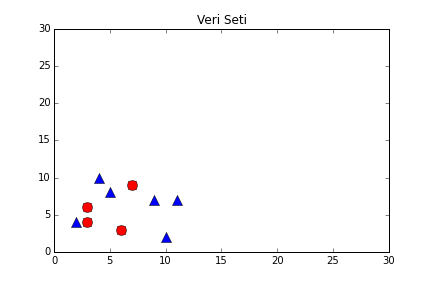
**1-KNN ile Sınıflandırma**

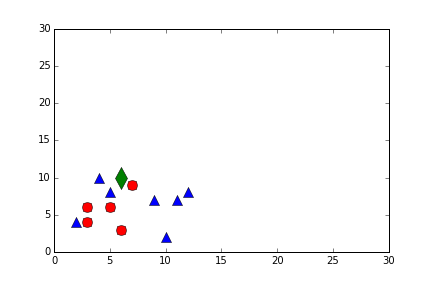
[Sınıflandırmada (classification)](http://www.bilgisayarkavramlari.com/2013/03/31/siniflandirma-classification/) kullanılan bu algoritmaya göre sınıflandırma sırasında [çıkarılan özelliklerden (feature extraction)](http://www.bilgisayarkavramlari.com/2008/12/01/ozellik-cikarimi-feature-extraction/), sınıflandırılmak istenen yeni bireyin daha önceki bireylerden k tanesine yakınlığına bakılmasıdır.

Örneğin k = 3 için yeni bir eleman sınıflandırılmak istensin. Bu durumda eski sınıflandırılmış elemanlardan en yakın 3 tanesi alınır. Bu elamanlar hangi sınıfa dahilse, yeni eleman da o sınıfa dahil edilir. Mesafe hesabından genelde [öklit mesafesi (euclid distance)](http://www.bilgisayarkavramlari.com/2007/11/24/oklit-mesafesi-euclidean-distance-euclidean-metric/) kullanılabilir.

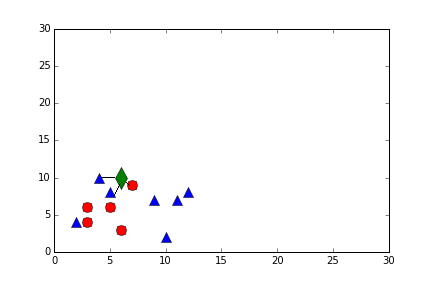


örneğin yukarıda verilen ve özelliklerine göre 2 boyutlu koordinat sistemine yerleştirilmiş olan örnekleri ele alalım. Bu örneklerin birbirinden ayrılması [doğrusal ayrıştırma (linear discrimination)](http://www.bilgisayarkavramlari.com/2008/10/02/dogrusal-ayrilabilirlik-linear-seperability/)problemidir ve buradaki yöntemlerle çözülür.

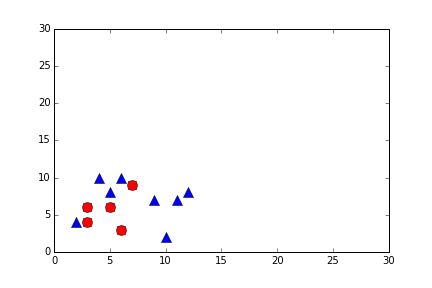
KNN yöntemine göre aşağıdaki şekilde yeni bir üyenin geldiğini düşünelim:

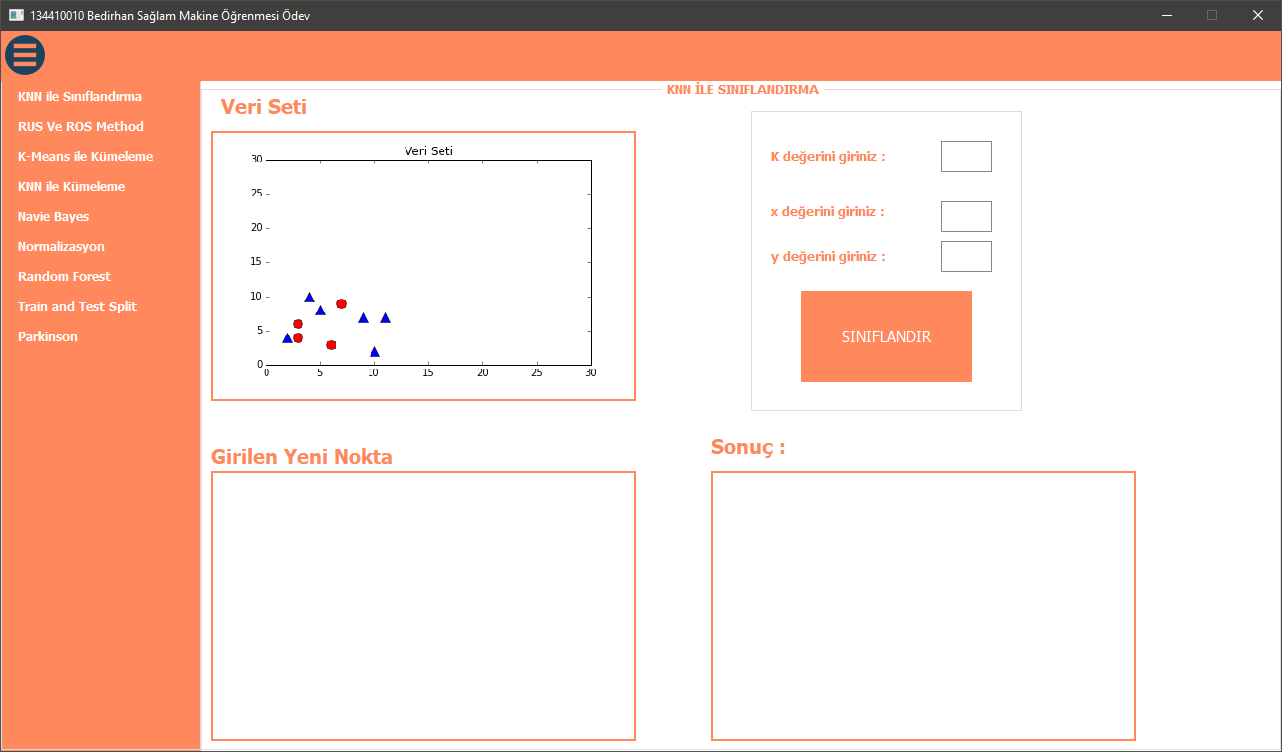


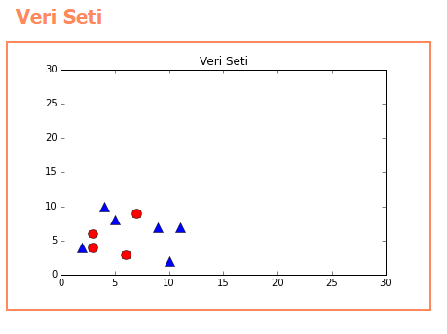
Yukarıdaki bu yeni gelen üyenin en yakın olduğu 3 üyeyi (3 nearest neighbors) tespit edelim.



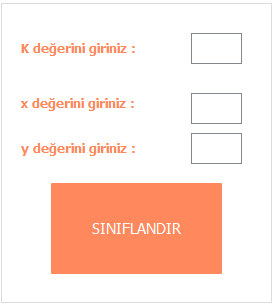
En yakın 3 üyenin iki tanesi mavi üçgen üyeler olduğuna göre yeni üyemizi bu şekilde sınıflandırabiliriz.

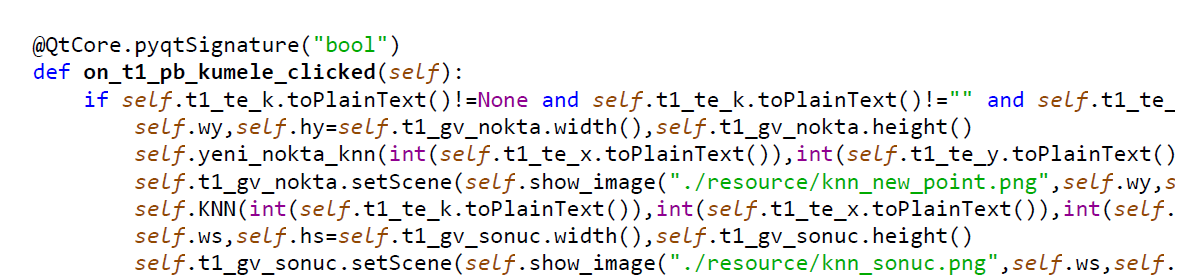


  
Yandaki şekilde yaptığımız uygulamada knn ile sınıflandırma alanını görüyorsunuz..

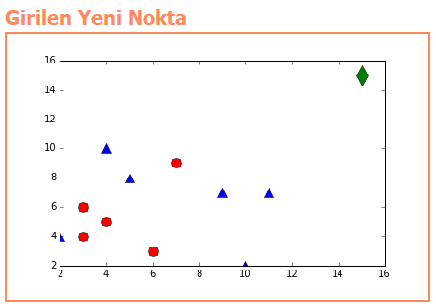
**Verisetini oluşturmak : knn.py dataload\_knn() kodlar:**

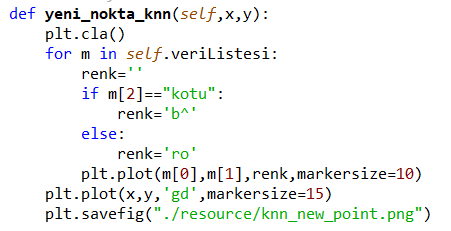
Yukarıdaki verisetini oluşturmak için **knn.py** dosyasında  
**dataload\_knn ()** fonksiyonunu çağırıyoruz bu fonksiyon verisetindeki iyi ve kötü olarak etiketlenmiş değerleri 2 boyutlu koordinat sisteminde çizdirerek , çizdirilen grafiğin kayıt edilmesi işini kotarıyor.

**Listeye yeni eleman eklemek :**

****Listeye yeni eleman eklemek için şekilde gördüğümüz alanı kullanıyoruz. Burada k değeri ve yeni noktanın koordinatları girildikten sonra sınıflandır butonuna tıklandıktan sonra sınıflandırma işlemi yapılıyor. Sınıflandır butonuna tıkladığımızda **main.py** de **on\_t1\_pb\_kumele\_clicked(self)** fonksiyonu çalışıyor.

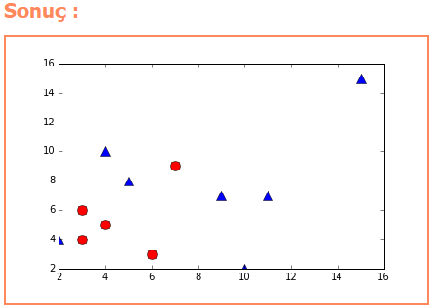
Bu fonksiyon main.py deki yeni\_nokta\_knn ve KNN fonksiyonlarını tetikliyor. K değerinin 3 , x ve y değerinin 15 girildiğini varsayalım.

**Main.py yeni\_nokta\_knn():**

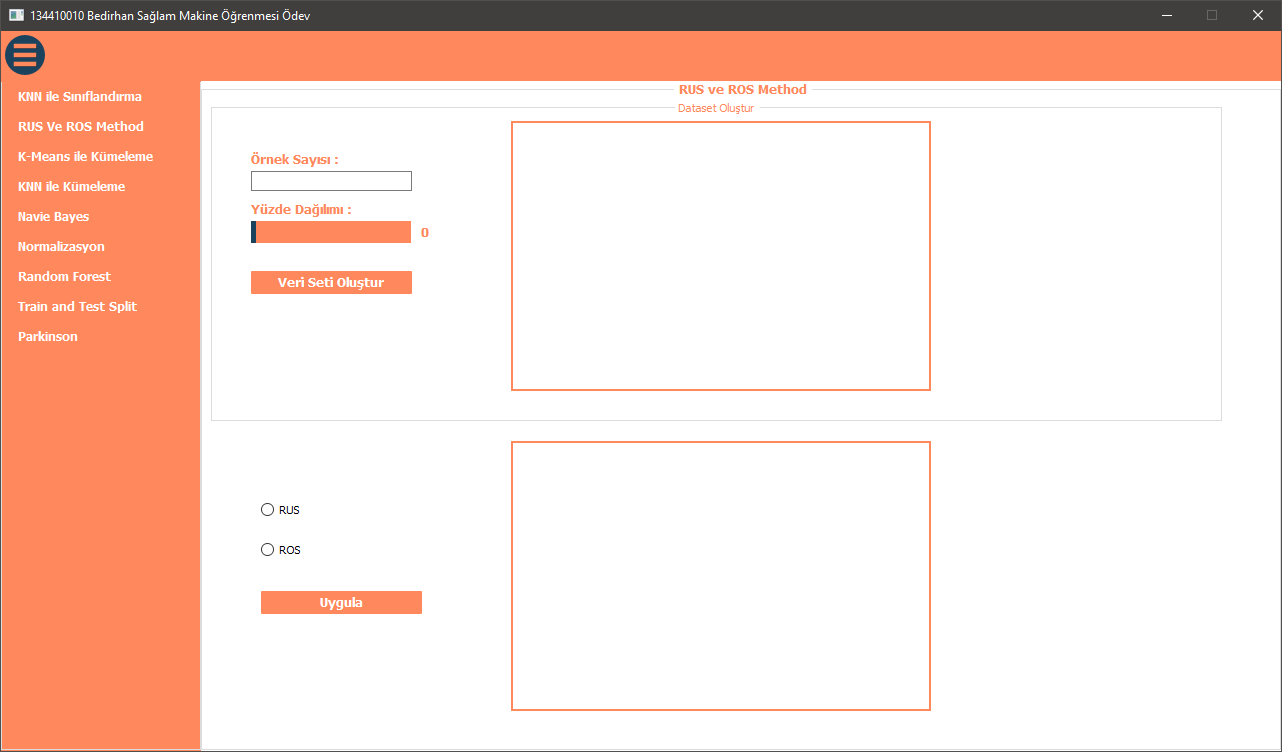


Fonksiyon çalıştıktan sonra girilen yeni nokta adında görselimiz oluşur , **on\_t1\_pb\_kumele\_clicked(self)** fonksiyonunda Show-image fonksiyonuyla bu resim gösterilir.

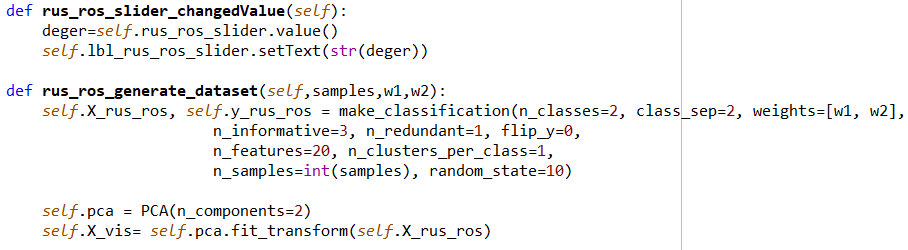
**Main.py KNN() :**

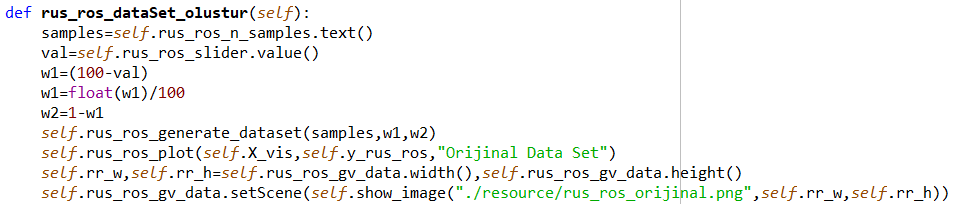
****Girilen değerlere göre en yakın komşular **KNN** fonksiyonunda hesaplanır ve yeni nokta sınıflandırılmış bir biçimde tekrar bir görsel çizdirilir bu görsel sonuç verisi olarak gösterilir.

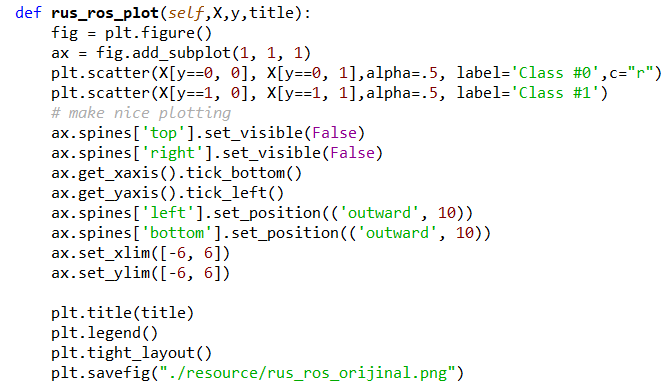
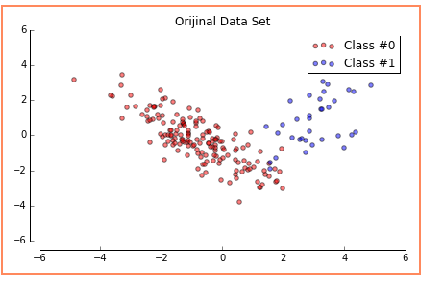
****

**2-Random Under Sampling & Random Over Sampling**

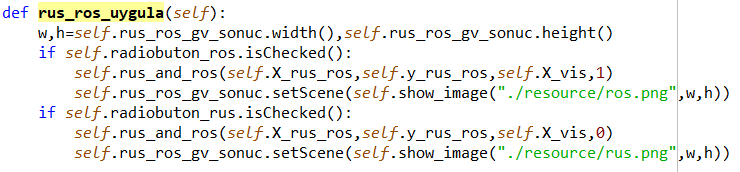
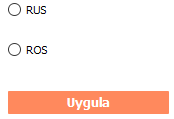
**Rastgele dataset oluştur:**

Burada örnek sayısı ve yüzde dağılımı verildikten sonra rastgele bir veriseti oluşturulur.Bu veriseti **Main.py** de **rus\_ros\_generate\_dataset** fonksiyonu ile oluşturulur. Örnek sayısını 200 ve yüzde dağılımını %20 olarak ayıralım. Veri setimizi oluşturalım.

