SKLEARN in PYTHON (Scikit Learn – Machine Learning)

```
# Sklearn'den hazır veri seti yüklemek için(Python'da her kütüphane içerisinde uygulama yapabilmemiz için veri setleri sunulmuştur.) ;
from sklearn.datasets import load_iris  (Hazır iris datasetinin yüklenmesi)
```

```
# Dizi veya matrisleri, rassal olarak train ve test gruplarına ayırmak için;
from sklearn.model_selection import train_test_split
```

```
# Bileşenleri -grupları veya boyutu azaltmak için Temel Bileşen Analizi;
from sklearn.decomposition import PCA

# Değişken sayısı gözlem sayısından çok olduğu ya da gözlem sayısı değişken sayısından çok olduğu durumda Kısmi En Küçük Kareler Regresyon(PLS);
from sklearn.cross_decomposition import PLSRegression
```

REGRESSION with SKLEARN (TAHMİNLEME):

Elimizdeki veri setleri üzerinden tahminler yapmak istersek aşağıdaki uygun istatistiksel modelleri kullanabiliriz. Örneğin; sigara içip içmemesine göre kanser olma tahmini, deneyim ve yaşa göre maaş tahmini gibi…

➢ Simple – Multiple Linear Regression

Doğrusal modeller, kısaca açıklayacak olursak giriş özelliklerinin doğrusal bir fonksiyonunu kullanarak bir tahminde bulunur.

```
from sklearn.linear_model import LinearRegression
```

➢ Polynomial Linear Regression

Basit ve çoklu regresyondan farklı olarak, noktalara en yakın geçecek biçimde paraboller çizerek tahmin etmeye çalışır.

```
from sklearn.preprocessing import PolynomialFeatures
```

➢ K Nearest Neighbors Regression

Yeni bir veri noktası hakkında tahminde bulunmak için, algoritma eğitim veri setindeki en yakın veri noktalarını bulur - "en yakın komşusu".

```
from sklearn.neighbors import KNeighborsRegressor
```

➢ Support Vector Machine Regression

"Destek Vektör Makinesi" (SVM), sınıflandırma veya regresyon problemleri için kullanılabilen denetimli bir makine öğrenmesi algoritmasıdır. Genellikle sınıflandırma problemlerinde kullanılır. Bu algoritmada, her bir veri maddesini belirli bir koordinatın değeri olan her özelliğin değeri ile birlikte n-boyutlu boşluğa (burada n sahip olduğunuz özelliklerin sayısı) bir nokta olarak çizilir. Ardından, iki sınıftan oldukça iyi ayrım yapan hiper-düzlemi bularak sınıflandırma gerçekleştirilir.

```
from sklearn.svm import SVR
```

Tolga Kaplan

➢ Decision Tree Regression

Karar ağacı, çok sayıda kayıt içeren bir veri kümesini, bir dizi karar kuralları uygulayarak daha küçük kümelere bölmek için kullanılan bir yapıdır. Basit karar verme adımları uygulanarak, büyük miktarlardaki kayıtları, çok küçük kayıt gruplarına bölerek kullanılan bir yapıdır.

```
from sklearn.tree import DecisionTreeRegressor
```

➢ Random Forest Regression

Tahminlerde bulunmak için, tüm ağaçların tahminlerini elde eder, daha sonra en çok oyu alan sınıfı tahmin eder. Karar Ağaçlarının böyle bir grubuna Rastgele Orman adı verilir ve sadeliğine rağmen, bugün mevcut en güçlü Makine Öğrenimi algoritmalarından biridir.

```
from sklearn.ensemble import RandomForestRegressor
```

