir yöntemdir. Ancak işlem süresi fazla olduğundan yalnızca güçlü modeller için uygulanmıştır.

Denklem 3 Modeller ve Grid Search Örnek Kod

```
1.  models = {
        'LogisticRegression': LogisticRegression(max_iter=1000, C=1.0, solver='liblinear',
penalty='l2'),
        'RandomForest': RandomForestClassifier(n_estimators=200, max_depth=10,
min_samples_split=4, max_features='sqrt'),
        'ExtraTrees': ExtraTreesClassifier(n_estimators=150, max_depth=10),
        'Bagging': BaggingClassifier(estimator=DecisionTreeClassifier(max_depth=5),
n_estimators=100),
        'GradientBoosting': GradientBoostingClassifier(n_estimators=200, learning_rate=0.05,
max_depth=4, subsample=0.8),
        'AdaBoost': AdaBoostClassifier(n_estimators=200, learning_rate=0.5),
        'SVM': SVC(C=1.0, kernel='rbf', gamma='scale', probability=True),
        'KNN': KNeighborsClassifier(n_neighbors=5, weights='distance', metric='minkowski'),
        'NaiveBayes': GaussianNB(),
        'LDA': LinearDiscriminantAnalysis(solver='lsqr', shrinkage='auto'),
        'MLP': MLPClassifier(hidden_layer_sizes=(100, 50), activation='relu', solver='adam',
max_iter=1500, alpha=0.0005),
    }

    results = {}
    trained_models = {}

    # GridSearch örnekleri (sadece bazı güçlü modeller için çok zaman alıyor)
    param_grids = {
        'RandomForest': {
            'n_estimators': [100, 200],
            'max_depth': [None, 10],
            'max_features': ['sqrt'],
            'min_samples_split': [2, 4]
        },
        'GradientBoosting': {
            'n_estimators': [100, 200],
            'learning_rate': [0.05, 0.1],
            'max_depth': [3, 4]
        },
        'SVM': {
            'C': [0.5, 1.0],
            'kernel': ['rbf', 'linear'],
            'gamma': ['scale', 'auto']
        }
    }
```

Bölüm 7: Model Performans Karşılaştırması:

Makine öğrenmesi modellerinin başarısı, sadece doğruluk (accuracy) üzerinden değerlendirilmemelidir. Özellikle tıbbi tanı gibi kritik uygulamalarda, duyarlılık (recall) ve özgüllük (specificity) gibi metrikler hayati öneme sahiptir. Bu bölümde; projede kullanılan sınıflandırma modellerinin çeşitli performans ölçütleri üzerinden karşılaştırması yapılmıştır.

1.18 Doğruluk (Accuracy):

Doğru tahmin edilen örneklerin, toplam örnek sayısına oranıdır.

7.1.1 Formül:

Denklem 4 Doğruluk Denklemi

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{(\boldsymbol{TP} + \boldsymbol{TN} + \boldsymbol{FP} + \boldsymbol{FN})}$$

- Yüksek doğruluk değeri, modelin genel başarısının yüksek olduğunu gösterir.
- Ancak veri dengesizse yanıltıcı olabilir (örneğin, tüm hücreler iyi huyluysa %90 doğruluk anlamlı değildir).

1.19 Kesinlik (Precision):

Modelin kötü huylu olarak sınıflandırdığı hücrelerin, gerçekten kötü huylu olma oranıdır.

7.1.2 Formül:

Denklem 5 Kesinlik Denklemi

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}}$$

- Yanlış alarm oranını azaltmak istenen durumlarda önemlidir.
- Özellikle **yanlış pozitif** tanılar maliyetliyse kritik bir metriktir.

1.20 Duyarlılık (Recall / Sensitivity):

Modelin gerçekten kötü huylu olan hücreleri yakalama başarısıdır.

7.1.3 Formül:

Denklem 6 Hassasiyet Denklemi

$$\boldsymbol{Recall = \frac{TP}{TP + FN}}$$

- **Eksik tanı** (False Negative) oranını düşürmek için kullanılır.
- Kanser tespiti gibi vakalarda **hayati öneme sahiptir**.

1.21 . F1 Skoru (F1 Score):

Kesinlik ve duyarlılığın harmonik ortalamasıdır.

7.1.4 Formül:

Denklem 7 F1-Skor Denklemi

$$\boldsymbol{F1 = 2x\frac{Precision\ x\ Recall}{Precision + Recall}}$$

- Dengeli bir genel performans ölçüsüdür.
- Hem yanlış pozitif hem yanlış negatif hataların minimize edilmesi istendiğinde tercih edilir.

Bölüm 8: ROC-AUC Skoru:

ROC eğrisi (Receiver Operating Characteristic Curve), sınıflandırma modellerinin başarımını değerlendirmek için kullanılan bir grafiksel araçtır.

- Y ekseni (True Positive Rate- TPR): Duyarlılık (sensitivity veya recall) yani modelin pozitif sınıfı doğru tahmin etme oranı.

  Denklem 8 ROC Y-ekseni Duyarlılık Denklemi

$$\boldsymbol{TPR = \frac{TP}{TP + FN}}$$

- X ekseni (False Positive Rate- FPR): 1- özgüllük (specificity), yani modelin negatif sınıfı yanlışlıkla pozitif tahmin ettiği oran.

  Denklem 9 ROC X-ekseni Özgüllük Denklemi

$$\boldsymbol{FPR = \frac{FP}{FP + TN}}$$

- ROC eğrisi, modelin sınıflandırma eşiği değiştikçe oluşan TPR ve FPR değerleriyle çizilir.
- Her eşik değeri için farklı bir TPR-FPR çifti oluşur ve bu noktalar birleştirilerek ROC eğrisi elde edilir.

AUC (Area Under Curve) ise ROC eğrisinin altında kalan alanı temsil eder ve modelin genel ayrım gücünü ölçer.

