KARABÜK ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSİTÜSÜ

YÜKSEK LİSANS BİYOMEDİKAL MÜHENDİSLİĞİ BÖLÜMÜ

Dersin Adı:BMM723Biyomedikal Mühendisliğinde Yapay Sinir Ağı Uygulamaları

Proje Adı: MLP ve KERAS ÇALIŞMASI

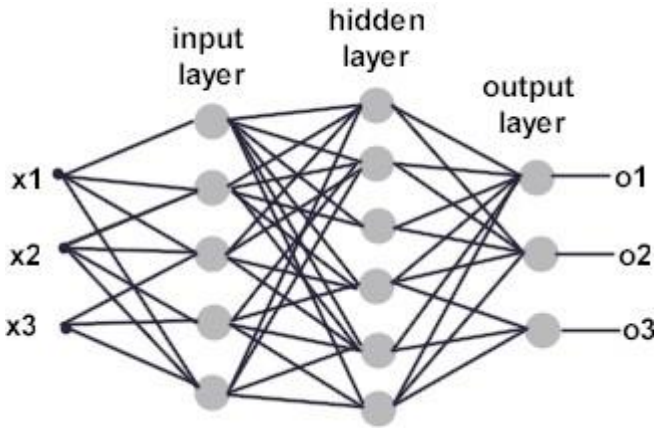
Öğrencinin Adı: TUĞBA TAŞBAŞI

Öğretim Elemanı: Dr.Öğr. Üyesi HAKAN YILMAZ

Özet Bilgi

1) MULTİ LAYER PERCEPTRON (MLP-ÇOK KATMANLI ALGILAYICILAR)

Beyin milyonlarca nörondan oluşur, bu nedenle bir sinir ağı, sadece farklı yollarla bağlanan ve farklı aktivasyon fonksiyonlarında çalışan perceptronların bir bileşimidir. Çok Katmanlı Algılayıcılar (MLP) XOR Problemi'ni çözmek için yapılan çalışmalar sonucu ortaya çıkmıştır. MLP özellikle sınıflandırma ve genelleme yapma durumlarında etkin çalışır. Çok Katmanlı Ağ'ların yapısı aşağıdaki gibidir.



Birçok giriş için bir nöron yeterli olmayabilir. Paralel işlem yapan birden fazla nörona ihtiyaç duyulduğunda katman kavramı devreye girer.Single Perceptron Model'den farklı olarak arada gizli (hidden) katman bulunmaktadır. Giriş katmanı gelen verileri alarak ara katmana gönderir. Gelen bilgiler bir sonraki katmana aktarılırlar. Ara katman sayısı en az bir olmak üzere probleme göre değişir ve ihtiyaca göre ayarlanır. Her katmanın çıkışı bir sonraki katmanın girişi olmaktadır. Böylelikle çıkışa ulaşılmaktadır. Her işlem elemanı yani nöron bir sonraki katmanda bulunan bütün nöronlara bağlıdır. Ayrıca katmandaki nöron sayısı da probleme göre belirlenir. Çıkış katmanı önceki katmanlardan gelen verileri işleyerek ağın çıkışını belirler. Sistemin çıkış sayısı çıkış katmanında bulunan eleman sayısına eşittir.

sklearn.neural_networkMLPClassifier

Parameters;

- **hidden_layer_sizes***tuple, length = n_layers - 2, default=(100,)*

Gizli katman boyutu

- **solver***{'lbfgs', 'sgd', 'adam'}, default='adam'*

- 'lbfgs', yarı Newton metotları ailesinde bir optimize edicidir.
- 'sgd' stokastik gradyan iniş anlamına gelir.
- 'adam', Kingma, Diederik ve Jimmy Ba tarafından önerilen stokastik gradyan tabanlı bir optimizer anlamına gelir

Varsayılan çözücü 'adam', hem eğitim süresi hem de geçerlilik puanı açısından nispeten büyük veri kümelerinde (binlerce eğitim örneği veya daha fazlası ile) oldukça iyi çalışır. Bununla birlikte, küçük veri kümeleri için 'lbfgs' daha hızlı birleşebilir ve daha iyi performans gösterebilir.

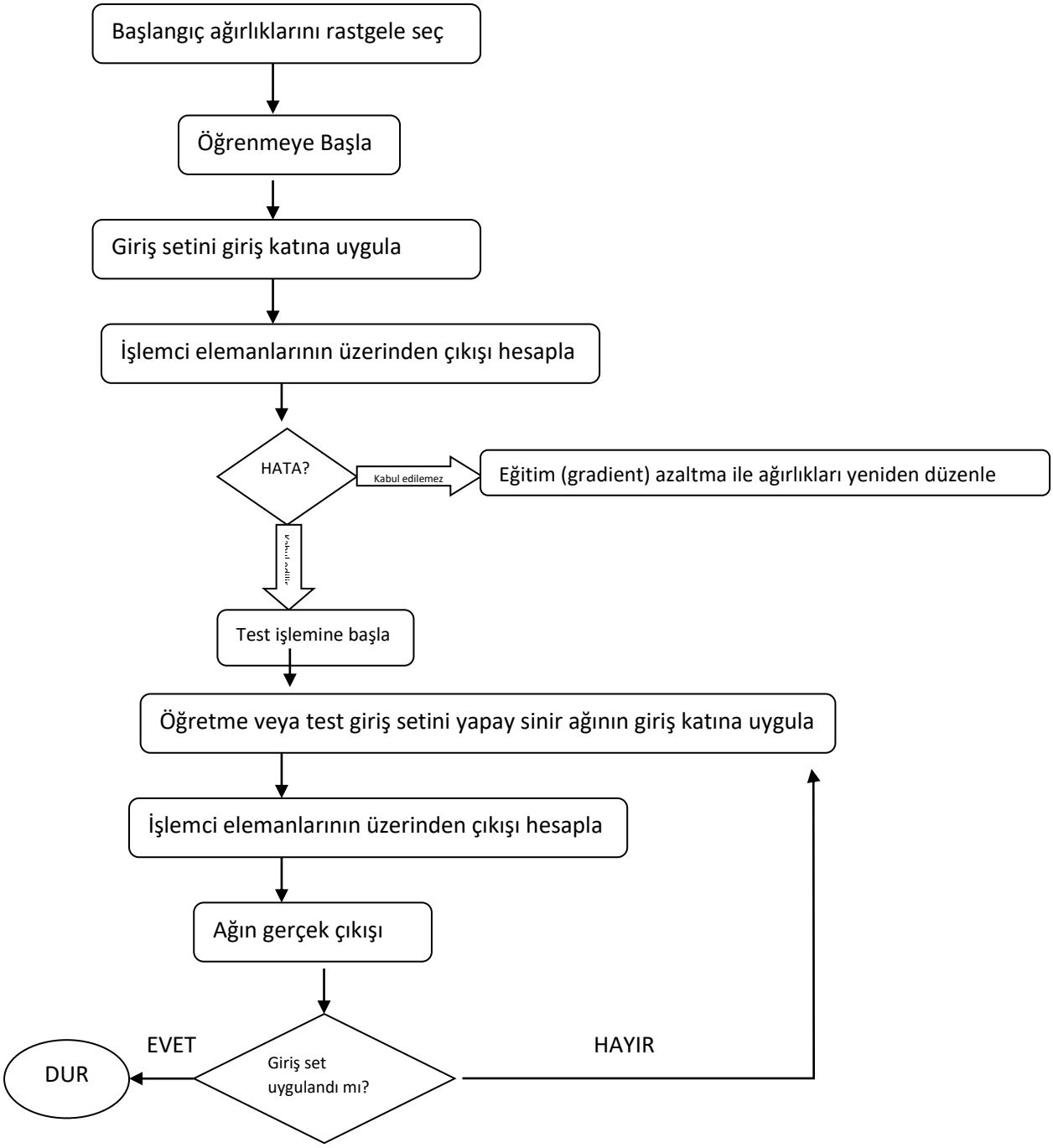
- **alpha*float, default=0.0001***

L2 ceza (düzenleme süresi) parametresi.