➢ Bagging Regression

Var olan bir eğitim setinden yeni eğitim setleri veya yeni alt kümeler(örneklemler) türeterek yeniden eğitmeyi amaçlayan bir yöntemdir. Yerine koyarak örnekleme yapılır. Eğitim kümesi Bagging'de n adet örnekten oluşan eğitim setinden yine n örnekli bir eğitim seti yerine koyarak **rastgele** seçimle üretilir. Seçilen her örnek tekrar geri eğitim setine geri konulur. Bazı örnekler yeni eğitim kümesinde yer almazken bazıları birden fazla kez yer alırlar.

```
from sklearn.ensemble import BaggingRegressor
```

➢ XGBoost Regression

Kaggle.com yarışmalarının en iyilerinin kullandığı algoritmalardandır. Temel olarak karar ağaçlarının benzeri bir yapısı vardır. Ancak çalışma hızı olarak hepsinden daha hızlı sonuçlar üretmektedir. Bu yöntem kullanılırken hiperparametrelerle oynanabilir ve tahminlerin doğruluk oranları arttırılabilir. Sklearn kütüphanesi dahilinde değildir. Ayrıca kurulmalıdır. Ancak Sklearn ile koordine çalışmaktadır.

```
import xgboost as xgb

xg_reg =xgb.XGBRegressor(learning_rate=0.1,max_depth=10, n_estimators=500)
```

➢ LightGBM Regression

Bu algoritmada gradient boost alt yapısıyla çalışmakla beraber çok iyi ve çok hızlı sonuçlar vermektedir. Hatta XGBoost'tan çok daha hızlı ve yakın sonuçlarla karşılaşılabilmektedir. Bu yöntem kullanılırken hyperparametrelerle oynanabilir ve tahminlerin doğruluk oranları arttırılabilir. Sklearn kütüphanesi dahilinde değildir. Ayrıca kurulmalıdır. Ancak Sklearn ile koordine çalışmaktadır.

```
import lightgbm as lgbm

lgbm_reg=lgbm.LGBMRegressor(learning_rate=0.1,max_depth=10,n_estimators=500,
min_child_weight=1.5)
```

Tolga Kaplan

➢ PLS Regression

Kısmi en küçük kareler regresyonu(PLS Regression), kısmi en küçük kareler analizi (KEKK) ve çoklu doğrusal regresyon analizinden oluşan çok değişkenli istatistiksel bir yöntemdir. Kısmi en küçük kareler yöntemi ile fazla sayıda olan ve aralarında çoklu doğrusal bağlantı bulunan açıklayıcı değişkenler, bağımlı ve açıklayıcı değişkendeki değişimi büyük ölçüde açıklayan daha az sayıda ve aralarında çoklu doğrusal bağlantı sorunu olmayan yeni değişkenlere (bileşen) indirgenmektedir. Elde edilen bileşenlere çoklu doğrusal regresyon analizi uygulanarak regresyon modeli oluşturulmaktadır.

```
from sklearn.cross_decomposition import PLSRegression
```

➢ Neural Network Regression

İnsan beyninin öğrenme şekline benzer şekilde çalışan bir algoritmadır. Girdileri ağırlıklarıyla çarpar, bunları toplar ve bunu bir fonksiyonun içine sokarak sonuçlandırmaya-tahmin etmeye çalışır.

```
from sklearn.neural_network import MLPRegressor
```

➢ Ridge Regression

Lineer Regresyonu düzenlemek için kullanılır. Ridge regresyonunda, katsayılar (w) sadece eğitim verilerini iyi tahmin edebilecek şekilde değil, aynı zamanda ek bir kısıtlamaya uyacak şekilde seçilmiştir. Ayrıca katsayıların büyüklüğünün mümkün olduğunca küçük olmasını istiyoruz; başka bir deyişle, tüm w girişleri sıfıra yakın olmalıdır. Sezgisel olarak, bu, her bir özelliğin hala iyi tahmin ederken, sonuç üzerinde mümkün olduğunca az etkiye sahip olması gerektiği anlamına gelir. Bu kısıtlama, düzenlenme olarak adlandırılan şeyin bir örneğidir. Düzenleme, fazla takılmamak için bir modeli açıkça kısıtlamak anlamına gelir. Ridge regresyon, L2 düzenlenmesi olarak da bilinir.

```
from sklearn.linear_model import Ridge
```

