**VERİ MADENCİLİĞİ ÇALIŞMASI**

# - Veri Seti

Eğer bir bankada kredi sorumlusuysanız, kredilerde temerrüde düşmesi muhtemel kişilerin göstergesi olan özellikleri tanımlayabilmek ve bu özellikleri iyi ve kötü kredi risklerini tanımlamak için kullanmak isteyebilirsiniz.

700 geçmiş müşteri hakkında bilgi **“bankakredi.csv”** içinde yer almaktadır. Değişkenler:

Yaş : Kişinin yaşı

Eğitim : Kişinin eğitim aldığı yıl

İstihdam : Kişinin istihdamda kaldığı süre

İkamet : Kişinin mevcut ikametgâhında kaldığı süre

Gelir : Hane halkı geliri

G/B : Gelir/Borç Oranı

Kart : Kredi Kartı Borcu

Diğer : Diğer borcu

Risk : Temettü riski % cinsinden

# Genel Açıklama

1) Yukarıdaki veri seti üzerinde Python’da yer alan aşağıdaki regresyon algoritmalarını kodlayacaksınız. Verinin %80’i eğitim %20’si test için kullanılacaktır. Her model için default değerlerle çalıştırdığınız kodlarınızı model başlığı altına kopyalayınız. Her model için farklı parametrelerle aldığınız sonuçların aşağıda belirtilen metrik sonuçlarını yine model başlığı altına eklenecektir.

Bu metrikler: MAE, MSE, RMSE değerleridir. Bu metrikler tahmin sonuçları için hesaplanacaktır

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

**2 - Modeller**

# 2-1-Basit Modeller

*2-1-1-* [*Support Vector Regression*](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html?highlight=svr#sklearn.svm.SVR)

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

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

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

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

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

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

* TABLOLARDA SONUÇLAR MEVCUTTUR.

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

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

Default değerlerle çalıştırıp sonuçları alınız. Kodunuzu ve metrik sonuçlarını buraya kopyalayınız.

from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()

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

y\_pred = model.predict(X\_test)

mae = metrics.mean\_absolute\_error(Y\_test, y\_pred)

mse = metrics.mean\_squared\_error(Y\_test, y\_pred)

rmse = np.sqrt(mse)

print(" MAE={}, MSE={}, RMSE={}".format(mae, mse, rmse))

MAE=21.87857142857143, MSE=760.2693771428571, RMSE=27.572982739320334

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

Ön tanımlı temel regresyon algoritması olarak DecisionTreeRegressor(max\_depth=3) kullanmaktadır. Default değerlerle çalıştırarak sonuçları alınız. Kodunuzu ve metrik sonuçlarını buraya kopyalayınız.

from sklearn.ensemble import AdaBoostRegressor

model = AdaBoostRegressor()

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

y\_pred = model.predict(X\_test)

mae = metrics.mean\_absolute\_error(Y\_test, y\_pred)

mse = metrics.mean\_squared\_error(Y\_test, y\_pred)

rmse = np.sqrt(mse)

print(" MAE={}, MSE={}, RMSE={}".format(mae, mse, rmse))

MAE=28.818185572839358, MSE=970.4164141154898, RMSE=31.151507413213405

Daha sonra <https://www.datacamp.com/community/tutorials/adaboost-classifier-python>yukardaki linki inceleyiniz. “Using Different Base Learners” başlığı altındaki gibi temel regresyon algoritması olarak SVR (Destek Vektör Regresyonu) algoritması kullanarak sonuçları alınız. Kodunuzu ve metrik sonuçlarını buraya kopyalayınız.

from sklearn.ensemble import AdaBoostRegressor

from sklearn.svm import SVR

from sklearn import metrics

import numpy as np

svr = SVR(kernel='linear')

abr = AdaBoostRegressor(n\_estimators=50, base\_estimator=svr, learning\_rate=1)

model = abr.fit(X\_train, Y\_train)

y\_pred = model.predict(X\_test)

mae = metrics.mean\_absolute\_error(Y\_test, y\_pred)

mse = metrics.mean\_squared\_error(Y\_test, y\_pred)

rmse = np.sqrt(mse)

print("MAE={}, MSE={}, RMSE={}".format(mae, mse, rmse))

MAE=31.507736992298486, MSE=1131.065026869621, RMSE=33.63131021636863

*2-2-3-*[***Bagging Regressor***](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html#sklearn.ensemble.BaggingRegressor)

Ön tanımlı temel regreson algoritması olarak DecisionTree kullanmaktadır. Default değerlerle çalıştırarak sonuçları alınız ve buraya kopyalayınız.

