**1 - Veri Seti**

**Veri Seti Kullanımı – 1**

<https://medium.com/gft-engineering/start-to-learn-machine-learning-with-the-iris-flower-classification-challenge-4859a920e5e3>

Yukarıda linkte yer alan proje örneğindeki gibi kütüphane olarak projenize ekledikten sonra kullanabilirsiniz

from sklearn.datasets import load\_iris

iris\_dataset = load\_iris()

**Veri Seti Kullanımı – 2**

Diğer bir kullanım ise sınıfta yazdığımız [programda](http://ders.hyesilyayla.com/ZekiSistemler/SiniftaYazilanKod/) yer aldığı gibi projenize ekleyebilirsiniz. Veri seti linkte yer alan rar’lı dosyanın içinde yer almaktadır.

1) Yukarıdaki veri seti üzerinde Python’da yer alan aşağıdaki sınıflama algoritmalarını kodlayacaksınız. Verinin %80’i eğitim %20’si test için kullanılacaktır. Her model için default değerlerle çalıştırdığınız kodlarınızı model başlığı altına kopyalayınız. Her model için farklı parametrelerle aldığınız sonuçların confusion matrix(karmaşıklık matrisi)’ini yine model başlığı altına eklenecektir. Ayrıca her modelin confusion matrix’inden elde edilen metrik değerleri hesaplanmalıdır.

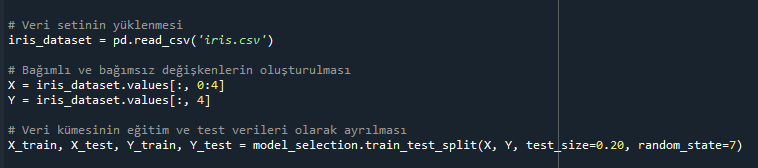
Bu metrikler: Accuracy, Precision, Recall, F1 Score, Support değerleridir.

Metrikler için classification\_report() fonksiyonundan faydalanabilirsiniz.

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report>

print(classification\_report(Y\_test, predictions)) formatını kullanabilirsiniz.

**CEVAP:**



**2 - Modeller**

**2-1-Basit Modeller**

*2-1-1-* [*Support Vector Classification*](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)

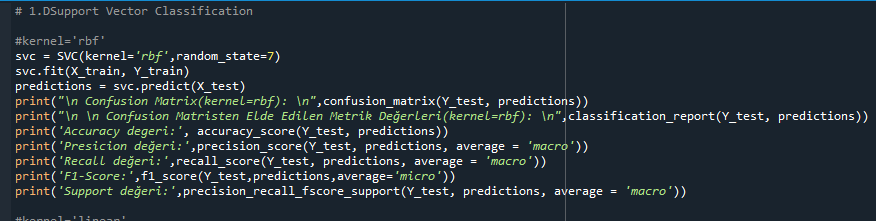
Bu yöntem farklı doğrular çizerek veriyi ayırma prensibine göre çalışır. Hangi tür doğru çizeceği de kernal parametresi ile belirlenmektedir. Default olarak RBF kullanıyor ama kernel parametresini değiştirerek sonuçları alıp 3 farklı kernal için ayrı confusion matrix(karmaşıklık matrisi) sonuçlarını oluşturmanız gerekmektedir.

