



**KAHRAMANMARAŞ ST İMAM NİVERSİTESİ  
BİLGİSAYAR MHENDİSLİĐİ**

**YAPAY ZEKÂ RAPORU**

17110131052 - Mehtap KL  
18110131011 - Banu KSE

## Özet

İnsanların yeni bilgileri öğrenmek ve karar almak için kullandığı düşünsel becerinin, matematik ve mantık kullanan bilgisayarlar tarafından simüle edilmesine yapay zeka denir. Yapay zeka, mevcut verilerle işlemler yapan, ardından hatalarından ders çıkartarak kendisini geliştiren sistemlerdir.

Temellerinin 1943'te atılmasıyla gelişmeye başlayan yapay zeka günümüzde hayatımızın her alanında bize kolaylık sağlamaktadır. İçerisinde makine öğrenmesi ve derin öğrenme disiplinlerini de barındırmaktadır. Daha hızlı ve daha doğru kararlar vermemize yardımcı olan yapay zeka, sürücüsüz araçlar, akıllı ev teknolojileri, sağlık verilerinin analizi gibi çeşitli alanlarda yer almaktadır. [1][2]

### 1. Giriş

Bu projede kırmızı şarabın, belli başlı özelliklerine bakılır. Bunlar; fixedacidity, volatileacidity, citricacid, residualsugar, chlorides, free sülfür dioxide, density, pH, sulphates, alcohol, quality. Bu özellikler sayesinde yapay sinir ağı eğitilerek, hangi şarabın kaliteli hangi şarabın kalitesiz olduğu belirlenmesi amaçlanmıştır.

Veri setimizde birden çok özellik bulunmaktadır. Yukarıda belirttiğim özelliklere bakılarak en son oluşturulan “quality” sütunun hepsini toplayıp ortalamasını aldıktan sonra o ortalamanın altında kalanları “0” kalitesiz, ortalamanın üzerinde kalanları “1” kaliteli olarak ayırdık. Böylece veri setimizi işlenecek duruma getirdik.

### 2. Ağ Eğitme

Yapay sinir ağlarında eğitme işlemi; veri setini tanımlama, ağ yapısını oluşturma ve eğitime sokarak belirli sonuçlar elde etme yöntemidir.

#### 2.1. Veri Seti

Veri seti Kaggle platformu üzerinden bulduğumuz açık, .csv uzantılı bir veri setidir. 12 öznitelikten oluşmakta olup toplam 1599 veri girişi vardır.

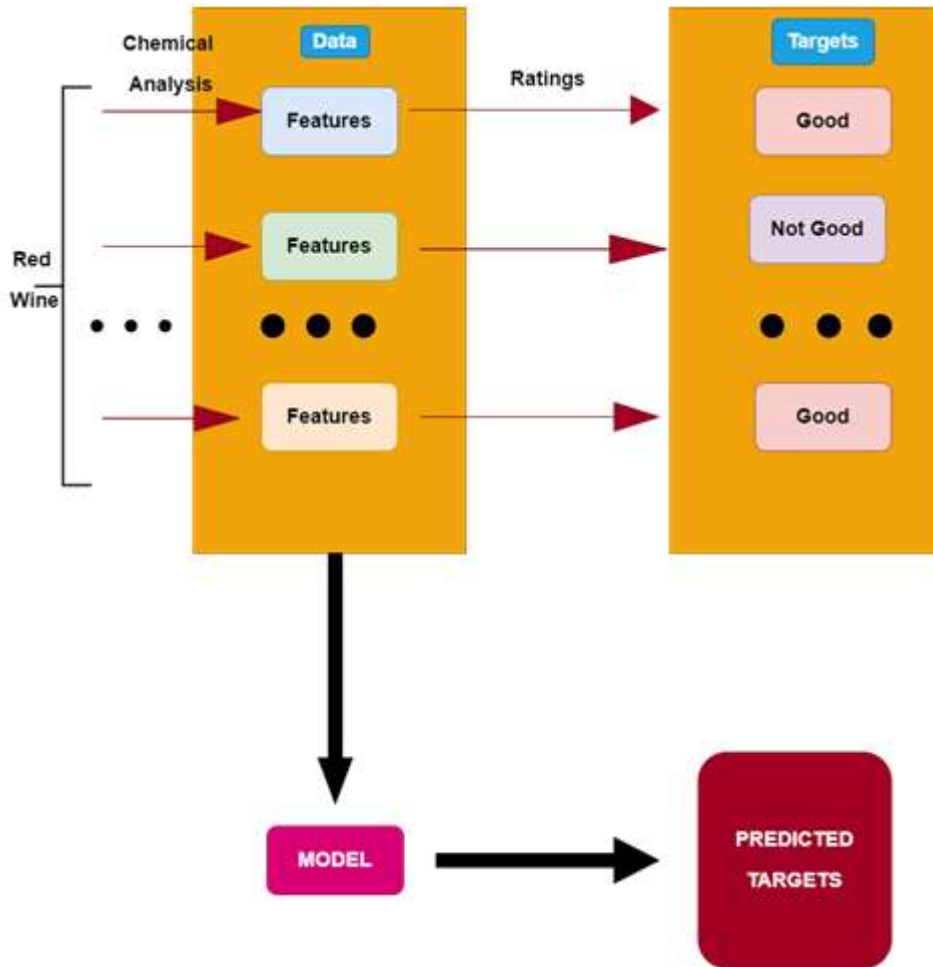
## 2.2. Ağ Yapısını Oluşturma

```
data = pd.read_csv('winequality-red.csv')
data.corr()['quality'].sort_values(ascending=False)

X=zscore(data.drop(['quality'],axis=1))
y=data['quality']

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=TEST_SIZE,random_state=42)
preds = []
algorithms = [
    "Nearest Neighbor Classification",
    "C-Support Vector Classification",
    "Logistic Regression",
    "Gaussian Naive Bayes",
    "Decision Tree Classifier",
    "Bagging Classifier",
    "RandomForestClassifier"
]
```

