LOTUS AI

YAPAY ZEKA VE BİLİŞİM ÇÖZÜMLERİ A.Ş.

# PARKİNSON SINIFLANDIRMA RAPORU

HAZIRLAYANIN;

ADI: SENANUR
SOYADI: BAYRAM

27/11/2024

## Giriş

Parkinson hastalığı, dünya çapında milyonlarca insanı etkileyen ve nörolojik bir bozukluk olarak, titreme, kas sertliği, hareket yavaşlaması ve postural instabilite gibi semptomlarla kendini gösterir. Hastalığın erken teşhisi, tedaviye yönelik stratejilerin geliştirilmesi ve hastaların yaşam kalitesinin iyileştirilmesi açısından büyük önem taşımaktadır. Ancak, Parkinson hastalığının tanısı genellikle klinik değerlendirmelere dayalıdır ve bu süreçte doktorlar, genellikle hastanın semptomlarını gözlemleyerek veya çeşitli testler aracılığıyla hastalığı teşhis etmeye çalışmaktadır. Bununla birlikte, klinik testler her zaman doğru sonuçlar veremeyebilir ve bu nedenle doğru teşhis için daha güvenilir ve hızlı bir çözüm arayışı önem kazanmaktadır.

Bu bağlamda, makine öğrenimi tekniklerinin, özellikle sınıflandırma algoritmalarının, Parkinson hastalığının teşhisinde kullanılması büyük bir potansiyele sahiptir. Makine öğrenimi, geniş veri setlerinden öğrenme ve bu verilerle doğru tahminler yapma kapasitesine sahip olup, sağlık alanında da özellikle hastalık sınıflandırma ve teşhis süreçlerinde önemli bir rol oynamaktadır.

Bu çalışmada, Parkinson hastalığının sınıflandırılmasında kullanılan çeşitli makine öğrenimi algoritmalarını inceleyeceğiz. Kullanılan veri seti, hastaların çeşitli biyometrik ve sağlık parametrelerine ilişkin bilgileri içermektedir. Veri setinde yer alan değişkenler arasında yaş, cinsiyet, etnik köken, eğitim seviyesi, vücut kitle indeksi (BMI), sigara içme alışkanlıkları, alkol tüketimi, fiziksel aktivite düzeyi, uyku kalitesi gibi parametreler bulunmaktadır. Ayrıca, hastaların daha önceki sağlık durumlarına dair bilgiler de yer almakta, örneğin hipertansiyon, diyabet, depresyon, felç, ailesel Parkinson geçmişi gibi faktörler de değerlendirilmektedir.

Bu proje, Parkinson hastalığının sınıflandırılmasında beş farklı makine öğrenimi algoritmasını kullanarak sınıflandırma modellemesi yapmayı amaçlamaktadır. Bu algoritmalar Naive Bayes, Support Vector Machines (SVM), Karar Ağaçları, Random Forest ve Gradient Boosted Trees'tir. Her bir algoritmanın başarı oranları ve genel performansı değerlendirilerek, hangi algoritmanın Parkinson hastalığının sınıflandırılmasında en uygun sonuçları verdiği analiz edilmiştir.

Veri setindeki çeşitli özellikler ve hastaların sağlık durumları göz önüne alındığında, bu tür sınıflandırma yöntemleri, hastaların hastalığa dair belirtilerini daha hızlı ve doğru bir şekilde değerlendirebilmek için önemli bir araç sunmaktadır. Ayrıca, makine öğrenimi tekniklerinin uygulanabilirliği, erken teşhisle birlikte tedavi süreçlerinin iyileştirilmesine ve daha etkili sağlık stratejilerinin geliştirilmesine katkı sağlayacaktır.

Bu çalışma, hem KNIME platformu hem de Python programlama dili kullanılarak gerçekleştirilmiştir. KNIME, görsel bir analiz ortamı sunarak kullanıcıların veri akışlarını kolayca modellemelerini ve çeşitli algoritmalarla veriyi analiz etmelerini sağlar. Python ise, özellikle makine öğrenimi kütüphanelerinin gücünden yararlanarak, daha derinlemesine analizler ve özelleştirilmiş modellerin oluşturulmasına olanak tanır. Her iki yöntem de bu projede kullanılmakta olup, sonuçlar karşılaştırılmış ve her iki platformda da Parkinson hastalığının doğru şekilde sınıflandırılması sağlanmıştır.

Sonuç olarak, bu proje, Parkinson hastalığının sınıflandırılmasında makine öğrenimi tekniklerinin etkinliğini ve uygulama alanlarını gözler önüne sermektedir. Ayrıca, çeşitli sınıflandırma algoritmalarının karşılaştırılması, bu tür hastalıkların tanısında daha güvenilir ve hızlı çözümler üretmeye yönelik önemli bir adım atılmasına katkı sağlamaktadır.

# 1. KNIME


İşte adım adım Knime workflowu açıklayalım:

1. **CSV Reader (CSV Okuyucu)**: Bu düğüm, veri kümesini okur ve KNIME çalışma alanına getirir. Veriler genellikle bir CSV (Comma-Separated Values) dosyasından okunur. Bu düğümle, dışarıdan alınan veriler, KNIME üzerinde işlenmeye hazır hale gelir.

2. **Missing Value (Eksik Değerler)**: Bu düğüm, veri kümesindeki eksik veya boş (NaN) değerleri yönetir. Eksik değerler genellikle veri setinin bir kısmı eksik olduğunda veya bir hücrede bilgi bulunmadığında ortaya çıkar. Bu düğüm ile sayısal eksik verileri ortalama ile string verileri ise "none" ile dolduruldu.