**kernel*{‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’}, default=’rbf’***

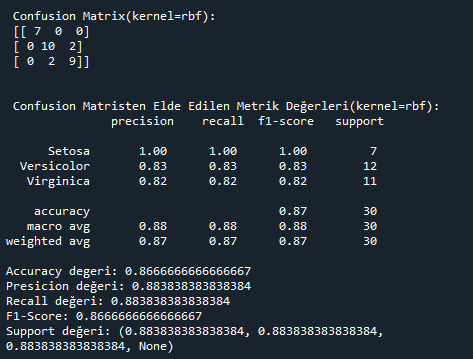
**CEVAPLAR**

1. **Default değer: kernel= rbf için**

**KOD:**

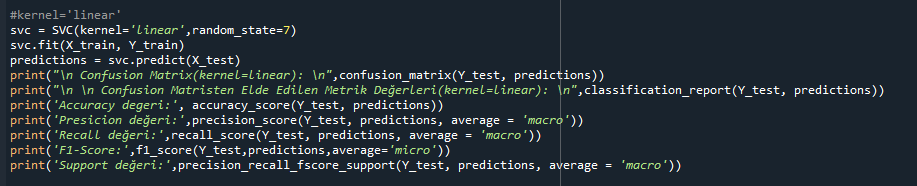


**EKRAN ÇIKTISI:**

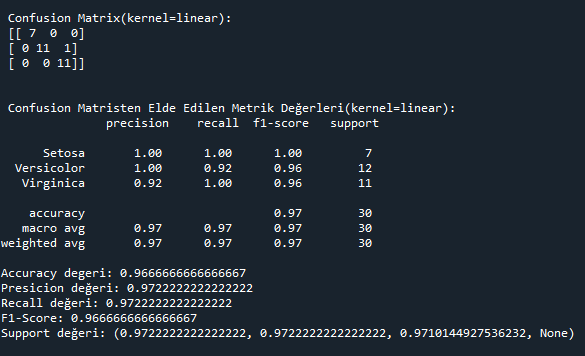


1. **Kernel=linear için**

**KOD:**

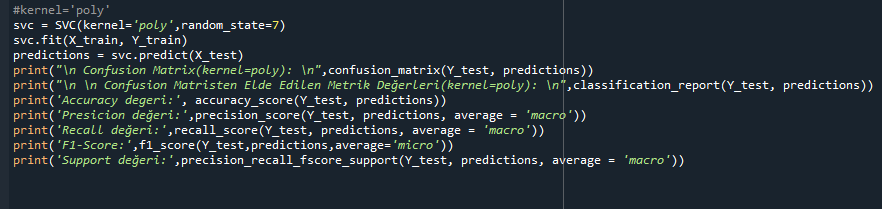


**EKRAN ÇIKTISI:**

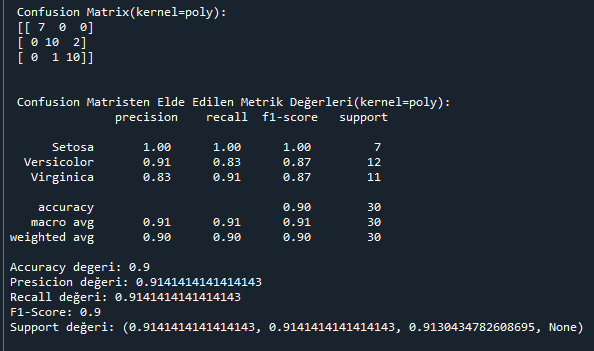


1. **Kernel=poly için**

**KOD:**

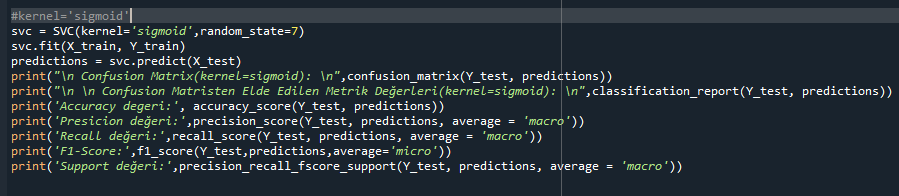


**EKRAN ÇIKTISI:**

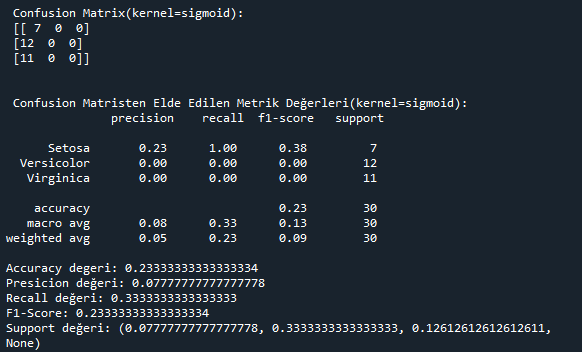


1. **Kernel=sigmoid için**

**KOD:**



**EKRAN ÇIKTISI:**



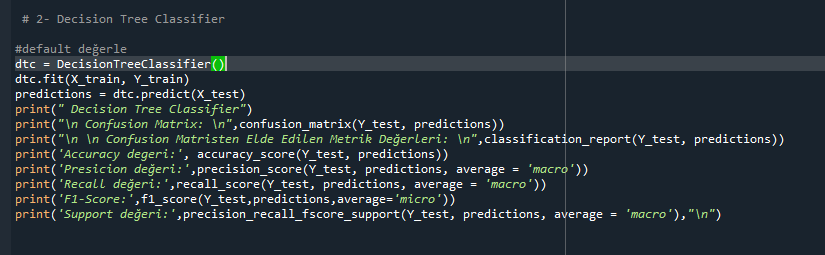
*2-1-2-* [*Decision Tree Classifier*](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html)

Default değerlerle çalıştırıp sonuçları aldıktan sonra en az 2 parametreyi değiştirerek 2 farklı daha sonuç (toplamda 3 adet) alarak her birinin confusion matrix(karmaşıklık matrisi) sonuçlarını oluşturmanız gerekmektedir.