- AUC = 1.0: Tüm pozitifleri negatiflerden tamamen ayırt eder.
- AUC = 0.5: Rastgele tahmin yapan model (hiçbir ayırt etme yeteneği yok).
- AUC <0.5: Model yanlış sınıflandırma yapıyor (negatifleri pozitif gibi sınıflıyor); bu durumda ters çevrilmiş sınıflarla daha iyi sonuç alınabilir.

1.22 ROC-AUC Ne İşe Yarar ve Neden Önemlidir?

- ROC-AUC, modelin sınıflar arasındaki ayrım yeteneğini gösterir. Sadece bir eşik değeri üzerinden değil, tüm olasılık eşiği değerleri dikkate alınarak performans ölçülür.
- Özellikle sınıf dengesizliği olan durumlarda ROC-AUC, doğruluk gibi yanıltıcı metriklere göre daha güvenilir sonuçlar verir.
- İkili sınıflandırma problemlerinde genellikle tercih edilen bir ölçüdür.

Bölüm 9: Confusion Matrix:

Confusion Matrix, bir sınıflandırma modelinin performansını değerlendirmek için kullanılan bir tablodur. Bu tablo, modelin doğru ve yanlış tahminlerini gerçek sınıflarla karşılaştırarak gösterir. İkili (binary) sınıflandırma problemlerinde dört ana bileşeni vardır:

- TP (True Positive): Kanserli hücreyi doğru tanıma
- TN (True Negative): İyi huylu hücreyi doğru tanıma
- FP (False Positive): Yanlış şekilde kanser teşhisi koyma
- FN (False Negative): Kanserli hücreyi gözden kaçırma

Denklem 10 Confusion Matrix Örnek Kod

```
1.  # Confusion Matrix
        cm = confusion_matrix(y_test, y_pred)
        tn, fp, fn, tp = cm.ravel()

        # Specificity ve Sensitivity hesapla
        specificity = tn / (tn + fp) if (tn + fp) > 0 else 0
        sensitivity = tp / (tp + fn) if (tp + fn) > 0 else 0

        # Modeli kaydet
        model_path = f'models/trained_models/{name}.joblib'
        joblib.dump(best_model, model_path)

        # Sonuçları sakla (genişletilmiş)
        results[name] = {
            'accuracy': accuracy,
            'precision': report['weighted avg']['precision'],
            'recall': report['weighted avg']['recall'],
            'f1': report['weighted avg']['f1-score'],
            'roc_auc': roc_auc,
            'specificity': specificity,
            'sensitivity': sensitivity,
            'confusion_matrix': {
                'tn': int(tn), 'fp': int(fp),
                'fn': int(fn), 'tp': int(tp)
            }
        }
```

1.23 Örnek Yorumlama:

Tablo 1 Örnek Confusion Matrix

	Gerçek Pozitif (Kanser)	Gerçek Negatif (İyi Huylu)
Tahmin: Pozitif	90 TP	10 FP
Tahmin: Negatif	5 FN	95 TN

Bu durumda:

- Accuracy = (90 + 95) / 200 = 0.925
- Precision = 90 / (90 + 10) = 0.9
- Recall = 90 / (90 + 5) = 0.947
- F1-Score ≈ 0.923

Görüldüğü gibi, 5 FN hatası çok düşük ve bu model güvenilir sayılabilir.

Bölüm 10: Model Karşılaştırması ve Yorumlar:

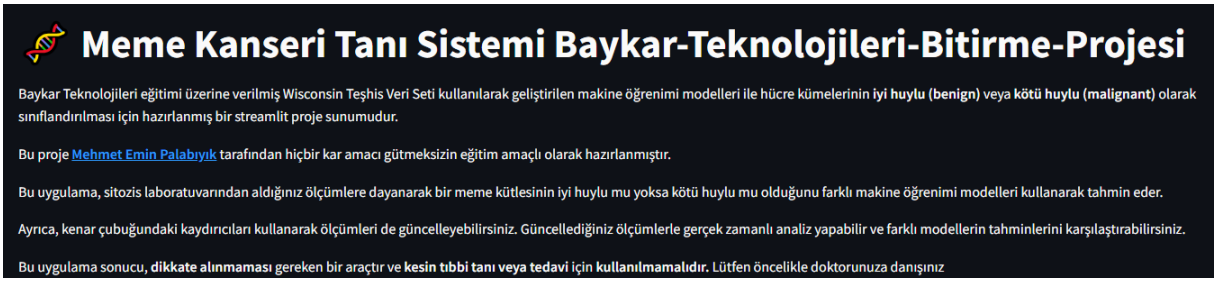
- **Logistic Regression:** Doğruluğu yüksek ancak sınırlı karmaşıklığa sahip olduğundan karmaşık sınırlarda performansı düşebilir.
- **Random Forest:** Hem doğruluk hem de ROC-AUC açısından en iyi modellerden biridir. Ensemble yapısı sayesinde aşırı öğrenme riski düşüktür.
- **SVM:** Özellikle iyi parametre ayarlandığında yüksek doğruluk ve AUC değeri sağlar.
- **KNN:** Komşu temelli olması nedeniyle küçük veri kümelerinde iyi çalışsa da, yüksek boyutlu veride etkisi azalır.
- **Gradient Boosting:** Duyarlılık ve F1 skorlarında güçlüdür. Özellikle yanlış negatifleri azaltma konusunda başarılıdır.
- **MLP (Yapay Sinir Ağı):** Non-lineer ilişkileri iyi öğrenir, ancak yüksek sayıda örnek ve parametre ayarı gerektirir.
- **Naive Bayes & LDA:** Basit yapıda olmalarına rağmen yüksek performans göstermiştir, özellikle zaman ve kaynak kısıtlı durumlar için idealdir.

```
models = {
        'LogisticRegression': LogisticRegression(max_iter=1000, C=1.0, solver='liblinear',
penalty='l2'),
        'RandomForest': RandomForestClassifier(n_estimators=200, max_depth=10,
min_samples_split=4, max_features='sqrt'),
        'ExtraTrees': ExtraTreesClassifier(n_estimators=150, max_depth=10),
        'Bagging': BaggingClassifier(estimator=DecisionTreeClassifier(max_depth=5),
n_estimators=100),
        'GradientBoosting': GradientBoostingClassifier(n_estimators=200, learning_rate=0.05,
max_depth=4, subsample=0.8),
        'AdaBoost': AdaBoostClassifier(n_estimators=200, learning_rate=0.5),
        'SVM': SVC(C=1.0, kernel='rbf', gamma='scale', probability=True),
        'KNN': KNeighborsClassifier(n_neighbors=5, weights='distance', metric='minkowski'),
        'NaiveBayes': GaussianNB(),
        'LDA': LinearDiscriminantAnalysis(solver='lsqr', shrinkage='auto'),
        'MLP': MLPClassifier(hidden_layer_sizes=(100, 50), activation='relu', solver='adam',
max_iter=1500, alpha=0.0005),
    }
```