- **activation*{'identity', 'logistic', 'tanh', 'relu'}, default='relu'***

- 'identity', op-olmayan etkinleştirme, doğrusal darboğaz uygulamak için yararlı, f (x) = x döndürür
- 'logistic', lojistik sigmoid işlevi, f (x) = 1 / (1 + exp (-x)) değerini döndürür.
- 'tanh',hiperbolik tan fonksiyonu olan 'tanh' f (x) = tanh (x) değerini döndürür.
- 'relu',rektifiye edilmiş doğrusal birim fonksiyonu olan 'relu', f (x) = max (0, x) döndürür

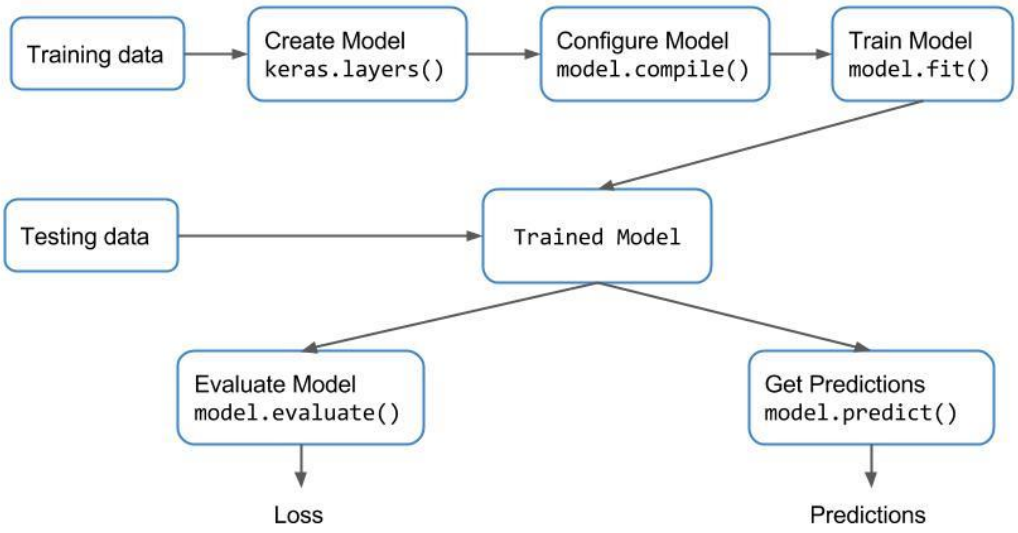
1.1) Çok Katmanlı Ağın Çalışma Şekli



2) KERAS

Keras, Python ile yazılmış ve TensorFlow, Theano veya CNTK'nin üzerinde çalışabilen üst düzey bir API'dir. Keras, Sinir Ağları oluşturmak ve eğitmek için başlık altındaki karmaşık ayrıntıların çoğunu gizleyen basit ve modüler bir API sağlar. Bu, Derin Öğrenme yolculuğunuza başlamanızı kolaylaştırır.

Keras, modelleri eğitmek ve değerlendirmek için çok basit bir iş akışı sağlar. Aşağıdaki şema ile açıklanmıştır:



Temel olarak, modeli oluşturuyoruz ve eğitim verilerini kullanarak eğitiyoruz. Model eğitildikten sonra, test verileriyle ilgili çıkarım yapmak için modeli alırız.

2.1) Keras Katmanları

Katmanlar, Sinir Ağının yapı taşları olarak düşünülebilir. Girdi verilerini işler ve daha sonra bunlara bağlı olan katmanlar tarafından kullanılan katman türüne bağlı olarak farklı çıktılar üretir. Gelecekteki yayınlarda her katmanın ayrıntılarını ele alacağız.
Keras, içinde bulunan birkaç çekirdek katmanı sağlar.

Girdideki her düğüm çıktıdaki her düğüme bağlı olduğundan, tam olarak bağlanmış katman olarak da adlandırılan yoğun katmanlar,
ReLU, tanh, sigmoid gibi aktivasyon fonksiyonlarını içeren aktivasyon katmanını,
Bırakma katmanı – antreman sırasında düzenlileştirme için kullanılır,
Düzleştir, Yeniden Şekillendir, vb.
Bu çekirdek katmanların yanı sıra, bazı önemli katmanlar

Evrişim katmanları – evrişim gerçekleştirmek için kullanılır,
Pooling katmanları – aşağı örnekleme için kullanılır,

Tekrarlayan katmanlar
Yerel olarak bağlı, normalleştirme vb.

2.2) Keras Modelleri

Keras, bir model tanımlamanın iki yolunu sunar:

Sequential, katmanları yığmak için kullanılır – En sık kullanılan.

Functional API, çoklu çıktılı modeller, paylaşılan katmanlar vs. gibi karmaşık model mimarileri tasarlamak için kullanılır.

```
from keras.models import Sequential
```

Sıralı(Sequential) bir model oluşturmak için, katman listesini listeye yapıcıya argüman olarak iletebilir veya model.add () fonksiyonunu kullanarak katmanları sırayla ekleyebiliriz.

Model tanımında dikkat edilmesi gereken önemli bir nokta, ilk katman için giriş şeklini belirtmemiz gerektiğidir. Bu, ilk Yoğun katmanla birlikte geçen input_shape parametresi kullanılarak yukarıdaki snippet'te yapılır. Diğer katmanların şekilleri derleyici tarafından çıkarılır.

2.3) Eğitim Sürecini Yapılandırma

Model hazır olduğunda, öğrenme sürecini yapılandırmamız gerekir. Bunun anlamı

Ağ ağırlıklarının nasıl güncellendiğini belirleyen bir Optimize Edici belirtin.
Maliyet fonksiyonunun veya kayıp fonksiyonunun türünü belirtin.
Eğitim ve test sırasında değerlendirmek istediğiniz metrikleri belirtin.
Arka uç kullanarak model grafiği oluşturun.
Diğer gelişmiş yapılandırmalar.

Bu, Keras'ta model.compile () fonksiyonu kullanarak yapılır. Kod pasajı kullanımı gösterir.

```
model.compile(optimizer='rmsprop', loss='mse', metrics=['mse', 'mae'])
```

Belirlenmesi gereken zorunlu parametreler, optimize edici ve kayıp fonksiyonudur.

2.3.1) Optimizer

Keras, aralarından seçim yapabileceğiniz çok sayıda optimize edici sağlar

2.3.2) Kayıp fonksiyonları

Denetimli bir öğrenme probleminde gerçek değerler ile öngörülen değerler arasındaki hatayı bulmak zorundayız. Bu hatayı değerlendirmek için kullanılabilecek farklı ölçümler olabilir. Bu metriğe genellikle kayıp fonksiyonu veya maliyet fonksiyonu veya amaç fonksiyonu denir. Hata ile ne yaptığınıza bağlı olarak, birden fazla kayıp fonksiyonu olabilir. Genel olarak kullanırız

İkili sınıflandırma problemi için ikili çapraz entropi,
Çok sınıflı bir sınıflandırma problemi için kategorik-çapraz entropi,
regresyon problemi için ortalama-kare-hata vb.

2.4) Eğitim

Model yapılandırıldıktan sonra eğitim sürecine başlayabiliriz. Bu, Keras'taki model.fit () fonksiyonu kullanılarak yapılabilir. Kullanım aşağıda açıklanmıştır.

```
model.fit(trainFeatures, trainLabels, batch_size=4, epochs = 100)
```

Sadece eğitim verilerini, batch büyüklüğünü ve çağ sayısını belirtmemiz gerekiyor. Keras, verilerin, belirtilen çağ sayısı için yinelemeli olarak optimize ediciye nasıl geçirileceğini otomatik olarak çözer. Bilginin geri kalanı önceki adımda zaten optimize ediciye verildi.

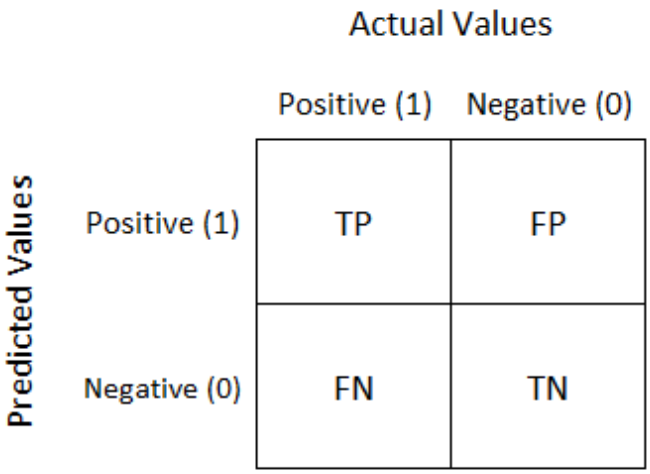
2.5) Modelin Değerlendirilmesi

Model eğitildiğinde, görünmeyen test verilerinin doğruluğunu kontrol etmemiz gerekir. Bu Keras'ta iki şekilde yapılabilir.

model.evaluate () – model.compile () adımında belirtilen kayıp ve metrikleri bulur. Hem test verilerini hem de etiketleri girdi olarak alır ve doğruluğun nicel bir ölçüsünü verir. Çapraz doğrulama yapmak ve en iyi modeli elde etmek için parametreleri daha ince ayar yapmak için de kullanılabilir.
model.predict () – Verilen test verisinin çıktısını bulur. Çıktıların kalitatif olarak kontrol edilmesinde yararlıdır.

3) Confusion matrix

Confusion Matrix, makine öğrenimi sınıflandırması için bir performans ölçümüdür. Çıktının iki veya daha fazla sınıf olabileceği makine öğrenimi sınıflandırma problemi için bir performans ölçümüdür. Öngörülen ve gerçek değerlerin 4 farklı kombinasyonunu içeren bir tablodur.



Recall, Precision, Specificity, Accuracy ve en önemlisi roc-auc eğrilerini çizdirmek için kullanılır.