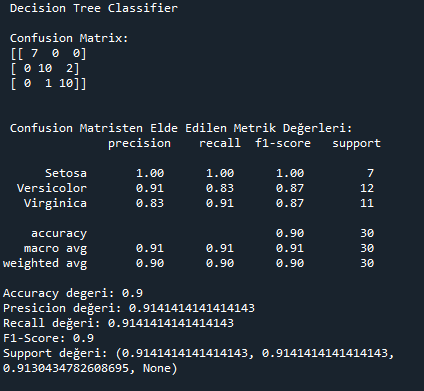
**CEVAPLAR**

1. **Default değerler ile çalıştırıldığında**

**KOD:**



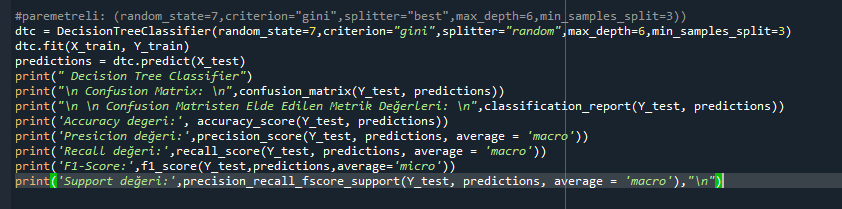
**EKRAN ÇIKTISI:**



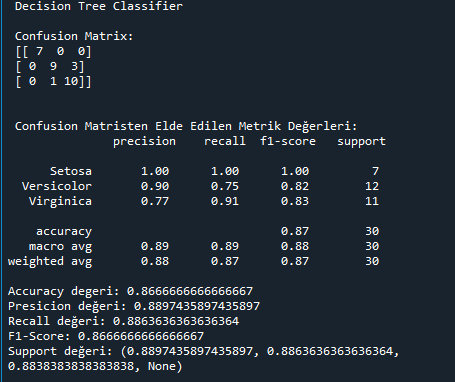
1. **PAREMETRELER DEĞİŞTİĞİNDE**

**Kullanılan Parametreler:** **(random\_state=7,criterion="gini",splitter="random",max\_depth=6,min\_samples\_split=3)**

**KOD:**



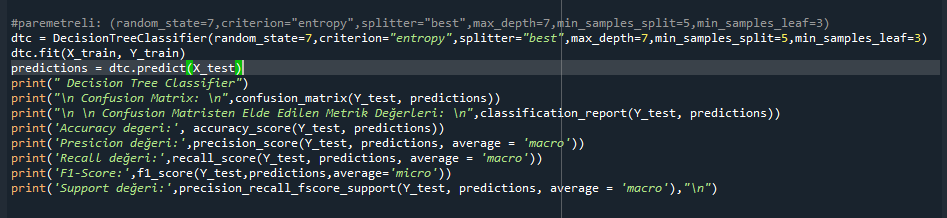
**EKRAN ÇIKTISI:**



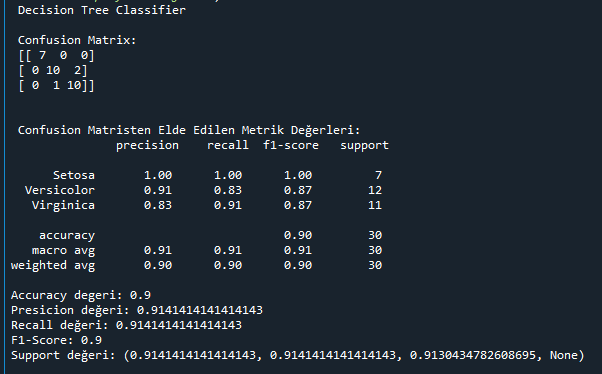
1. **PAREMETRELER DEĞİŞTİĞİNDE**

**Kullanılan Parametreler: (random\_state=7,criterion="entropy",splitter="best",max\_depth=7,min\_samples\_split=5,min\_samples\_leaf=3)**

**KOD:**



**EKRAN ÇIKTISI:**



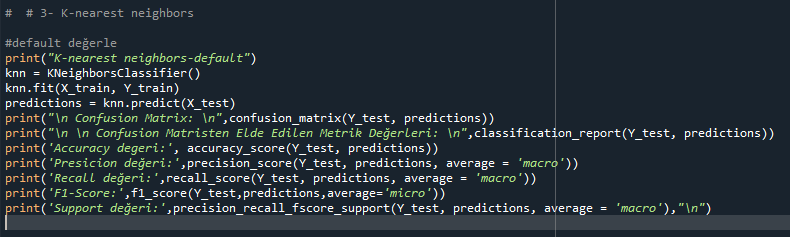
*2-1-3-* [*K-nearest neighbors*](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)

Default değerlerle çalıştırıp sonuçları aldıktan sonra en az 2 parametreyi değiştirerek 2 farklı daha sonuç (toplamda 3 adet) alarak her birinin confusion matrix(karmaşıklık matrisi) sonuçlarını oluşturmanız gerekmektedir.

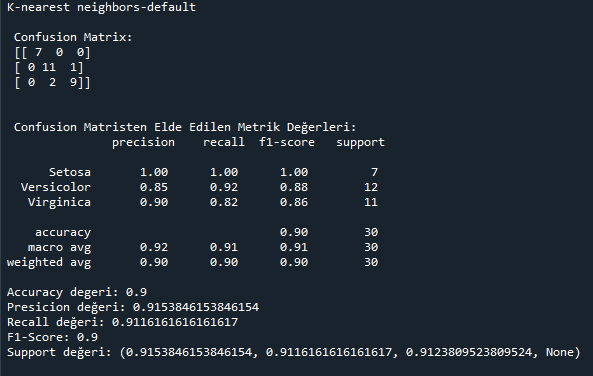
**CEVAPLAR**

1. **Default değerler ile çalıştırıldığında**

**KOD:**



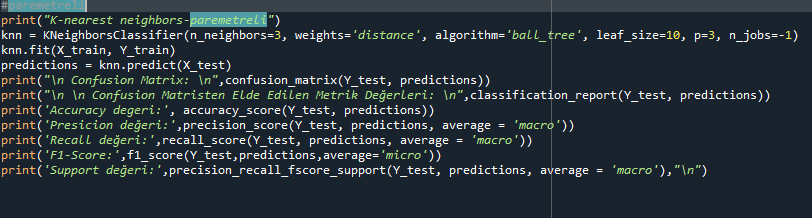
**EKRAN ÇIKTISI:**



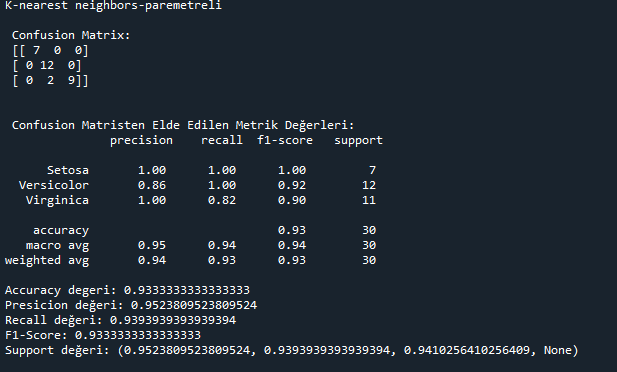
1. **PAREMETRELER DEĞİŞTİĞİNDE**

**Kullanılan paremetreler: (n\_neighbors=3, weights='distance', algorithm='ball\_tree', leaf\_size=10, p=3, n\_jobs=-1)**