Bölüm 11: Streamlit ile Geliştirilen Web Uygulaması:

Proje kapsamında geliştirilen web tabanlı kullanıcı arayüzü, kullanıcıların kolayca veri girişi yaparak farklı sınıflandırma modelleri üzerinden tahmin almasına ve aynı zamanda bu modellerin performanslarını kapsamlı şekilde karşılaştırmasına olanak sağlar. Uygulama, Python'un popüler web framework'ü Streamlit kullanılarak hazırlanmıştır.

1.24 Uygulama Giriş Ekranı:

- Proje Başlığı
- Projenin Geliştiricisi
- Projenin Amacı ve Kullanım Alanı: Bu bölümde projenin temel hedefi olan, hücre özelliklerine göre iyi ya da kötü huylu olup olmadığını tahmin etme süreci özetlenmiştir.
- 

Şekil 1 Meme Kanseri Tanı Sistemi

Denklem 11 Streamlit Anasayfa Örnek Kod

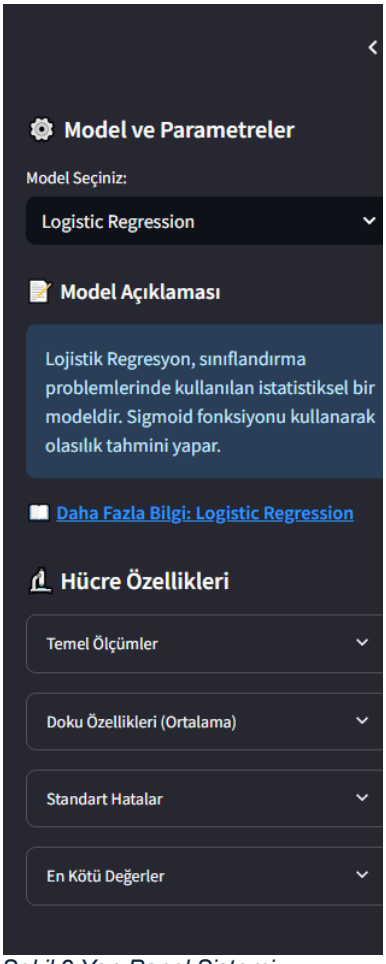
```
1.  # Streamlit sayfa ayarlarını yapılandırma
st.set_page_config(
    page_title="Meme Kanseri Tanı Sistemi",
    layout="wide",
    initial_sidebar_state="expanded",
    page_icon="□"
)

# Ana başlık ve açıklama bölümü
with st.container():
    st.title("□ Meme Kanseri Tanı Sistemi Baykar-Teknolojileri-Bitirme-Projesi")
    st.markdown("""
                Baykar Teknolojileri eğitimi üzerine verilmiş Wisconsin Teşhis Veri Seti
kullanılarak geliştirilen makine öğrenimi modelleri ile hücre kümelerinin
                **iyi huylu (benign)** veya **kötü huylu (malignant)** olarak sınıflandırılması
için hazırlanmış bir streamlit proje sunumudur. \n
                Bu proje [**Mehmet Emin Palabıyık**](https://www.linkedin.com/in/emin-
35bnry72698265/) tarafından hiçbir kar amacı gütmeksizin eğitim amaçlı olarak hazırlanmıştır. \n
                Bu uygulama, sitozis laboratuvarından aldığınız ölçümlere dayanarak bir meme
kütlesinin iyi huylu mu yoksa kötü huylu mu olduğunu farklı makine öğrenimi modelleri kullanarak
tahmin eder. \n
                Ayrıca, kenar çubuğundaki kaydırıcıları kullanarak ölçümleri de
güncelleyebilirsiniz.
                Güncellediğiniz ölçümlerle gerçek zamanlı analiz yapabilir ve farklı modellerin
tahminlerini karşılaştırabilirsiniz.\n
                Bu uygulama sonucu, **dikkate alınmaması** gereken bir araçtır ve **kesin tıbbi
tanı veya tedavi** için **kullanılmamalıdır.** Lütfen öncelikle doktorunuza danışınız\n
                """)
```

1.25 Sol Panel – Kullanıcı Etkileşim Alanı:

Uygulamanın sol tarafında bir kontrol paneli (sidebar) yer almaktadır. Bu bölümde:

- **Model Seçimi**: Kullanıcı, uygulamada bulunan farklı sınıflandırma modellerinden istediğini seçebilir (örneğin, Lojistik Regresyon, Random Forest, SVM vb.).
    - Seçilen model için açıklama alanı bulunmaktadır
- **Kullanıcı Verisi Girişi**: Hücreye ait ölçüm değerleri (örneğin radius_mean, texture_mean gibi) manuel olarak girilebilir. Bu girişler, gerçek zamanlı tahmin sürecini başlatır.
    - Hücre ölçümleri daha kolay kullanım için gruplara ayrılmıştır
    - Kullanıcı için "rastgele değer" ve değiştirilmiş değerleri eski haline getirmek için "orijinal değer" butonları eklenmiştir