3. **Normalizer (Normalizasyon)**: Bu düğüm, veri kümesindeki sayısal özelliklerin değer aralıklarını standardize eder. Özellikle makine öğrenimi algoritmalarında, farklı ölçeklerdeki özelliklerin normalize edilmesi, modelin daha doğru ve verimli çalışmasını sağlar. Bu problemde min-max kullanılarak geniş değer aralığına sahip sütunlar için normalizasyon yapılmıştır.

4. **Number to String (Sayılardan Metne Dönüştürme)**: Bu düğüm, sayısal değerleri metin formatına dönüştürür. Bazı algoritmaların, özellikle sınıflandırma gibi görevlerde, hedef değişkeni bir kategori olarak işlemeye ihtiyaç duyduğu durumlarda bu adım gereklidir. Örneğin bu problemde K-NN için string veriye ihtiyaç vardır.

5. **Partitioning (Veri Bölme)**: Bu düğüm, veriyi eğitim ve test veri setlerine böler. Eğitim seti, modelin eğitilmesi için kullanılırken, test seti modelin doğruluğunu test etmek için kullanılır. Bu problem için %80-%20 oranla veri ikiye bölünmüştür.

6. **Gradient Boosted Trees Learner (Gradient Boosted Trees Öğrenici)**: Bu düğüm, Gradient Boosting algoritmasını kullanarak model eğitir. Gradient Boosting, zayıf öğrenicilerden güçlü bir öğrenici oluşturmak için birden fazla karar ağacının sırasıyla eğitildiği bir yöntemdir. Bu model, doğruluğu yüksek sınıflandırmalar için etkili bir tekniktir.

7. **Gradient Boosted Trees Predictor (Gradient Boosted Trees Tahmin Edici)**: Bu düğüm, Gradient Boosting modelinin test verisi üzerinde tahminler yapmasını sağlar. Eğitilen model, yeni gelen veriler üzerinde sınıflandırma yaparak tahmin sonuçları üretir.

8. **Random Forest Learner (Random Forest Öğrenici)**: Random Forest algoritması, birden fazla karar ağacından oluşan topluluk yöntemidir. Her ağaç, verinin bir alt kümesiyle eğitilir ve sınıflandırmalar çoğunluk oylamasıyla yapılır. Bu düğüm, Random Forest modelini eğitir.

9. **Random Forest Predictor (Random Forest Tahmin Edici)**: Bu düğüm, eğitilmiş Random Forest modelini kullanarak test verisi üzerinde sınıflandırma tahminleri yapar.

10. **Decision Tree Learner (Karar Ağacı Öğrenici)**: Karar Ağacı algoritması, veri üzerinde yapılan bölünmelere göre sınıflandırma yapan bir modeldir. Bu düğüm, karar ağacını eğitir ve model oluşturur.

11. **Decision Tree Predictor (Karar Ağacı Tahmin Edici)**: Eğitilmiş karar ağacını kullanarak, test verisi üzerinde sınıflandırma tahminleri yapar.

12. **SVM Learner (Destek Vektör Makineleri Öğrenici)**: Bu düğüm, SVM algoritmasını kullanarak modelin eğitimini yapar. SVM, sınıflar arasındaki sınırı en iyi şekilde ayırmaya çalışan bir yöntemdir. Özellikle doğrusal olmayan sınıflandırmalar için etkilidir.

13. **SVM Predictor (Destek Vektör Makineleri Tahmin Edici)**: Bu düğüm, eğitilmiş SVM modelini kullanarak test verisi üzerinde tahmin yapar.

14. **Naive Bayes Learner (Naive Bayes Öğrenici)**: Naive Bayes algoritması, Bayes teoremine dayalı olarak çalışır ve özellikle büyük veri setlerinde etkili olur. Bu düğüm, Naive Bayes modelini eğitir.

15. **Naive Bayes Predictor (Naive Bayes Tahmin Edici)**: Eğitilen Naive Bayes modelini kullanarak test verisi üzerinde sınıflandırma tahminleri yapar.