**KOD:**



**EKRAN ÇIKTISI:**

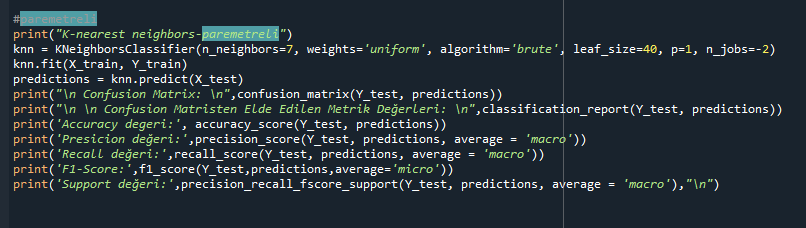


1. **PAREMETRELER DEĞİŞTİĞİNDE**

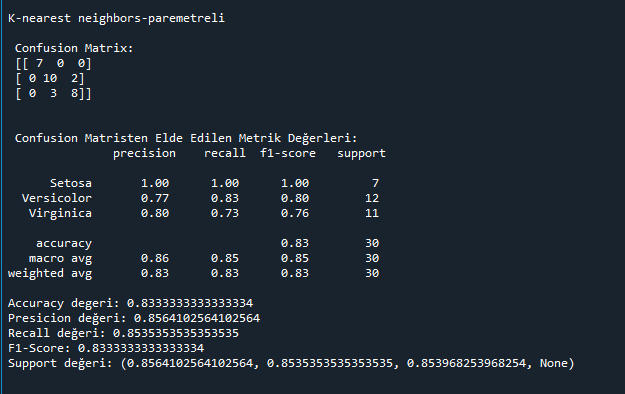
**Kullanılan paremetreler:**

**(n\_neighbors=7, weights='uniform', algorithm='brute', leaf\_size=40, p=1, n\_jobs=-2)**

**KOD:**



**EKRAN ÇIKTISI:**



[**2-2-Ensemble(Kollektif) Modeller**](https://www.datacamp.com/community/tutorials/ensemble-learning-python)

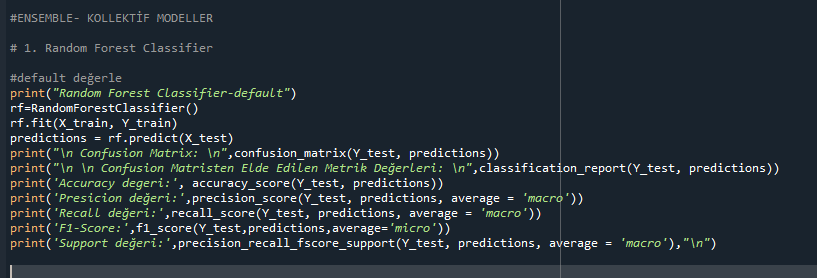
*2-2-1-* [***Random Forest Classifier***](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier)

Default değerlerle çalıştırıp sonuçları alınız. Kodunuzu ve confusion matrix(karmaşıklık matrisi) sonuçlarını buraya kopyalayınız.

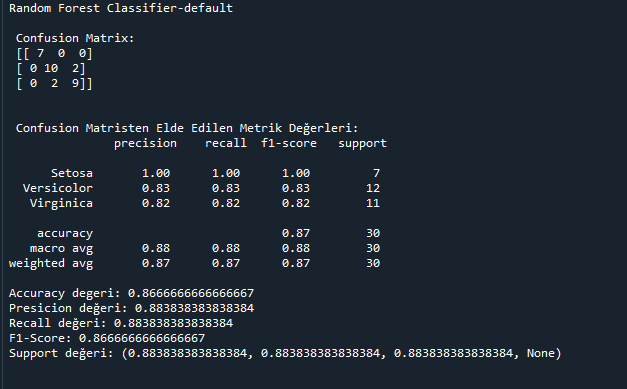
**CEVAP**

**Default değerler ile çalıştırıldığında**

**KOD:**



**EKRAN ÇIKTISI:**



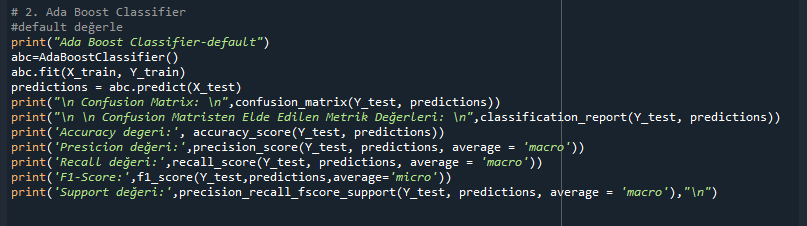
*2-2-2-* [***Ada Boost Classifier***](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier)

Ön tanımlı temel sınıflandırma algoritması olarak DecisionTreeClassifier(max\_depth=1) kullanmaktadır. Default değerlerle çalıştırarak sonuçları alınız. Kodunuzu ve confusion matrix(karmaşıklık matrisi) sonuçlarını buraya kopyalayınız.