Denklem 12 Yan Panel Örnek Kod




```
# Yan çubuk başlığı ve model seçim alanı
st.sidebar.header("⚙️ Model ve Parametreler")

# Kullanıcının ML modelini seçmesi için
dropdown menü
selected_model = st.sidebar.selectbox(
    "Model Seçiniz:",
    list(MODEL_DESCRIPTIONS.keys()),  # Mevcut
tüm model isimlerini listele
    format_func=lambda x: x  # Model ismini
olduğu gibi göster
)

# Seçilen modelin açıklamasını yan çubukta
gösterme
st.sidebar.markdown("### 📝 Model Açıklaması")
st.sidebar.info(MODEL_DESCRIPTIONS[selected_mod
el])

# Model hakkında daha detaylı bilgi için harici
link
st.sidebar.markdown(f"📖 **[Daha Fazla Bilgi:
{selected_model}]({MODEL_LINKS[selected_model]}
)**")

# Hücre özellikleri için slider'lar bölümü
st.sidebar.header("🔬 Hücre Özellikleri")
```

Daha detaylı kod için lütfen kaynak koduna bakınız.

- *Şekil 2 Yan Panel Sistemi*

1.26 Gerçek Zamanlı Analiz Sistemi:

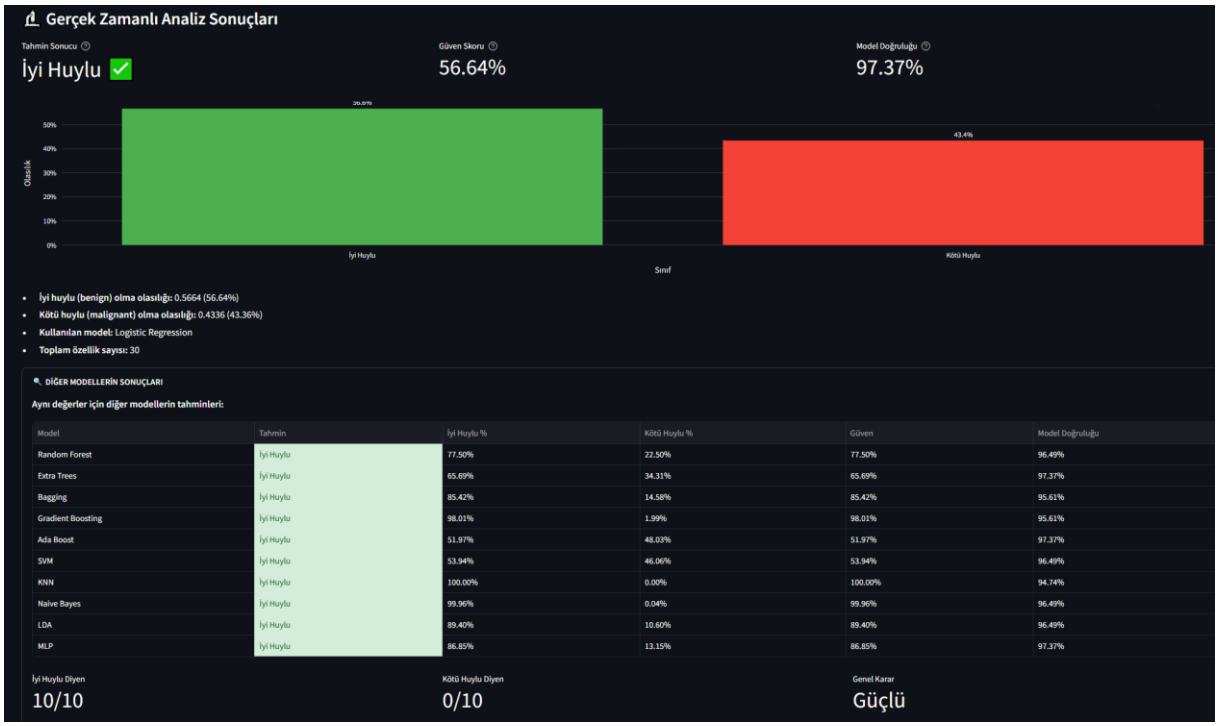
Bu sistem, kullanıcı veri girişi yaptıktan sonra seçilen model üzerinden anında tahmin üretir. Bu sayede kullanıcı:

- Hücrenin iyi huylu mu kötü huylu mu olduğunu,
- Seçilen modelin doğruluk oranını (% cinsinden),
- Kullanılan modelin adını ve yapısını görebilir.

1.27 Diğer Modellerin Sonuçlarını Gösteren Tablo:

Gerçek zamanlı analiz sonucunda, tüm modellerin aynı kullanıcı girdisi için ne tür bir çıktı verdiği tablo halinde gösterilir. Bu tablo sayesinde:

- Kullanıcı, sadece bir modele bağlı kalmadan diğer modellerin tahminlerini görebilir.
- Hangi modellerin daha tutarlı sonuçlar verdiği kolayca gözlemlenebilir.



Şekil 3 Gerçek Zamanlı Analiz

Örnek kod çok uzun olduğu için konulmamıştır, lütfen kaynak kodunu inceleyiniz.

Kodlarım herhangi bir problem(error) oluşmasına karşın hata-işleme(error-handling) mantığı ile dene-yakala(try-catch) sistemi ile yazılmıştır.

1.28 Model Radar-Chart Görselleştirmesi:

Bu grafik, kullanıcının girdiği hücre değerlerinin, normalleştirilmiş bir biçimde **çok boyutlu olarak görselleştirilmesini** sağlar. Bu kısımda "**Alejandro AO- Software & Ai**" isimli youtube kullanıcısının, "**Streamlit Python Course: Build a Machine Learning App to Predict Cancer"** videosundan esinlenilmiştir.