True Positive(TP):

Olumlu tahmin ettiniz ve bu doğru.

True Negative(TN):

Olumsuz tahmin ettiniz ve bu doğru.

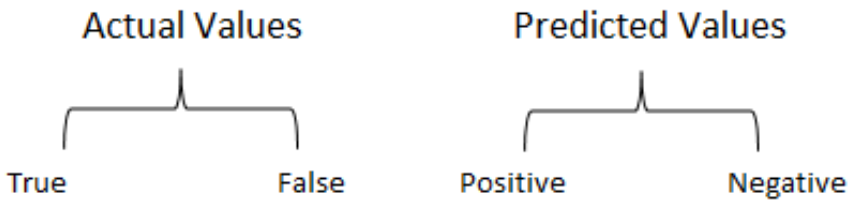
False Positive(FP): (Type 1 Error)

Olumlu tahmin ettiniz ve bu yanlış.

False Negative(FN): (Type 2 Error)

Olumsuz tahmin ettiniz ve bu yanlış.

Öngörülen değerleri Pozitif ve Negatif, gerçek değerleri Doğru ve Yanlış olarak tanımlarız.



3.1) Recall

Tüm pozitif sınıflardan ne kadar doğru tahmin ettiğimizi. Mümkün olduğunca yüksek olmalıdır.

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

3.2) Precision

Doğru tahmin ettiğimiz tüm pozitif sınıflardan, kaç tanesi gerçekten olumlu ve **accuracy** olacak.Tüm sınıflardan ne kadar doğru tahmin ettiğimiz durumda olacak. Mümkün olduğunca yüksek olmalıdır.

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

3.3) f1_score

İki modeli düşük precision ve yüksek recall ile karşılaştırmak zordur veya tersi de geçerlidir. Bu yüzden onları karşılaştırılabilir hale getirmek için F1-Skoru kullanıyoruz. F1-skoru Recall ve Precision aynı anda ölçmeye yardımcı olur. Aşırı değerleri daha fazla cezalandırarak Arithmetic Mean yerine Harmonic Mean kullanır. F1 skoru 1'de en iyi değerine ve en kötü skoru 0'da olur. Hassasiyet ve hatırlamanın F1 skoruna göreceli katkısı eşittir.

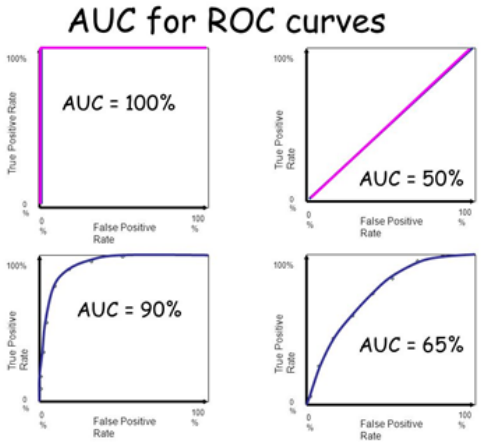
$$F - measure = \frac{2*Recall*Precision}{Recall + Precision}$$

4) ROC Eğrileri (ROC Curve)

Bir sınıflandırma probleminin performansının değerlendirilmesinde AUC — ROC eğrisinden yararlanırız. Özellikle dengesiz veri setlerinin bulunduğu durumlarda, makine öğrenmesi algoritmalarının performansını değerlendirmek için en yaygın kullanılan ölçümlerden biridir. Ve modelin tahmininde ne kadar iyi olduğunu açıklar. ROC farklı sınıflar için bir olasılık eğrisidir. Tipik bir ROC eğrisinde X ekseninde Yanlış Pozitif Oran (FPR) ve Y ekseninde Gerçek Pozitif Orana (TPR) vardır. Eğri altındaki alan (AUC) model becerisinin başka bir deyişle model performansının bir özeti olarak kabul edilebilir.

5)AUC(ROC Eğrisi Altındaki Alan- Alıcı Çalışma Karakteristikleri)

Bu alanın kapsamı AUC'dir. Kapsanan alan ne kadar büyükse, makine öğrenme modelleri o kadar iyi verilen sınıfları ayırt etmede daha iyidir. AUC için ideal değer 1'dir.



Roc →Doğru pozitiflerin yanlış pozitiflere olan kesri. Bu iki metriği X ve Y eksenlerine yerleştirerek çizginin altında kalan alanı hesaplamaya çalışıyoruz. (AUC-Area Under Curve)

AUC → Çizginin altında kalan alan ne kadar büyükse modelin başarı oranı o kadar yüksek demektir.

6) k-Fold Cross-Validation(Çapraz Doğrulama)

```
sklearn.model_selection.KFold
```

Çapraz doğrulama, sınırlı bir veri örneğinde makine öğrenme modellerini değerlendirmek için kullanılan bir yeniden örnekleme prosedürüdür.
Prosedür, belirli bir veri örneğinin bölüneceği grup sayısını ifade eden k adlı tek bir parametreye sahiptir. Bu nedenle, prosedüre genellikle k-kat çapraz geçerlilik denir. K için belirli bir değer seçildiğinde, k = 10'un 10 kat çapraz geçerlilik kazanması gibi modele referansta k yerine kullanılabilir.
Çapraz doğrulama, esas olarak, bir makine öğrenimi modelinin görünmeyen veriler üzerindeki becerisini tahmin etmek için uygulamalı makine öğreniminde kullanılır. Yani, modelin eğitimi sırasında kullanılmayan veriler hakkında tahminlerde bulunmak için kullanıldığında modelin genel olarak nasıl performans göstermesi beklendiğini tahmin etmek için sınırlı bir örnek kullanmak.
Bu popüler bir yöntemdir çünkü anlaşılması kolaydır ve model becerisinin genellikle basit bir tren / test bölümü gibi diğer yöntemlere göre daha az önyargılı veya daha az iyimser bir tahminiyle sonuçlanır.
Genel prosedür aşağıdaki gibidir:

- Veri kümesini rastgele karıştırın.
- Veri kümesini k gruplarına bölme
- Her bir benzersiz grup için:
- Grubu beklemeye alma veya veri kümesini test etme
- Kalan grupları bir eğitim veri seti olarak kabul edin
- Eğitim setine bir model takın ve test setinde değerlendirin
- Değerlendirme puanını koruyun ve modeli atın
- Model değerlendirme puanlarının örneklemini kullanarak model becerisini özetler

Önemli olarak, veri örneğindeki her gözlem ayrı bir gruba atanır ve prosedür süresince bu grupta kalır. Bu, her bir numuneye, bekleme seti 1 kez kullanılma ve model k-1 kez eğitilmesi için fırsat verildiği anlamına gelir.
Bu yaklaşım, gözlem kümesinin yaklaşık olarak eşit büyüklükteki k gruplarına veya kıvrımlara rastgele bölünmesini içerir. İlk kat bir doğrulama seti olarak kabul edilir ve yöntem geri kalan k - 1 katlarına uyar.

Özetlemek gerekirse, k-kat çapraz doğrulamasında k seçimi ile ilişkili bir sapma-varyans değiş tokuşu vardır. Tipik olarak, bu hususlar göz önüne alındığında, k = 5 veya k = 10 kullanılarak k-kat çapraz doğrulaması gerçekleştirilir, çünkü bu değerlerin ampirik olarak ne çok yüksek sapmadan ne de çok yüksek sapmadan muzdarip test hata oranı tahminleri sağladığı gösterilmiştir.

Cross-Validation API

K-kat çapraz doğrulamayı manuel olarak uygulamak zorunda değiliz. Scikit-learn kütüphanesi, belirli bir veri örneğini ayıracak bir uygulama sağlar.
KFold () scikit-öğrenme sınıf kullanılabilir. Tartışma olarak, numunenin karıştırılıp karıştırılmayacağına dair bölünme sayısını ve karışıklıktan önce kullanılan yalancı sayı üreteci tohumunu alır.

Variations on Cross-Validation

- **Train/Test Split**: Bir uç noktaya bakıldığında k, 2 (1 değil) olarak ayarlanabilir; böylece modeli değerlendirmek için tek bir tren / test bölünmesi oluşturulur.
- **LOOCV**: Başka bir uç noktaya bakıldığında k veri kümesindeki toplam gözlem sayısına ayarlanabilir, böylece her bir gözlem veri kümesinin dışında tutulması için bir şans verilir. Buna bir kerelik izinsiz çapraz doğrulama veya kısaca LOOCV denir.
- **Stratified**: Verilerin kıvrımlara bölünmesi, her bir kıvrımın, sınıf çıktı değeri gibi belirli bir kategorik değere sahip gözlemlerin aynı oranına sahip olmasını sağlamak gibi kriterler ile yönetilebilir. Buna tabakalı çapraz doğrulama denir.
- **Repeated**: Bu, k-kat çapraz doğrulama prosedürünün n kez tekrarlandığı yerdir; burada, önemli bir şekilde, her tekrarlamadan önce veri örneği karıştırılır ve bu da numunenin farklı bir bölünmesine neden olur.