➢ Lasso Regression

Lineer regresyonu düzenlemek için Ridge'e bir alternatif olan bir yöntemdir. Ridge regresyonda olduğu gibi, Lasso kullanmak da katsayıları sıfıra yakın, ancak L1 düzenlileştirme denilen biraz farklı bir şekilde kısıtlar. Lasso kullanırken, L1 düzenlileşmesinin sonucu, bazı katsayıların tamamen sıfır olmasıdır. Bu, bazı özelliklerin model tarafından tamamen göz ardı edildiği anlamına gelir. Bu bir otomatik özellik seçimi şekli olarak görülebilir. Bazı katsayıların tamamen sıfır olması genellikle bir modelin yorumlanmasını kolaylaştırır ve modelinizin en önemli özelliklerini ortaya çıkarabilir.

```
from sklearn.linear_model import Lasso
```

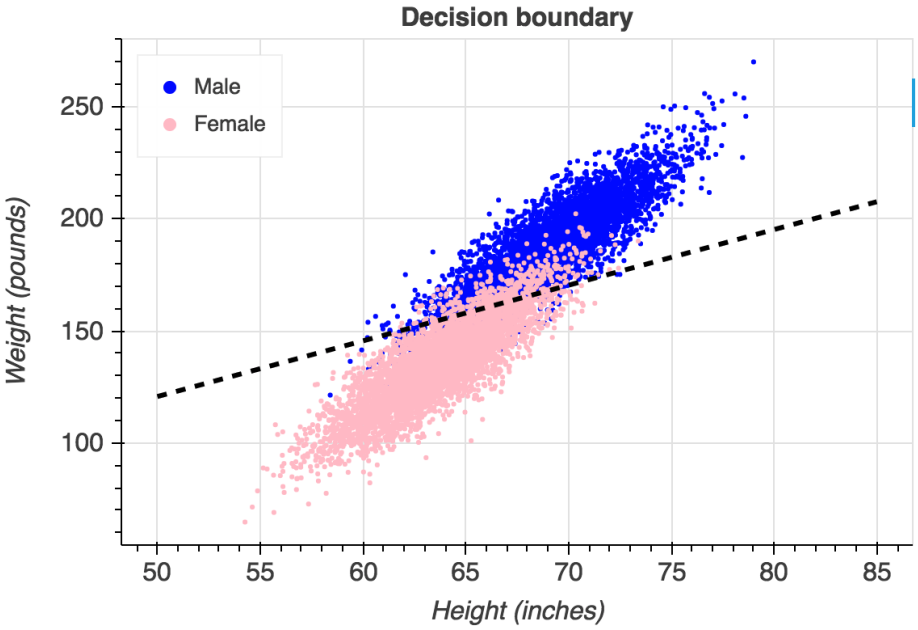
➢ ElasticNet Regression

Elastic Net ilk olarak değişken seçimi verilere çok fazla bağımlı olan ve dolayısıyla kararsız olan Lassoo'ya yapılan eleştirinin bir sonucu olarak ortaya çıktı. ElasticNet burada çözüm olarak doğmuş, her iki dünyanın da en iyisini elde etmek için Ridge regresyon ve Lassoo'nun yeteneklerini almış bir yöntemdir.

```
from sklearn.linear_model import ElasticNet
```

CLASSIFICATION with SKLEARN (SINIFLANDIRMA):

Elimizdeki veri setlerinin, hedef sınıflarını makinaya öğreterek, sınıfı hakkında bilgimiz olmayan verilerin sınıflarını aşağıdaki istatistiksel modeller yardımıyla tahmin ettirmiş oluruz. Örneğin; cinsiyetlere göre, medeni durumu, hastalığın var olup olmama durumu gibi…



➢ Logistic Regression

Belirli bir sınıfa ait olma olasılığını tahmin etmek için yaygın olarak kullanılır (Örneğin, bu e-postanın spam olma olasılığı nedir?). Model tahmini olasılığı % 50'den büyükse, örneğin o sınıfa ait olduğunu (“1” olarak adlandırılan pozitif sınıf olarak adlandırılır) veya başka bir durumda olmadığını (“0” etiketli yani, negatif sınıfa ait olduğunu tahmin eder). Bu onu ikili bir sınıflandırıcı yapar.