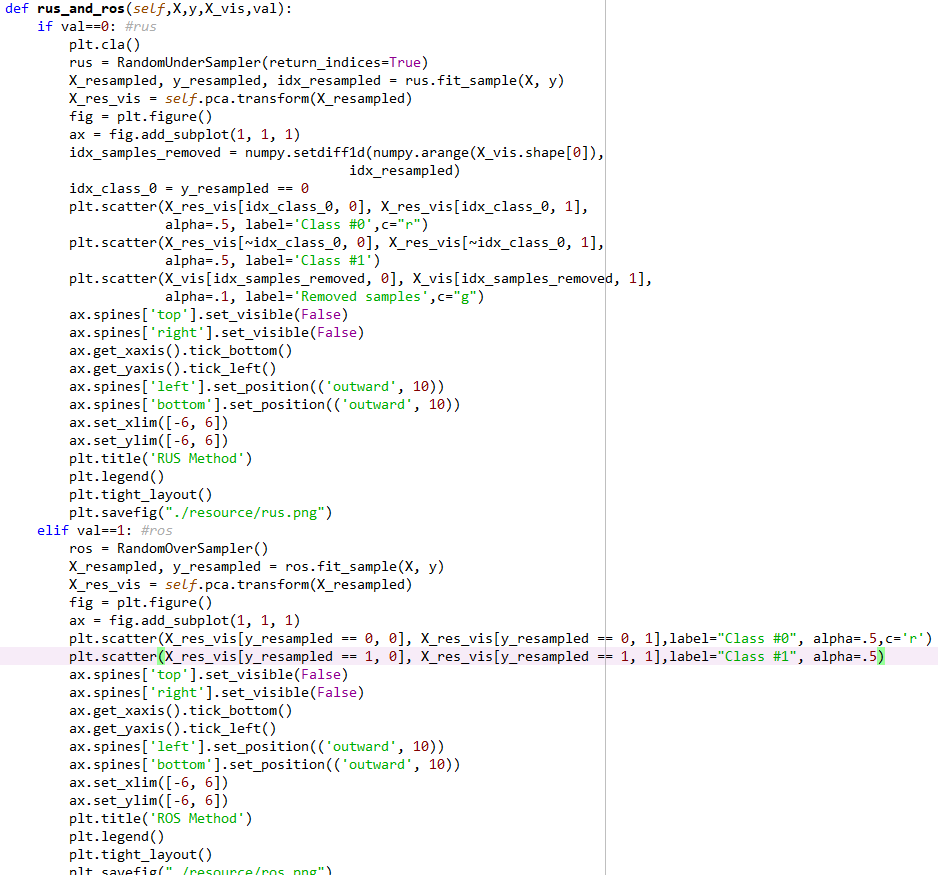
Veri seti oluştur butonuna tıkladığımızda **rus\_ros\_dataSet\_olustur** fonksiyonu tetiklenir.

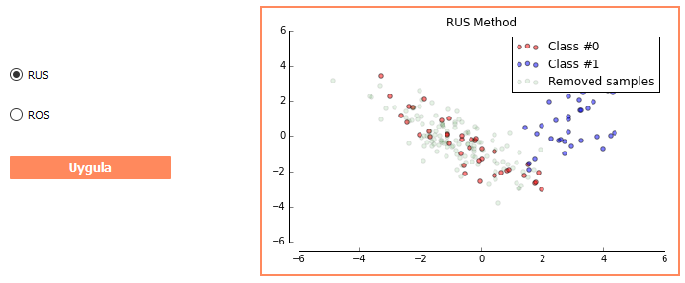
Dataset oluşturulduktan sonra rus\_ros\_plot fonksiyonu ile datasetimiz çizdirilir.  
Çizdirilen data set show\_image fonksiyonu ile gösterilir.

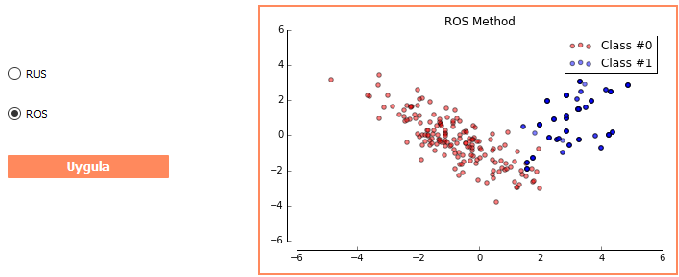
**RUS & ROS Uygula :**

****Verisetimizi rastgele oluşturduktan sonra uygulamak istediğimiz methodu seçip uygula butonuna tıklıyoruz. Uygula butonuna tıkladığımızda Main.py de **rus\_ros\_uygula** fonksiyonu tetikleniyor.

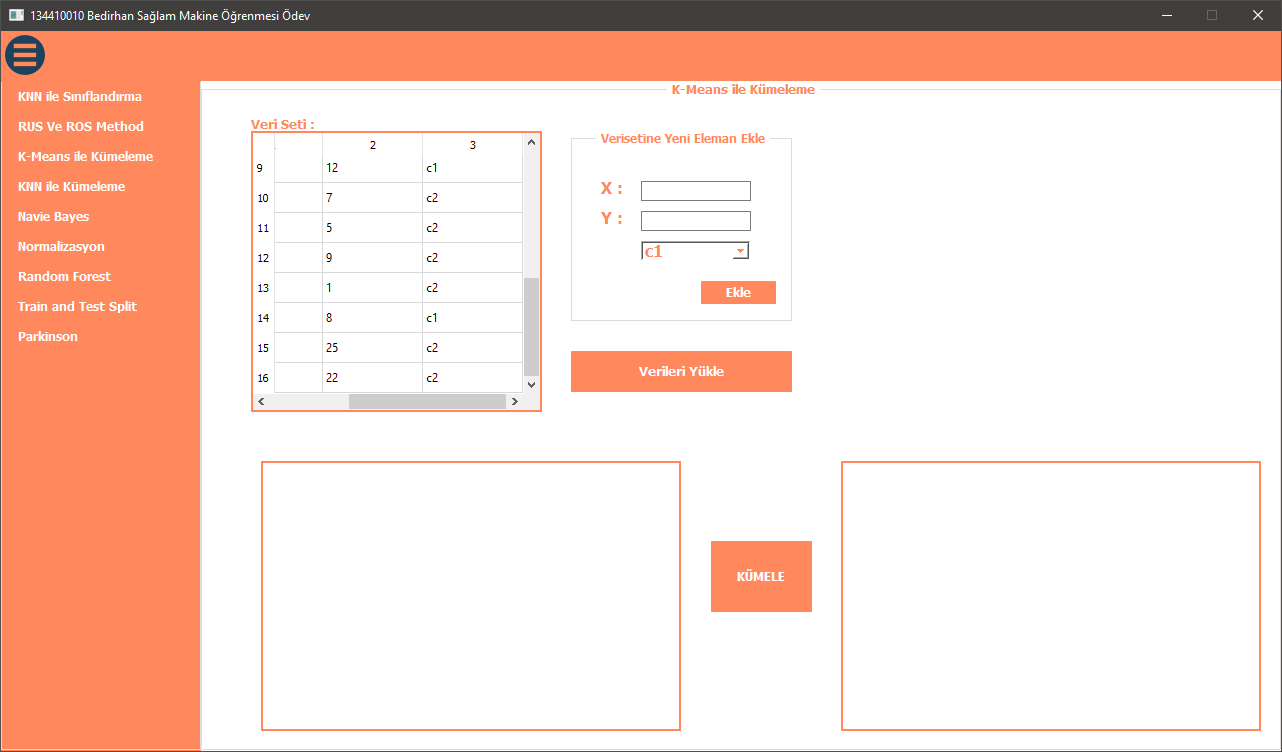
**rus\_ros\_uygula** fonksiyonu   
rus\_and\_ros fonksiyonunu tetikliyor ve rus veya ros işlemi yapılıyor.

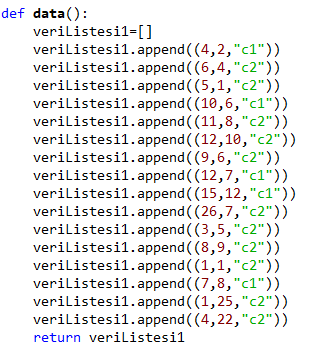
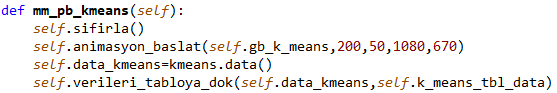
Main.py rus\_and\_ros fonksiyonu :

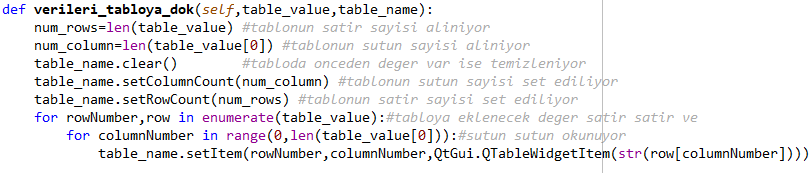


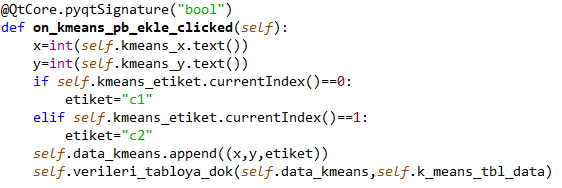


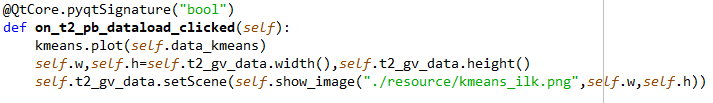
**3- K- Means ile Kümeleme**

Kümeleme (clustering) kullanılan algoritmalardan birisidir. Amaç [özellik çıkarımı (Feature extraction)](http://www.bilgisayarkavramlari.com/2008/12/01/ozellik-cikarimi-feature-extraction/)yapılmış bir grup verinin birden fazla küme özelliğine göre hangi kümeye ait olduğunun bulunmasıdır.  
Kullanılan matematiksel yöntem her sınıf için merkez belirlenen noktaya uzaklığa (aynı zamanda bu hata miktarıdır) göre yeni kümelerin yerleştirilmesidir.  
**Algoritma temel olarak 4 aşamadan oluşur:**  
1-Küme merkezlerinin belirlenmesi  
2-Merkez dışındaki örneklerin mesafelerine göre sınıflandırılması  
3-Yapılan sınıflandırmaya göre yeni merkezlerin belirlenmesi (veya eski merkezlerin yeni merkeze kaydırılması)  
4-Kararlı hale (stable state) gelinene kadar 2. ve 3. adımların tekrarlanması

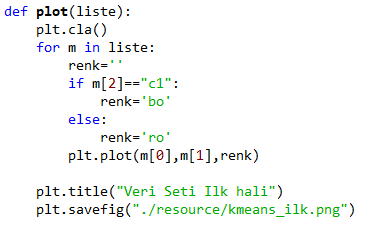
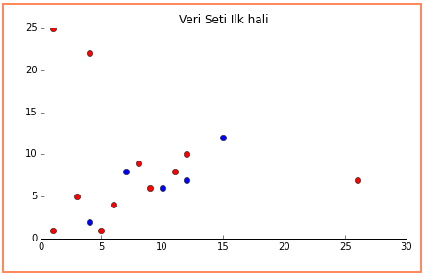
Veriseti hamburger menüdeki K-Means ile kümeleme butonuna tıklandığında Main.py **mm\_pb\_kmeans** fonksiyonu tetikleniyor ve **verileri\_tabloya\_dok** fonksiyonu ile veriler tabloya yükleniyor. Bu veriseti **kmeans.py** nin **data** fonksiyonundan çekiliyor.  
**kmeans.py data :**



**Verisetine yeni eleman ekle:**  
Burada x,y değerleri ve etiket seçilerek verisetine yeni değer eklenmesi sağlanmıştır. Değerleri girdikten sonra Ekle butonuna tıkladığımızda **on\_kmeans\_pb\_ekle\_clicked** fonksiyonu çalışıyor.Yeni veri tabloya ekleniyor

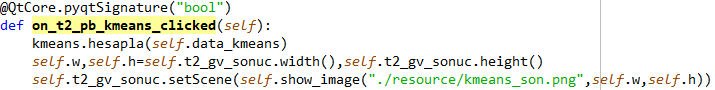
Verileri yükle butonuna tıkladığımızda , tabloda var olan veriler görselleştiriliyor ve graphicsviewde gösteriliyor.Verileri yükle butonuna tıkladığımızda **on\_t2\_pb\_dataload\_clicked** fonksiyonu tetikleniyor

Bu fonksiyon çağırıldığında kmeans.py deki plot fonksiyonu aktif oluyor ve kmeans\_ilk adında bir figür kayıt ediliyor. Bu figür graphicview da gösteriliyor.

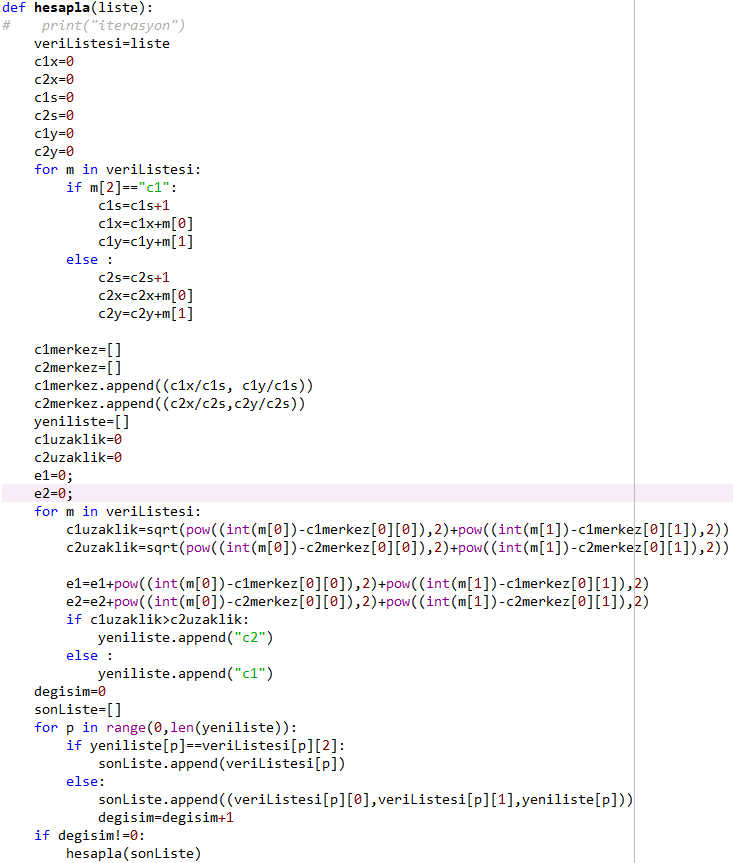
**Kmeans.py plot fonksiyonu: Ana ekranda gösterilen figür :**

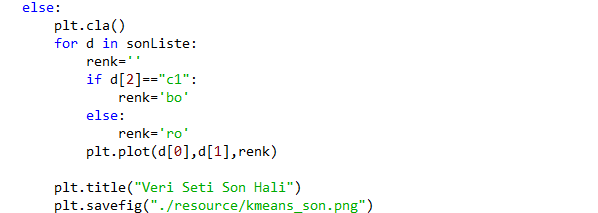
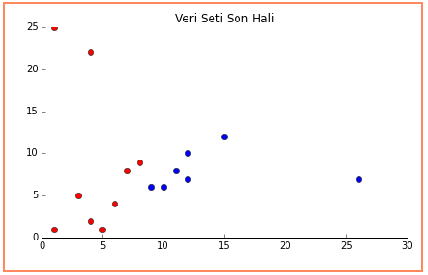