Neden Normalizasyon Yapıldı?

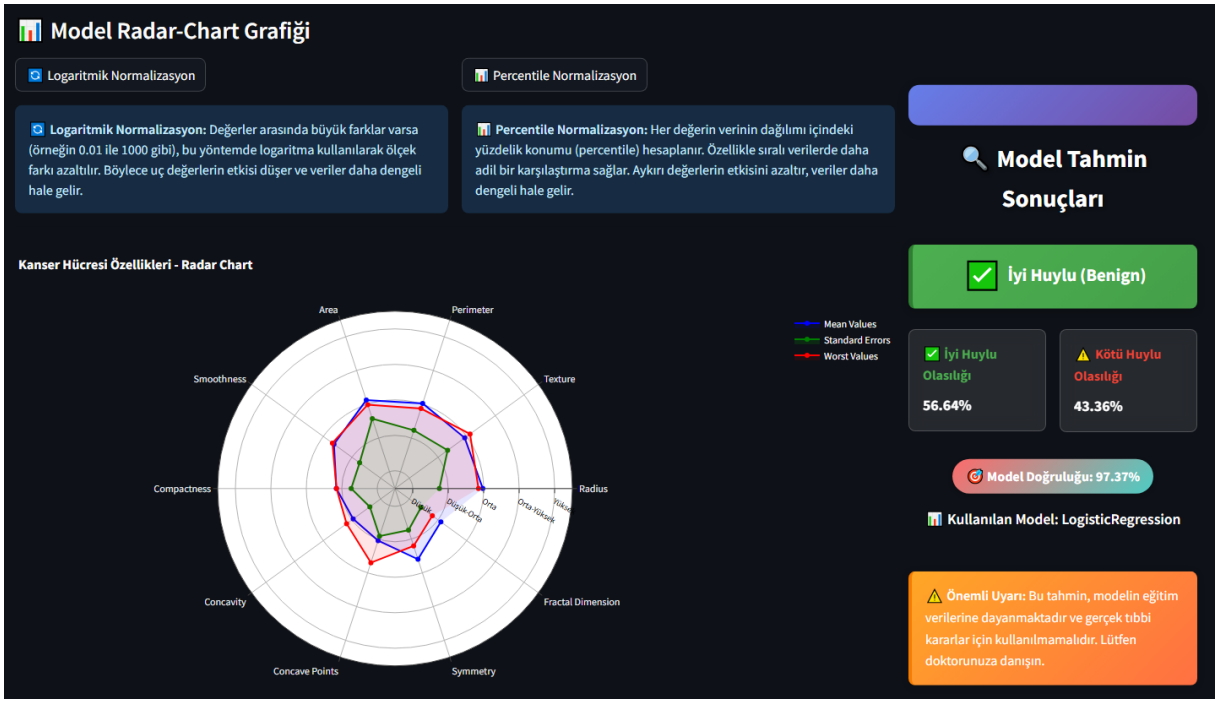
Girdi özellikleri arasında ciddi ölçek farkları bulunmaktadır. Örneğin:

- "fractal_dimension_mean" gibi değerler 0.82 gibi küçük aralıklarda,
- "area_worst" gibi değerler ise 2000'in üzerinde olabilir.

Bu durum, grafiklerde baskın özelliklerin diğerlerini gölgede bırakmasına neden olur. Bu nedenle:

- **Logaritmik Normalizasyon:** Özellikle geniş değer aralığına sahip öznitelikler için uygulanır. Bu yöntem, verinin logaritmasını alarak büyüklük farklarını küçültür.
- **Yüzdelik (Percentile) Normalizasyon:** Her bir özelliği %0–100 aralığına çeker. Bu sayede her bir özellik radar grafiğinde eşit görünürlük kazanır.

Bu iki normalizasyon, birlikte kullanılarak grafik okunabilirliği ve anlamlılığı artırılmıştır. Radar-Chart'ın hemen yanında, kullanıcıya özet bilgi veren bir panel bulunmaktadır. **Gerçek Zamanlı Analiz Sistemi kapatıldığında** bu kutucuk üzerinden de temel bilgiler erişilebilir olmaya devam eder.



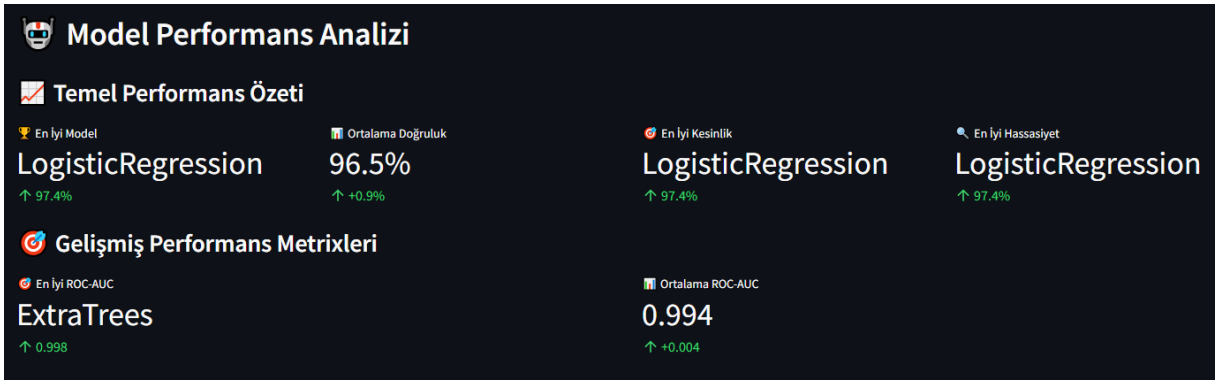
Şekil 4 Model Radar-Chart Grafiği

1.29 Model Performans Özeti:

Bu bölümde, projenin eğitim ve test süreçlerinden elde edilen **temel performans özetleri** sunulmaktadır:

- En Yüksek Doğruluğa Sahip Model
- Ortalama Doğruluk Oranı
- En İyi Kesinlik (Precision) Değeri
- En Yüksek Ortalama ROC-AUC Skoru

Bu özet, kullanıcıya hangi modelin hangi metrikte öne çıktığını kısa sürede gösterir.