7) Grid Search Cross Validation

`sklearn.model_selection.GridSearchCV`

Bir tahminci için belirtilen parametre değerleri üzerinde kapsamlı arama.

Önemli üyeler formda, tahmin.

GridSearchCV bir “fit” ve “score” yöntemi uygular. Ayrıca, kullanılan tahmin ediciye uygulandıklarında “predict”, “predict_proba”, “decision_function”, “transform” ve “inverse_transform” de uygular.

Bu yöntemleri uygulamak için kullanılan tahmin edicinin parametreleri, bir parametre ızgarası üzerinde çapraz doğrulanmış ızgara aramasıyla optimize edilir.

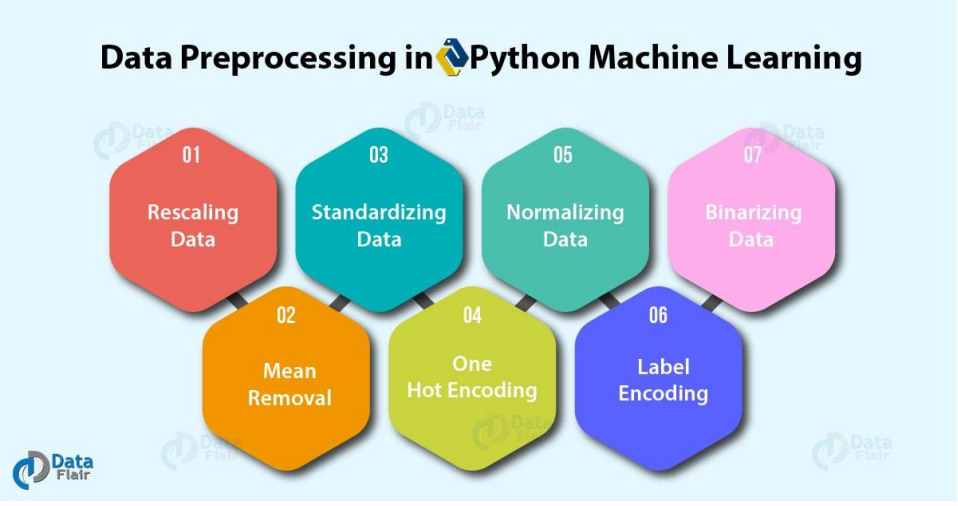
- Sadece iyi başarım veren sonuca odaklanılmaması gerekmektedir. Konfigürasyon değerlerine karar vermek için grid search'deki trendler gözlenmelidir. Bu nedenle grid search sonucundaki tüm sonuçlar gözden geçirilmelidir. Yani parametreler arasındaki ilişkiler, eğilimler gözlenerek sabit parametreler ya da aralıklar gözlenmelidir.
- Grid search'de farklı parametre kombinasyonlarının denenebilmesi için paralel işlemler koşturulabilir. Böylece zamandan tasarruf sağlanmış olur.
- Derin ağları eğitmek uzun zaman alan işlem olduğu için, hiper parametre seçimi için veri kümesinin bir alt kümesi üzerinde çalışılabilir böylece zamandan tasarruf edilmiş olur. Bu işlemde amaç hiper parametreler için en uygun değerleri bulmak değil, genel bir kanı oluşturmaktır. Böylece hiper parametrelerin hangi aralıklarına odaklanacağı belirlenir.
- Hiper parametreler için başlangıçta değerleri büyük tutmak, daha sonra uygun bulunan değer aralıklarına yoğunlaşmak zaman kazancı sağlayacaktır. Örneğin ağdaki nöron sayısı için başlangıçta [10, 100, 500, 1000] gibi bir aralık seçip daha sonra sonuçlara göre kapsamı genişleterek [100, 150, 200, 250, 300, 350 , 400] gibi derinlemesine ilerlemek zaman tasarrufu sağlayacaktır.

`sklearn.metrics`.classification_report

sklearn.metrics.**classification_report**(*y_true, y_pred, *, labels=None, target_names=None, sample_weight=None, digits=2, output_dict=False, zero_division='warn'*)
Bildirilen ortalamalar macro average (averaging the unweighted mean per label), weighted average (averaging the support-weighted mean per label), ve sample average (only for multilabel classification)içerir. Mikro ortalama (toplam gerçek pozitiflerin, yanlış negatiflerin ve yanlış pozitiflerin ortalaması), yalnızca sınıfların bir alt kümesine sahip çoklu etiket veya çoklu sınıf için gösterilir, aksi takdirde doğrulukla karşılık gelir.

Makine Öğrenimi algoritmaları

Veri önişleme ile ham verileri temiz bir veri kümesine dönüştürüyoruz.



Data Set 1-QSAR biodegradation

Data set 1 bilgileri:

1055 kimyasal maddeyi 2 sınıfa (hazır ve hazır biyolojik olarak parçalanamaz) sınıflandırmak için kullanılan 41 özellik (moleküler tanımlayıcı) için değerler içeren veri settir.

QSAR biyodegradasyon veri seti Milano Chemometrics ve QSAR Araştırma Grubunda (UniversitÃ Stli Milano - Bicocca, Milano, İtalya) oluşturulmuştur. Bu sonuçlara yol açan araştırma, Hibe Anlaşması n uyarınca Avrupa Topluluğunun Yedinci Çerçeve Programı'ndan [FP7 / 2007-2013] fon almıştır.

Marie Curie ITN Çevre Kemoinformatik (ECO) projesinin 238701. Veriler, kimyasal yapı ve moleküllerin biyodegradasyonu arasındaki ilişkilerin araştırılması için QSAR (Kantitatif Yapı Aktivite İlişkileri) modellerini geliştirmek için kullanılmıştır.

1055 kimyasalın biyolojik bozunma deneysel değerleri, Ulusal Teknoloji ve Japonya Değerlendirme Enstitüsü'nün (NITE) web sayfasından toplanmıştır.

41 moleküler tanımlayıcı ve 1 deneysel sınıf:

1) SpMax_L: Laplace matrisinden önde gelen özdeğer..
2) J_Dz (e): Sanderson elektronegatifliği ile ağırlıklandırılmış Barysz matrisinden Balaban benzeri indeks.
3) nHM: Ağır atom sayısı.
4) F01 [N-N]: Topolojik uzaklık 1'de N-N frekansı.
5) F04 [C-N]: Topolojik mesafe 4'te C-N frekansı.
6) NssssC: ssssC tipi atom sayısı.
7) nCb-: İkame edilmiş benzen C (sp2) sayısı.
8) C%: C atomlarının yüzdesi.
9) nCp: Terminal birincil C (sp3) sayısı.
10) nO: Oksijen atomlarının sayısı.
11) F03 [C-N]: Topolojik mesafe 3'te C-N frekansı.
12) SdssC: dssC E-durumlarının toplamı.
13) HyWi_B (m): Kütle ağırlıklı Burden matrisinden Hyper-Wiener benzeri indeks (log fonksiyonu).
14) LOC: Merkezleme endeksi.
15) SM6_L: Laplace matrisinden 6. derecenin spektral momenti.
16) F03 [C-O]: Topolojik mesafede C - O frekansı 3.
17) Ben: Ortalama atom Sanderson elektronegatifliği (Karbon atomuna göre ölçeklendirilmiş).
18) Mi: Ortalama ilk iyonlaşma potansiyeli (Karbon atomuna göre ölçeklendirilmiş).
19) nN-N: N hidrazin sayısı.
20) nArNO2: Nitro gruplarının sayısı (aromatik).
21) nCRX3: CRX3 sayısı.
22) SpPosA_B (p): Polarizasyonla ağırlıklandırılmış Yük matrisinden normalize edilmiş spektral pozitif toplam.
23) nCIR: Devre sayısı.
24) B01 [C-Br]: Topolojik mesafede C - Br varlığı / yokluğu 1 25) B03 [C-Cl]: Topolojik mesafede C - Cl varlığı / yokluğu 3 26) N-073: Ar2NH / Ar3N / Ar2N-Al / R..N..R.
27) SpMax_A: Bitişiklik matrisinden önde gelen özdeğer (Lovasz-Pelikan indeksi).