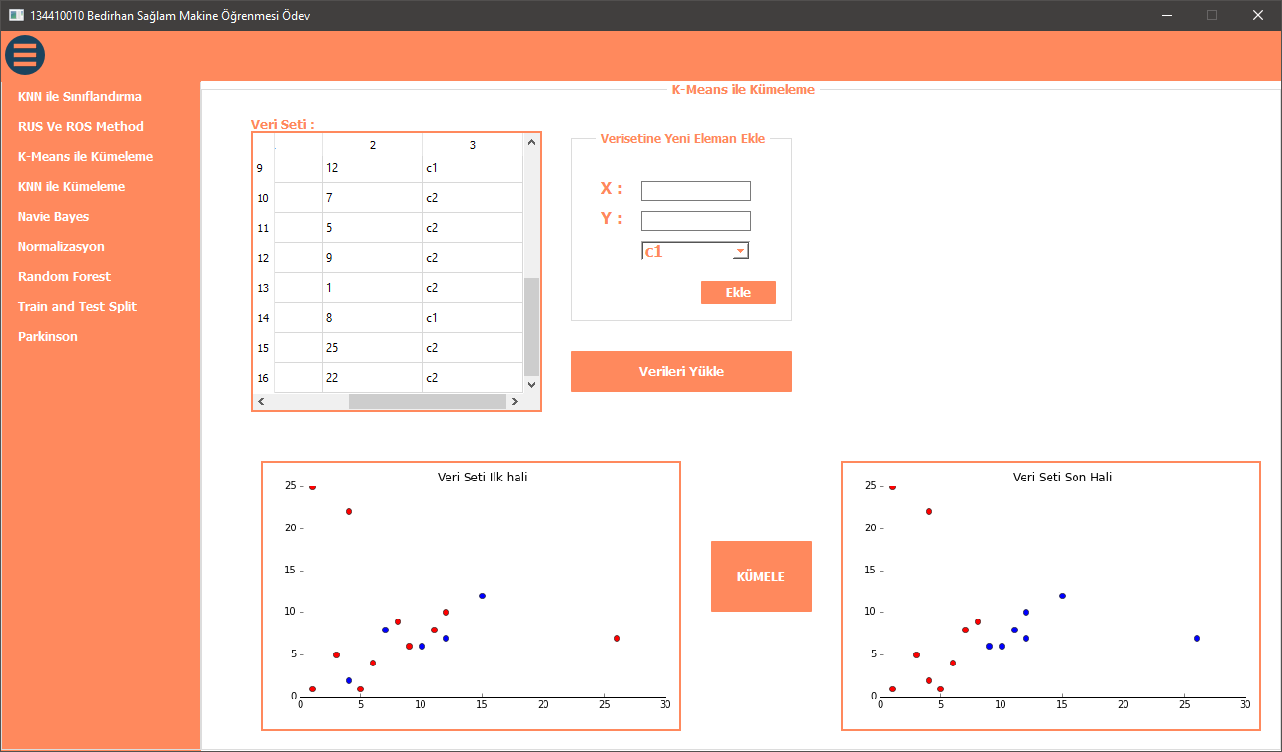
Kümele butonuna tıklandığı zaman kmeans kümeleme işlemi başlar ve **on\_t2\_pb\_kmeans\_clicked** fonksiyonu tetiklenir

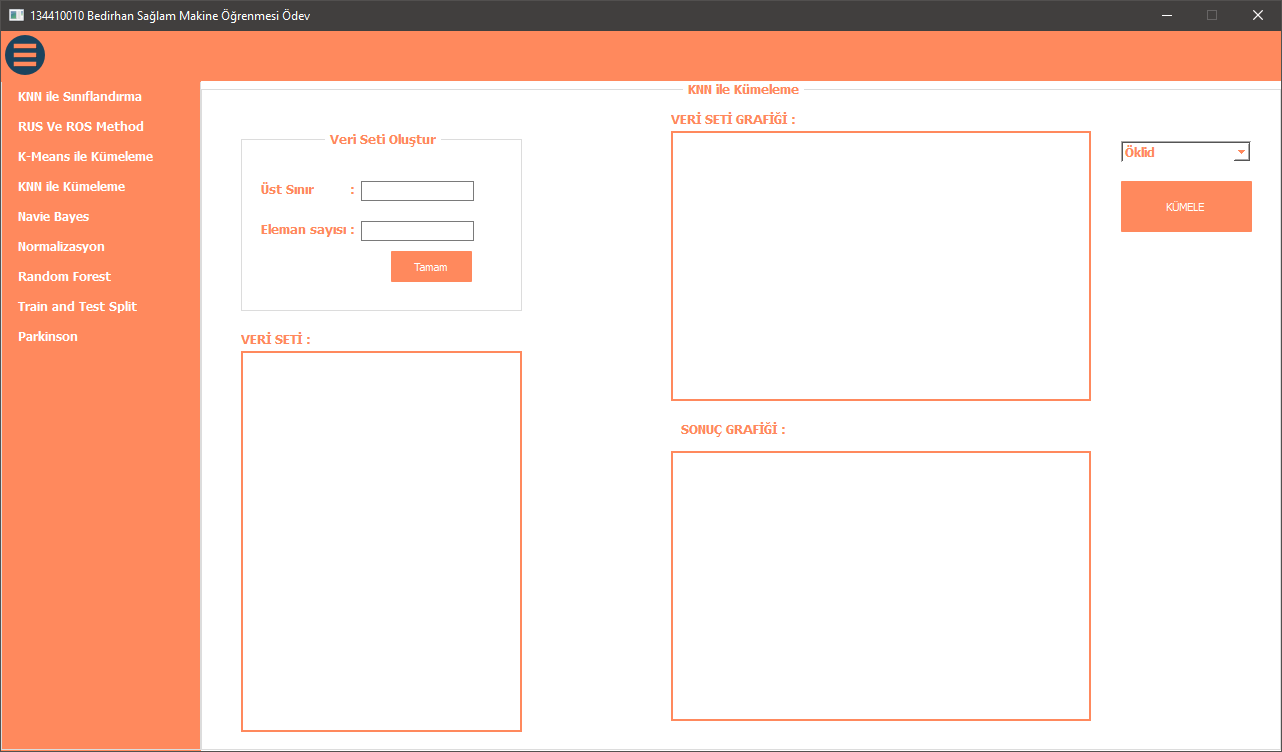


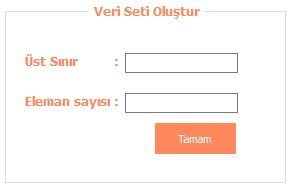
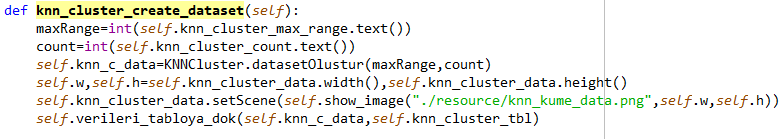
Bu fonksiyon kmeans.py de hesapla fonksiyonunu tetikler ve hesapla fonksiyonu çalışır. Hesapla fonksiyonu kümeleme işlemi yaptıktan sonra kmeans\_son adında bir figür oluşturur ve kayıt eder. **on\_t2\_pb\_kmeans\_clicked** fonksiyonu bu figürü graphicsview da gösterir.

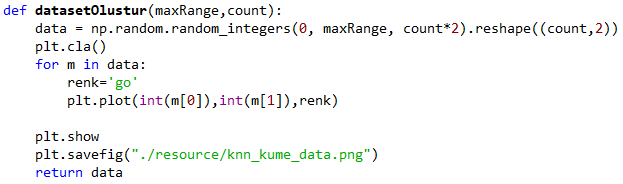


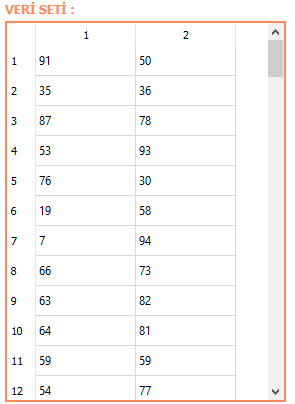
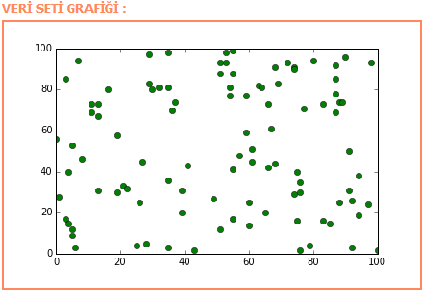


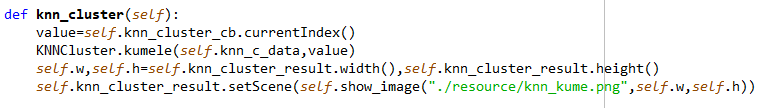
**K-Means Kümeleme Son Hali :**

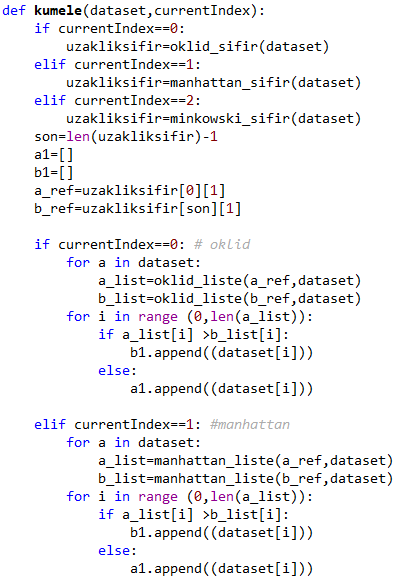
**4- KNN ile Kümeleme**

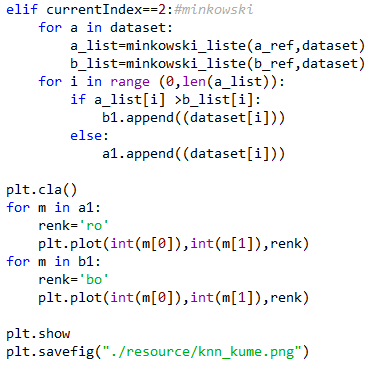
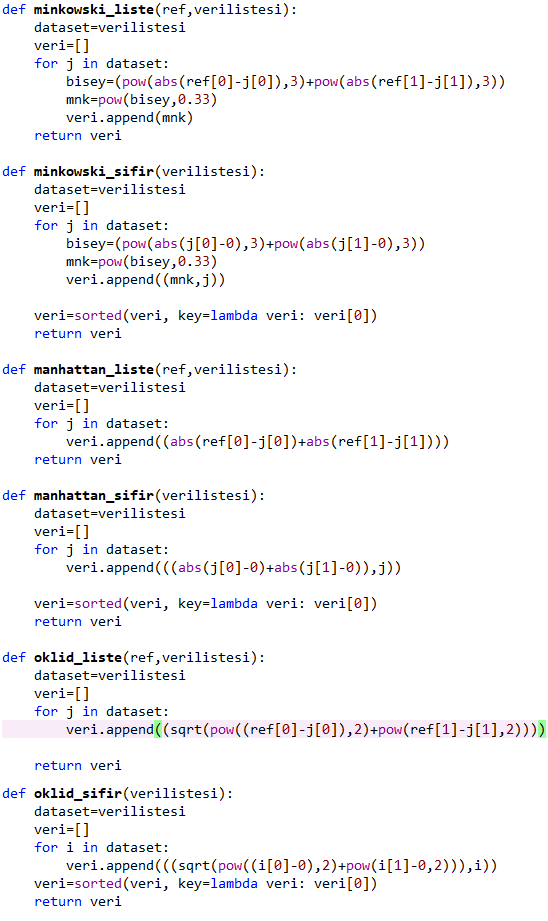
Veri seti oluştur bölümünde rastgele eleman sayısı kadar eleman oluşur.Üst sınırın 100, eleman sayısının 100 olduğu bir dataset oluşturalım. Değerleri girdikten sonra tamam butonuna tıkladığımızda rastgele veriler oluşur bu veriler formdaki tabloya ve graphich view a görsel olarak düşer. Tamam butonuna tıklandığında **knn\_cluster\_create\_dataset** fonksiyonu tetiklenir.

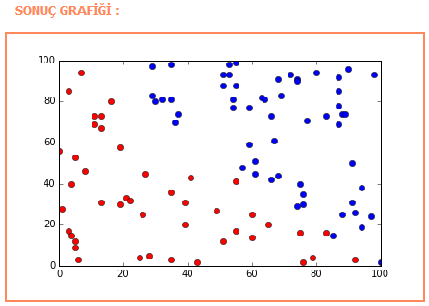
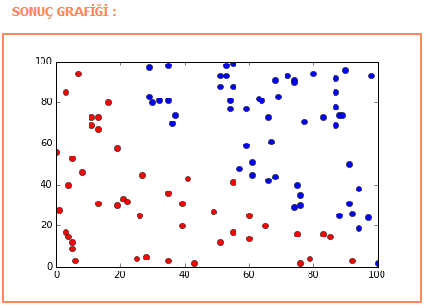
Bu fonksiyon KNNCluster.py dosyasındaki datasetOlustur fonksiyonunu çalıştırır. Dataset oluştur bir dataset oluşturarak bunu plt ile figür haline dönüştürür bu figür knn\_kume\_data adında bir figür oluşturur.  
Bu dataset verileri tabloya dök fonksiyonuyla tabloya çekilir ve show image fonksiyonuyla graphics view da gösterilir.

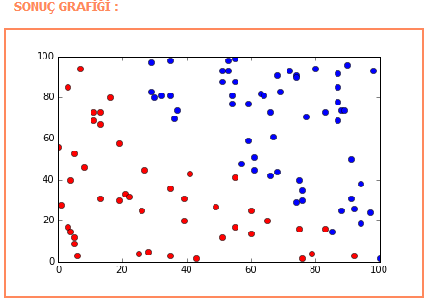


Comboboxtan kümeleme yöntemlerinden herhangi birini seçip kümele butonuna tıkladığımızda **knn\_cluster** fonksiyonu tetiklenir.

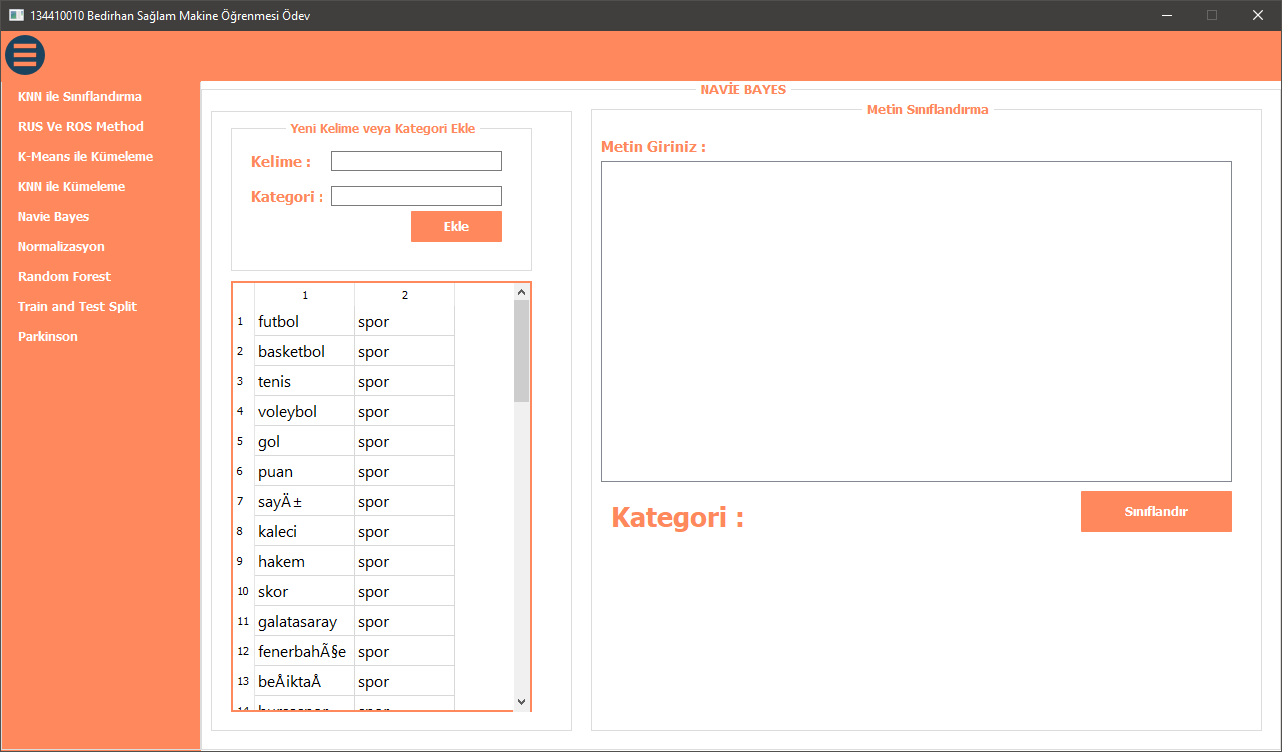
knn\_cluster fonksiyonu KNNCluster.py dosyasından kümele fonksiyonunu çağırır , bu fonksiyon çağılırken comboboxtaki seçili değere göre Öklid,manhattan veya minkowski uzaklıkları kümeleme de baz alınır.