Şekil 5 Performans Analizi

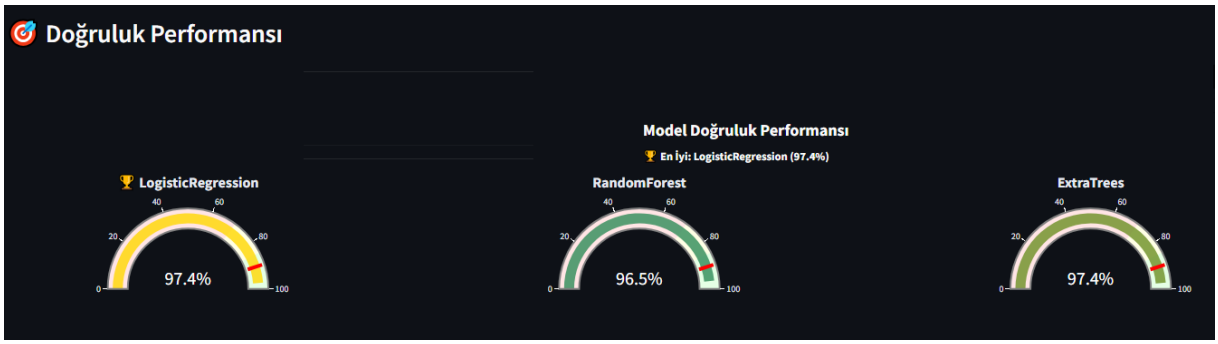
Denklem 13 Model Performans Örnek Kod

```
def create_performance_summary_cards(metrics_df):
    # En yüksek doğruluğa sahip modeli al
    best_model = metrics_df.loc[metrics_df['Doğruluk'].idxmax()]
    # Ortalama doğruluğu hesapla
    avg_accuracy = metrics_df['Doğruluk'].mean()
    # En yüksek kesinliğe sahip modeli al
    best_precision = metrics_df.loc[metrics_df['Kesinlik'].idxmax()]
    # En yüksek hassasiyete sahip modeli al
    best_recall = metrics_df.loc[metrics_df['Hassasiyet'].idxmax()]

    # 4 sütunlu kart yapısı oluştur
    col1, col2, col3, col4 = st.columns(4)

    with col1:
        # En iyi modeli ve doğruluğunu göster
        st.metric(
            label="🏆 En İyi Model",
            value=best_model['Model'],
            delta=f"{best_model['Doğruluk']:.1%}"
        )

    with col2:
        # Ortalama doğruluk ve en iyi modelle farkı göster
        st.metric(
            label="📊 Ortalama Doğruluk",
            value=f"{avg_accuracy:.1%}",
            delta=f"{best_model['Doğruluk'] - avg_accuracy:+.1%}"
        )
     # ......... kod devamı kaynak kodlarında bulunmaktadır.
```

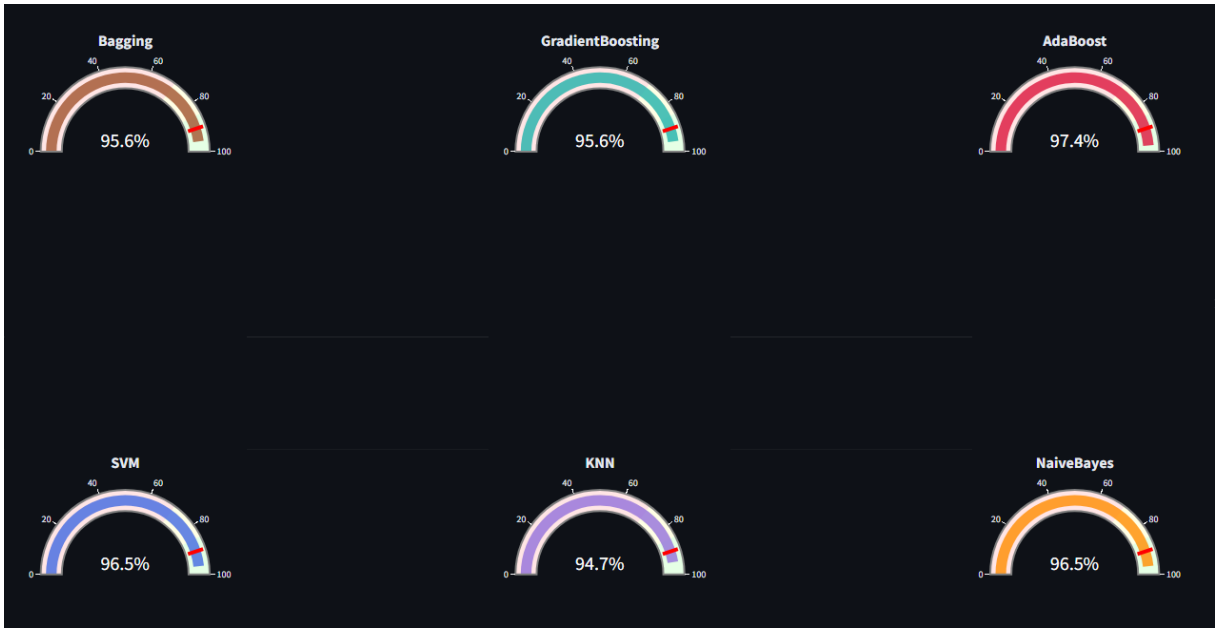
1.30 Doğruluk Performansı (Quart Chart):