28) Psi_i_1d: Gerçek durum sahte bağlantı endeksi - tip 1d.
29) B04 [C-Br]: Topolojik mesafede C - Br varlığı / yokluğu 4 30) SdO: dO E-durumlarının toplamı.
31) TI2_L: Laplace matrisinden ikinci Mohar endeksi.
32) nCrt: Halka tersiyer C sayısı (sp3).
33) C-026: R - CX – R.
34) F02 [C-N]: Topolojik mesafe 2'de C - N frekansı.
35) nHDon: H-bağları (N ve O) için verici atom sayısı.
36) SpMax_B (m): Kütle ile ağırlıklandırılmış Yük matrisinden önde gelen özdeğer.
37) Psi_i_A: Gerçek durum sahte bağlantı endeksi - S tipi ortalama 38) nN: Azot atomlarının sayısı.
39) SM6_B (m): Kütle ile ağırlıklandırılmış Yük matrisinden 6. derecenin spektral momenti.
40) nArCOOR: Ester sayısı (aromatik).
41) nX: Halojen atom sayısı.
42) Class, deneysel sınıf: hazır biyolojik olarak parçalanabilir (RB-1) ve hazır biyolojik olarak parçalanamaz (NRB-2).

print(df.info())

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1055 entries, 0 to 1054
Data columns (total 42 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   V1      1055 non-null   float64
 1   V2      1055 non-null   float64
 2   V3      1055 non-null   int64
 3   V4      1055 non-null   int64
 4   V5      1055 non-null   int64
 5   V6      1055 non-null   int64
 6   V7      1055 non-null   int64
 7   V8      1055 non-null   float64
 8   V9      1055 non-null   int64
 9   V10     1055 non-null   int64
 10  V11     1055 non-null   int64
 11  V12     1055 non-null   float64
 12  V13     1055 non-null   float64
 13  V14     1055 non-null   float64
 14  V15     1055 non-null   float64
 15  V16     1055 non-null   int64
 16  V17     1055 non-null   float64
 17  V18     1055 non-null   float64
 18  V19     1055 non-null   int64
 19  V20     1055 non-null   int64
 20  V21     1055 non-null   int64
 21  V22     1055 non-null   float64
 22  V23     1055 non-null   int64
 23  V24     1055 non-null   int64
 24  V25     1055 non-null   int64
 25  V26     1055 non-null   int64
```

```
 26  V27     1055 non-null   float64
 27  V28     1055 non-null   float64
 28  V29     1055 non-null   int64
 29  V30     1055 non-null   float64
 30  V31     1055 non-null   float64
 31  V32     1055 non-null   int64
 32  V33     1055 non-null   int64
 33  V34     1055 non-null   int64
 34  V35     1055 non-null   int64
 35  V36     1055 non-null   float64
 36  V37     1055 non-null   float64
 37  V38     1055 non-null   int64
 38  V39     1055 non-null   float64
 39  V40     1055 non-null   int64
 40  V41     1055 non-null   int64
 41  Class   1055 non-null   int64
dtypes: float64(17), int64(25)
memory usage: 346.3 KB
None
```

Veri Seti Hazırlık Çalışmaları 1

Veri setinde Class sütunundaki 1 ve 2 değerleri saydırıldı .1 değerleri 0 a 2 ve varsa başka değerler 1 e dönüştürüldü. Standardizing veri setine uygulandı..En iyi sonucu veren veri ölçekleme yöntemi ;

- ```
  from sklearn.preprocessing import StandardScaler
  Scaler=StandardScaler()
  x=Scaler.fit_transform(x_data)
  ```

StandardScaler yönteminde amaç verilerinizi, dağılımın ortalama değeri 0 ve standart sapma değeri 1 olacak şekilde değiştirecek olmasıdır. Verilerin dağılımı göz önüne alındığında, veri kümesindeki her değer, çıkarılan örnek ortalama değerine sahip olacaktır. Tüm veri kümesinin standart sapması ile bölünmüştür.

Hesaplama yöntemi;

Standardization:

$$z = \frac{x - \mu}{\sigma}$$

with mean:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

and standard deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Min-Max scaling:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

bu şekildedir.
  ```

Data Set 2- thyroid-disease

Data Set 2 bilgileri

hypothyroid, primary hypothyroid, compensated hypothyroid,

secondary hypothyroid,

1. Yaş: Sürekli.
2. Cinsiyet: M, F.
3. Tiroksin Üzerinde: F, T.
4. Tiroksinde Sorgulama: F, T.
5. İlaç Tedavisi Hakkında: F, T.
6. Hasta: F, T.
7. Hamile: F, T.
8. Tiroid Cerrahisi: F, T.
9. I131 Tedavisi: F, T.
10. Sorgu Hipotiroidi: F, T.
11. Sorgu Hipertiroidi: F, T.
12. Lityum: F, T.
13. Guatr: F, T.
14. Tümör: F, T.
15. Hipofitüiter: F, T.
16. Psişik: F, T.
17. Ölçülen TSH: F, T.
18. TSH: Sürekli.
19. Ölçülen T3: F, T.
20. T3: Sürekli.
21. Ölçülen TT4: F, T.
22. TT4: Sürekli.
23. Ölçülen T4U: F, T.
24. T4U: Sürekli.
25. Ölçülen FTI: F, T.
26. FTI: Sürekli.
27. Ölçülen TBG: F, T.
28. TBG: Sürekli.
29. Sevk Kaynağı: WEST, STMW, SVHC, SVI, SVHD, Diğer.

```
print(df.info())
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2800 entries, 0 to 2799
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     2800 non-null   object
```

```
 1   sex                        2800 non-null   object
 2   on_thyroxine               2800 non-null   object
 3   query_on_thyroxine         2800 non-null   object
 4   on_antithyroid_medication  2800 non-null   object
 5   sick                       2800 non-null   object
 6   pregnant                   2800 non-null   object
 7   thyroid_surgery            2800 non-null   object
 8   I131_treatment             2800 non-null   object
 9   query_hypothyroid          2800 non-null   object
 10  query_hyperthyroid         2800 non-null   object
 11  lithium                    2800 non-null   object
 12  goitre                     2800 non-null   object
 13  tumor                      2800 non-null   object
 14  hypopituitary              2800 non-null   object
 15  psych                      2800 non-null   object
 16  TSH_measured               2800 non-null   object
 17  TSH                        2800 non-null   object
 18  T3_measured                2800 non-null   object
 19  T3                         2800 non-null   object
 20  TT4_measured               2800 non-null   object
 21  TT4                        2800 non-null   object
 22  T4U_measured               2800 non-null   object
 23  T4U                        2800 non-null   object
 24  FTI_measured               2800 non-null   object
 25  FTI                        2800 non-null   object
 26  TBG_measured               2800 non-null   object
 27  TBG                        2800 non-null   object
 28  referral_source            2800 non-null   object
 29  Class                      2800 non-null   object
dtypes: object(30)
memory usage: 656.4+ KB
None
```

Veri Seti Hazırlık Çalışmaları 2

1)

- df.head()

DataFrame.head(*self: ~FrameOrSeries, n: int = 5*)

İlk *n* satırı döndür.Bu işlev , konuma göre nesne için ilk *n* satırı döndürür . Nesnenizde doğru türde veri olup olmadığını hızlı bir şekilde test etmek için kullanışlıdır.

Parameters :n:*int, default 5(*Seçilecek satır sayısı.Varsayılan değer 5)

Returns :same type as caller**(**Arayan nesnesinin ilk *n* satırı.)

2)Class sütunundaki noktalama işaretlerini kaldırmak için

- string.replace(old, new, count)

- df['Class'] = df['Class'].str.replace('[^\w\s]','')

3)Class sütunundaki sayıları kaldırmak için

- df['Class'] = df['Class'].str.replace('\d','')

4)Class sütununda bulunan tekrar eden kelimelerin sayıları hesaplamak için

- df["Class"].value_counts()

5)TBG sütununun hepsi nan-value bunu drop etmek için

- df["TBG"].value_counts()

6)Sütun adlarını öğrenmek için

- df.columns

7)DataFrame boyutunu öğrenmek için

- df.shape

8) Her sütundaki eksik değerleri saymak için

- df.isnull().sum()

9) Bazı veriler normalde örn. yaş, TSH, T3, TT4, T4U, FTI, TBG için sayısal tip olabilir

ama tüm veriler nesne türüne benziyor

Bu sorun '?' karakterler

Bu karaktere dönüştür np.nan

- df.dtypes #Data türü
- mymap = {"?":np.NaN}

- df=df.applymap(lambda s: mymap.get(s) if s in mymap else s)

10)Eksik verileri gözlemlemek için

- df.isnull().sum() # missing datas can be observed

11)Cinsiyetleri F ve M ,1 ve 0 yapmak için

Diğer yerlerde görülen t ve f değerlerini 1,0 yapmak için

- df[['sex']]=df[['sex']].replace(to_replace={'F':1,'M':0})

- df[['on_thyroxine','query_on_thyroxine','on_antithyroid_medication','lithium','goitre','tumor','hypopituitary','psych']]=df[['on_thyroxine','query_on_thyroxine','on_antithyroid_medication','lithium','goitre','tumor','hypopituitary','psych']].replace(to_replace={'t':1,'f':0})

- df[['TSH_measured','T3_measured','TT4_measured','T4U_measured','FTI_measured','TBG_measured','referral_source']]=df[['TSH_measured','T3_measured','TT4_measured','T4U_measured','FTI_measured','TBG_measured','referral_source']].replace(to_replace={'t':1,'f':0})

- df[['sick','pregnant','thyroid_surgery','I131_treatment','query_hypothyroid','query_hyperthyroid']]=df[['sick','pregnant','thyroid_surgery','I131_treatment','query_hypothyroid','query_hyperthyroid']].replace(to_replace={'t':1,'f':0})

12)Tekrar kontrol etmek için istenmeyen verileri

- df.head()
- df.dtypes
- df.isnull().sum()

13)Object olan olan veri türlerini floata dönüştürmek için

- df["age"] = df["age"].astype(float)
- df["TSH"] = df["TSH"].astype(float)
- df["T3"] = df["T3"].astype(float)
- df["TT4"] = df["TT4"].astype(float)
- df["T4U"] = df["T4U"].astype(float)
- df["FTI"] = df["FTI"].astype(float)
- df.dtypes

14)Boşlukları ortalama ile doldurmak için

- for i in ['age','sex','TSH','T3','TT4','T4U','FTI']:
- df[i].fillna(df[i].mean(),inplace=True)
- df.isnull().sum()

15)Class sütununu drop edip y adında dizi oluşturuldu

```
df['Class'].value_counts() # 4 output
y= df['Class']
```

16)referal_source,Class,TBG sütunları drop edildikten sonra x_data adında yeni DaraFrame oluşturmak için

```
x_data=df.drop(['referral_source','Class','TBG'],axis=1)
```

17) Standardizing veri setine uygulandı.

```
from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()
x=Scaler.fit_transform(x_data)
```

adımları uygulanmıştır.