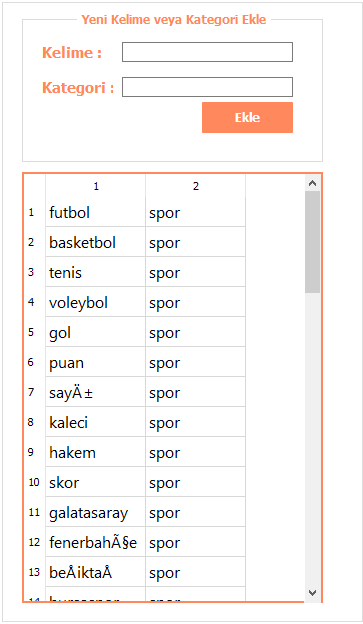
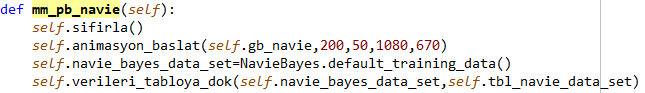
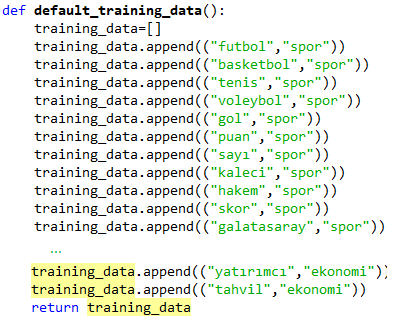
Kumele fonksiyonu gelen değere göre gerekli fonksiyonları çağırarak hesaplamaları yapar.

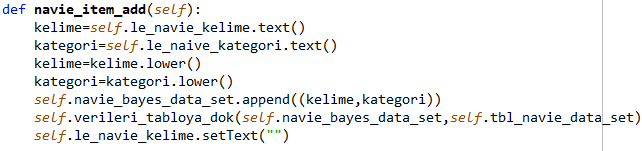
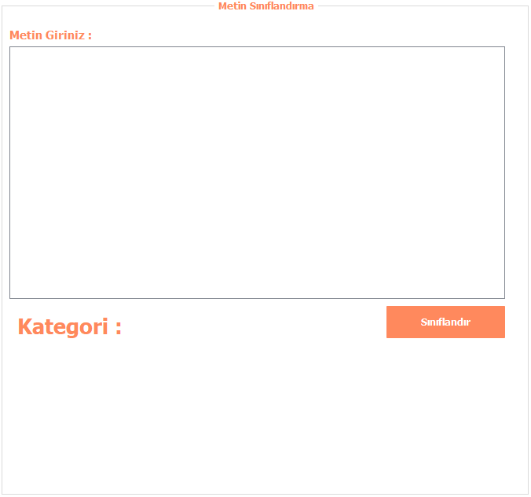
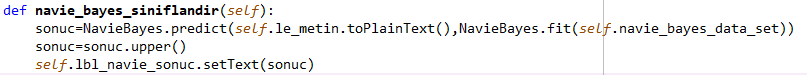
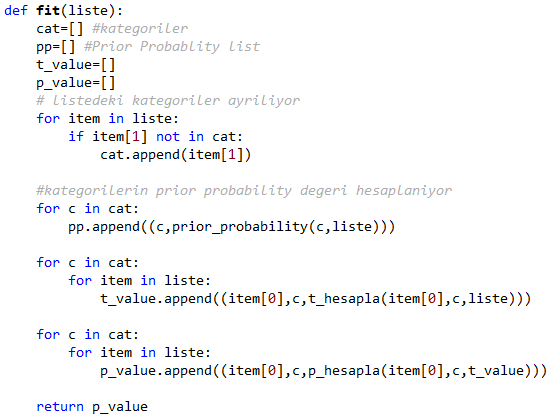
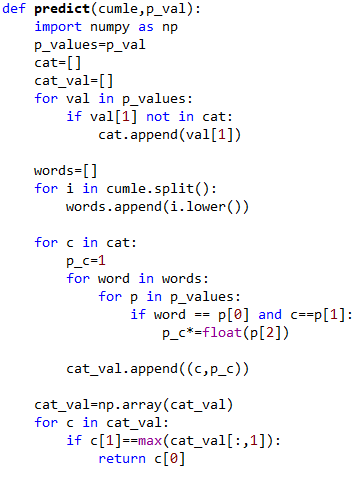
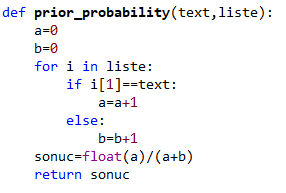
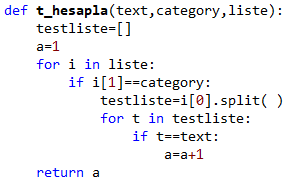
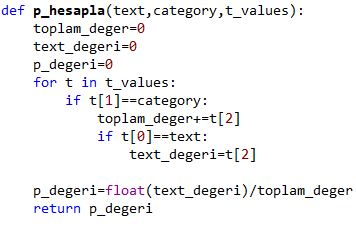
Gerekli işlemler yapıldıktan sonra kümelenen figür kayıt edilir ve graphicsview da bu figür gösterilir.  
**Öklid Sonuç Grafiği : Manhattan Sonuç Grafiği :**

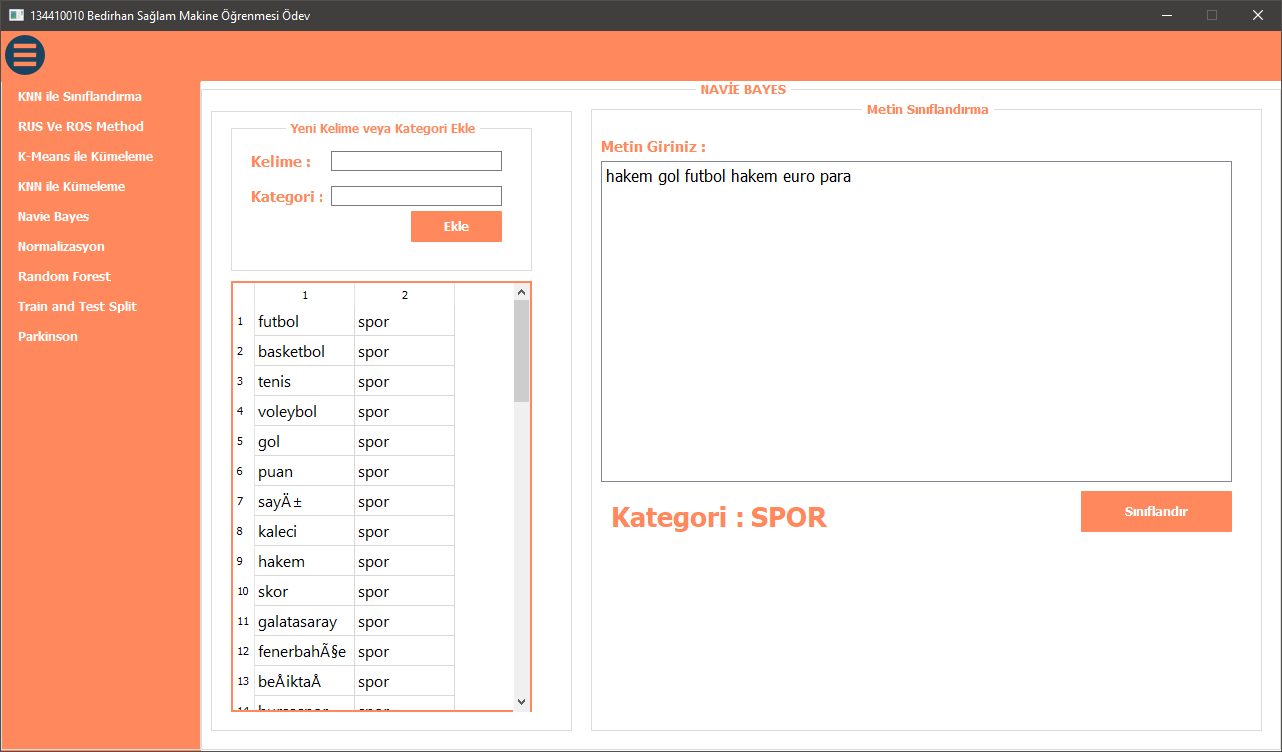
**Minkowski Sonuç Grafiği:**

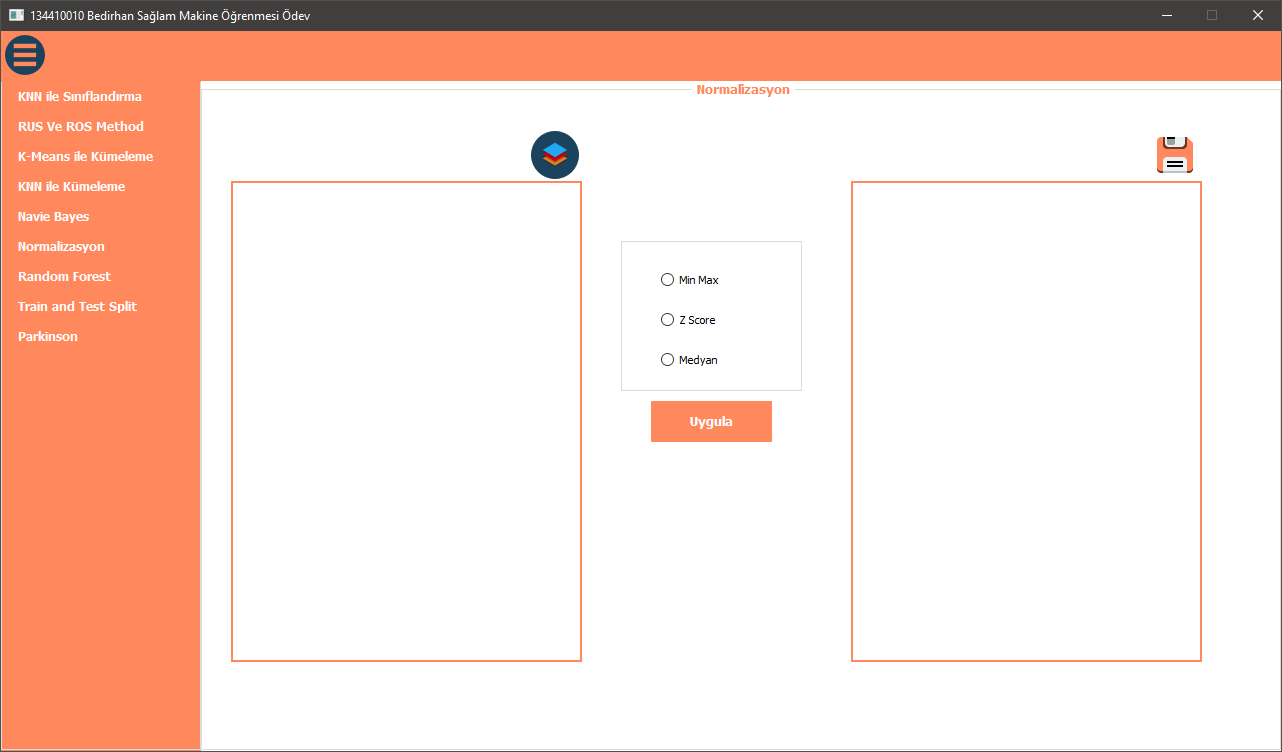
**KNN KÜMELEME SONUÇ:**

**5- Navie Bayes:**

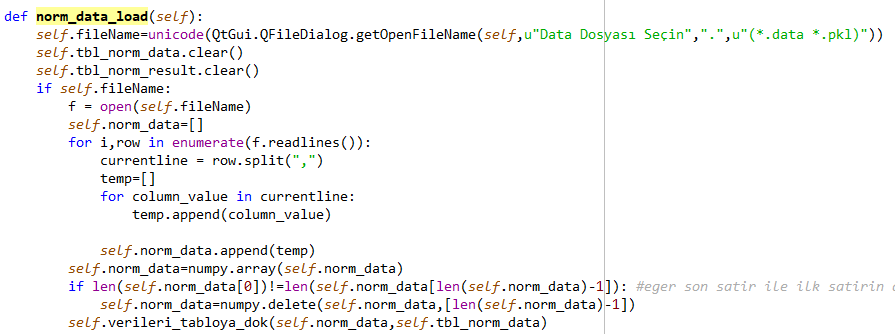
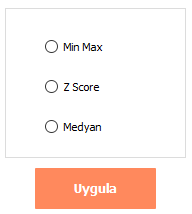
Naive Bayes ile metin sınıflandırma yaparken default olarak spor ve ekonomi verilerinden oluşan 2 kategori tabloya çekiliyor.Farklı kategoriler veya aynı kategoriler için farklı kelimeler eklemek istersek yeni kelime ekle bölümünden kelimeyi ve kategoriyi yazıp ekle butonuna basıyoruz.Tablodaki veriler default olarak hamburger menüdeki naviebayes butonuna tıklandığında **mm\_pb\_navie** fonksiyonunu tetikliyor. Bu fonksiyon NavieBayes .py dosyasındaki default\_training\_data fonksiyonunu çalıştırıyor.Bu fonksiyon tablodaki verileri geri döndürüyor.  
  
Bu default verilere ek yeni veriler eklemek istersek kelimeyi kategoriyi girip ekle butonuna tıklamamız gerekir.Ekle butonuna tıkladıktan sonra **navie\_item\_add** fonksiyonu çalışır.  
Bu fonksiyonla birlikte yeni veriler verisetine eklenmiş olur.

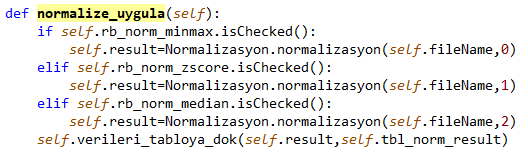
Metin Sınıflandırma bölümünde metin girilir ve bu metinin sınıflandırılması için sınıflandır butonuna tıklanır. Sınıflandır butonuna tıklandıktan sonra navie\_bayes\_siniflandir fonksiyonu tetiklenir.Bu fonksiyon **NavieBayes.py** dosyasının **predict** fonksiyonunu çalıştırır.Predict fonksiyonu fit fonksiyonunda eğitim bilgilerini çekerek yeni gelen cümlenin hangi sınıfa ait olduğunu tahmin eder. Fit fonksiyonu gerekli diğer fonksiyondan bilgiler alarak sınıflandırmayı gerçekleştirir.

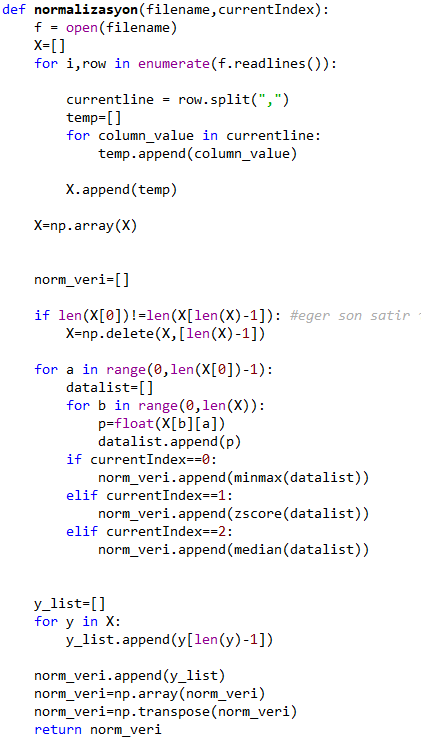
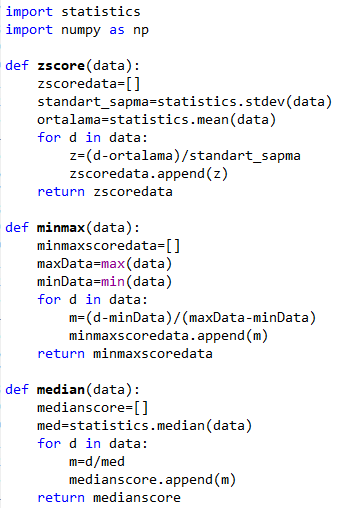
**Navie Bayes Sonuç :**