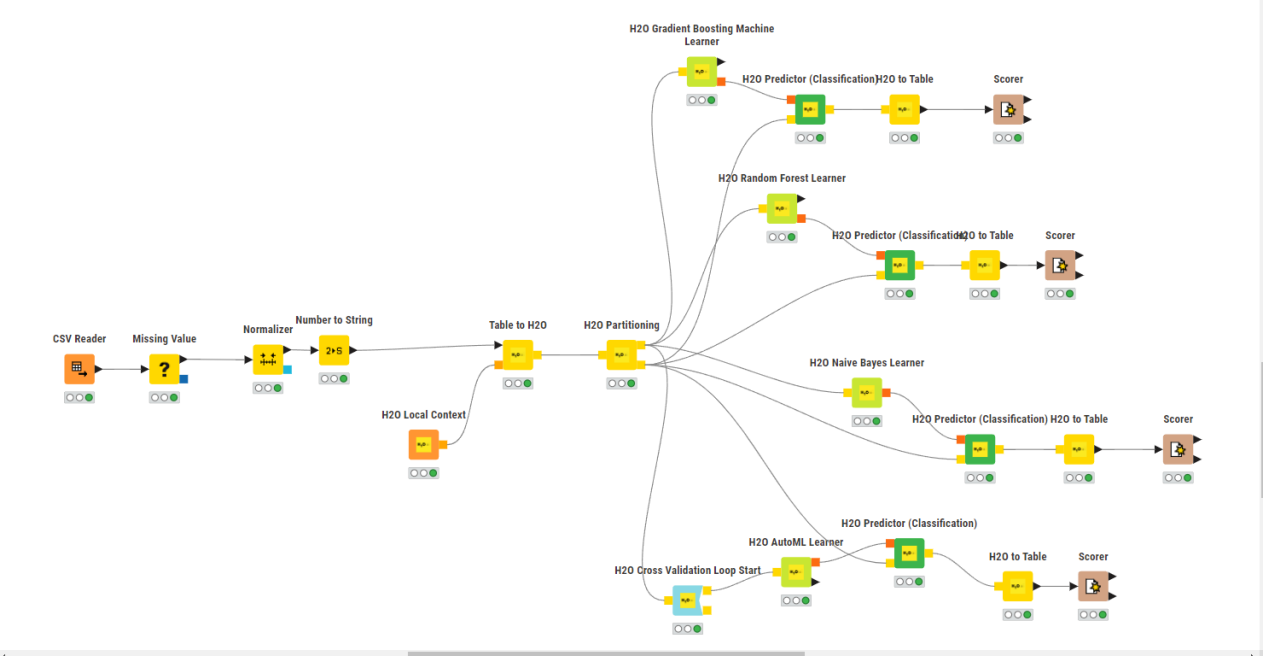
16. **K Nearest Neighbor (K En Yakın Komşu)**: KNN, her veri noktasının etrafındaki K en yakın komşusuna bakarak sınıflandırma yapan bir algoritmadır. Bu düğüm, KNN modelini kullanarak verileri sınıflandırır.

17. **Scorer (Değerlendirici)**: Bu düğüm, modelin doğruluğunu ve başarımını değerlendirir. Modelin tahmin sonuçlarını gerçek etiketlerle karşılaştırarak, doğruluk oranı gibi performans metriklerini hesaplar.

**Scorer Değerleri:**

| ALGORİTMALAR | HASTA | | SAĞLIKLI | |
|---|---|---|---|---|
| | Doğru Tahmin | Yanlış Tahmin | Doğru Tahmin | Yanlış Tahmin |
| Gradient BoostingTrees | 158 | 12 | 238 | 13 |
| Random Forest | 170 | 0 | 251 | 0 |
| K-NN | 95 | 75 | 200 | 51 |
| Decision Tree | 153 | 17 | 232 | 19 |
| Naive Bayes | 92 | 81 | 170 | 78 |

Bir de AutoML kullanarak tekrar sonuçları inceleyelim:




1. **H2O Gradient Boosting Machine (GBM) Learner ve Predictor**:
    - **Learner**: Gradient Boosting yöntemi, çoklu zayıf tahmincileri (decision trees) kullanarak güçlü bir tahmin modeli oluşturur. Bu, her bir ağacı önceki hataları düzeltmek için iteratif olarak optimize eder.
    - **Predictor**: GBM ile eğitilmiş modelden tahmin alır.
    - Çıkış **Scorer** düğümüne bağlanarak doğruluk ve **Confusion Matrix** hesaplanır.

2. **H2O Random Forest Learner ve Predictor**:
    - **Learner**: Random Forest algoritması, birçok karar ağacı (decision tree) oluşturur ve nihai tahmin için bu ağaçların ortalamasını (regresyonda) veya oy çokluğunu (sınıflandırmada) kullanır.
    - **Predictor**: Bu model, Random Forest ile tahminler üretir.
    - **Scorer** düğümü ile tahmin sonuçları değerlendirilir ve modelin doğruluğu ölçülür.

3. **H2O Naive Bayes Learner ve Predictor**:
    - **Learner**: Naive Bayes algoritması, olasılık temelli bir yöntemdir ve özellikle metin sınıflandırma gibi görevlerde yaygın kullanılır. Veri özelliklerinin birbirinden bağımsız olduğunu varsayar.
    - **Predictor**: Naive Bayes ile tahminler yapılır.
    - Yine, sonuçlar **Scorer** ile değerlendirilir.

4. **H2O AutoML Learner ve Predictor**:
    - **Learner**: Bu düğüm, H2O'nun otomatik makine öğrenimi (AutoML) özelliğini kullanır. Farklı algoritmalar arasında seçim yapar, hiperparametre optimizasyonu yapar ve en iyi modeli belirler.
    - **Cross-Validation Loop Start**: AutoML, k-katlı çapraz doğrulama ile çalışır. Bu, modeli farklı veri parçalarında test ederek daha genel bir performans değerlendirmesi sağlar.
    - **Predictor**: AutoML tarafından seçilen en iyi modeli kullanarak tahminler yapılır.
    - Çıktı, diğer modellerle aynı şekilde **Scorer** ile değerlendirilir.

5. **Table to H2O ve H2O Partitioning**:
    - **Table to H2O**: Veriler H2O formatına dönüştürülür. H2O düğümleri sadece H2O veri formatını kabul ettiği için bu dönüşüm zorunludur.
    - **H2O Partitioning**: Veriyi eğitim ve test setlerine ayırır. Bu düğüm, tüm modellerin aynı eğitim ve test setiyle çalışmasını sağlar.

6. **Scorer Düğümleri**:
    - Her bir öğrenme yöntemi için tahmin sonuçları **Scorer** düğümüne bağlanır. **Scorer** düğümü, modelin doğruluğunu, hatalarını ve **Confusion Matrix**'i hesaplar.
    - Her modelin performansı bağımsız olarak değerlendirilir ve karşılaştırılabilir.