Şekil 1 - Ağ Yapısı Oluşturma



Şekil 2 - Ağın Şematik Yapısı

Yukarıda tasarladığımız kodda öncelikle setimizin yolunu verdik. Elimizde bulunan 12 öznelik (Features) sayesinde vardığımız 2 sonuç çıkışımız olarak tanımlanmıştır.

## 2.3. Eğitim Sonuçları

Eğitim sonuçları olarak ROC eğrisi, confusionmatrix ve MSE (MeanSquaredError) sonuçlarına ulaşacağız.

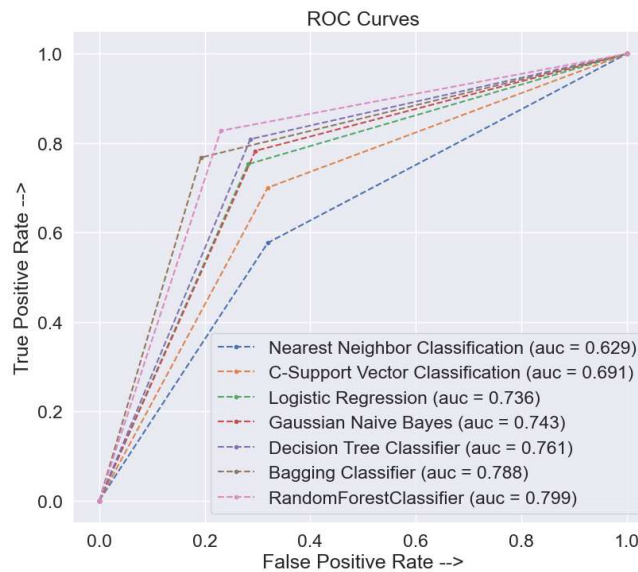
### 2.3.1. ROC Eğrisi

ROC eğrisi tanı testleri gibi sonuçları var/yok, yüksek/normal şeklinde iki sonuçlu ise testlerin başarısının belirlenmesi yöntemidir. İkili sınıflandırma sistemlerinde ayırım eşik değerinin farklılık gösterdiği durumlarda, hassasiyetin kesinliğe olan oranıyla ortaya çıkmaktadır. [3]

ROC eğrisi kodu ve gösterimi aşağıdaki gibidir.

```
def plot_ROCs(test, preds, labels):  
    plt.figure(figsize=(5, 5), dpi=100)  
    for i in range(len(labels)):  
        fpr, tpr, threshold3 = roc_curve(test, preds[i])  
        auc_tree = auc(fpr, tpr)  
        plt.plot(fpr, tpr, marker='.', linestyle='--', label=labels[i] + ' (auc = %0.3f)' % auc_tree)  
    plt.title("ROC Curves")  
    plt.xlabel('False Positive Rate -->')  
    plt.ylabel('True Positive Rate -->')  
    plt.legend()  
    plt.show()
```