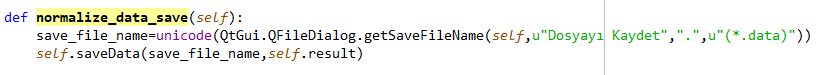
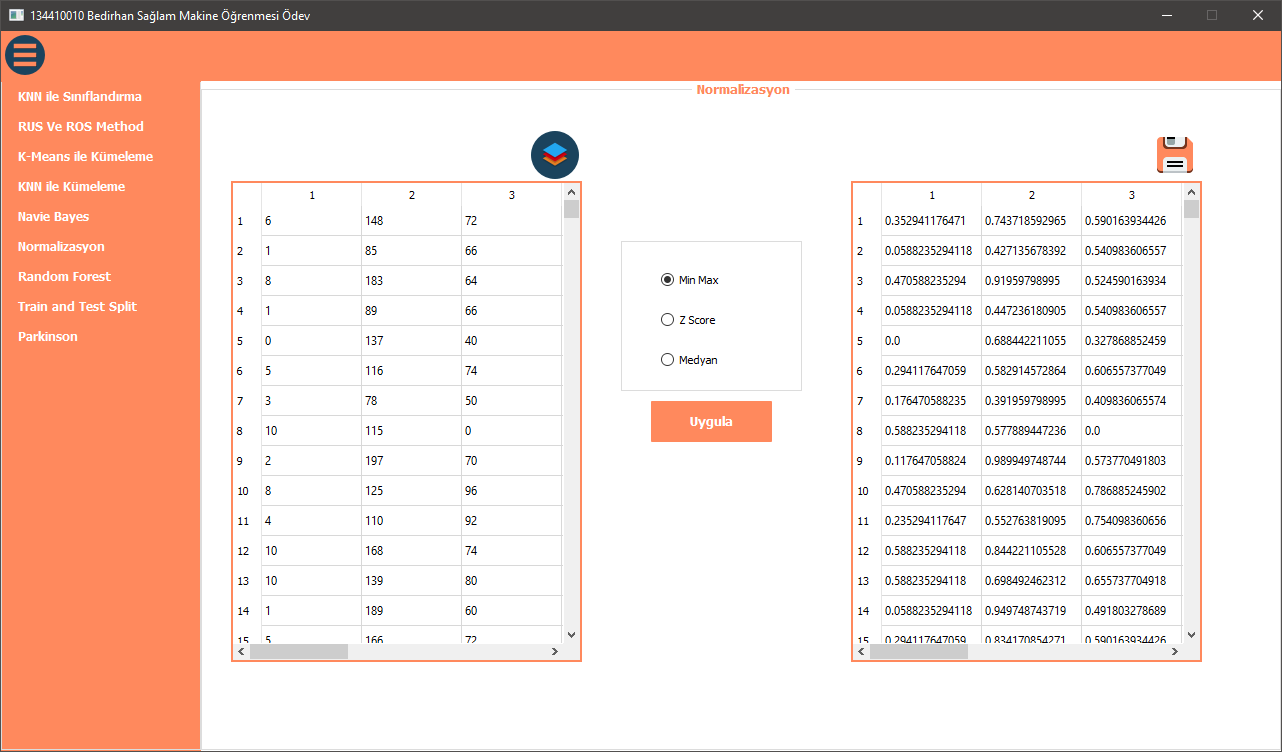
**6- Normalizasyon**

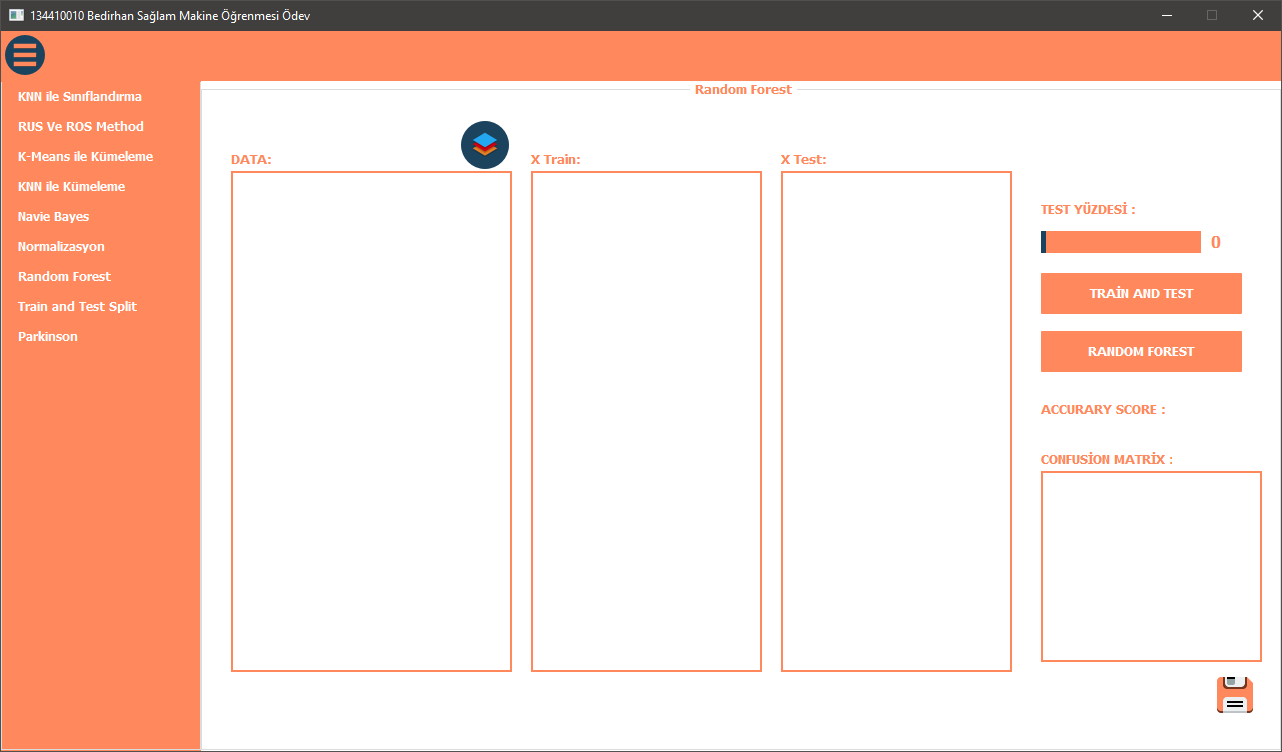
Verileri dosyadan yükle butonuna tıkladıktan sonra **norm\_data\_load** fonksiyonu tetiklenir.Bu fonksiyon bir filedialog açar ve filedialog ile verileri dosyadan çekeriz çekilen veriler tabloda gösterilir.

****

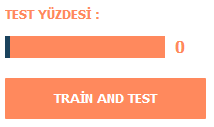
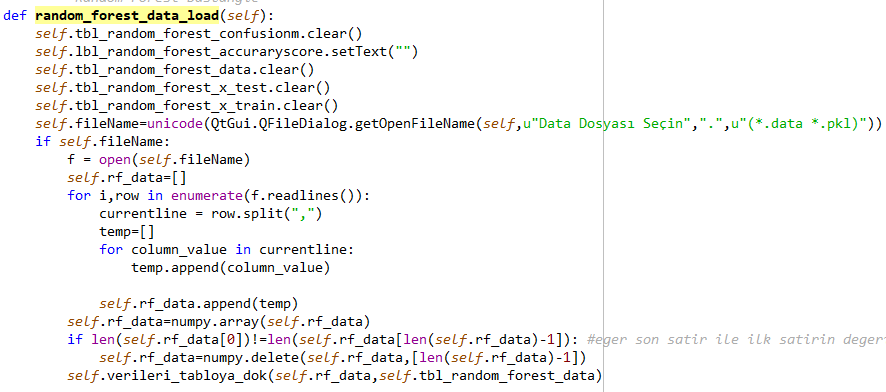
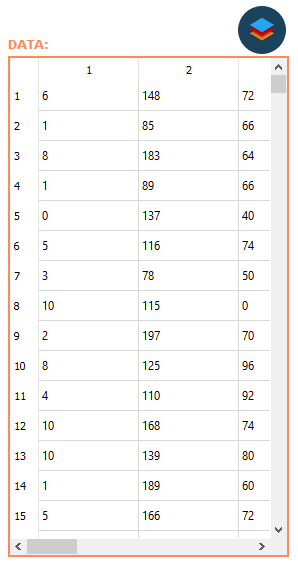
Bu bölümden uygulamak istediğimiz normalizasyon yöntemini seçip uygula butonuna tıklıyoruz. Uygula butonuna tıkladığımızda main.py deki **normalize\_uygula** fonksiyonu tetiklenir. Bu fonksiyon **Normalizasyon.py** dosyasındaki **normalizasyon** fonksiyonunu çalıştırır.

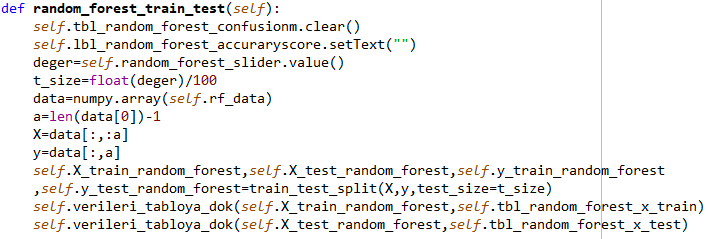
Yandaki normalizasyon fonksiyonu aldığı veriyi seçilen normalizasyon yöntimine göre normalize edip normalize edilmiş datayı geri döndürür. Geri dönen değer **normalize\_uygula** fonksiyonundaki verileri tabloya dök fonksiyonuyla tabloya aktarılır.

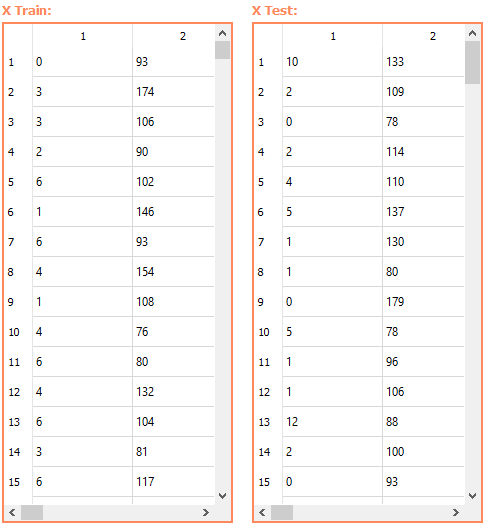
  
Seçtiğimiz yönteme göre normalize edilen veriyi kayıt etmek için bu butonu kullanıyoruz. Kayıt et butonuna tıkladığımızda normalize\_data\_save fonksiyonu tetiklenir. Bir filedialog açılır ve kayıt etmek istediğimiz konuma istediğimiz isimle dosyamızı .data uzantılı olarak kayıt edebiliriz.

**7-Random Forest :**

Verileri dosyadan yükle butonuna tıkladığımızda main.py deki random\_forest\_data\_load fonksiyonu tetiklenir ve bir filedialog açılır bir data dosyası seçtikten sonra seçtiğimiz data dosyası tabloya yüklenir.

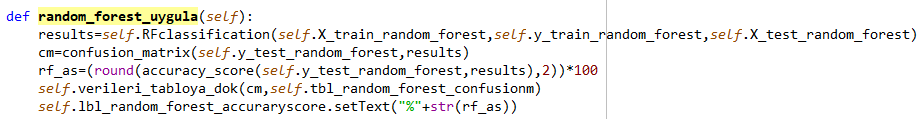
  
Data tabloya yüklendikten sonra train ve test olarak veriyi ayırmak için train test bölümünü kullanırız.

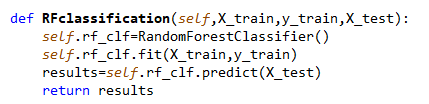
Bu bölümde test yüzdesini seçtikten sonra train and test butonuna tıklarız. Bu buton **main.py** deki **random\_forest\_train\_test** fonksiyonunu tetikler.

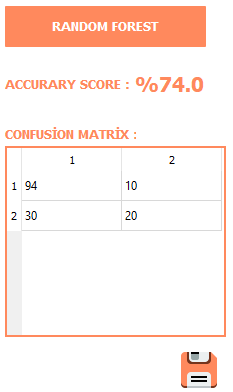


Bu fonksiyon verileri train ve test olarak belirlediğimiz oranda ayırır.

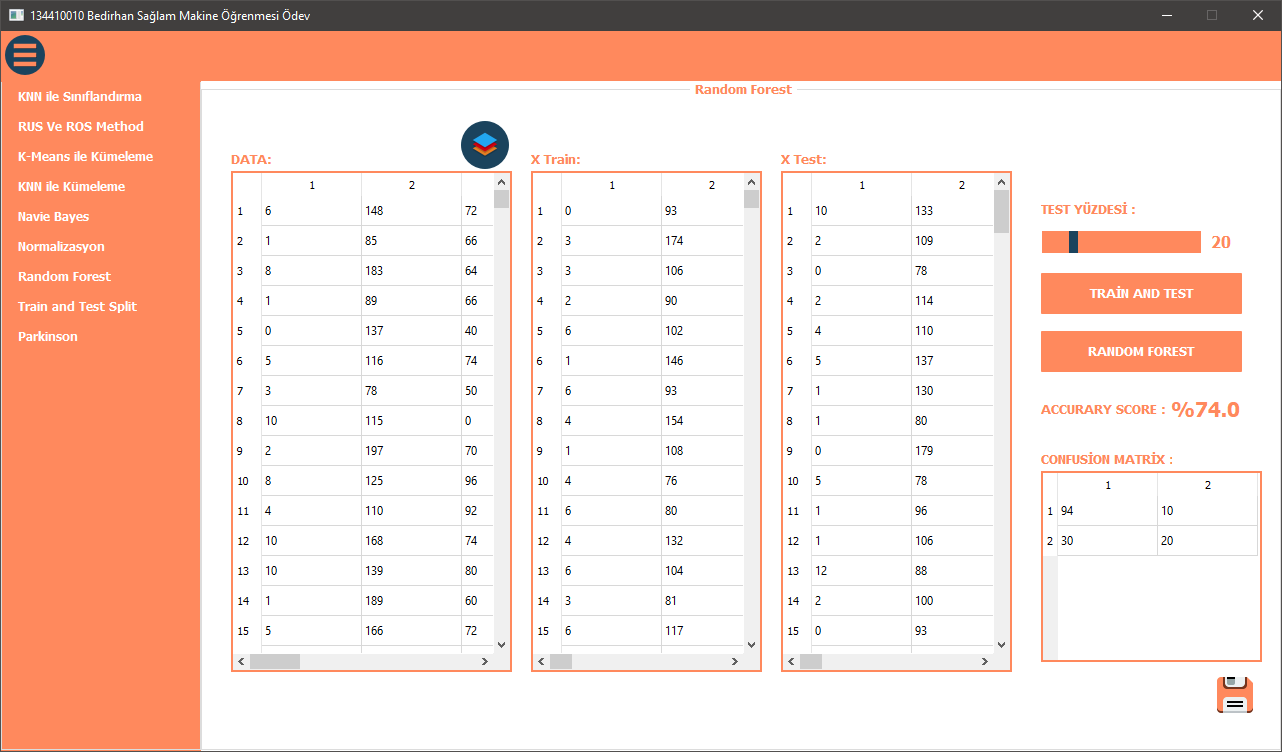
Veriyi yükleyip train ve test olarak ayırdıktan sonra en son yapılacak işlem olarak random forest butonuna tıklayarak  
**random\_forest\_uygula** fonksiyonunu çalıştırırız.

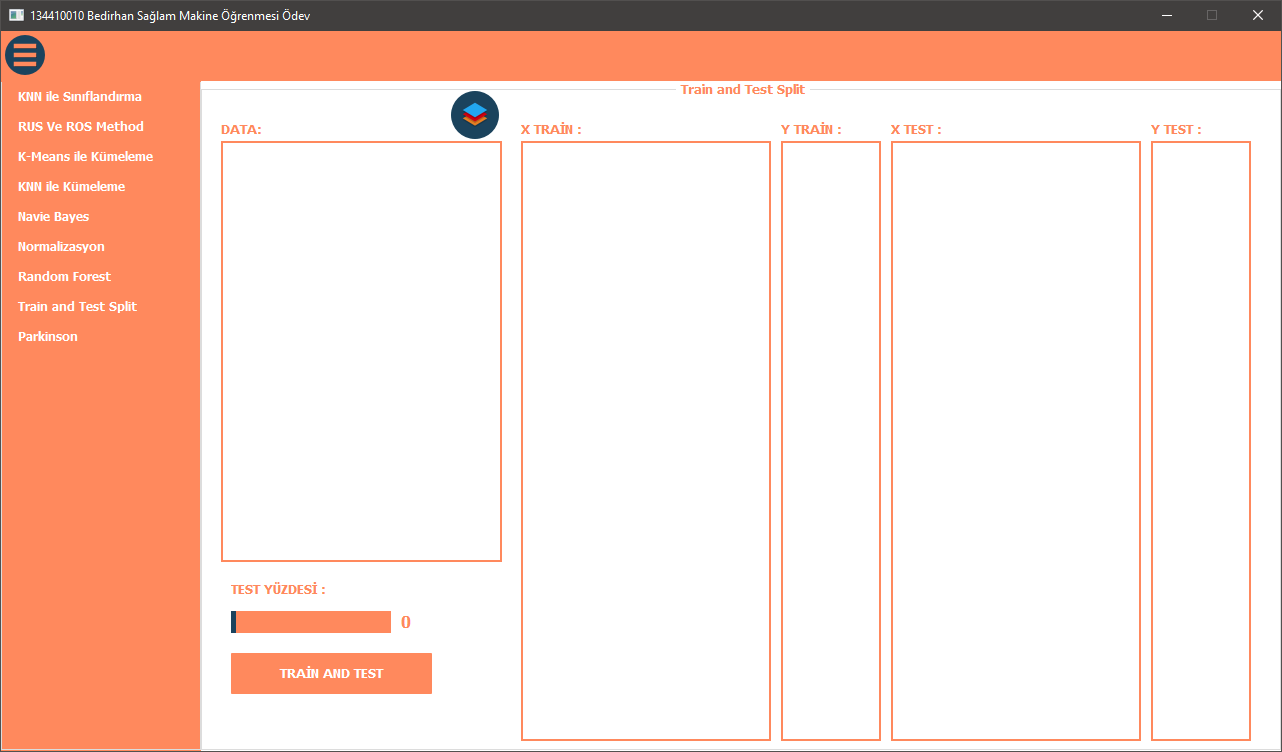
Bu fonksiyon sınıflandırma ve test yapabilmek için Main.py deki RFclassification fonksiyonunu çalıştırır.  
RFclassification fonksiyonu train ve test verilerini alıp işleyerek bir sonuç değeri döndürür .  
Bu sonuç değeriyle y\_test değerini karşılaştırarak karmaşıklık matrisini (confusion matrix) ve doğruluk puanını (accurary score) hesaplayabiliriz.