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.linear_model import LogisticRegressionCV
```

➢ K Nearest Neighbors Classification

KNN Sınıflandırıcısı, bir veri seti içerisindeki en yakın komşular yöntemiyle, bir K yani sınıf sayısı belirtilerek veya belirlenerek bunların sınıflandırılması sağlanır.

```
from sklearn.neighbors import KNeighborsClassifier
```

➢ Support Vector Machine Classification

SVM sınıflandırıcısını, sınıflar arasında mümkün olan en geniş alana (paralel kesikli çizgilerle temsil edilir) ayıracak şekilde sınıflandırır. Buna geniş marj sınıflaması denir.

```
from sklearn.svm import SVC
```

➢ Naive Bayes Classification

Hedef özellik seviyeleri için arka olasılıkların ve hedef özellik seviyesi verilen bir durumda tanımlayıcı özellikler arasındaki koşullu bağımsızlık varsayımı altında hesaplandığı bir (MAP) sınıflandırma tahmini verir.

```
from sklearn.naive_bayes import GaussianNB
```

➢ Decision Tree Classification

Karar ağacı yöntemiyle belirli sorular sorar ve bunların sonucunda sınıflandırma yapar.

```
from sklearn.tree import DecisionTreeClassifier
```

➢ Random Forest Classification

Birden çok karar ağacı kullanılarak sınıflandırmalar yapar.

```
from sklearn.ensemble import RandomForestClassifier
```

➢ XGBoost Classification

Karar ağaçları tabanlı bir sınıflandırma yöntemidir. Ancak daha hızlı çalışmasıyla bilinir.Bu yöntem kullanılırken hyperparametrelerle oynanabilir ve tahminlerin doğruluk oranları arttırılabilir. Sklearn kütüphanesi dahilinde değildir. Ayrıca kurulmalıdır. Ancak Sklearn ile koordine çalışmaktadır.

```
import xgboost as xgb

xg_reg =xgb.XGBClassifier(learning_rate=0.1,max_depth=10, n_estimators=500)
```

➢ LightGBM Classification

Bu algoritmada gradient boost alt yapısıyla çalışmakla beraber çok iyi ve çok hızlı sonuçlar vermektedir. Hatta XGBoost'tan çok daha hızlı ve yakın sonuçlarla karşılaşılabilmektedir. Bu yöntem kullanılırken hyperparametrelerle oynanabilir ve tahminlerin doğruluk oranları arttırılabilir. Sklearn kütüphanesi dahilinde değildir. Ayrıca kurulmalıdır. Ancak Sklearn ile koordine çalışmaktadır.