Şekil 3 - ROC Eğrisi Gösterim Kodu



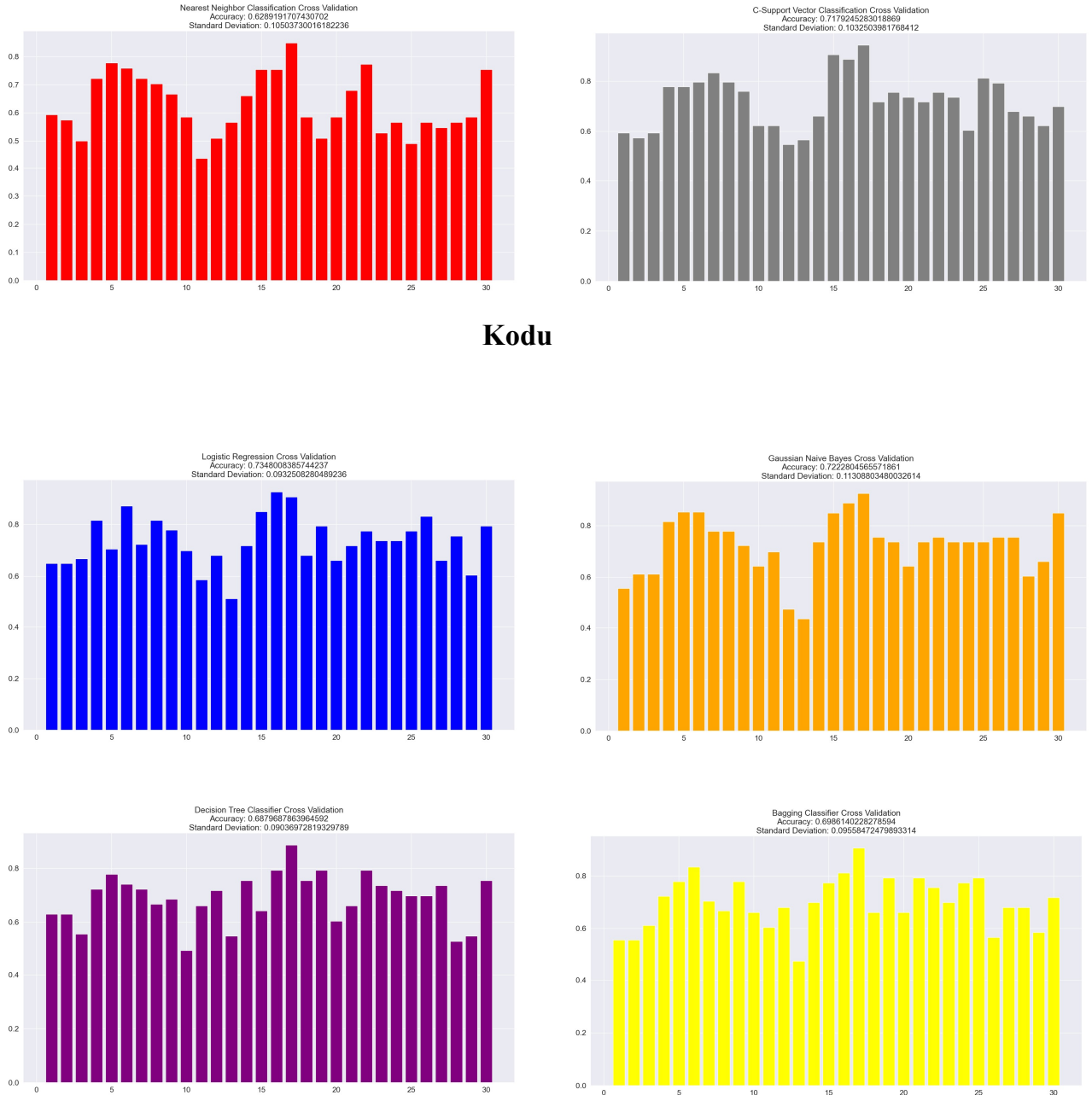
Şekil 4 - ROC Eğrisi Gösterimi

### 2.3.2. Cross Validation

Cross Validation, oluşturulan modeli birden fazla kez, farklı train test verileriyle fit ederek daha objektif ve doğru bir sonuç elde etmemizi sağlar. Cross Validation'a ait kod bloğu ve çıktılar aşağıdaki gibidir.

```
def cross_valid_graph(model, X, y, model_name, color='gray', cv=30):  
    scores = cross_val_score(model, X, y, cv=cv)  
    fig, axes = plt.subplots(figsize=(35,10), dpi=100)  
    plt.bar(range(1, len(scores) + 1), height=scores, color=color)  
    plt.title(model_name + " Cross Validation\nAccuracy: " +  
             str(scores.mean()) + "\nStandard Deviation: " + str(scores.std()))  
    plt.show()
```

Şekil 5 - Cross Validation Gösterim



Kodu



**Şekil 6 - Cross Validation Sonuçları**

### 2.3.3. ConfusionMatrix

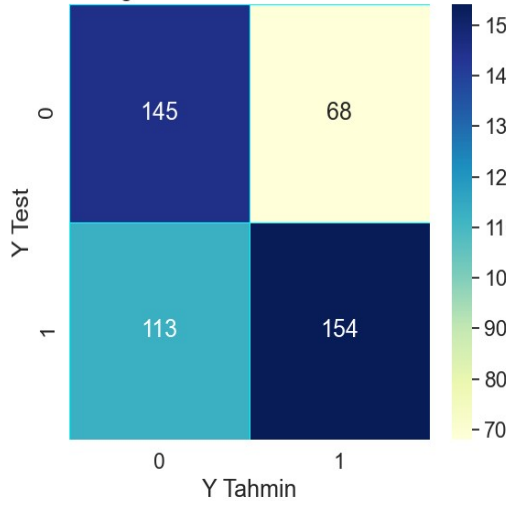
Sınıflandırma modellerinin performanslarını ölçmek için kullanılan bir makine öğrenmesi yöntemidir. Tahminlerin ve gerçek değerlerin karşılaştırıldığı hata matrisi yaygın olarak kullanılmaktadır.[4]

Confusionmatrix kodu her sınıf için ayrı ayrı oluşturulmuş olup gösterimi aşağıdaki gibidir.

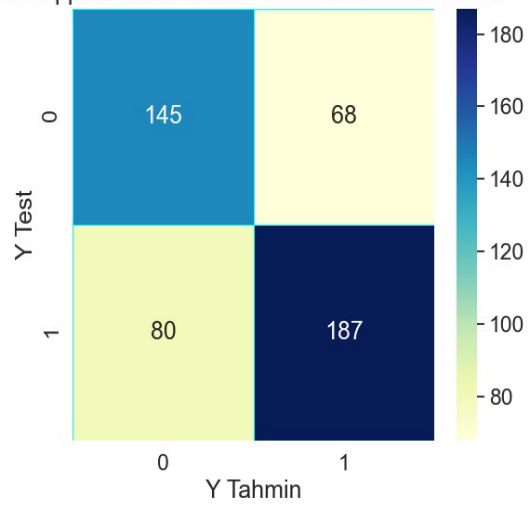
```
def plot_confusion_matrix(Ytest, Ypred, label):  
    confusionMatrix = confusion_matrix(Ytest, Ypred)  
    f, ax = plt.subplots(figsize=(5, 5))  
  
    sns.heatmap(confusionMatrix, annot=True, linewidth=0.7,  
                linecolor='cyan', fmt='g', ax=ax, cmap="YlGnBu")  
    plt.title(label + " Confusion Matrix")  
    plt.xlabel('Y Tahmin')  
    plt.ylabel('Y Test')  
    plt.show()
```