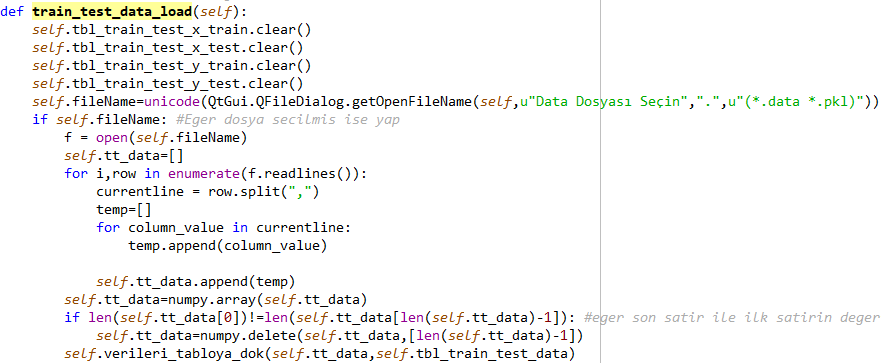


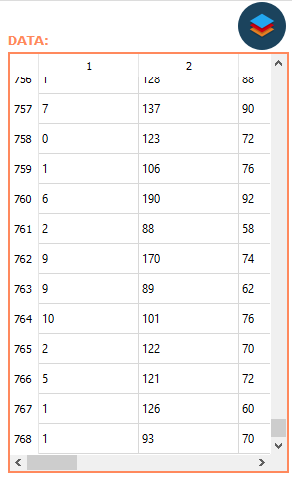
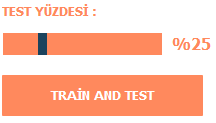


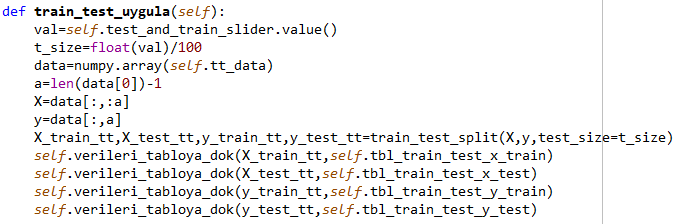
Oluşturulan modeli kayıt etmek istersek kayıt et butonuna tıklayarak random\_forest\_model\_save fonksiyonunu tetikleriz. Bu fonksiyon bir file dialog açar ve istediğimiz dizine istediğimiz isimle .pkl uzantılı model dosyalarını kayıt edebiliriz.

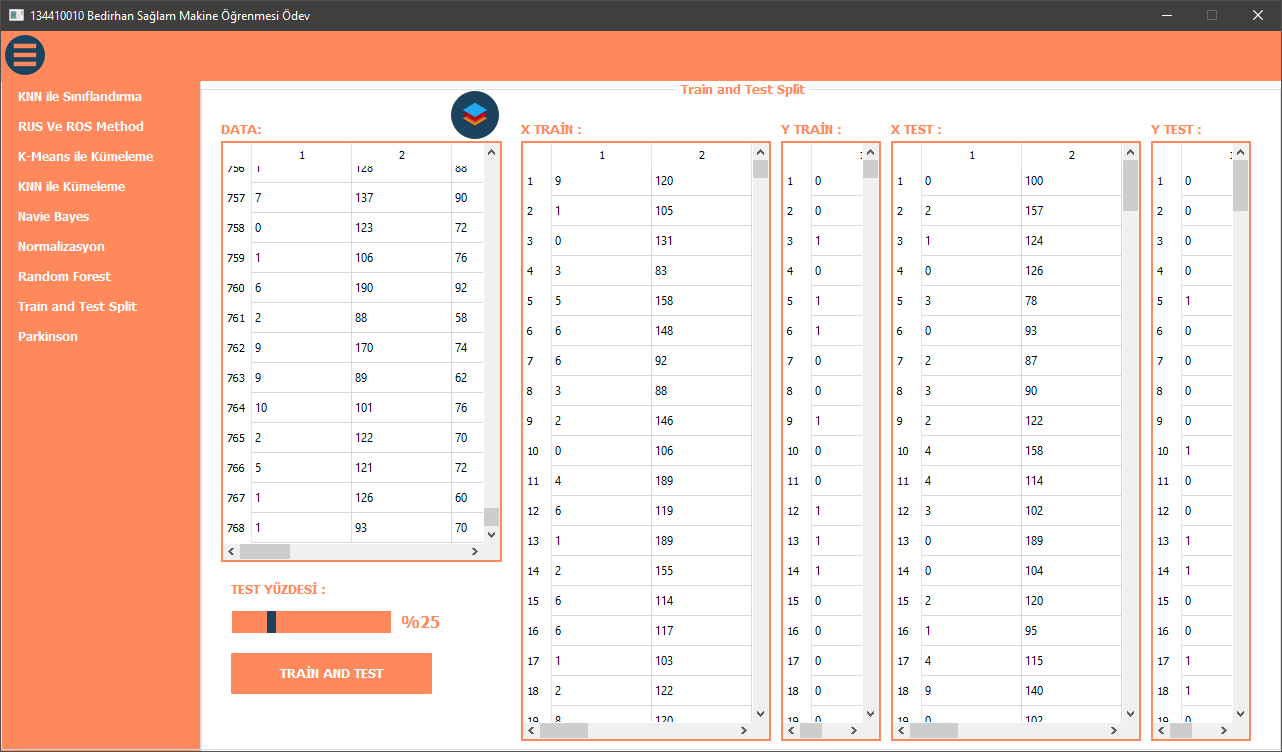
**Random Forest Sonuç :**

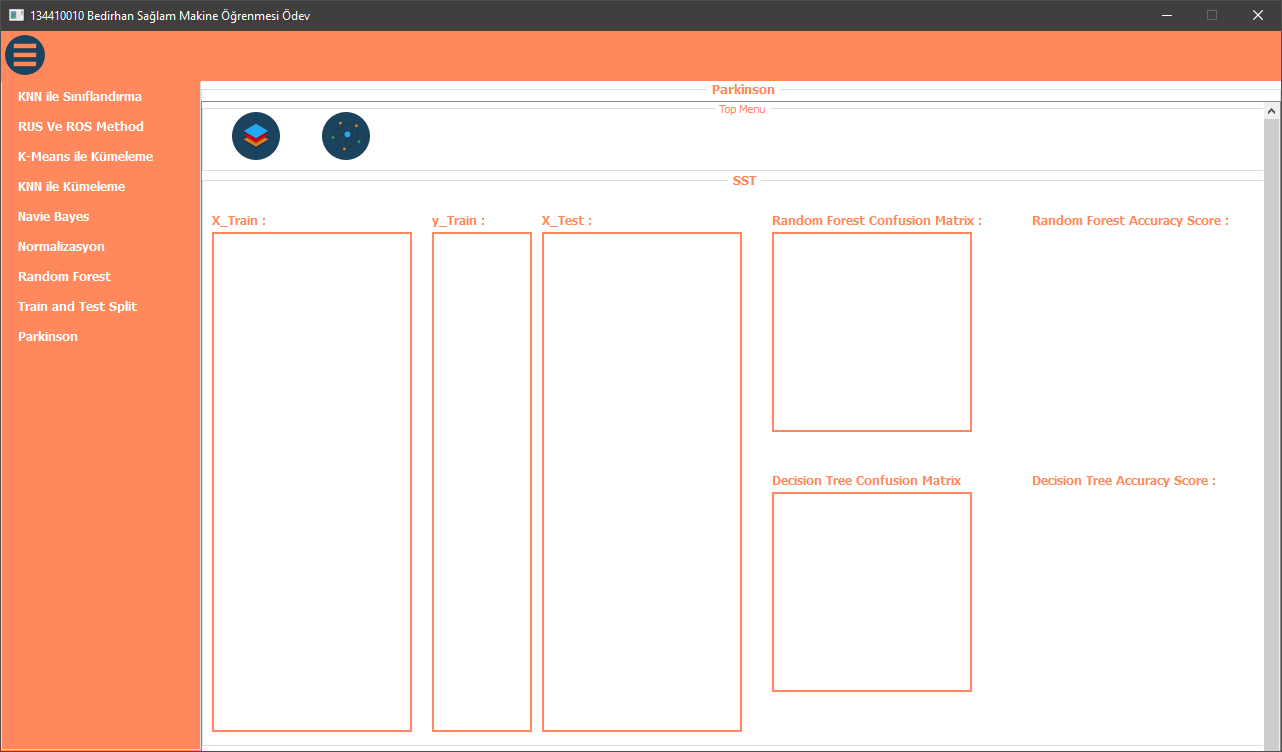
**8- Train & Test Split**

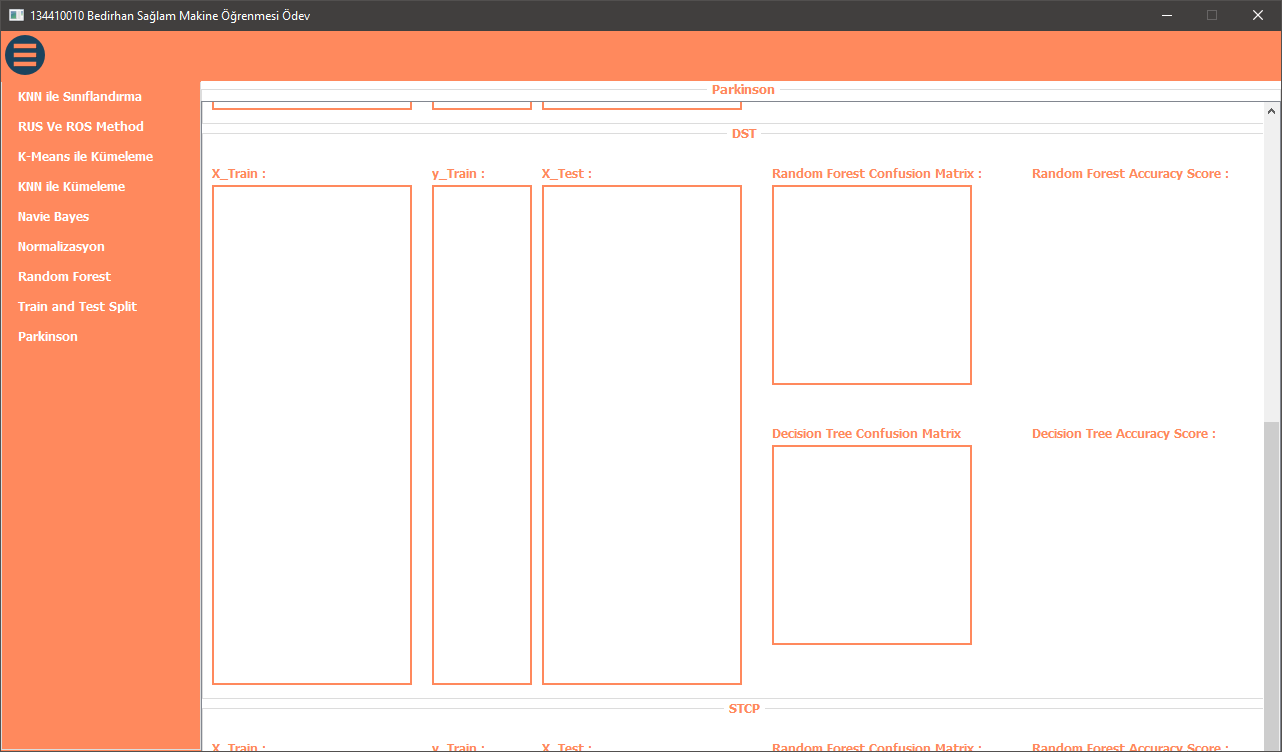
Verileri dosyadan yükle butonuna tıklayarak , **Main.py** deki **train\_test\_data\_load** fonksiyonunu tetikleriz bu fonksiyon bir filedialog açarak istediğimiz .data uzantılı dosyayı tabloya dökmemize yarar.

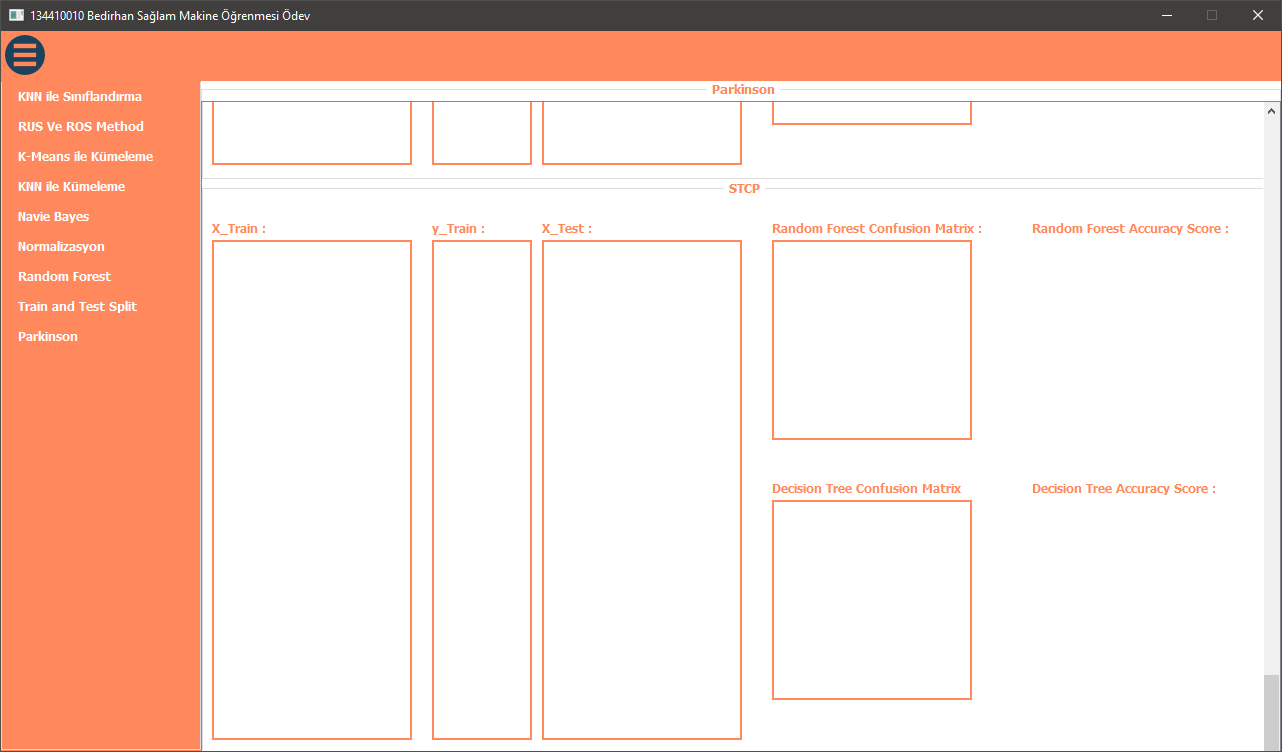
Veriler tabloya döküldükten sonra , test yüzdesini sliderdan belirliyoruz daha sonra train and test butonuna tıklayarak **Main.py** deki **train\_test\_uygula** fonksiyonu tetiklenir bu fonksiyon gelen veriyi , istenen yüzdeye göre train ve test olarak ayırıp sonuçları gerekli tablolara bastırır.