```
import lightgbm as lgbm

lgbm_reg=lgbm.LGBMClassifier(learning_rate=0.1,max_depth=10,n_estimators=500,
min_child_weight=1.5)
```

➢ Neural Network Classification

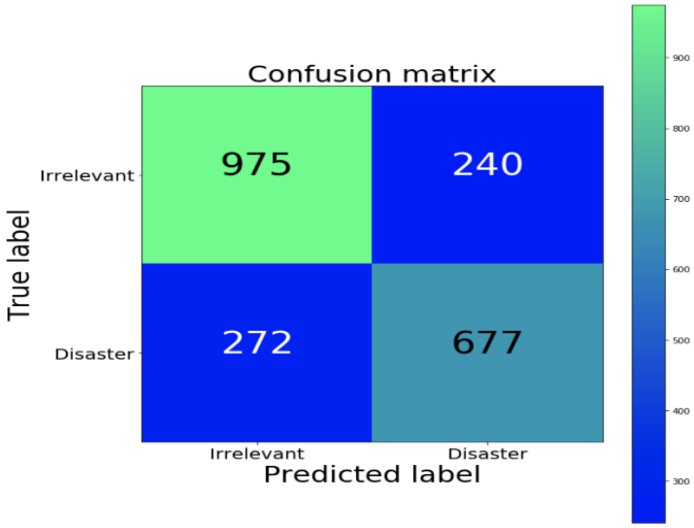
İnsan beyninin öğrenme şekline benzer şekilde çalışan bir algoritmadır. Girdileri ağırlıklarıyla çarpar, bunları toplar ve bunu bir fonksiyonun içine sokarak sınıfları tahmin etmeye çalışır.

```
from sklearn.neural_network import MLPClassifier
```

➢ Confusion Matrix

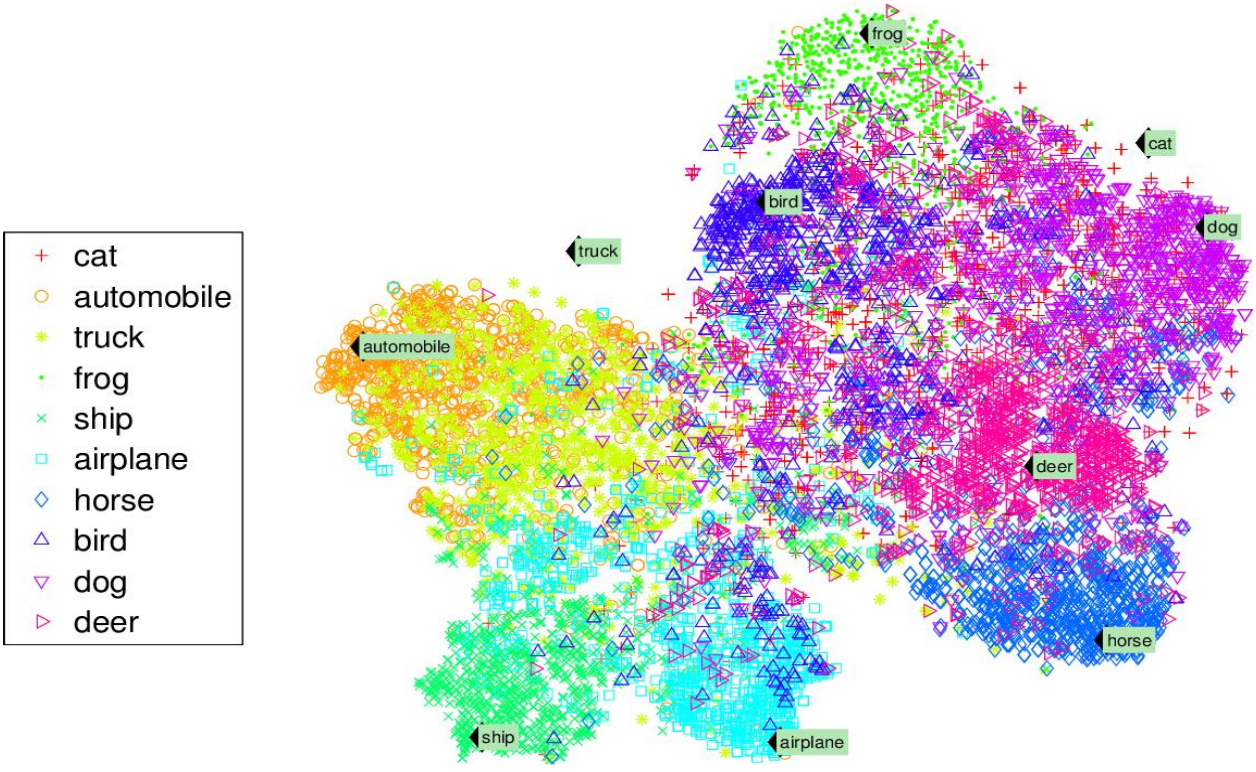
Confusion matrix, verideki var olan durum ile tahmin modelimizin doğru ve yanlış sayısını gösterir.