**Bu Workflow'un Avantajı:**

- Farklı algoritmaların performansını aynı veri seti üzerinde test ederek kıyaslama yapmanızı sağlar.
- AutoML, manuel olarak hiperparametre optimizasyonu yapmadan en iyi modeli seçer.
- Çapraz doğrulama (Cross-Validation), modellerin genelleştirilebilirliğini artırır.

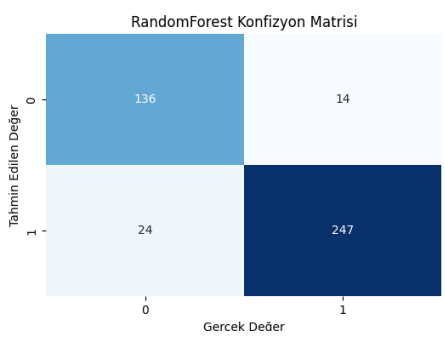
| ALGORİTMALAR | HASTA | | SAĞLIKLI | |
|---|---|---|---|---|
| | Doğru Tahmin | Yanlış Tahmin | Doğru Tahmin | Yanlış Tahmin |
| H2O Gradient Boosting | 209 | 54 | 331 | 2 |
| H2O Random Forest | 186 | 13 | 372 | 25 |
| H2O Naive Bayes | 108 | 21 | 364 | 103 |
| H2O AutoML Learning | 194 | 13 | 372 | 17 |

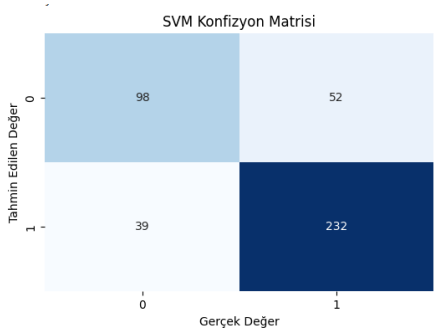
Sonuçları elde edilmiştir.

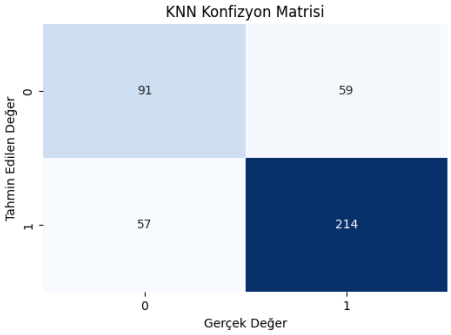
## 2. PYTHON

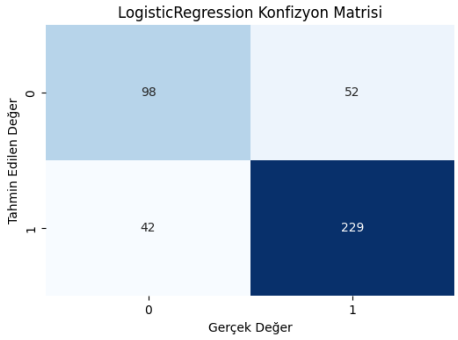
Projenin bu aşamasında dört farklı yol izlenerek sonuçlar elde edilmiştir. Bunlardan ilki sınıflandırıcı + hiperparametre optimizasyonu, ikincisi sınıflandırıcı + 5 fold cross validation + hiperparametre optimizasyonu, üçüncüsü sınıflandırıcı + 10 cross validation + hiperparametre optimizasyonu ve son olarak sınıflandırıcı + feature selection + 10 fold cross validation + hiperparametre optimizasyonudur.

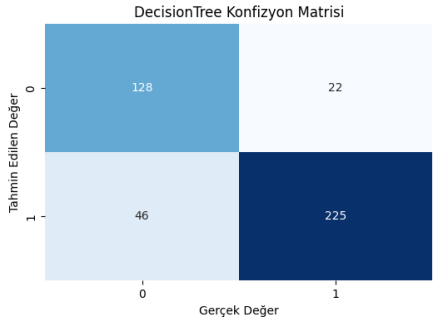
**1.Yol:** Bu yol parkinson hastalığı veri setini kullanarak çeşitli sınıflandırma algoritmalarının performansını değerlendirmek için tasarlanmıştır. İlk olarak, veri seti okunur ve hedef değişken ile bağımsız değişkenler ayrılır. Kategorik veriler sayısal verilere dönüştürülür ve veriler ölçeklendirilir, ardından eğitim ve test setlerine ayrılır. Belirli sınıflandırıcılar için hiperparametre optimizasyonu, GridSearchCV kullanılarak gerçekleştirilir. Her model, en iyi hiperparametrelerle eğitilir ve test verisi üzerindeki tahminleri değerlendirilir. Sonuçlar, her model için konfizyon matrisleri ile birlikte doğruluk skoru olarak yazdırılır. Ayrıca, her modelin konfizyon matrisleri görselleştirilir. Nihayetinde, en yüksek doğruluk oranına sahip model belirlenir ve kullanıcıya sunulur. Bu süreç, modelin performansını karşılaştırmaya ve en iyi sınıflandırıcıyı seçmeye olanak tanır. Sonuçlar ise şu şekildedir:

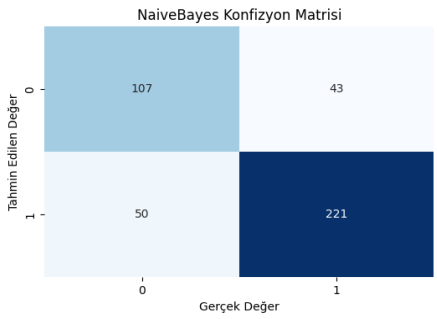


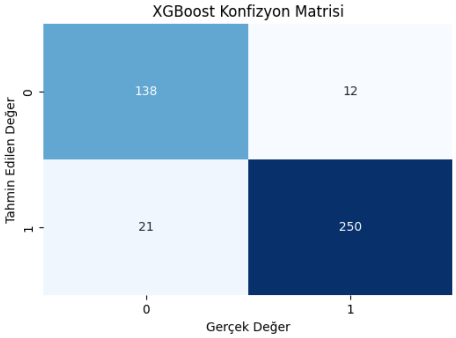



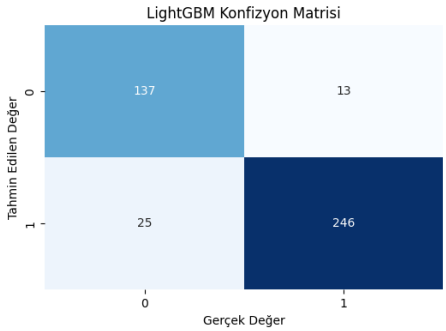



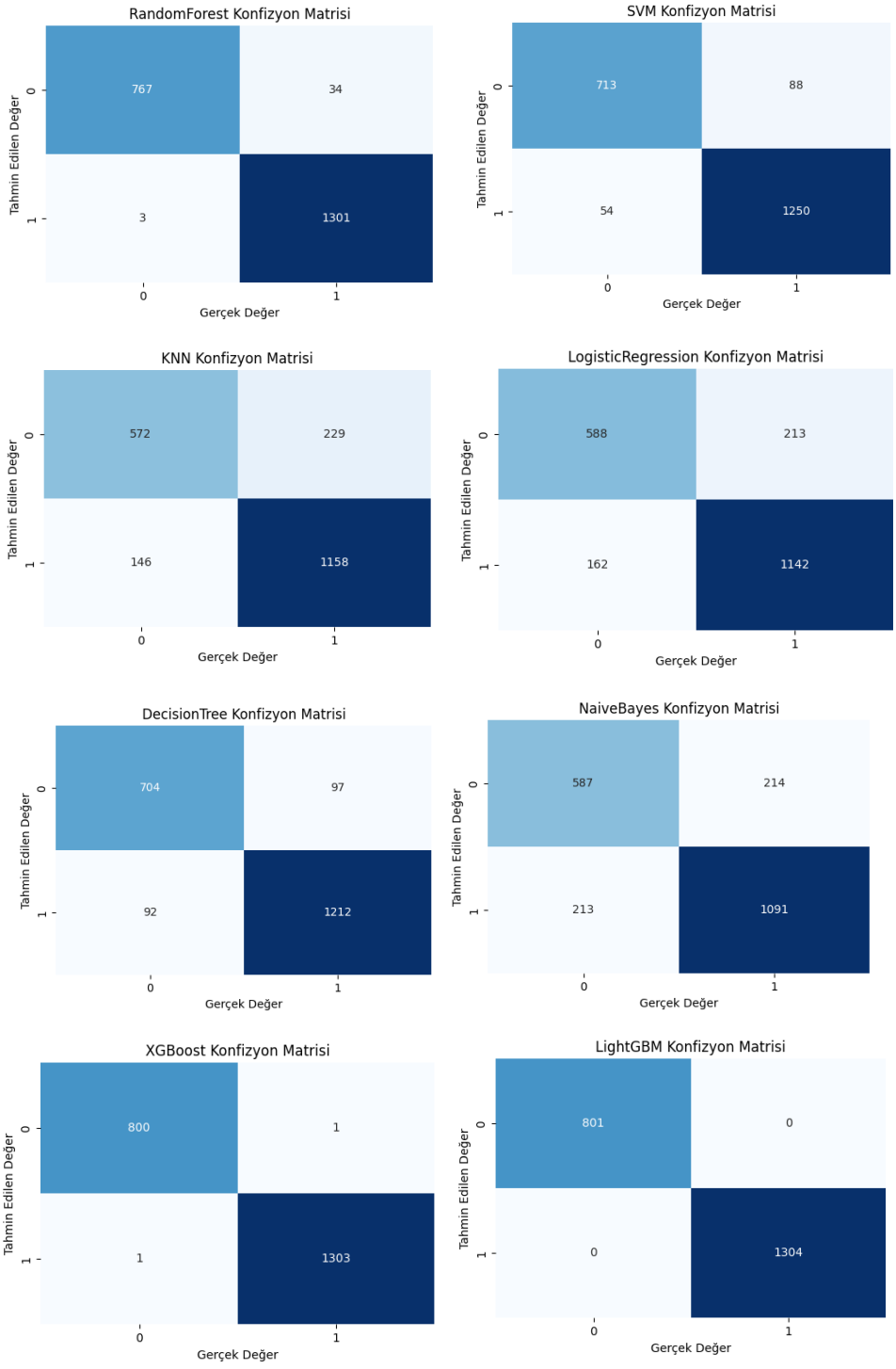




| **Sınıfladnırıcılar** | Random Forest | SVM | K-NN | Logistic Regression | Decision Tree | Naive Bayes | XGBoost | LightGBM |
|---|---|---|---|---|---|---|---|---|
| **Accuraccy** | 0.9097 | 0.7838 | 0.7245 | 0.7767 | 0.8385 | 0.7791 | 0.9216 | 0.9097 |