**Train & Test Sonuç :**



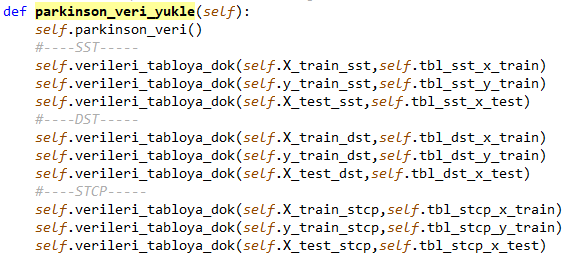
**9- Parkinson**

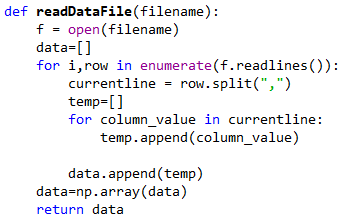
**Parkinson DST:**

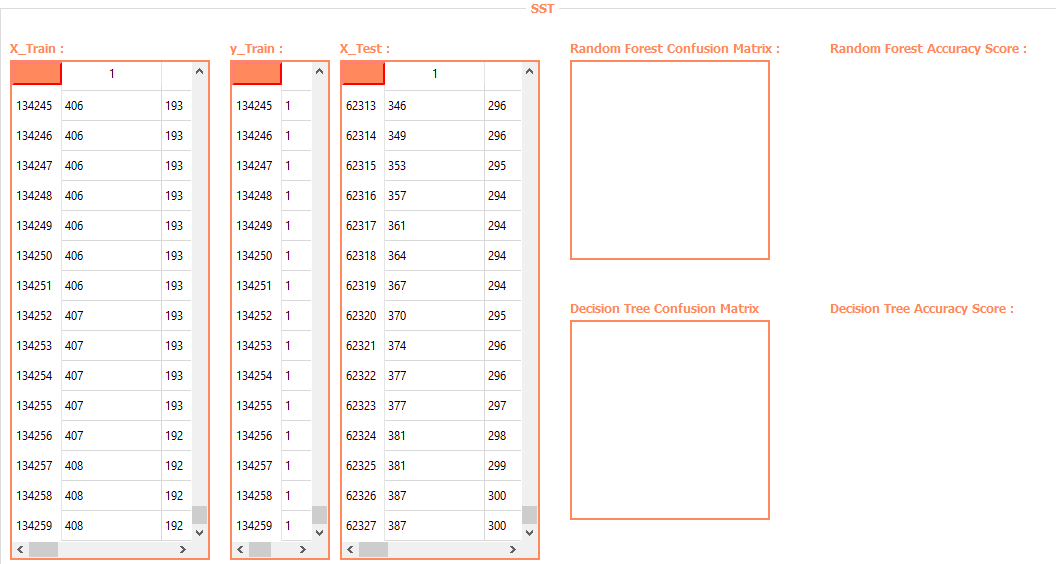
**Parkinson STCP:**

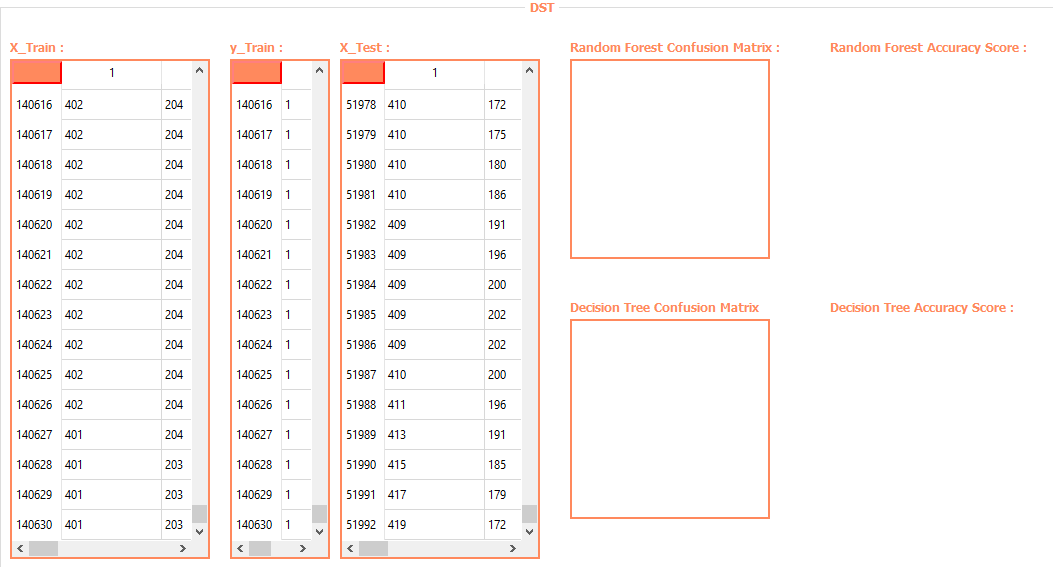
****

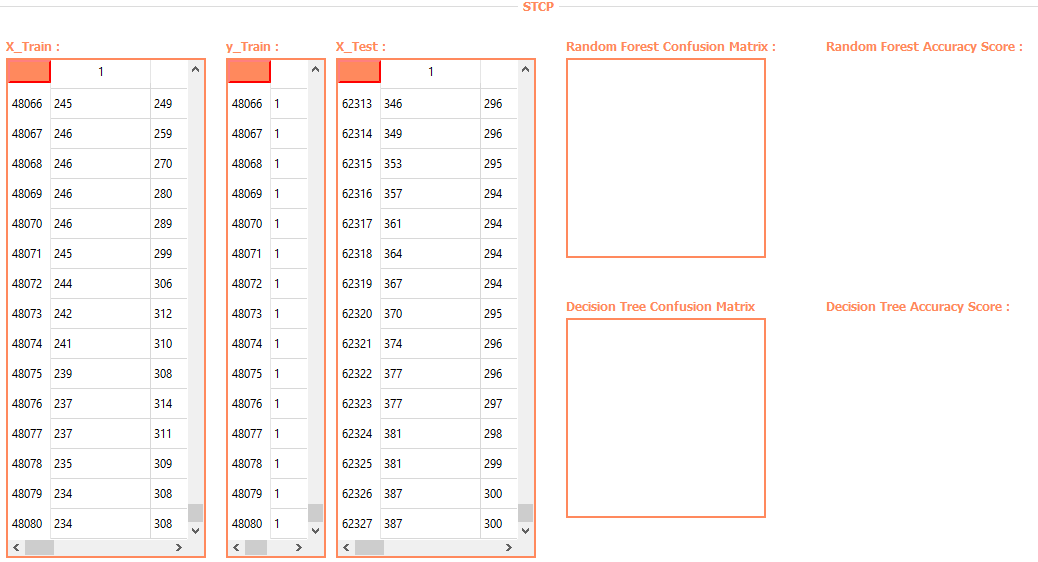
Verileri yükle butonuna tıkladıktan sonra **Main.py** deki **parkinson\_veri\_yukle** fonksiyonu tetiklenir bu fonksiyon , **main.py deki parkinson\_veri** fonksiyonunu çağırır. Buradan dönen değerleri ilgili tablolara basar.

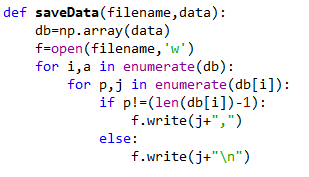
****

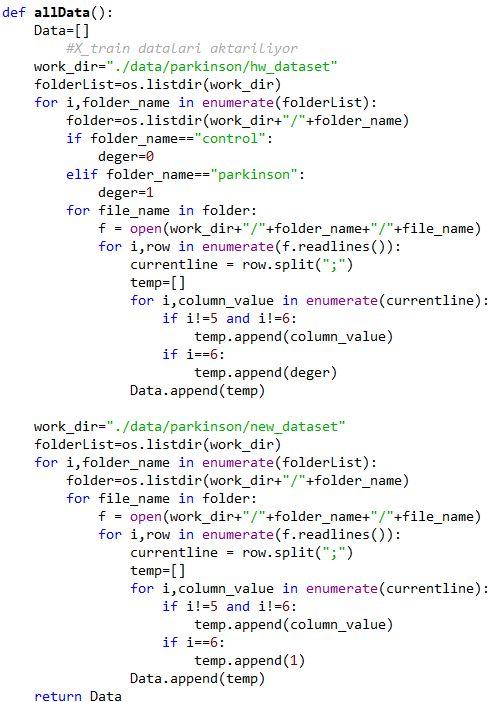
**ParkinsonDataSet.py** dosyasındaki **readDataFile** fonksiyonu ile gerekli datalar okunuyor ve gerekli değişkenlere atanıyor. .data uzantılı bu dosyalar daha önceden oluşturuldu.

SST DATA:

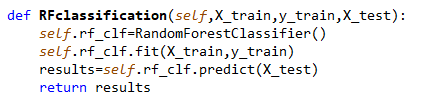
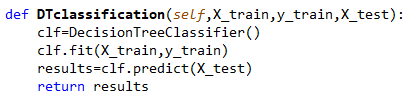
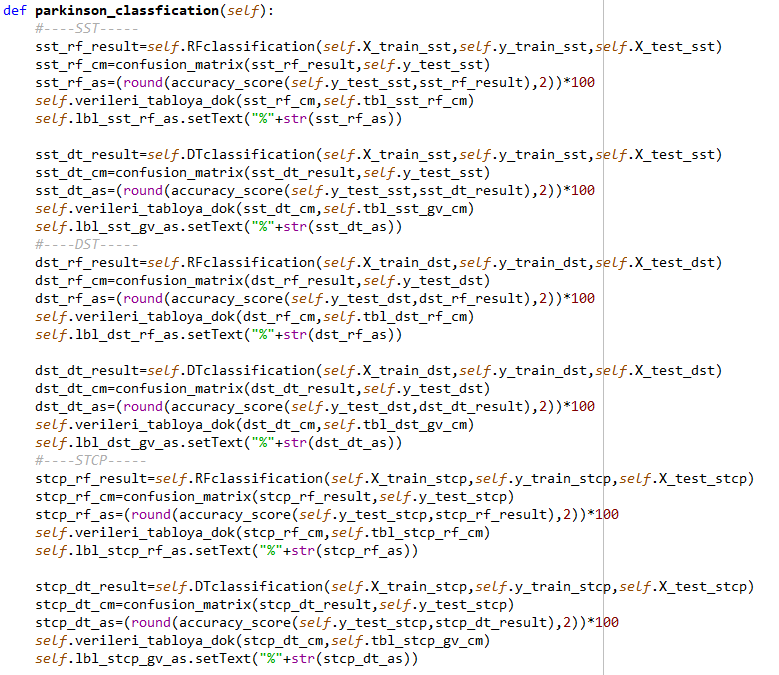
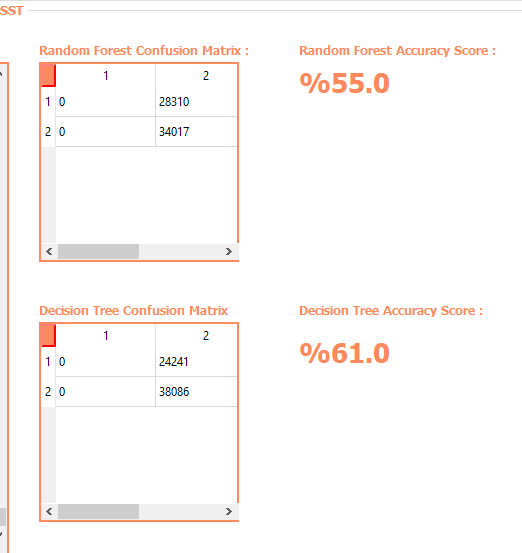
DST DATA:

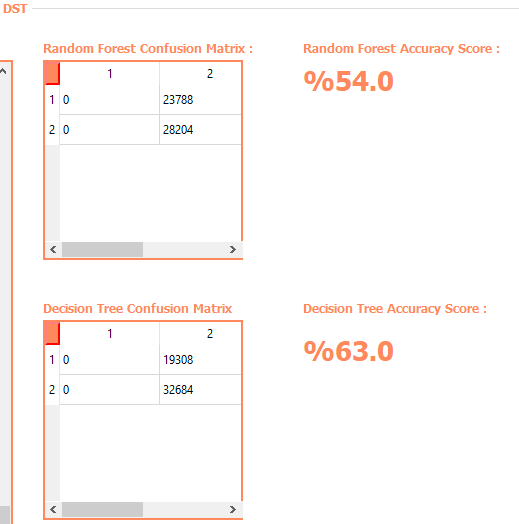
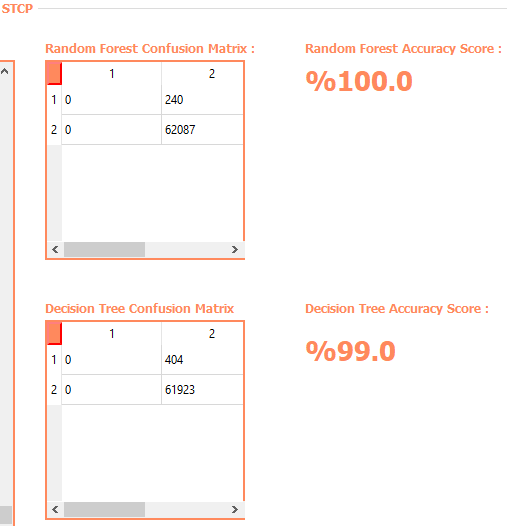
STCP DATA:

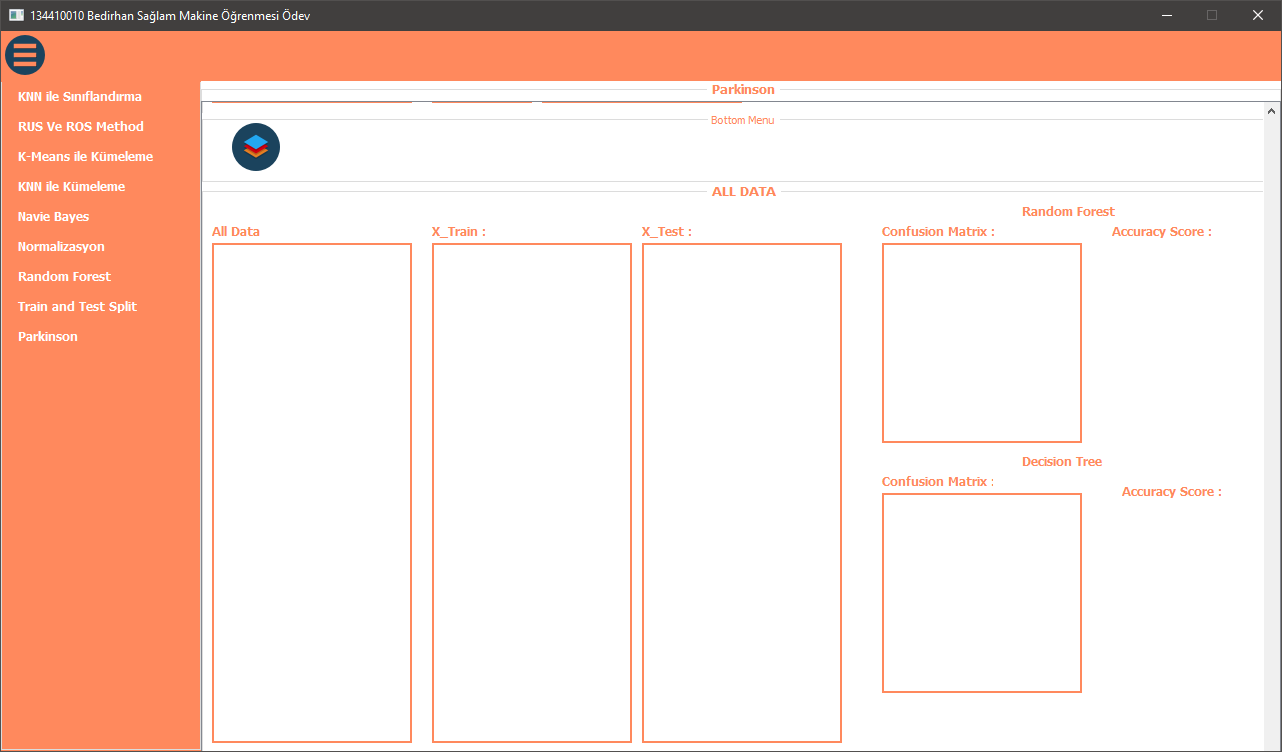
.data uzantılı veri dosyalarını oluşturmak için ParkinsonDataSet.py de saveData fonksiyonu kullanıldı. Dosya adını ve kayıt edilecek veriyi gönderip dosyayı kayıt ediyoruz.

Örneğin saveData(all.data,aldata()) ifadesini kullanarak all.data isimli bütün Parkinson verilerinin tutulduğu .data uzantılı verisetimiz hazırlanıyor. Diğer bütün verisetleri bu şekilde hazırlanarak anaformda bu datasetleri çekilmiştir. .data uzantıları olmadanda verileri direkt klasörden çekmek mümkündür. Main.py deki parkinson\_veri fonksiyonundaki yorum satırına alınmış kodları açıp verileri direkt klasördende çekebiliriz.  
**ParkinsonDataSet.py dataset oluşturulan fonksiyonlar :**

  
  
Datasetler yüklendikten sonra verileri sınıflandır butonuna tıklayıp **Main.py** deki **parkinson\_classfication** fonksiyonu tetiklenir. Bu fonksiyon çekilen verileri **RFclassification** ve **DTclassification** göndererek sınıflandırma yapar ve sonuçları test değerleriyle karşılaştırır , gelen sonuçlara göre confusion matrixler ve accuary scorelar ilgili tablolara set eder.