```
from sklearn.metrics import confusion_matrix
```



CLUSTERING with SKLEARN (KÜMELEME):

Veri setimizi oluşturan verilerimizin hiçbir sınıfa ait olmadığı durumda bunları aşağıdaki istatistiksel kümeleme yöntemleriyle sınıflandırırız.



Tolga Kaplan

- **K Means Clustering**

Merkez olarak belirlenen noktaya olan uzaklığa göre yeni kümelerin oluşturulmasıdır.

```
from sklearn.cluster import KMeans
```

- **Hierarchical Clustering**

Agglomerative *(Parçadan bütüne)* ve Divisive *(Bütünden parçaya)* olarak iki farklı varyasyonu vardır. Agglomerative varyasyonunu anlatmaya çalışırsak ilk önce tüm veriler bir küme haline getirilir yani N tane eleman varsa N tane küme oluşur. Daha sonra birbirine mesafe olarak yakın olan kümeler birleşerek yeni bir küme oluşturur. Bu durum sistem kararlı oluncaya kadar devam eder. Divisive ise Agglomerative' in tam tersidir. İlk başta tüm veriler tek bir küme oluşturulur. Daha sonra bu küme parçalanarak kümeleme işlemi yapılır.

```
from sklearn.cluster import AgglomerativeClustering
```

- **Affinity Propagation**

Küme sayısını (k), benzer olan noktalarla yayılım göstererek belirleyen bir kümeleme yöntemidir.

```
from sklearn.cluster import AffinityPropagation
```

- **Spectral Clustering**

Spektral kümeleme, basit uygulaması ve birçok grafik tabanlı kümelemede ümit verici performansı nedeniyle giderek daha popüler hale gelmiştir. Standart lineer cebir yazılımı ile verimli bir şekilde çözülebilir.

```
from sklearn.cluster import SpectralClustering
```

- **DBSCAN**

DBSCAN, mesafe ölçümüne (genellikle Öklid mesafesi) ve minimum noktaya dayanarak birbirine yakın noktaları birlikte gruplandırır. Ayrıca düşük yoğunluklu bölgelerde bulunan noktaları outliers olarak işaretler.

```
from sklearn.cluster import DBSCAN
```

- **Birch**

Hierarchical kümeleme yöntemine benzerdir. Kaç adet küme oluşturulacağını (K) belirtmeye gerek yoktur.