**Şekil 7 - ConfusionMatrix Kod Bloğu**

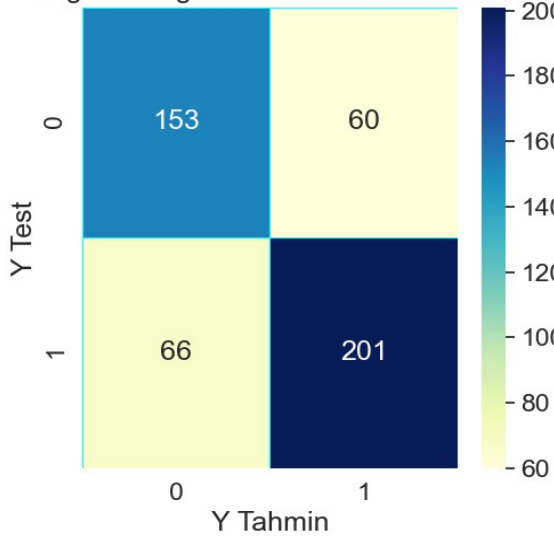
Nearest Neighbor Classification Confusion Matrix



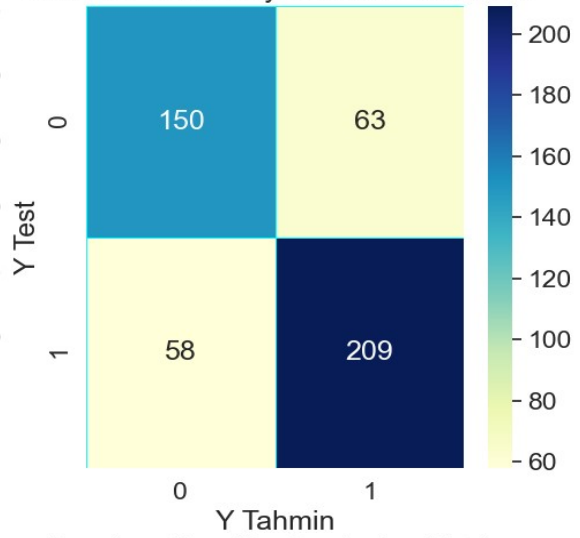
C-Support Vector Classification Confusion Matrix