Çalışma 1 MLP-QSAR biodegradation

1-)Gerekli kütüphaneler import edildi. Veri ön işleme yapıldı.

2) Standardizing veri setine uygulandı.

```
from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()
```

3) Veri kümesii test ve train şekinde bölündü

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train,
y_test=train_test_split(x,y,test_size=0.4,random_state=42)
```

4)mlpc_model oluşturuldu.

5)Model Tuning-GridSearchCv uygulandı parametler genel kullanıma göre seçildi.

Fitting 5 folds for each of 16 candidates, totalling 80 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.

[Parallel(n_jobs=-1)]: Done 37 tasks | elapsed: 1.1min

[Parallel(n_jobs=-1)]: Done 80 out of 80 | elapsed: 1.8min finished

6)Validasyon Yöntemi-KFold uygulandı. Skor hesaplandı. (n_splits=10)

1. 0.7452830188679245
2. 0.7924528301886793
3. 0.8113207547169812
4. 0.839622641509434

5. 0.9339622641509434
6. 0.8666666666666667
7. 0.9142857142857143
8. 0.7523809523809524
9. 0.8095238095238095
10. 0.8857142857142857

7)En iyi parametler hesaplandı.

{'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (100, 100, 100), 'solver': 'adam'}

8)mlpc_tuned skoru hesaplandı.

score: 0.8933649289099526

9) confusion matrix, hesaplandı

cm: [[255 25]

 [25 117]]

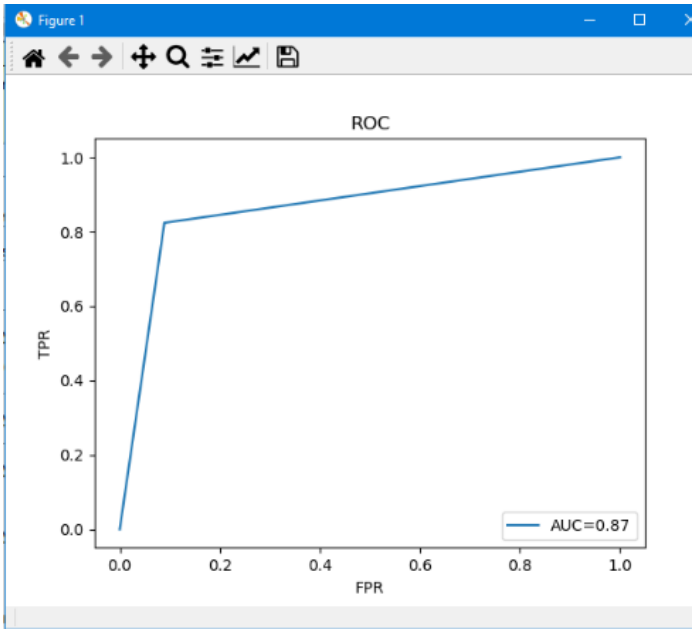
10)f1 score hesaplandı

f1: 0.823943661971831

11)print(metrics.classification_report(y_test,y_pred))

	precision	recall	f1-score	support
0	0.91	0.91	0.91	280
1	0.82	0.82	0.82	142
accuracy			0.88	422
macro avg	0.87	0.87	0.87	422
weighted avg	0.88	0.88	0.88	422

12) roc ve auc çizdirildi.



Çalışma 2 Keras-QSAR biodegradation

1-)Gerekli kütüphaneler import edildi. Veri ön işleme yapıldı.

2)Veri ölçeklendirme(Normalizasyon) yapıldı. Diğer yöntemlerden daha iyi sonuç verdi max-min.

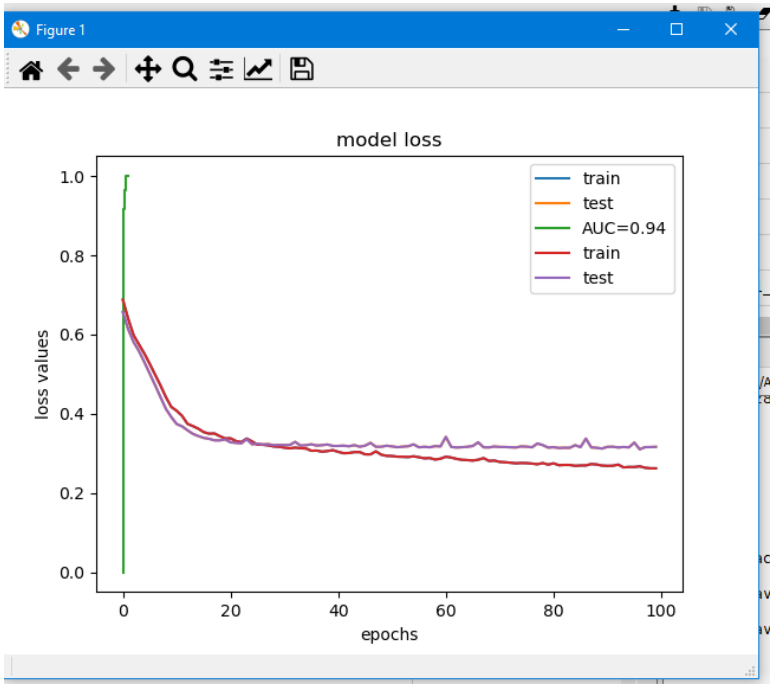
- x=(x_data-np.min(x_data))/(np.max(x_data)-np.min(x_data))

3) Veri kümesii test ve train şekinde bölündü

- from sklearn.model_selection import train_test_split
- x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.4,random_state=42)

4)Keras Ağı oluşturuldu.

5) Plot loss during training



6)accuracy skor hesaplandı

Accuracy:0.8696682464454977

7)f1-skor hesaplandı

f1: 0.8196721311475411

8)GridSearch uygulandı

9)keras best parametre hesaplandı

{‘epochs’:100,’optimizer’:’RMSprop’}

10)keras modeli oluşturuldu best parametler girildi

11)KFold hesaplandı,accuracy skor hesaplandı

Accuracies [0.89411765 0.87058824 0.89285714 0.88095238 0.85714286]

12)Genel skorlar hesaplandı

- acc: 0.8696682464454977
- cm:

  [[246 34]

  [21 121]]

- f1: 0.8148148148148148
- precision recall f1-score support

  0 0.92 0.88 0.90 280

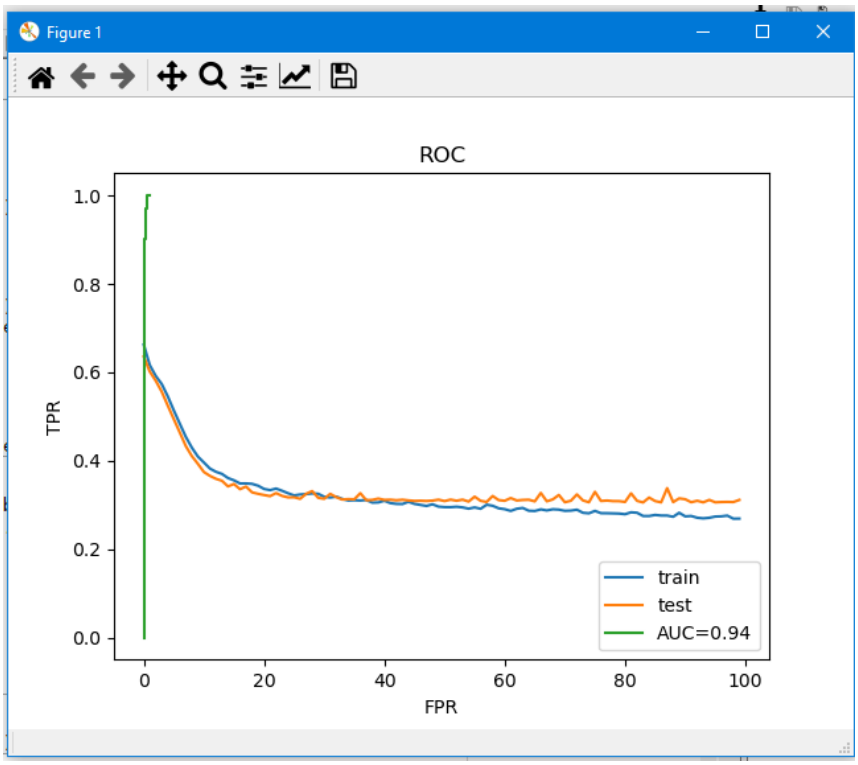
  1 0.78 0.85 0.81 142

  accuracy 0.87 422

  macro avg 0.85 0.87 0.86 422

  weighted avg 0.87 0.87 0.87 422

13) roc-auc grafiği çizdirildi



Çalışma 3 MLP- thyroid-disease

1-)Gerekli kütüphaneler import edildi. Veri ön işleme yapıldı.

2) Standardizing veri setine uygulandı.