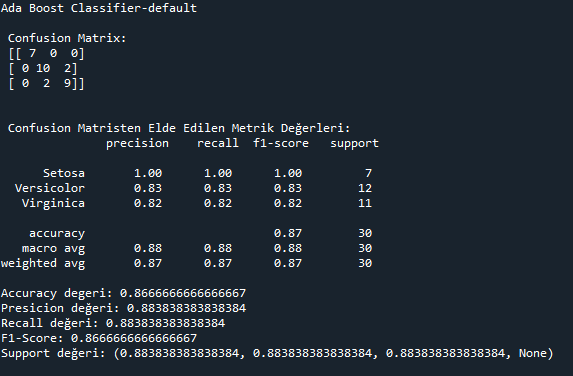
**CEVAPLAR**

1. **Default değerler ile çalıştırıldığında**

**KOD:**



**EKRAN ÇIKTISI:**



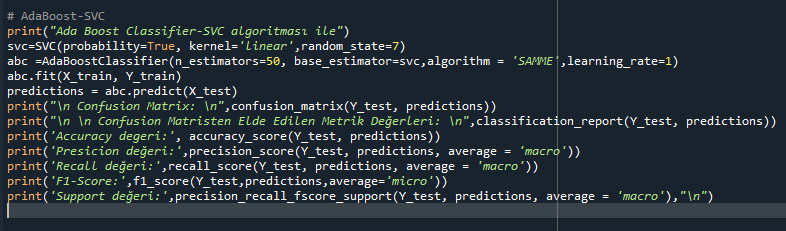
Daha sonra

<https://www.datacamp.com/community/tutorials/adaboost-classifier-python>

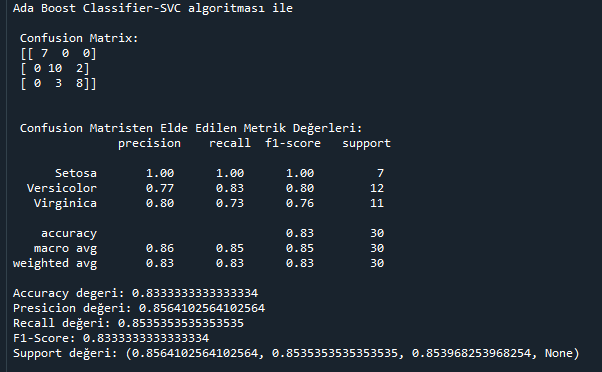
yukardaki linki inceleyiniz. “Using Different Base Learners” başlığı altındaki gibi temel sınıflandırma algoritması olarak SVC (Destek Vektör Makinesi Sınıflandırma) algoritması kullanarak sonuçları alınız. Kodunuzu ve confusion matrix(karmaşıklık matrisi) sonuçlarını buraya kopyalayınız.

1. **Temel sınıflandırma algoritması olarak SVC (Destek Vektör Makinesi Sınıflandırma) algoritması kullanılarak elde edilen sonuçlar.**

**KOD:**



**EKRAN ÇIKTISI:**



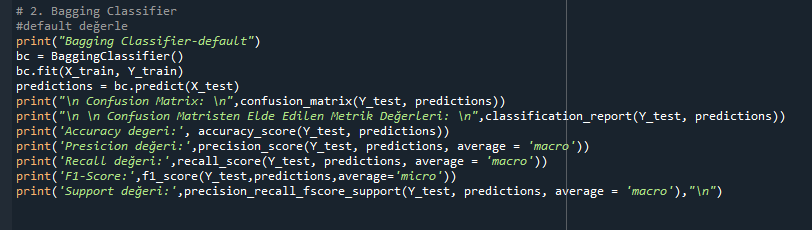
*2-2-3-* ***[Bagging Classifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html" \l "sklearn.ensemble.BaggingClassifier" \o "sklearn.ensemble.BaggingClassifier)***

Ön tanımlı temel sınıflandırma algoritması olarak DecisionTreeClassifier(max\_depth=1) kullanmaktadır. Default değerlerle çalıştırarak sonuçları alınız ve buraya kopyalayınız.