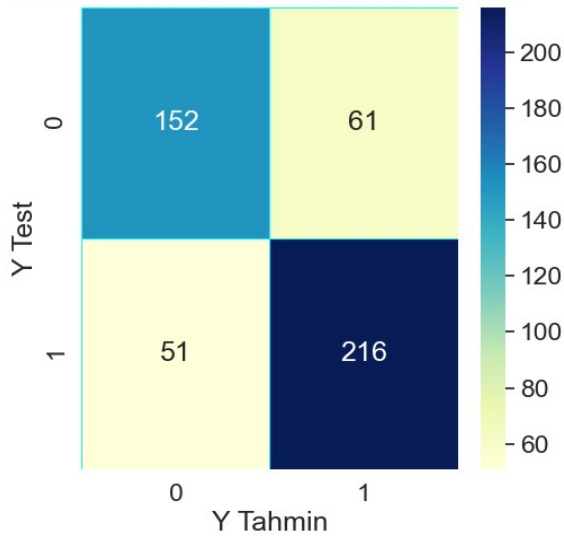
Logistic Regression Confusion Matrix



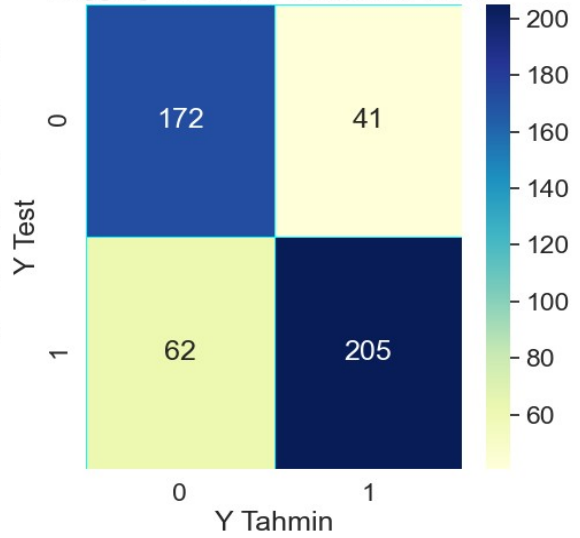
Gaussian Naive Bayes Confusion Matrix

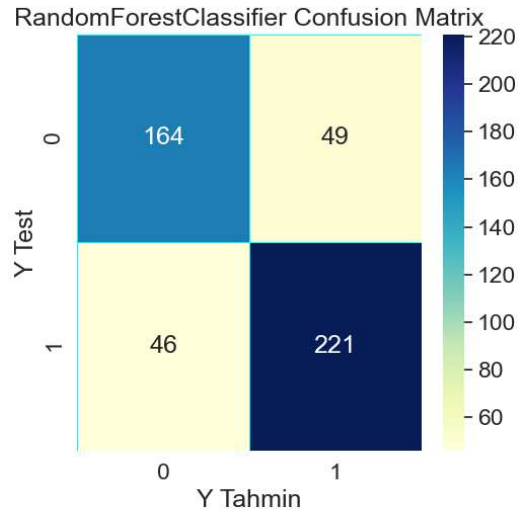


Decision Tree Classifier Confusion Matrix



Bagging Classifier Confusion Matrix





**Şekil 8 - ConfusionMatrix Sonuçları**

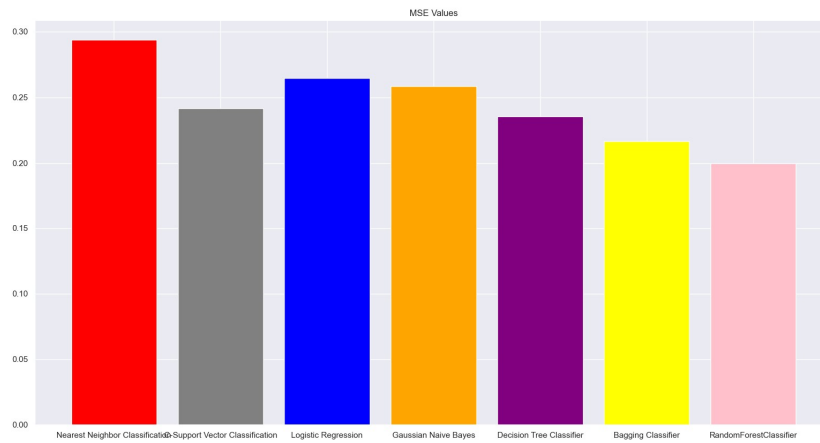
### 3. MSE (MeanSquaredError)

MSE, oluşturulan modelin, tahminleyicinin performansını ölçer. Pozitif ve 0'a yakın bir değer olmalıdır. [12]

Aşağıda işlem yaptığımız algoritmaların MSE kodu ve değerleri verilmiştir.

```
def log_MSEs(test, preds, labels):
    mse = []
    for i in range(len(preds)):
        mse.append(mean_squared_error(test, preds[i]))
        #print(labels[i] + " MSE: " + str(mse))
    fig, axes = plt.subplots(figsize=(35,10), dpi=100)
    plt.bar(labels, height=mse,
            color=['red', 'gray', 'blue', 'orange', 'purple', 'yellow', 'pink'])
    plt.title("MSE Values")
    plt.show()
```

**Şekil 9 - MSE Kod Bloğu**



**Şekil 10 - MSE Çıktıları**



#### 4. NearestNeighborClassification

En yakın komşu olarak bilinen KNN algoritması gözetimli öğrenme metodlarından birisidir. Basit sınıflandırma işleminde kullanılıp, çıktısı bir sınıf üyeliğidir. Nesne, en yakın komşuları arasında en yaygın olan sınıfa verilir.[5]

KNN algoritmasının kod bloğu aşağıdaki gibidir.

```
knnModel = KNeighborsClassifier(n_neighbors=8, metric='manhattan')
knnModel.fit(X_train, y_train)
Ypred_knn = knnModel.predict(X_test)
preds.append(Ypred_knn)

cross_valid_graph(knnModel, X, y, algorithms[0], 'red')
plot_confusion_matrix(y_test, Ypred_knn, algorithms[0])
scores = [[algorithms[0], knnModel.score(X_test, y_test)]]
```

**Şekil 11 - KNN Algoritması Gösterimi**

#### 5. SupportVectorClassification

Destek vektör algoritması olarak bilinen bir algoritmadır. İki sınıfı bir doğru veya düzlem ile birbirinden ayırma işlemi yapmaya çalışırken sınırındaki elemanları baz alır. Alanındaki bir çok yönteme göre daha yüksek başarı verir.[6]

SVM algoritmasının kod bloğu ve çıktısı aşağıdaki gibidir.

```
svcModel = SVC(C=10, random_state=42)
svcModel.fit(X_train, y_train)
Ypred_svc = svcModel.predict(X_test)
preds.append(Ypred_svc)

cross_valid_graph(svcModel, X, y, algorithms[1], 'gray')
scores.append([algorithms[1], svcModel.score(X_test, y_test)])
plot_confusion_matrix(y_test, Ypred_svc, algorithms[1])
plot_SVM(data, 0, 2, svcModel, algorithms[1])
```

**Şekil 12 - SVM Algoritması Gösterimi**

Kırmızı şarap kalitesini içeren veri setini farklı algoritmalarla eğittik. Bu algoritmalarından birisi de SVC algoritmasıydı. Veri setinin iki sütununu örnek olarak alıp ‘Destek Vektör Makinesi Grafiği’ ninplot görselini çizdireceğiz.

Öncelikle verimizin eğitimini tekrar yapmamız gerekiyor. Çünkü SVM grafiği, 2 adet X sütununu destekliyor. Bu yüzden istediğimiz iki sütunu ‘row’ değişkenleriyle belirtip, train\_test\_split fonksiyonuna gönderiyoruz.

Bu sayede yeni X, y değişkenlerimizin test, train verilerini almış oluyoruz. Daha sonra 2 sütunlu X verimizi yeniden boyutlandırıp (StandardScaler) eğitim sonucunu alıyoruz. Böylece SVM grafiğini çizdirmemiz için gerekli olan veriler elimize geçmiş oluyor.

İki adet olarak aldığımız X değerini 'meshgrid' kullanarak ikiye ayırdık. Daha sonra grafiğimizi oluşturmak için plt figürünün içine yolladık. Figürün fiziksel özelliklerini de tanımladıktan sonra 'scatter' kullanarak, noktalı veriye göre çizgi çekilmesini sağladık. Daha sonra figürümüzü ekrana çıkarttık.