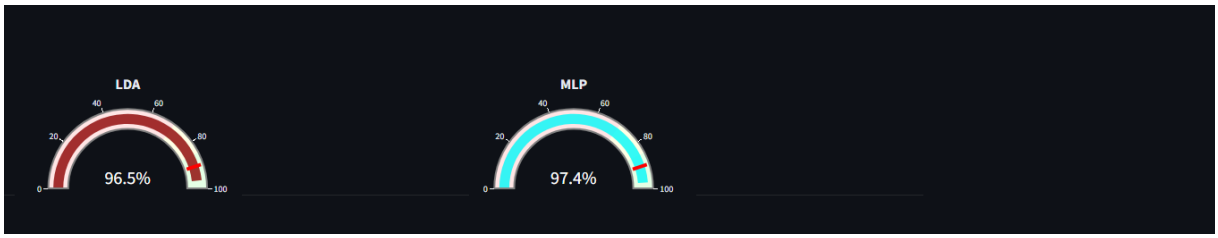
Bu grafik, uygulamada kullanılan tüm modellerin doğruluk oranlarını tek bir görselde karşılaştırmalı olarak sunar.

- **Quart Bar Chart** kullanılarak sade bir görünüm sağlanmıştır.
- En başarılı modeller üstte vurgulanır.



Şekil 6 Doğruluk Performansı

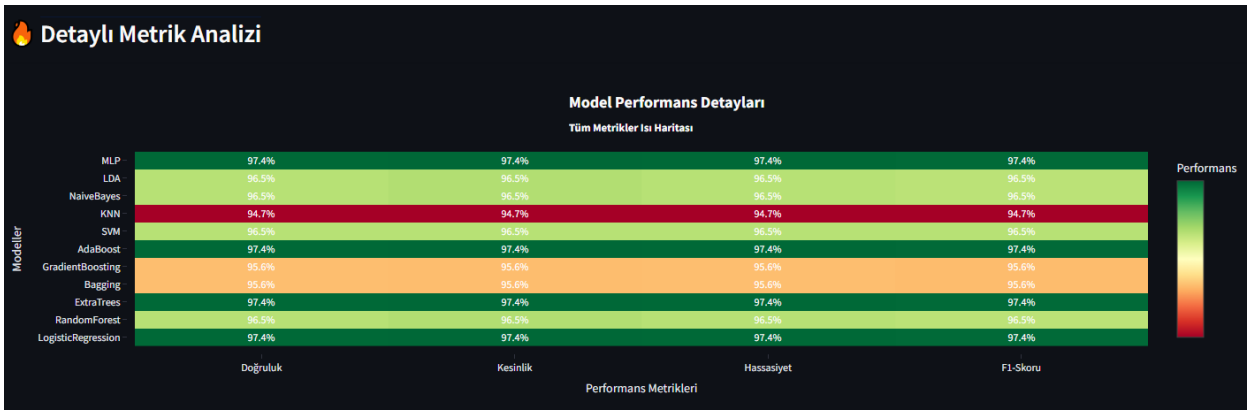




1.31 Detaylı Metrik Analiz (Yatay Bar Chart):

Modellerin **Doğruluk, Kesinlik, Duyarlılık ve F1 Skoru** gibi metrikler bazında karşılaştırılması, yatay çubuk grafiklerle görselleştirilmiştir.

- Her bir modelin güçlü ve zayıf yönlerini kolayca ayırt edebilir.
- Hangi metriğe göre seçim yapmak istiyorsa, doğrudan o metrik üzerinden karar verebilir.



Şekil 7 Metrik Analizi

Denklem 14 Metrik Analizi Örnek Kod

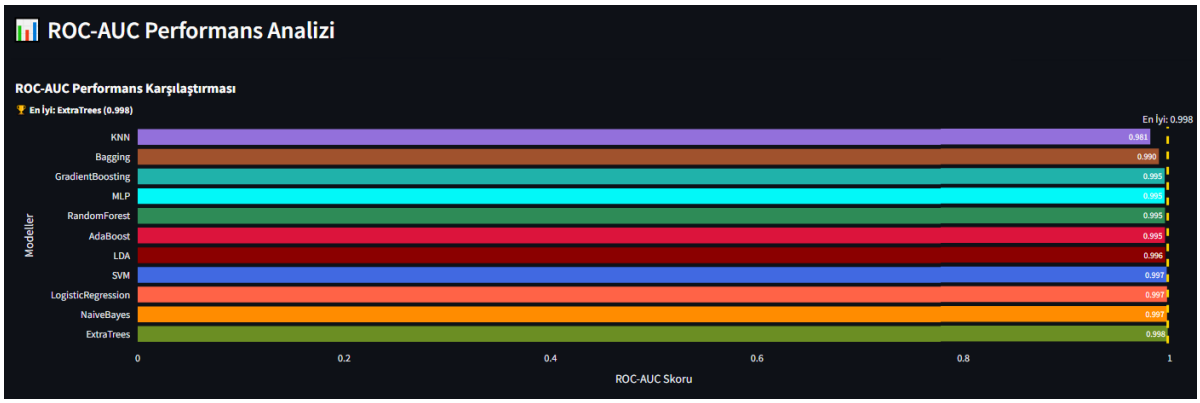
```
1.  def create_detailed_metrics_heatmap(metrics_df):
    metrics_only = metrics_df.set_index('Model')[['Doğruluk', 'Kesinlik', 'Hassasiyet', 'F1-
Skoru']]
    fig = go.Figure(data=go.Heatmap( #Isı haritası oluştur
        z=metrics_only.values,
        x=metrics_only.columns,
        y=metrics_only.index,
        colorscale='RdYlGn',
        text=[[f'{val:.1%}' for val in row] for row in metrics_only.values],
        texttemplate="%{text}",
        textfont={"size": 12, "color": "white"},
        colorbar=dict(
            title="Performans",
            tickvals=[0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
            ticktext=['50%', '60%', '70%', '80%', '90%', '100%']
        ),
        hoverongaps=False,
        hovertemplate="<b>%{y}</b><br>" +
                      "Metrik: %{x}<br>" +
                      "Değer: %{text}<br>" +
                      "<extra></extra>"
    ))
    fig.update_layout(
        title={
            'text': "<b>Model Performans Detayları</b><br><sub>Tüm Metrikler Isı Haritası</sub>",
            'x': 0.45,
            'font': {'size': 18}
        },
        xaxis_title="Performans Metrikleri",
        yaxis_title="Modeller",
        height=400,
        width=500
    )
return fig
```