```
from sklearn.cluster import Birch
```

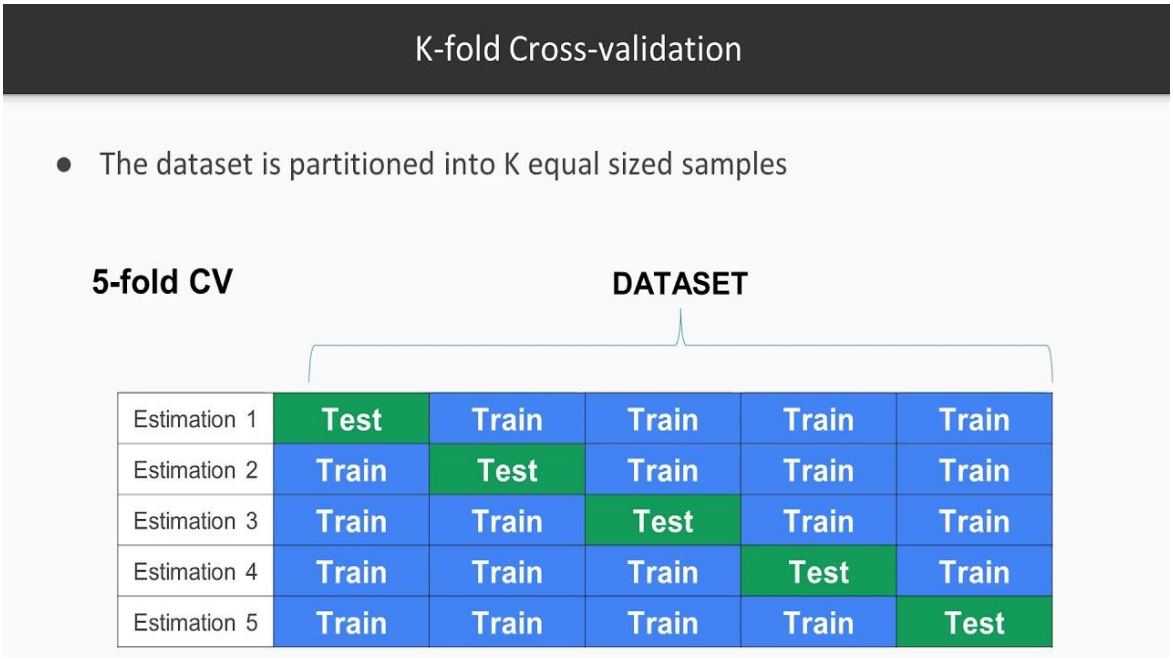
VALIDATION with SKLEARN (ONAYLAMA - DOĞRULAMA):

Veri madenciliğinde uygulanan yöntemlerin başarılarının sınanması için aşağıdaki algoritmaları kullanırız.

➢ K- Fold Cross Validation

Veri setini, aşağıdaki görselde görüldüğü üzere parçalayarak, train-test yöntemiyle çarpraz başarı sınaması yapar.

```
from sklearn.cross_validation import cross_val_score
```



➢ Grid Search Validation

Bilindiği üzere bir çok tahminleyici ve sınıflayıcı algoritması mevcut. Genellikle bizler bu algoritmalar arasından en yüksek doğrulukla tahminleme yapanı seçiyoruz. Rastgele algoritma için parametreleri seçmek yorucu olabileceğinden,(parametrelerin değerlerini rastgele seçmek yerine), belirli bir model için en iyi parametreleri otomatik olarak bulan bir algoritma geliştirmek daha iyi bir yaklaşım olacaktır. Grid Search böyle bir algoritmadır.

```
from sklearn.model_selection import GridSearchCV
```

➢ Leave One Out

Verileri, train / test setlerine bölmek için train / test endeksleri sağlar. Her bir örnek test seti (singleton) olarak kullanılırken, kalan örnekler eğitim setini oluşturur.

```
from sklearn.model_selection import LeaveOneOut
```

Tolga Kaplan

- **Shuffle Split**

Shuffle Split, Confusion MatrixSplit dizileri veya matrisleriyle rasgele train ve test alt kümelerine ayırarak tahmin doğrulaması yapmaya yarar.

```
from sklearn.model_selection import ShuffleSplit
```

- **Stratified K- Fold Cross Validation**

Tabakalı K-Katlamalı Çarpraz Doğrulama, verileri train / test setlerine bölmek için train / test endeksleri sağlar. Bu çapraz doğrulama nesnesi, katmanlı katmanları döndüren bir K- Fold varyasyonudur. Katlamalar, her sınıf için örneklerin yüzdesi korunarak yapılır.

```
from sklearn.model_selection import StratifiedKFold
```

- **Time Series Split**

Zaman serisi verileriyle, geleneksel çapraz doğrulama kullanmamızı sağlar.

```
from sklearn.model_selection import TimeSeriesSplit
```

NOT: Yukarıdaki algoritmaları kullanırken, modelinize uygun olup olmadığını araştırınız.

Tolga Kaplan