Kod bloğu aşağıdaki gibidir.

```
def plot_SVM(data, row_1, row_2, model, title):
    #dataset, row1st, row2nd, model, header
    X_set, x_, y_set, y_ = train_test_split(
        data.iloc[:, row_1: row_2].values,
        data.iloc[:, -1].values,
        test_size = TEST_SIZE,
        random_state = 42
    )

    X_set = StandardScaler().fit_transform(X_set)
    mmodel = model.fit(X_set, y_set)

    X1, X2 = np.meshgrid(
        np.arange(X_set[:, 0].min() - 1, X_set[:, 0].max() + 1, 0.01),
        np.arange(X_set[:, 1].min() - 1, X_set[:, 1].max() + 1, 0.01)
    )

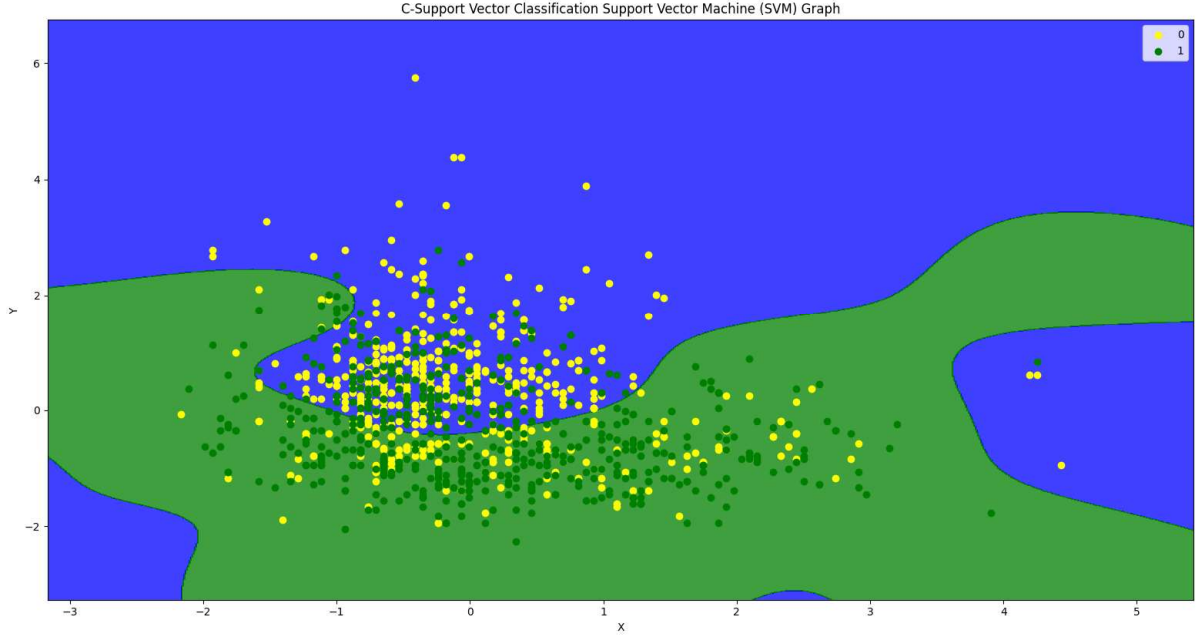
    plt.contourf(
        X1, X2,
        mmodel.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
        alpha = 0.75,
        cmap = ListedColormap(('blue', 'green'))
    )

    plt.xlim(X1.min(), X1.max())
    plt.ylim(X2.min(), X2.max())

    for i, j in enumerate(np.unique(y_set)):
        plt.scatter(
            X_set[y_set == j, 0],
            X_set[y_set == j, 1],
            c = ListedColormap(('yellow', 'green'))(i), label = j
        )

    plt.title(title + " Support Vector Machine (SVM) Graph")
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.legend()
    plt.show()
```

Şekil 13 - SVM Grafiği Kod Bloğu



Şekil 14 - SVM Grafiği

## 6. LogisticRegression

Lojistik Regresyon, istatistiksel bir yöntemdir. Bir veya birden fazla bağımsız değişken bulunduran veri kümesini analiz etmek için kullanılır. [7]

Lojistik regresyonun kod bloğu aşağıdaki gibidir.

```
logisticModel = LogisticRegression(C=10, random_state=42)
logisticModel.fit(X_train, y_train)
Ypred_logistic = logisticModel.predict(X_test)
preds.append(Ypred_logistic)

cross_valid_graph(logisticModel, X, y, algorithms[2], 'blue')
scores.append([algorithms[2], logisticModel.score(X_test, y_test)])
plot_confusion_matrix(y_test, Ypred_logistic, algorithms[2])
```

Şekil 15 - Lojistik Regresyon Kod Bloğu

## 7. NaiveBayes

Tembel bir öğrenme algoritması olup Bayes teoremine dayanır. Elimizde bulunan örneklerin hangi kategoriye ait olduğunu elde ettiği oranla tahmin eder. Aynı zamanda dengesiz veri kümelerinde de çalışabilir.[8]

NaiveBayes algoritması kod bloğu aşağıdaki gibidir.

```
naiveBayesModel = GaussianNB()
naiveBayesModel.fit(X_train, y_train)
Ypred_naiveBayes = naiveBayesModel.predict(X_test)
preds.append(Ypred_naiveBayes)

cross_valid_graph(naiveBayesModel, X, y, algorithms[3], 'orange')
scores.append([algorithms[3], naiveBayesModel.score(X_test, y_test)])
plot_confusion_matrix(y_test, Ypred_naiveBayes, algorithms[3])
```