from sklearn.ensemble import BaggingRegressor

model = BaggingRegressor()

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

y\_pred = model.predict(X\_test)

mae = metrics.mean\_absolute\_error(Y\_test, y\_pred)

mse = metrics.mean\_squared\_error(Y\_test, y\_pred)

rmse = np.sqrt(mse)

print(" MAE={}, MSE={}, RMSE={}".format(mae, mse, rmse))

MAE=22.279999999999998, MSE=819.1454285714286, RMSE=28.620716772495907

Daha sonra <https://www.datacamp.com/community/tutorials/adaboost-classifier-python>yukardaki linki inceleyiniz. “Using Different Base Learners” başlığı altındaki gibi temel regresyon algoritması olarak SVR (Destek Vektör Regresyonu) algoritması kullanarak sonuçları alınız ve buraya kopyalayınız.

from sklearn.ensemble import BaggingRegressor

from sklearn.svm import SVR

from sklearn import metrics

import numpy as np

svr = SVR(kernel='linear')

bgr = BaggingRegressor(base\_estimator=svr, n\_estimators=50, random\_state=0)

model = bgr.fit(X\_train, Y\_train)

y\_pred = model.predict(X\_test)

mae = metrics.mean\_absolute\_error(Y\_test, y\_pred)

mse = metrics.mean\_squared\_error(Y\_test, y\_pred)

rmse = np.sqrt(mse)

print("MAE={}, MSE={}, RMSE={}".format(mae, mse, rmse))

MAE=21.503022776667667, MSE=898.2850010078868, RMSE=29.97140305370916

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

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

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

[learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html#sklearn.ensemble.VotingClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html#sklearn.ensemble.VotingClassifier)  <https://scikit>-[learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html#sklearn.ensemble.VotingRegress or](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html#sklearn.ensemble.VotingRegressor)

Linklerinde yer alan örnek kodlardaki gibi çalıştırıp sonuçları alınız. Kodunuzu ve metrik sonuçlarını buraya kopyalayınız.

from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, BaggingRegressor, VotingRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

import numpy as np

rf = RandomForestRegressor()

abr = AdaBoostRegressor()

bgr = BaggingRegressor()

tahmin = VotingRegressor(estimators=[

    ('rf', rf),

    ('abr', abr),

    ('bgr', bgr),

])

model = tahmin.fit(X\_train, Y\_train)

y\_pred = model.predict(X\_test)

mae = metrics.mean\_absolute\_error(Y\_test, y\_pred)

mse = metrics.mean\_squared\_error(Y\_test, y\_pred)

rmse = np.sqrt(mse)

print("MAE={}, MSE={}, RMSE={}".format(mae, mse, rmse))

MAE=24.032702613941233, MSE=807.9534645212061, RMSE=28.424522239102036

# Özet Tablo ve Yorum

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

|  |  |  |  |
| --- | --- | --- | --- |
| Modeller | MAE | MSE | RMSE |
| SVR(kernel='rbf') | 22,75 | 1194,32 | 34,56 |
| SVR-Linear | 21,21 | 884,56 | 29,74 |
| SVR-Poly | 22,19 | 1103,15 | 33,21 |
| DecisionTreeRegressor-default | 26,95 | 1629,95 | 40,37 |
| DecisionTreeRegressor(max\_depth=5) | 22,06 | 979,05 | 31,28 |
| DecisionTreeRegressor(min\_samples\_split=2) | 24,36 | 1209,45 | 34,78 |
| KNeighborsRegressor-default | 23,41 | 940,14 | 30,66 |
| KNeighborsRegressor(n\_neighbors=8) | 23,54 | 935,49 | 30,58 |
| KNeighborsRegressor(weights='uniform') | 23,41 | 940,14 | 30,66 |
| RandomForestRegressor-default | 21,87 | 760,26 | 27,57 |
| RandomForestRegressor (criterion='poisson') | 21,69 | 748,35 | 27,35 |
| RandomForestRegressor- (n\_estimators=500) | 21,90 | 747,32 | 27,33 |
| AdaBoostRegressor-default | 28,81 | 970,41 | 31,15 |
| BaggingRegressor-default | 22,27 | 819,14 | 28,62 |
| VotingRegressor-(RandomForest, AdaBoost, BaggingRegressor) | 24,03 | 807,95 | 28,42 |

* Min MAE ; SVR-LİNEAR
* Min MSE ; RandomForestRegressor- (n\_estimators=500)
* Min RMSE ; RandomForestRegressor- (n\_estimators=500) ise modelinde görünüyor.