```
from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()
```

3) Makine öğrenimi algoritmaları kategorik verilerle doğrudan çalışamaz.
Kategorik veriler sayılara dönüştürülmelidir.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(y)
print(integer_encoded)
# binary encode
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
y = onehot_encoder.fit_transform(integer_encoded)

y[0:3]
```

Bu, bir dizi sınıflandırma tipi problemi ile çalışırken ve Uzun Kısa Süreli Bellek tekrarlayan sinir ağları gibi derin öğrenme yöntemlerini kullanmayı planladığınızda geçerlidir. One hot encoding kategorik değişkenlerin ikili vektörler olarak temsilidir. Bu öncelikle kategorik değerlerin tamsayı değerlerle eşlenmesini gerektirir.
Daha sonra, her bir tamsayı değeri, 1 ile işaretlenmiş tamsayı dizini hariç tüm sıfır değerler olan bir ikili vektör olarak temsil edilir.
'0' '1' '2' ve '3'değerlerine sahip bir dizi etiketimiz olduğunu varsayın.
Daha sonra, her bir tam sayı değerini temsil etmek için bir 4 lü vektör oluşturabiliriz. Vektör, olası 4 tamsayı değeri için 4 uzunluğa sahip olacaktır.

4)mlpc model oluşturuldu test-train ile birlikte

5)accuracy skor hesaplandı

0.9678571429

6)GridSearch uygulandı best parametler hesaplandı

{'activation': 'logistic', 'alpha': 0.1,

'hidden_layer_sizes': (10, 10, 10), 'solver': 'lbfgs'}

7)Best parametreler girildi

{'activation': 'logistic', 'alpha': 0.1,

'hidden_layer_sizes': (10, 10, 10), 'solver': 'lbfgs'}

8)KFold uygulandı skor hesaplandı

- 0.9803571428571428
- 0.975
- 0.9642857142857143
- 0.9875
- 0.9821428571428571

9)Genel skorlar hesaplandı

acc:0.975

f1:0.9761904761904762

cm: [[806 1 1 3]

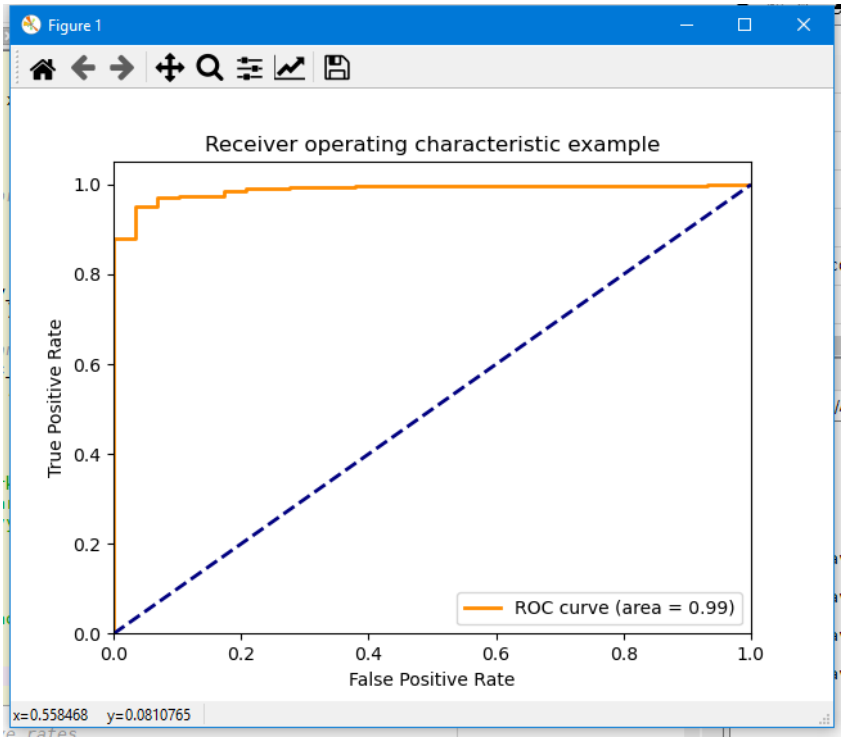
[1 4 3 0]

[3 3 4 0]

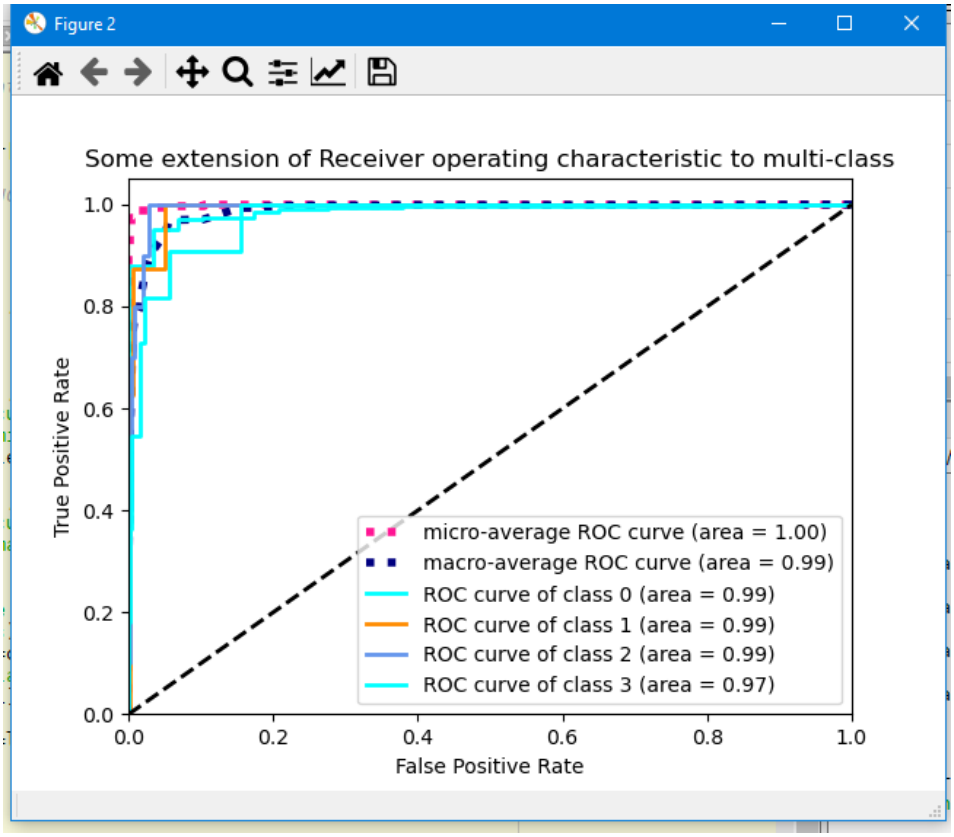
[6 0 0 5]]

	precision	recall	f1-score	support
0	0.99	0.99	0.99	811
1	0.56	0.62	0.59	8
2	0.50	0.40	0.44	10
3	0.62	0.45	0.53	11
micro avg	0.98	0.98	0.98	840
macro avg	0.67	0.62	0.64	840
weighted avg	0.97	0.98	0.97	840
samples avg	0.98	0.98	0.98	840

10)roc-auc grafiği çizdirildi(0 sınıfı buradaki sayıyı 1-2-3 olarak değiştirildiğinde diğer sınıflar içinde grafik elde edilebilir)



11)roc-auc grafiği tüm sınıflar için çizdirildi.



Çalışma 4 Keras- thyroid-disease

1-)Gerekli kütüphaneler import edildi. Veri ön işleme yapıldı.

2)Standardizing veri setine uygulandı.

```
from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()
```

3) Makine öğrenimi algoritmaları kategorik verilerle doğrudan çalışamaz.
Kategorik veriler sayılara dönüştürülmelidir.