Şekil 16 - NaiveBayes Kod Bloğu

## 8. DecisionTree

Yaygın olarak kullanılan gözetimli öğrenme algoritmalarından biridir. Karar düğümleri ve yaprak düğümlerinden oluşur. Ağaç dalı hiyerarşisi gibi dallanır.[9]

DecisionTree algoritması kod bloğu aşağıdaki gibidir.

```
decisionTreeModel = DecisionTreeClassifier(random_state=42)
decisionTreeModel.fit(X_train, y_train)
Ypred_decisionTree = decisionTreeModel.predict(X_test)
preds.append(Ypred_decisionTree)

cross_valid_graph(decisionTreeModel, X, y, algorithms[4], 'purple')
scores.append([algorithms[4], decisionTreeModel.score(X_test, y_test)])
plot_confusion_matrix(y_test, Ypred_decisionTree, algorithms[4])
```

Şekil 17 - DecisionTree Kod Bloğu

## 9. Bagging

Var olan bir eğitim setinden yeni eğitim setleri türeterek temel öğrenciyi yeniden eğitmeyi amaçlayan bir yöntemdir.[10]

Bagging algoritması kod bloğu aşağıdaki gibidir.

```
baggingModel = BaggingClassifier(base_estimator=decisionTreeModel,
                                n_estimators=10, random_state=17)
baggingModel.fit(X_train, y_train)
Ypred_bagging = baggingModel.predict(X_test)
preds.append(Ypred_bagging)

cross_valid_graph(baggingModel, X, y, algorithms[5], 'yellow')
scores.append([algorithms[5], baggingModel.score(X_test, y_test)])
plot_confusion_matrix(y_test, Ypred_bagging, algorithms[5])
```

Şekil 18 - Bagging Kod Bloğu

## 10.RandomForestTree

Denetimli bir algoritma olan RandomForestTree, diğer algoritmalara göre daha yüksek doğruluk oranı verir. Hem sınıflandırma hem de regresyon analizi için uygulanabilir. Bu algoritma rastgele bir orman yaratıyor. Ağaç sayısı arttıkça doğruluk oranıda artar.[11]

RandomForest algoritması kod bloğu aşağıdaki gibidir.

```
randomForestTreeModel = RandomForestClassifier(random_state=42)
randomForestTreeModel.fit(X_train, y_train)
Ypred_randomForestTree = randomForestTreeModel.predict(X_test)
preds.append(Ypred_randomForestTree)

cross_valid_graph(randomForestTreeModel, X, y, algorithms[6], 'pink')
scores.append([algorithms[6], randomForestTreeModel.score(X_test, y_test)])
plot_confusion_matrix(y_test, Ypred_randomForestTree, algorithms[6])
```

Şekil 19 - RandomForestTree Kod Bloğu

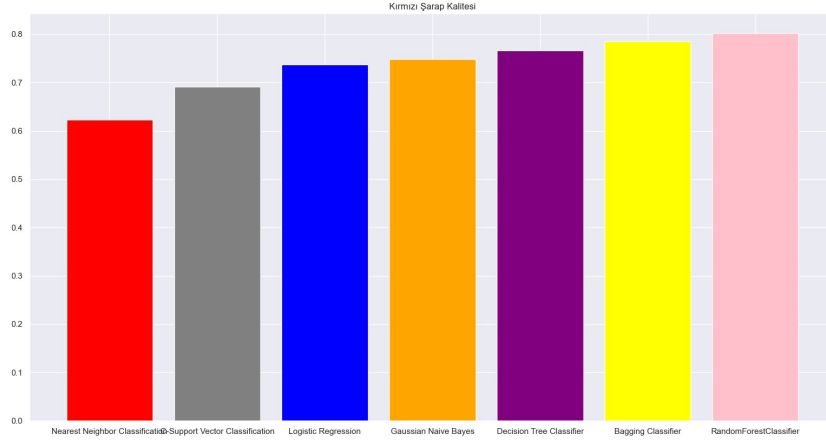
## 11.SONUÇ

Red Wine Quality adında bir veri seti seçtik. Kurduğumuz yapay sinir ağlarında modellemesini, eğitimini gerçekleştirdik. Daha sonra eğitim sonuçlarını, ROC eğrisini, confusionmatrix'ini ve MSE değerini grafiksel sonuç olarak elde ettik.