**2.Yol:** Yeni yolun önceki yoldan farkı, değerlendirme yöntemleri ve görselleştirmelerin daha kapsamlı bir şekilde uygulanmasıdır. Önceki yolda eğitim ve test verileri ayrılarak modeller, yalnızca bir test seti üzerinde doğruluk oranlarına göre değerlendirilirken, yeni yolda tüm veri üzerinde 5 katlı çapraz doğrulama yapılır. Bu, modelin farklı veri bölümleri üzerindeki performansını test ederek daha tutarlı ve güvenilir bir doğruluk ölçümü sağlar. Ayrıca, çapraz doğrulama doğruluk oranlarının ortalamaları hesaplanarak modeller arasında bir kıyaslama yapılır.

Yeni yol, hiperparametre optimizasyonu sürecini de geliştirir. GridSearchCV ile en iyi hiperparametreler her model için seçildikten sonra, bu modeller hem çapraz doğrulama hem de tüm veri üzerinde tahminler yaparak performanslarını ölçer. Ek olarak, her bir modelin sonuçları bir konfizyon matrisi ile görselleştirilir ve bu matriste doğru ve yanlış sınıflandırmalar kolayca analiz edilebilir. Sonuçlar ise şu şekildedir:

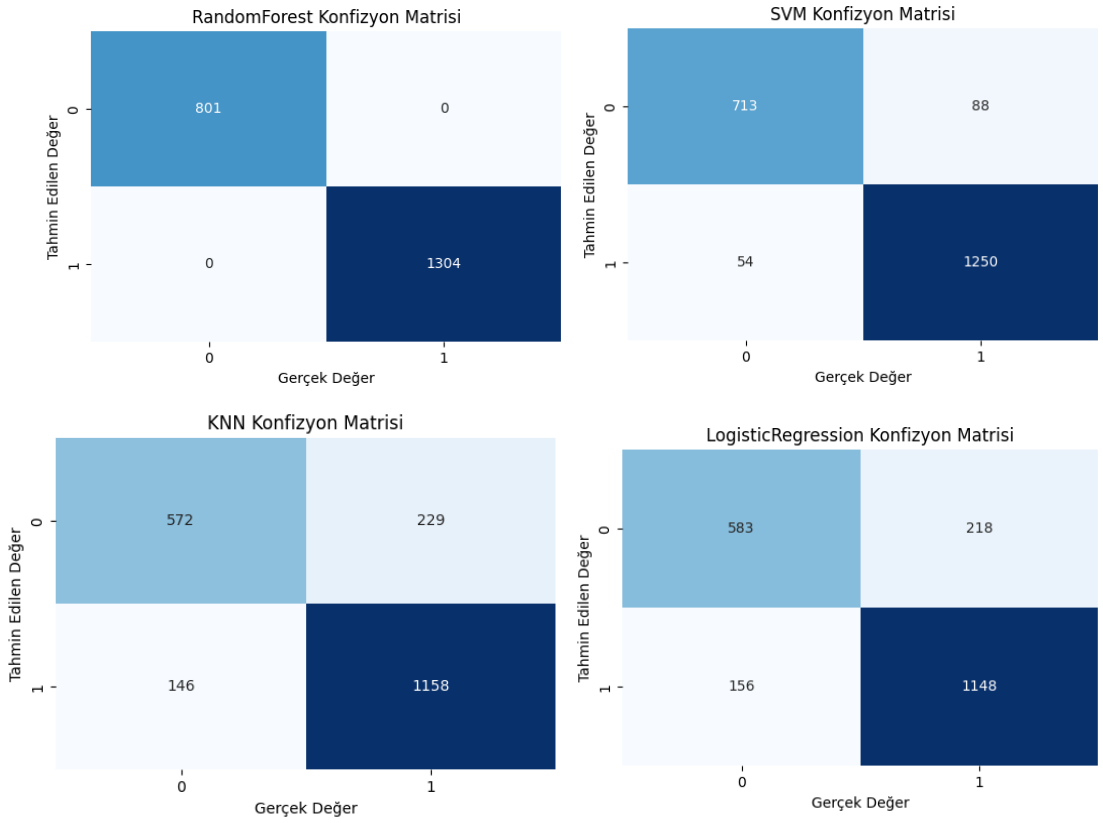


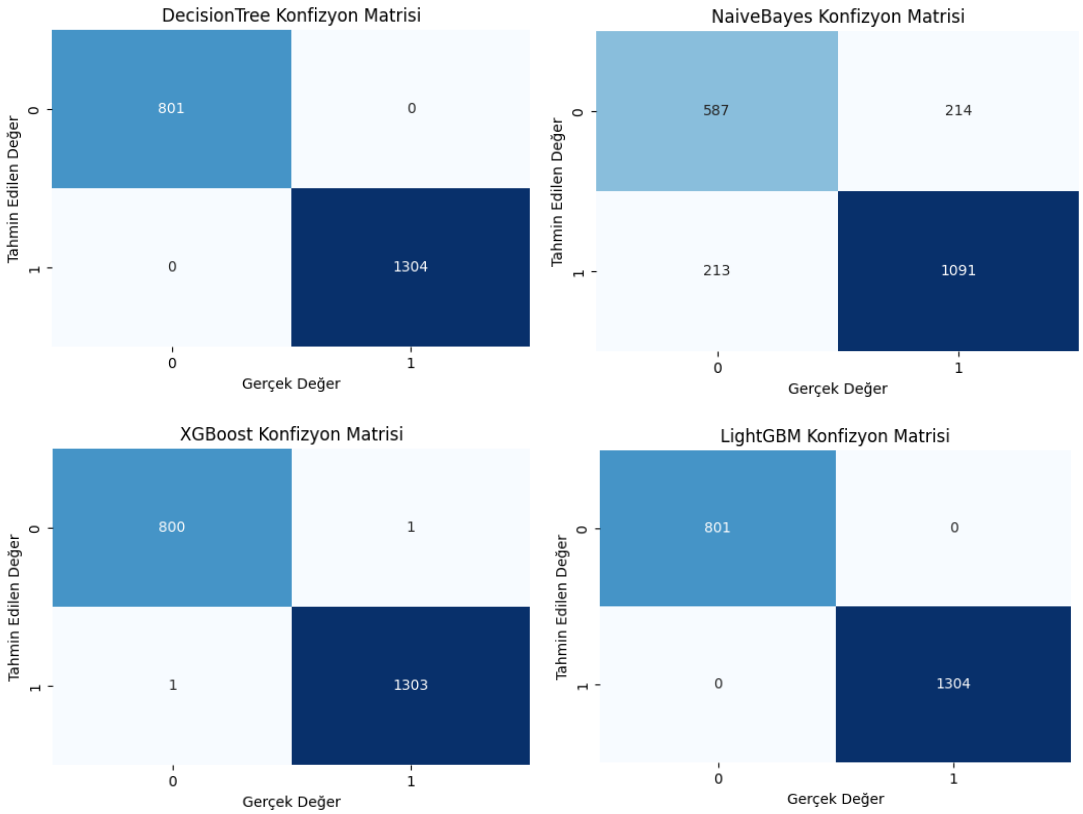
| Sınıfladnırıcılar | Random Forest | SVM | K-NN | Logistic Regression | Decision Tree | Naive Bayes | XGBoost | LightGBM |
|---|---|---|---|---|---|---|---|---|
| **Accuraccy** | 0.9221 | 0.8252 | 0.7387 | 0.8114 | 0.8803 | 0.7843 | 0.9230 | 0.9254 |