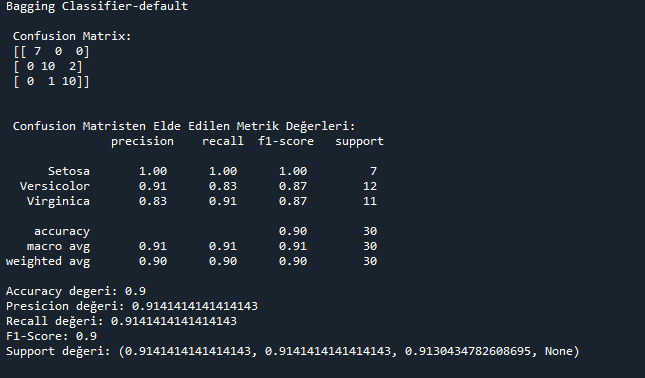
**CEVAPLAR:**

1. **Default değerler kullanıldığında**

**KOD:**



**EKRAN ÇIKTISI:**



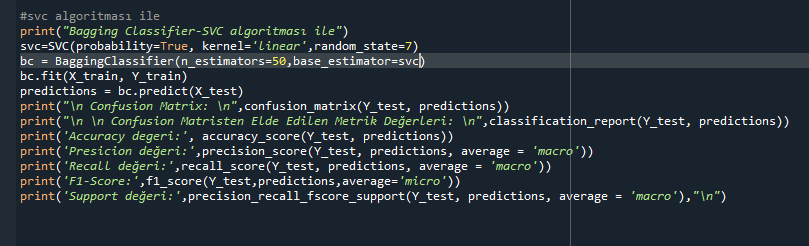
Daha sonra

<https://www.datacamp.com/community/tutorials/adaboost-classifier-python>

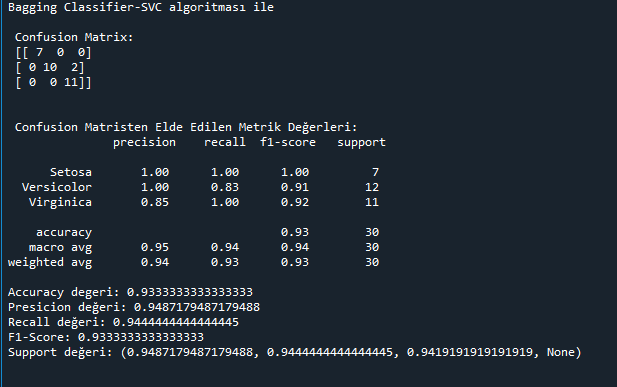
yukardaki linki inceleyiniz. “Using Different Base Learners” başlığı altındaki gibi temel sınıflandırma algoritması olarak SVC (Destek Vektör Makinesi Sınıflandırma) algoritması kullanarak sonuçları alınız ve buraya kopyalayınız.

1. **Temel sınıflandırma algoritması olarak SVC (Destek Vektör Makinesi Sınıflandırma) algoritması kullanıldığında elde edilen sonuçlar**

**KOD:**



**EKRAN ÇIKTISI**:



*2-2-4-* [***Voting Classifer***](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html#sklearn.ensemble.VotingClassifier)

Bu model oylama mantığına dayanmaktadır. Birden fazla basit model birlikte çalışarak oylama yaparlar ve tek sonuç oluştururlar. 2-1 başlığında verdiğim 3 basit modeli

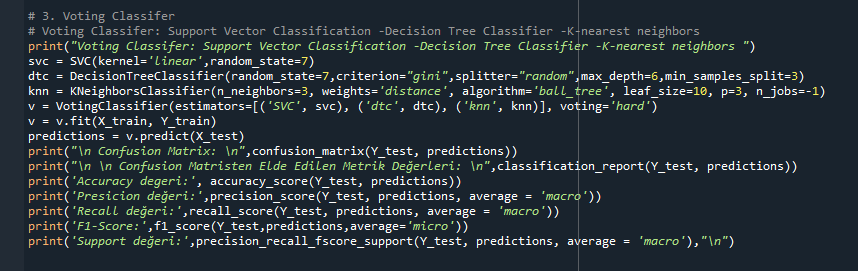
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html#sklearn.ensemble.VotingClassifier>

Linkinde yer alan örnek koddaki gibi çalıştırıp sonuçları alınız. Kodunuzu ve confusion matrix(karmaşıklık matrisi) sonuçlarını buraya kopyalayınız.

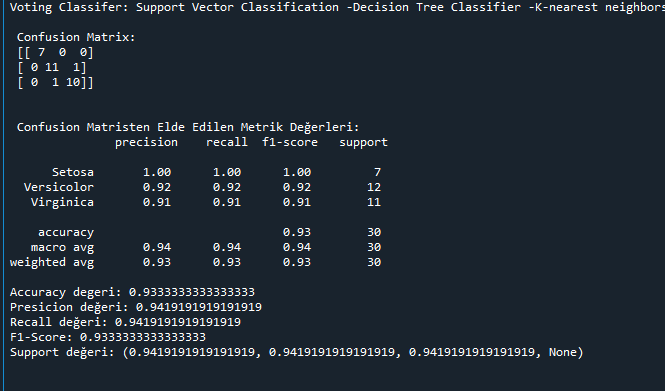
**CEVAP:**

[***Voting Classifer***](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html#sklearn.ensemble.VotingClassifier)***:* Support Vector Classification -Decision Tree Classifier -K-nearest neighbors**

**KOD:**



**EKRAN ÇIKTISI:**



2) Toplamda 7 farklı sınıflandırma tekniği kullanarak sonuçları alacaksınız. Bazı modellerde birden fazla sonuç oluşturmanızı istedim. Tüm sonuçlarınızı( basit modellerde 9, kollektif yöntemlerde 6 yani toplam 15) bir tabloda listelemeniz gerekmektedir. Tabloyu kendi çalışmalarınıza göre doldurunuz.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modeller | Accuracy | Precision | Recall | F1 Score | Support |
| SVC-default(rbf) | 0.8667 | 0.8838 | 0.8838 | 0.8667 | 0.8838 |
| SVC- linear | 0.9667 | 0.9722 | 0.9722 | 0.9667 | 0.9722 |
| SVC-poly | 0.9 | 0.9141 | 0.9141 | 0.9 | 0.9141 |
| Decision Tree Classifier-default | 0.9 | 0.9141 | 0.9141 | 0.9 | 0.9141 |
| Decision Tree Classifier- paremetreli: (random\_state=7,criterion="gini",splitter="random",  max\_depth=6,min\_samples\_split=3) | 0.8667 | 0.8897 | 0.8864 | 0.8667 | 0.8897 |
| Decision Tree Classifier-(random\_state=7,criterion="entropy",splitter="best"  ,max\_depth=7,min\_samples\_split=5,min\_samples\_leaf=3) | 0.9 | 0.9141 | 0.9141 | 0.9 | 0.9141 |
| K-nearest neighbors-default | 0.9 | 0.9154 | 0.9116 | 0.9 | 0.9154 |
| K-nearest neighbors-(n\_neighbors=3, weights='distance', algorithm='ball\_tree', leaf\_size=10, p=3, n\_jobs=-1) | 0.9333 | 0.9524 | 0.9394 | 0.9333 | 0.9524 |
| K-nearest neighbors-paremetreli: (n\_neighbors=7, weights='uniform', algorithm='brute', leaf\_size=40, p=1, n\_jobs=-2) | 0.8333 | 0.8564 | 0.8535 | 0.8333 | 0.8564 |
| Random Forest Classifier-default | 0.8667 | 0.8838 | 0.8838 | 0.8667 | 0.8838 |
| Ada Boost Classifier-default | 0.8667 | 0.8838 | 0.8838 | 0.8667 | 0.8838 |
| Ada Boost Classifier-SVC | 0.8333 | 0.8564 | 0.8535 | 0.8333 | 0.8564 |
| Bagging Classifier-default | 0.9 | 0.9141 | 0.9141 | 0.9 | 0.9141 |
| Bagging Classifier-SVC | 0.9333 | 0.9487 | 0.9444 | 0.9333 | 0.9419 |
| Voting Classifer | 0.9333 | 0.9419 | 0.9419 | 0.9333 | 0.9419 |