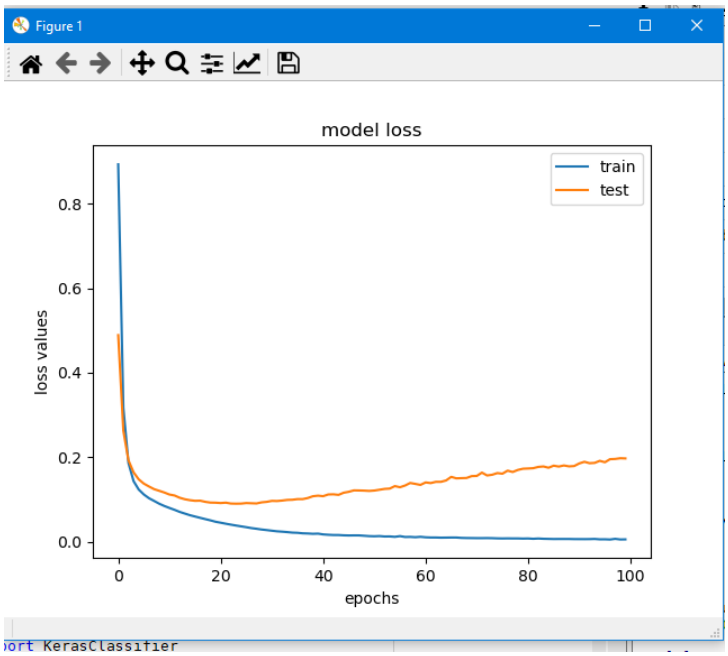
```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(y)
print(integer_encoded)
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
y = onehot_encoder.fit_transform(integer_encoded)
y[0:3]
```

4)) Veri kümesii test ve train şekinde bölündü.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train,
y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

5)Keras ağı oluşturuldu.

6)Plot loss during training.



7)Oluşturulan modelin skorları hesaplandı

Accuracy :0.9773809523809524

8)GridSearchCV uygulandı

9)Best parametreler belirlendi

{‘epochs’:200,’optimizer’:RMSprop’}

10)KFold uygulandı skorlar hesaplandı

Accuracies [0.96428571 0.94642857 0.93452381 0.98809524 0.96428571]

11)Genel skorlar hesaplanıldı

- accuracy: 0.9726190476190476
- cm: [[807 2 1 1]

  [1 4 3 0]

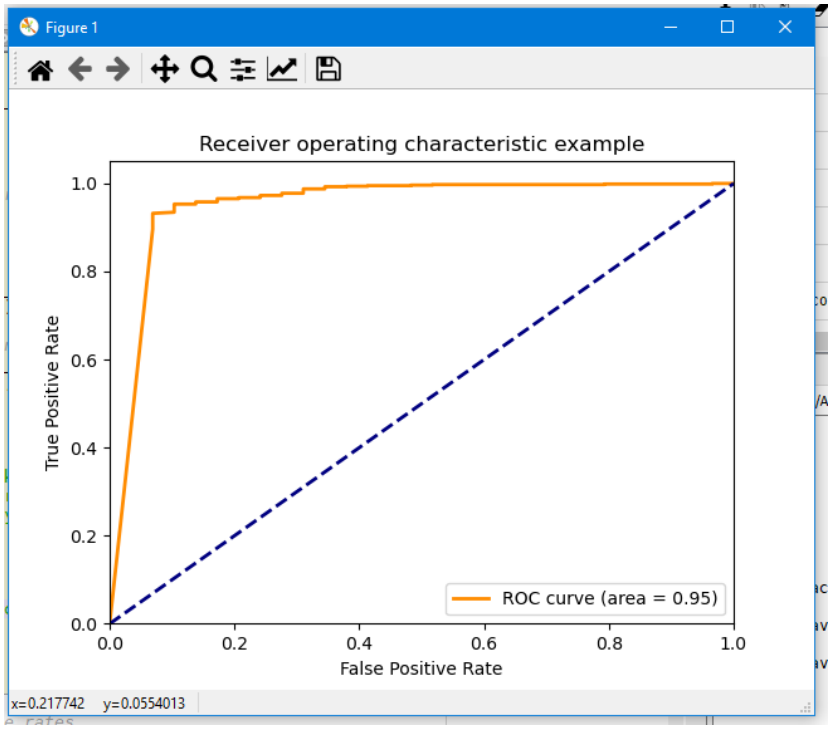
  [3 4 3 0]

  [8 0 0 3]
- f1: 0.9726190476190477

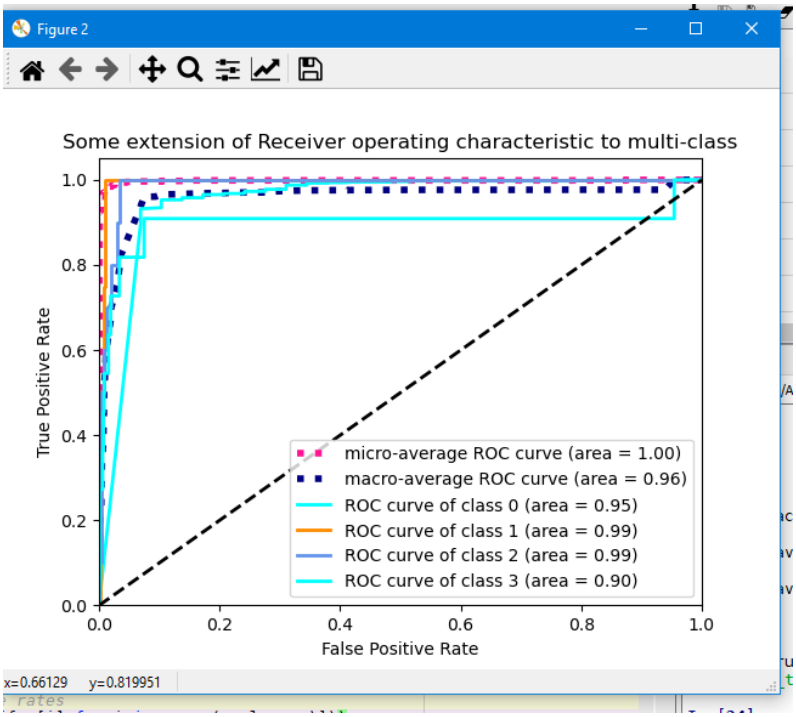
- precision recall f1-score support

0	0.99	1.00	0.99	811
1	0.40	0.50	0.44	8
2	0.43	0.30	0.35	10
3	0.75	0.27	0.40	11
accuracy			0.97	840
macro avg	0.64	0.52	0.55	840
weighted avg	0.97	0.97	0.97	84

12)roc-auc grafiği çizdirildi(0 sınıfı buradaki sayıyı 1-2-3 olarak değiştirildiğinde diğer sınıflar içinde grafik elde edilebilir)



13)roc-auc grafiği tüm sınıflar için çizdirildi.



KAYNAKÇA

1. https://gitlab.com/hakanyilmaz/bmm723
2. https://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation
3. https://archive.ics.uci.edu/ml/machine-learning-databases/thyroid-disease/
4. https://medium.com/@isikhanelif/multi-layer-perceptron-mlp-nedir-4758285a7f15
5. http://ilhanumut.trakya.edu.tr/mlp.pdf
6. http://www.nuhazginoglu.com/2018/05/15/cok-katmanli-algilayicilar-multi-layer-perceptron/
7. https://devhunteryz.wordpress.com/2018/07/05/yapay-sinir-agi-egitimi-cok-katmanli-perceptronmulti-layer-perceptron/
8. https://medium.com/turkce/keras-ile-derin-%C3%B6%C4%9Frenmeye-giri%C5%9F-40e13c249ea8
9. http://www.veridefteri.com/2019/02/06/keras-ile-derin-ogrenmeye-giris/
10. https://www.tensorflow.org/guide/keras/overview
11. https://stackoverflow.com/
12. https://medium.com/@gulcanogundur/model-se%C3%A7imi-k-fold-cross-validation-4635b61f143c
13. https://scikit-learn.org/stable/modules/cross_validation.html
14. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
15. https://campus.datacamp.com/courses/winning-a-kaggle-competition-in-python/dive-into-the-competition?ex=9
16. https://github.com/encodedANAND/K-Fold-Cross-Validation
17. https://github.com/kentmacdonald2/k-Folds-Cross-Validation-Example-Python
18. https://www.udemy.com/course/python-egitimi/
19. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
20. https://scikit-learn.org/stable/auto_examples/model_selection/plot_grid_search_digits.html
21. https://stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python/
22. https://www.kaggle.com/questions-and-answers/30560
23. https://machinelearningmastery.com/multi-output-regression-models-with-python/
24. https://benalexkeen.com/feature-scaling-with-scikit-learn/
25. http://danlec.com/st4k#questions/51378105
26. https://abdalimran.github.io/2019-06-01/Drawing-multiple-ROC-Curves-in-a-single-plot
27. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
28. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
29. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
30. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
31. https://www.datatechnotes.com/2020/02/multi-output-regression-example-with.html
32. https://data-flair.training/blogs/python-ml-data-preprocessing/