**3.Yol:** Bu yol, Parkinson hastalığı teşhisi için çeşitli makine öğrenmesi yollarını test etmek ve karşılaştırmak amacıyla yazılmıştır. İlk olarak, Parkinson hastalığı verisi parkinsons_disease_data.csv dosyasından yüklenir ve hedef yol olarak "Diagnosis" (teşhis) sütunu seçilir. Verinin özellikleri (X) ve hedef yolu (y) ayrıldıktan sonra, kategorik sütunlar sayısal yollara dönüştürülür ve veriler, özelliklerin farklı ölçeklerinden kaynaklanabilecek sorunları önlemek için standartlaştırılır.

Yolda, 10 katlı çapraz yol (cross-validation) kullanılarak, farklı makine öğrenmesi algoritmaları (RandomForest, SVM, KNN, LogisticRegression, DecisionTree, NaiveBayes, XGBoost, LightGBM) hiperparametre aralıklarıyla birlikte tanımlanır. GridSearchCV kullanılarak her bir yolun hiperparametreleri optimize edilir ve her yolun doğruluk skoru çapraz yol ile hesaplanır. Bu sayede, her yolun performansı daha güvenilir bir şekilde değerlendirilir.

Her bir yolun hiperparametrelerinin en iyisi bulunarak, yolun doğruluk skoru çapraz yol ile test edilir. Yolların performansları karşılaştırıldığında, her yol için en iyi hiperparametreler ve çapraz yol doğruluk oranları ekrana yazdırılır. Ayrıca, her yolun konfizyon matrisinin görselleştirilir. Konfizyon matrisi, yolun tahmin ettiği sınıfların doğruluğunu ve hangi sınıflarda hatalar yapıldığını gösterir. Son olarak, çapraz yol doğruluk skorlarına göre en iyi performans gösteren yol belirlenir ve bu yolun ismi ile doğruluk oranı ekrana yazdırılır. Bu süreç, Parkinson hastalığının teşhisi için en uygun yolun seçilmesini sağlar. Sonuçlar şu şekildedir:



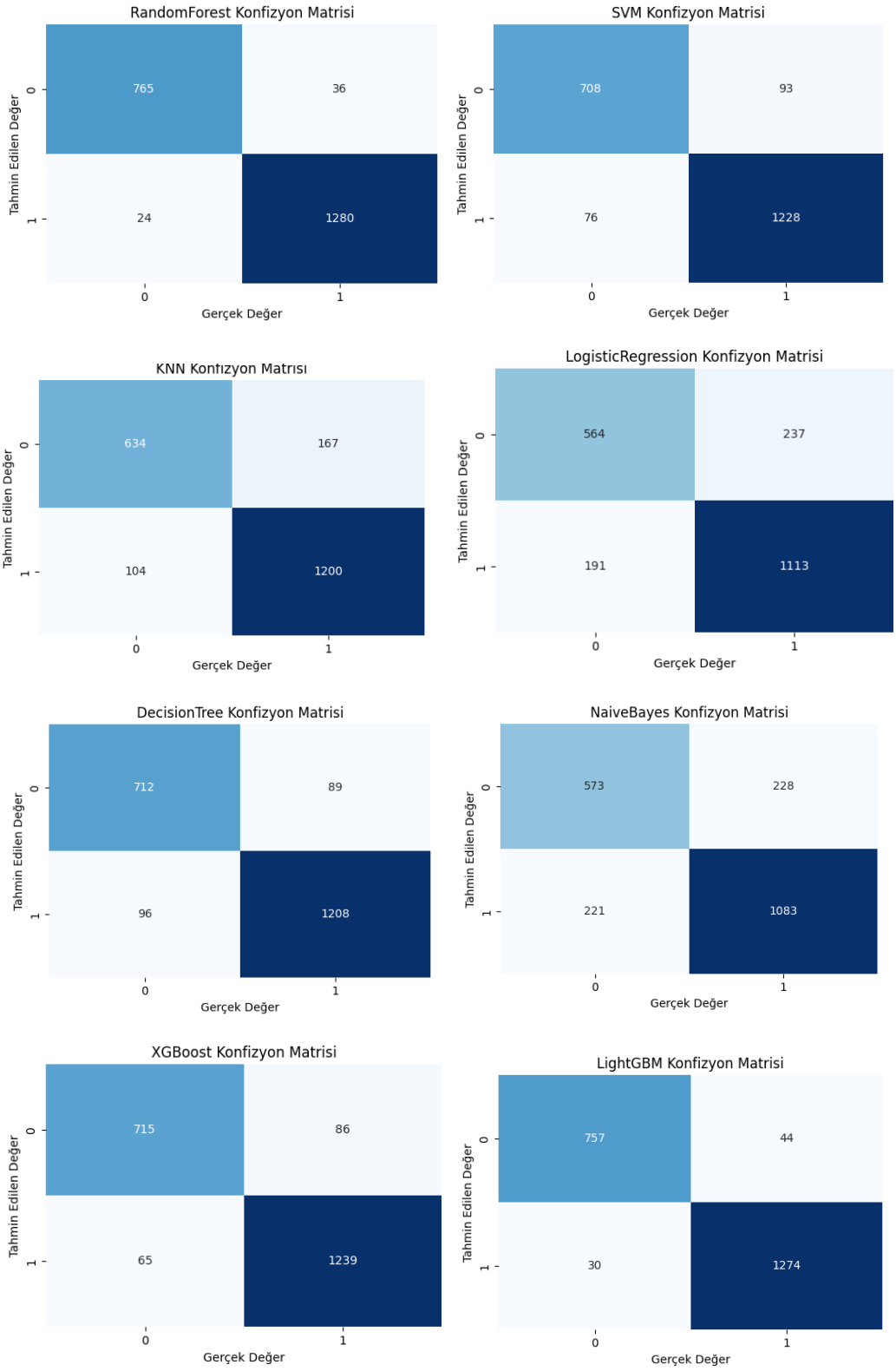



| **Sınıfladnırıcılar** | Random Forest | SVM | K-NN | Logistic Regression | Decision Tree | Naive Bayes | XGBoost | LightGBM |
|---|---|---|---|---|---|---|---|---|
| **Accuraccy** | 0.9201 | 0.8304 | 0.7387 | 0.8128 | 0.8698 | 0.7886 | 0.9268 | 0.9273 |

**4.Yol:** Bu yol, Parkinson hastalığının teşhisini yapmak için çeşitli makine öğrenmesi algoritmalarını test etmek, karşılaştırmak ve en iyi sonuçları veren modeli seçmek amacıyla yazılmıştır. İlk olarak, veri seti bir CSV dosyasından okunur ve hedef değişken ("Diagnosis") ile bağımsız değişkenler ayrılır. Gereksiz sütunlar ("PatientID" ve "DoctorInCharge") veri setinden çıkarılır. Kategorik veriler, algoritmalar tarafından işlenebilir hale getirilmek üzere sayısal verilere dönüştürülür. Ardından, tüm özellikler standartlaştırılır (ortalama sıfır, standart sapma bir) ve veri daha tutarlı hale getirilir.

Özellik çıkarımı aşamasında, veri setine temel istatistiksel bilgiler eklenir: her bir örneğin ortalaması, standart sapması, maksimum ve minimum değerleri hesaplanır ve bu özellikler modele dahil edilir. Ayrıca, PCA (Principal Component Analysis) ile verilerin boyutu azaltılır ve ilk 15 bileşen seçilerek veriye eklenir. Son olarak, SelectKBest kullanılarak en iyi 10 özellik seçilir.

Çeşitli makine öğrenmesi algoritmaları (Random Forest, SVM, KNN, Lojistik Regresyon, Karar Ağaçları, Naive Bayes, XGBoost ve LightGBM) için hiperparametre optimizasyonu yapılır. GridSearchCV kullanılarak her modelin en iyi hiperparametre kombinasyonu bulunur ve çapraz doğrulama (CV) ile her bir modelin doğruluğu ölçülür. En iyi doğruluk oranını veren modelin parametreleri ve performansı kaydedilir. Son olarak, her modelin konfizyon matrisi çizilerek görsel olarak değerlendirilir, böylece modelin hangi sınıflarda doğru ya da yanlış tahminler yaptığı analiz edilir. Sonuçlar ise şu şekildedir:




| **Sınıfladnırıcılar** | Random Forest | SVM | K-NN | Logistic Regression | Decision Tree | Naive Bayes | XGBoost | LightGBM |
|---|---|---|---|---|---|---|---|---|
| **Accuraccy** | 0.8997 | 0.8641 | 0.8328 | 0.7952 | 0.8907 | 0.7852 | 0.9054 | 0.9021 |