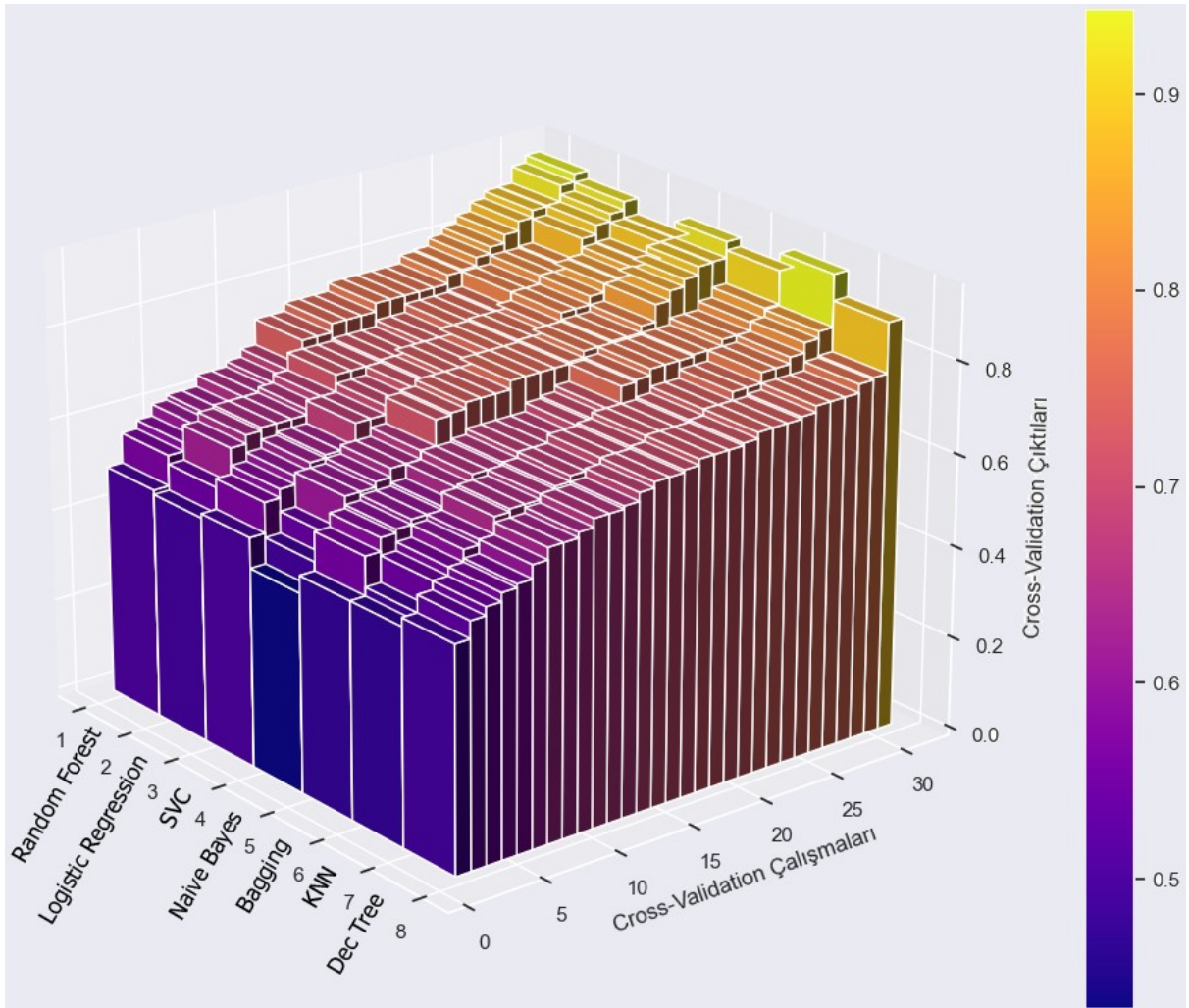
Çalışmamızın diğer kısmında birden çok algoritma kullandık. Kullandığımız algoritmaları modelleyip, eğititik. Bu eğitim sonucunda tüm algoritmaların sonuçlarını histogram grafiğine aktardık. Histogram grafiği aşağıda verilmiştir.

- **Kırmızı:**NearestNeighborClassification
- **Gri:**C-SupportVectorClassification
- **Mavi:** LogisticRegression
- **Turuncu:** GaussianNaiveBayes
- **Mor:** DecisionTreeClassifier
- **Sarı:** BaggingClassifier
- **Pembe:** RandomForestClassifier





**Şekil 20 - Final Sonuç Grafiği (Accuracy)**



**Şekil 21 – Tüm Algoritmaların Cross Validation Sonuçlarının 3D Grafiği**

## KAYNAKLAR

- [1]<https://azure.microsoft.com/tr-tr/overview/what-is-artificial-intelligence/#how>
- [2] [https://tr.wikipedia.org/wiki/Yapay\\_zek%C3%A2#Tarih%C3%A7e](https://tr.wikipedia.org/wiki/Yapay_zek%C3%A2#Tarih%C3%A7e)
- [3] <https://www.leyastat.com/roc-analizi-ve-roc-egrisi/>
- [4]<https://www.veribilimiokulu.com/hata-matrisini-confusion-matrix-yorumlama/>
- [5]<https://veribilimcisi.com/2017/07/20/k-en-yakin-komsu-k-nearest-neighborsknn/>
- [6]<https://www.veribilimiokulu.com/support-vector-machine-svm-ile-siniflandirma-python-ornek-uygulamasi/>
- [7] <https://veribilimcisi.com/2017/07/18/lojistik-regresyon/>
- [8]<https://medium.com/@ekrem.hatipoglu/machine-learning-classification-naive-bayes-part-11-4a10cd3452b4>
- [9][https://erdincuzun.com/makine\\_ogrenmesi/decision-tree-karar-agaci-id3-algoritmasi-classification-siniflama](https://erdincuzun.com/makine_ogrenmesi/decision-tree-karar-agaci-id3-algoritmasi-classification-siniflama)
- [10]<https://kadirguzel.medium.com/kolektif-%C3%B6%C4%9Frenme-ve-bagging-algoritmas%C4%B1-e8ea3d932b72>
- [11]<https://medium.com/data-science-tr/makine-%C3%B6%C4%9Frenmesi-dersleri-5-bagging-ve-random-forest-2f803cf21e07>
- [12]<https://veribilimcisi.com/2017/07/14/mse-rmse-mae-mape-metrikleri-nedir/>