**YORUMLAR**

Accuracy değeri doğru sınıflandırılmış örnek sayısının toplam örnek sayısına oranını ifade eder kullanılan modeller arasında en yüksek accuracy değerini 0,9667 değeri ile destek vektör makinesi “linear” paremetresiyle vermiştir. En düşük accuracy değerini ise 0.83 değeri ile K-nearest neighbors ( paremetreli) ve Ada Boost Classifier(SVC) modelleri vermiştir. Yani accuracy değerine göre karar verilmesi durumunda en iyi model olarak destek vektör makinesi (linear) seçilebilir.

Precision değeri tüm sınıflardan, doğru olarak ne kadar tahmin edildiğinin bir ölçüsüdür. Mümkün olduğu kadar yüksek olmalıdır.En yüksek precision değerini 0,9722 değeri ile destek vektör makinesi “linear” paremetresiyle vermiştir. En düşük precision değerini 0.8564 değeri ile K-nearest neighbors ( paremetreli) ve Ada Boost Classifier(SVC) modelleri vermiştir. Yani precision değerine göre karar verilmesi durumunda en iyi model olarak destek vektör makinesi (linear) seçilebilir.

Recall değeri Positive olarak tahmin etmemiz gereken işlemlerin ne kadarını Positive olarak tahmin ettiğimizi gösteren bir metriktir. Kullanılan modeller arasında en yüksek recall değerini 0,9722 değeri ile destek vektör makinesi “linear” paremetresiyle vermiştir. En düşük recall değerini ise 0.8535 değeri ile K-nearest neighbors ( paremetreli) ve Ada Boost Classifier(SVC) modelleri vermiştir. Yani recall değerine göre karar verilmesi durumunda en iyi model olarak destek vektör makinesi (linear) seçilebilir.

F1 score değeri gerçek pozitif değerlerin oranının (recall) ve hassasiyetin (precision) harmonik ortalamasıdır. Sınıflandırıcının ne kadar iyi performans gösterdiğinin bir ölçüsüdür. Kullanılan modeller arasında en yüksek F1 score değerini 0,9667 değeri ile destek vektör makinesi “linear” paremetresiyle vermiştir. En düşük F1 score değerini ise 0.8333 değeri ile K-nearest neighbors ( paremetreli) ve Ada Boost Classifier(SVC) modelleri vermiştir. F1 score değerine göre karar verilmesi durumunda en iyi model olarak destek vektör makinesi (linear) seçilebilir.

Kullanılan modeller arasında en yüksek destek değerini 0,9722 değeri ile destek vektör makinesi “linear” paremetresiyle vermiştir. En düşük destek değerini ise 0.8564 değeri ile K-nearest neighbors ( paremetreli) ve Ada Boost Classifier(SVC) modelleri vermiştir. Yani destek değerine göre karar verilmesi durumunda en iyi model olarak destek vektör makinesi (linear) seçilebilir.

Kullanılan modeller arasında en yüksek başarıya sahip olan model linear paremetresi kullanılarak oluşturulan destek vektör makinesi modelidir.

KODLARIN TAMAMI:

import pandas as pd

from sklearn import model\_selection

#metrikler

from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import f1\_score

from sklearn.metrics import precision\_recall\_fscore\_support

#basit modeller

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

#ensemble modeller

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn.ensemble import VotingClassifier

from sklearn.ensemble import AdaBoostClassifier

iris\_dataset = pd.read\_csv('iris.csv')

X = iris\_dataset.values[:, 0:4]

Y = iris\_dataset.values[:, 4]

X\_train, X\_test, Y\_train, Y\_test = model\_selection.train\_test\_split(X, Y, test\_size=0.20, random\_state=7)

#BASİT MODELLER

# 1. DSupport Vector Classification

kernel='rbf'

svc = SVC(kernel='rbf',random\_state=7)

svc.fit(X\_train, Y\_train)

predictions = svc.predict(X\_test)

print("\n Confusion Matrix(kernel=rbf): \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri(kernel=rbf): \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'))

#kernel='linear'

svc = SVC(kernel='linear',random\_state=7)

svc.fit(X\_train, Y\_train)

predictions = svc.predict(X\_test)

print("\n Confusion Matrix(kernel=linear): \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri(kernel=linear): \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'))

#kernel='poly'

svc = SVC(kernel='poly',random\_state=7)

svc.fit(X\_train, Y\_train)

predictions = svc.predict(X\_test)

print("\n Confusion Matrix(kernel=poly): \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri(kernel=poly): \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'))

#kernel='sigmoid'

svc = SVC(kernel='sigmoid',random\_state=7)

svc.fit(X\_train, Y\_train)

predictions = svc.predict(X\_test)

print("\n Confusion Matrix(kernel=sigmoid): \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri(kernel=sigmoid): \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'))