  
**SST Random Forest ve Decision Tree Sonuçlar :**

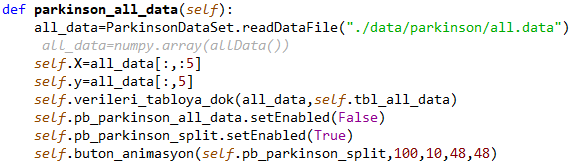
**DST Random Forest ve Decision Tree Sonuçlar : STCP Random Forest ve Decision Tree Sonuçlar :**

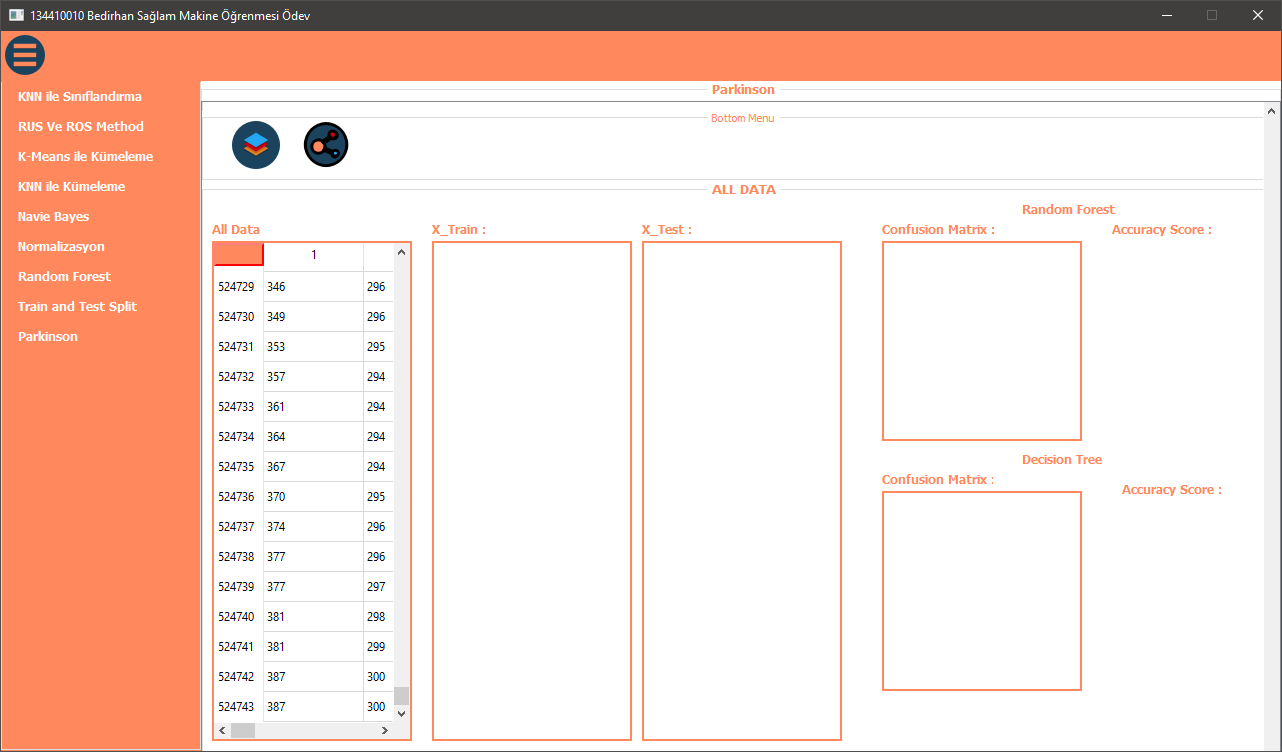
 **Parkinson Bölüm 2 Bütün Veriler :**

C:\Users\Bedirhan\Desktop\MakineOgrenmesiOdev\icons\data_load.pngBu bölümde olaylar sırayla gerçekleşiyor ; animasyonlarla ve enabled fonksiyonlarıyla kontrollü bir ilerleme oluyor.

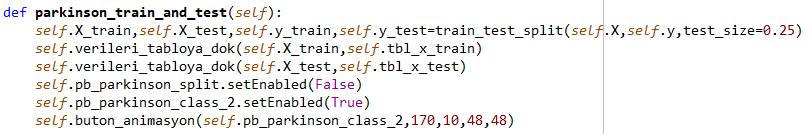
C:\Users\Bedirhan\Desktop\MakineOgrenmesiOdev\icons\reload.pngC:\Users\Bedirhan\Desktop\MakineOgrenmesiOdev\icons\class.pngC:\Users\Bedirhan\Desktop\MakineOgrenmesiOdev\icons\split.png Dataları all.data dosyasından yükle  
  
Yüklenen dataları %25 e %75 split et  
  
  
Split edilmiş verileri sınıflandır , sınıflandırma başarılarını ölç

Tekrar baştan yap (Verileri split etme bölümünde random split yapıldığı için her splitte farklı datalar kullanılıyor)

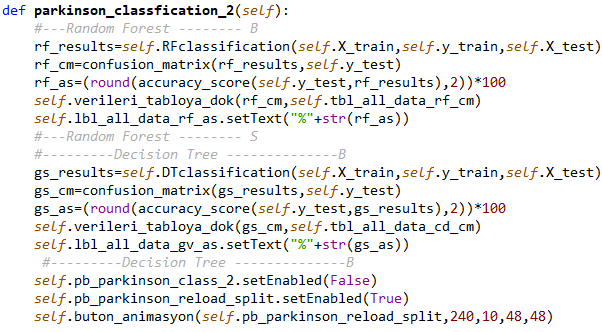
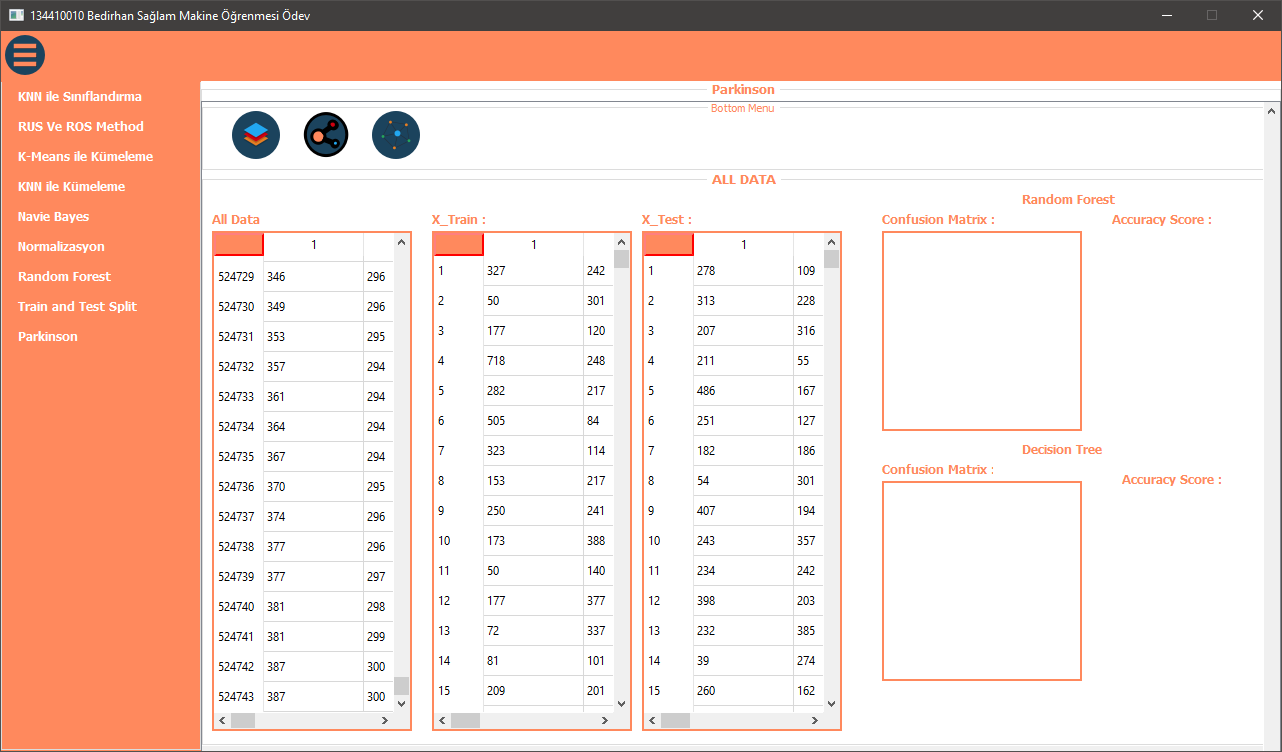
**C:\Users\Bedirhan\Desktop\MakineOgrenmesiOdev\icons\data_load.pngDataları all.data dosyasından yükle :** Bu butona tıkladığımızda parkinson\_all\_data fonksiyonu tetiklenir.

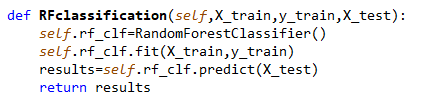
  
Datalar all.data dosyasından okunur X ve y değerleri ayrıştırılır. Bütün veriler gerekli tabloya aktarılır ve bu buton artık pasif yapılır.

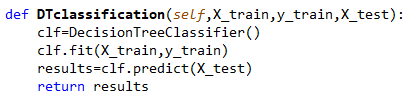
C:\Users\Bedirhan\Desktop\MakineOgrenmesiOdev\icons\split.png

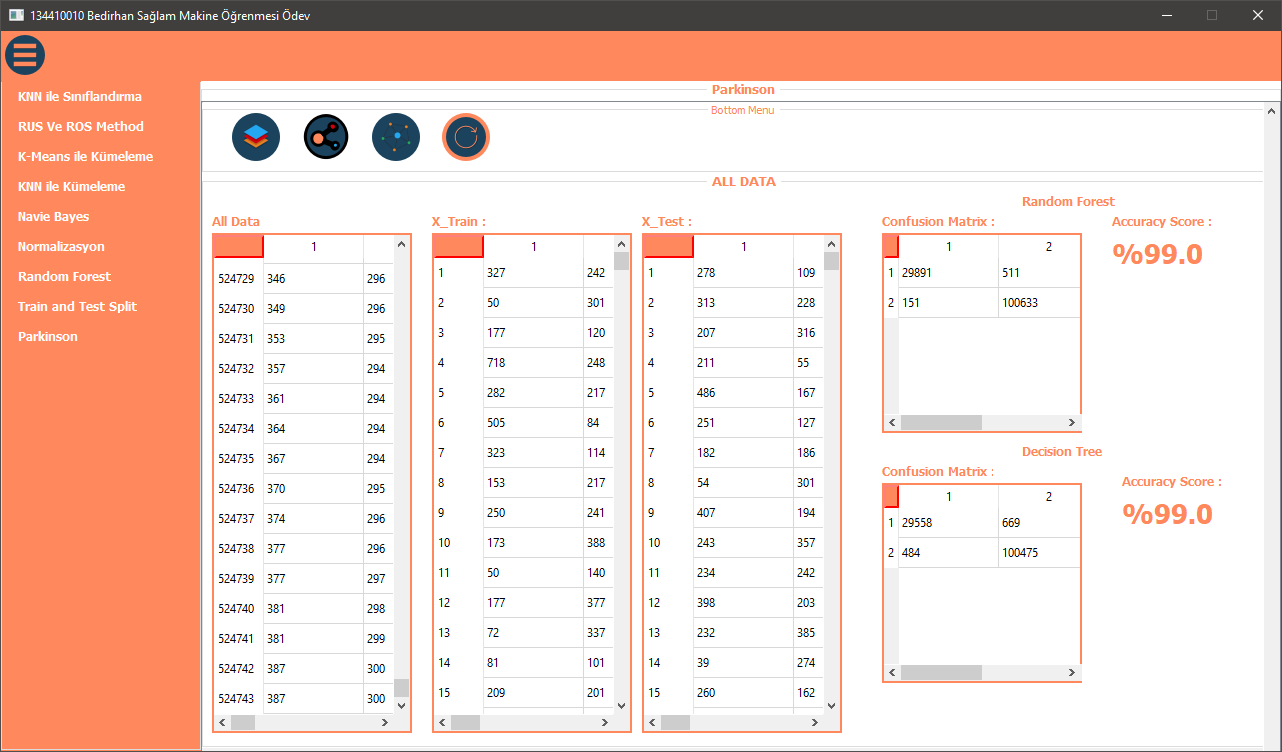
**Yüklenen dataları %25 e %75 split et :** Split butonuna tıklandığında **Main.py** deki **parkinson\_train\_and\_test** fonksiyonu tetiklenir

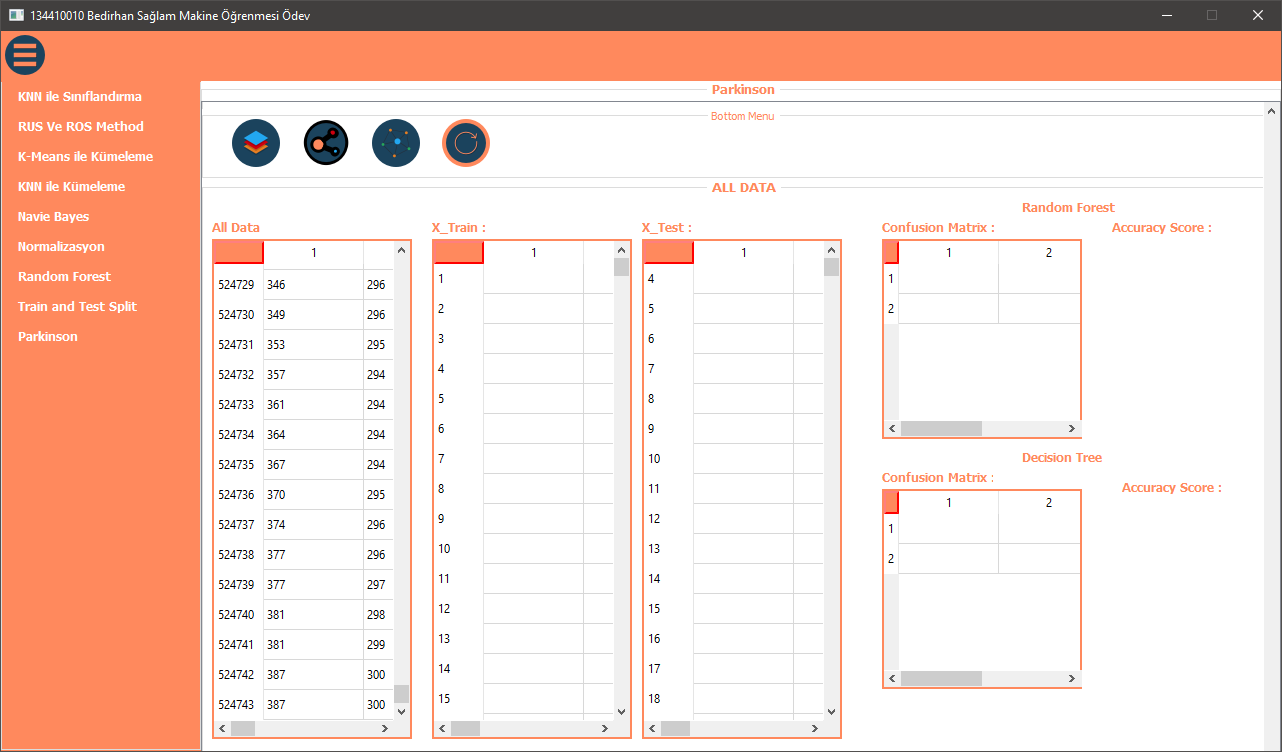
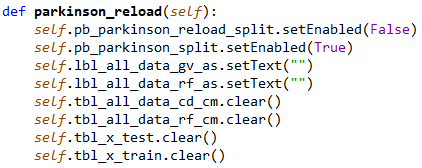
**Main.py** deki **parkinson\_all\_data** fonksiyonunda alınan X ve y burada train ve test için kullanılır.Veriler randomize olarak ayrıştırılır , ilgili tabloya ilgili veriler yüklenir. Bu buton pasif yapılır yeni işlem için sonraki buton aktif edilir.

  
**Verileri sC:\Users\Bedirhan\Desktop\MakineOgrenmesiOdev\icons\class.pngınıflandır , sınıflandırma başarılarını ölç , karmaşıklık matrisini göster :** Bu butona tıklanıldığında **Main.py** deki **parkinson\_classfication\_2** fonksiyonu tetiklenir.Bu fonksiyon split edilen verileri RFclassification ve DTclassification fonksiyonlarına gönderir gelen sonuçlarla test verilerini karşılaştırarak karmaşıklık matrisini ve doğruluk puanını ölçer, bu verileri gerekli tablolara ve labellara aktarır.





C:\Users\Bedirhan\Desktop\MakineOgrenmesiOdev\icons\reload.png

**Bütün işlemleri 2.adımdan başlayarak tekrar yap:** Bu butona tıkladığımız zaman **Main.py** deki **parkinson\_reload** fonksiyonu tetiklenir. Bu fonksiyon split butonunu tekrar aktif ederek işlemleri yeniden yapmamıza olanak sağlar ve bulunan verileri temizler.