1.32 ROC-AUC Performans Analizi:

Her modelin ROC-AUC skorlarını gösteren özel bir grafik yer almaktadır. AUC değeri, bir modelin sınıfları ne kadar iyi ayırt ettiğini gösterir.

Bu grafik sayesinde:

- Kullanıcı, en güvenilir sınıflandırmayı hangi modelin yaptığını tespit edebilir.



Şekil 8 ROC-AUC Performans Analizi

Denklem 15 ROC-AUC Performans Örnek Kod

```
1.  def create_roc_auc_bar_chart(metrics_df, model_colors):
    model_results = load_model_results()
    roc_data = []
    for model_name, results in model_results.items():
        if 'roc_auc' in results and results['roc_auc'] is not None:
            roc_data.append({
                'Model': model_name,
                'ROC-AUC': results['roc_auc']
            })

    fig.update_traces(
        texttemplate='%{text:.3f}',
        textposition='inside',
        textfont=dict(color='white', size=12)
    )
    fig.update_layout(
        xaxis_title="ROC-AUC Skoru",
        yaxis_title="Modeller",
        height=400,
        showlegend=False,
        xaxis=dict(range=[0, 1]),
        margin=dict(t=80, b=40, l=120, r=40)
    )

    fig.add_vline(
        x=best_roc_model['ROC-AUC'],
        line_dash="dash",
        line_color="gold",
        line_width=3,
        annotation_text=f"En İyi: {best_roc_model['ROC-AUC']:.3f}",
        annotation_position="top"
    )

    return fig
```

1.33 Confusion Matrix Analizi:

Bu bölümde, kullanıcı **birden fazla modeli seçerek**, her biri için confusion matrix'lerini görebilir.

- Gerçek ve tahmin edilen sınıfların dağılımı sunulur.
- Yanlış sınıflandırmalar (False Positive / False Negative) doğrudan izlenebilir.
- Her modelin hata eğilimlerini doğrudan görebilir.
- Yanlış negatiflerin hangi modelde daha az olduğunu karşılaştırabilir.
- Doğruluk (Accuracy) yerine daha hassas metrikler olan Recall / Sensitivity gibi ölçütleri öne çıkarabilir.
- Model seçiminde daha bilinçli kararlar alabilir; sadece yüksek doğruluk oranı değil, kritik hata türleri de değerlendirilmiş olur.

Bu analiz, özellikle yanlış negatif oranını düşük tutmak isteyen kullanıcılar için faydalıdır.

Şekil 9 Confusion Matrix Analizi

1.34 Uyarı ve Yasal Bilgilendirme:

Sayfanın en altında dikkat çekici bir uyarı yer almaktadır:

- “Bu uygulama, yalnızca eğitim, araştırma ve demonstrasyon amacıyla geliştirilmiştir.
- Sunulan tahminler, makine öğrenimi algoritmaları tarafından sağlanan *istatistiksel tahminlerdir* ve **herhangi bir şekilde kesin tıbbi teşhis, tedavi önerisi veya profesyonel sağlık hizmeti** olarak yorumlanmamalıdır.
- Lütfen bu uygulamadan elde ettiğiniz sonuçları kendi başınıza tıbbi bir karar vermek için kullanmayınız.
- Özellikle sağlık durumunuzla ilgili endişeleriniz varsa, mutlaka **nitelikli bir sağlık profesyoneline veya doktora danışınız**.
- Uygulama geliştiricisi, bu yazılımın kullanımından doğabilecek doğrudan veya dolaylı sonuçlardan **hiçbir şekilde sorumlu değildir”
- Bu açıklama, etik ve yasal sorumluluklardan kaçınmak adına uygulamanın en önemli bileşenlerinden biridir.”

Denklem 16 Uyarı Örnek Kod

```
st.warning("""
🕵 **Önemli Uyarı:**
Bu uygulama, yalnızca eğitim, araştırma ve demonstrasyon amacıyla geliştirilmiştir.
Sunulan tahminler, makine öğrenimi algoritmaları tarafından sağlanan *istatistiksel tahminlerdir*
ve **herhangi bir şekilde kesin tıbbi teşhis, tedavi önerisi veya profesyonel sağlık hizmeti**
olarak yorumlanmamalıdır.
Lütfen bu uygulamadan elde ettiğiniz sonuçları kendi başınıza tıbbi bir karar vermek için
kullanmayınız.
Özellikle sağlık durumunuzla ilgili endişeleriniz varsa, mutlaka **nitelikli bir sağlık
profesyoneline veya doktora danışınız**.
Uygulama geliştiricisi, bu yazılımın kullanımından doğabilecek doğrudan veya dolaylı sonuçlardan
**hiçbir şekilde sorumlu değildir**.
""")
```

Bölüm 12: Referanslar:

Mangasarian, O. L., Street, W. N., & Wolberg, W. H. (1995). Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), 570–577.

Polat, K., & Güneş, S. (2007). Breast cancer diagnosis using least square support vector machine. Digital Signal Processing, 17(4), 694–701.

Breast Cancer Wisconsin (Diagnostic) Data Set

Streamlit Python Course: Build a Machine Learning App to Predict Cancer

Radar Charts in Python