# 2- Decision Tree Classifier

# default değerle

dtc = DecisionTreeClassifier()

dtc.fit(X\_train, Y\_train)

predictions = dtc.predict(X\_test)

print(" Decision Tree Classifier")

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

#paremetreli: (random\_state=7,criterion="gini",splitter="best",max\_depth=6,min\_samples\_split=3))

dtc = DecisionTreeClassifier(random\_state=7,criterion="gini",splitter="random",max\_depth=6,min\_samples\_split=3)

dtc.fit(X\_train, Y\_train)

predictions = dtc.predict(X\_test)

print(" Decision Tree Classifier")

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

#paremetreli: (random\_state=7,criterion="entropy",splitter="best",max\_depth=7,min\_samples\_split=5,min\_samples\_leaf=3)

dtc = DecisionTreeClassifier(random\_state=7,criterion="entropy",splitter="best",max\_depth=7,min\_samples\_split=5,min\_samples\_leaf=3)

dtc.fit(X\_train, Y\_train)

predictions = dtc.predict(X\_test)

print(" Decision Tree Classifier")

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

# 3- K-nearest neighbors

#default değerle

print("K-nearest neighbors-default")

knn = KNeighborsClassifier()

knn.fit(X\_train, Y\_train)

predictions = knn.predict(X\_test)

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

#paremetreli

print("K-nearest neighbors-paremetreli")

knn = KNeighborsClassifier(n\_neighbors=3, weights='distance', algorithm='ball\_tree', leaf\_size=10, p=3, n\_jobs=-1)

knn.fit(X\_train, Y\_train)

predictions = knn.predict(X\_test)

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

#paremetreli

print("K-nearest neighbors-paremetreli")

knn = KNeighborsClassifier(n\_neighbors=7, weights='uniform', algorithm='brute', leaf\_size=40, p=1, n\_jobs=-2)

knn.fit(X\_train, Y\_train)

predictions = knn.predict(X\_test)

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

#ENSEMBLE- KOLLEKTİF MODELLER

# 1. Random Forest Classifier

# #default değerle

print("Random Forest Classifier-default")

rf=RandomForestClassifier()

rf.fit(X\_train, Y\_train)

predictions = rf.predict(X\_test)

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

# 2. Ada Boost Classifier

#default değerle

print("Ada Boost Classifier-default")

abc=AdaBoostClassifier()

abc.fit(X\_train, Y\_train)

predictions = abc.predict(X\_test)

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

# AdaBoost-SVC

print("Ada Boost Classifier-SVC algoritması ile")

svc=SVC(probability=True, kernel='linear',random\_state=7)

abc =AdaBoostClassifier(n\_estimators=50, base\_estimator=svc,algorithm = 'SAMME',learning\_rate=1)

abc.fit(X\_train, Y\_train)

predictions = abc.predict(X\_test)

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

# 2. Bagging Classifier

#default değerle

print("Bagging Classifier-default")

bc = BaggingClassifier()

bc.fit(X\_train, Y\_train)

predictions = bc.predict(X\_test)

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

#svc algoritması ile

print("Bagging Classifier-SVC algoritması ile")

svc=SVC(probability=True, kernel='linear',random\_state=7)

bc = BaggingClassifier(n\_estimators=50,base\_estimator=svc)

bc.fit(X\_train, Y\_train)

predictions = bc.predict(X\_test)

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

# 3. Voting Classifer

# Voting Classifer: Support Vector Classification -Decision Tree Classifier -K-nearest neighbors

print("Voting Classifer: Support Vector Classification -Decision Tree Classifier -K-nearest neighbors ")

svc = SVC(kernel='linear',random\_state=7)

dtc = DecisionTreeClassifier(random\_state=7,criterion="gini",splitter="random",max\_depth=6,min\_samples\_split=3)

knn = KNeighborsClassifier(n\_neighbors=3, weights='distance', algorithm='ball\_tree', leaf\_size=10, p=3, n\_jobs=-1)

v = VotingClassifier(estimators=[('SVC', svc), ('dtc', dtc), ('knn', knn)], voting='hard')

v = v.fit(X\_train, Y\_train)

predictions = v.predict(X\_test)

print("\n Confusion Matrix: \n",confusion\_matrix(Y\_test, predictions))

print("\n \n Confusion Matristen Elde Edilen Metrik Değerleri: \n",classification\_report(Y\_test, predictions))

print('Accuracy degeri:', accuracy\_score(Y\_test, predictions))

print('Presicion değeri:',precision\_score(Y\_test, predictions, average = 'macro'))

print('Recall değeri:',recall\_score(Y\_test, predictions, average = 'macro'))

print('F1-Score:',f1\_score(Y\_test,predictions,average='micro'))

print('Support değeri:',precision\_recall\_fscore\_support(Y\_test, predictions, average = 'macro'),"\n")

Yorumlarınızı da tablo altına eklemeyi unutmayınız ☺

**Yararlanabileceğiniz siteler:**

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html#sklearn.metrics.confusion_matrix>

<https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics>

<https://medium.com/gft-engineering/start-to-learn-machine-learning-with-the-iris-flower-classification-challenge-4859a920e5e3>

<https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html>

<https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205>

<https://www.datacamp.com/community/tutorials/ensemble-learning-python>

<https://projector-video-pdf-converter.datacamp.com/6280/chapter3.pdf>

Not: Modellerin adlarına tıkladığınızda modellerin web sayfasına yönlendirileceksiniz. Bu sayfalarda modellerin parametrelerini görebilirsiniz. Default(ön tanımlı) parametre değerlerini değiştirerek sonuçları iyileştirmeye çalışınız.