Aşağıda, Python ile şimdiye kadar yazdığımız tahmin algoritmalarının şablonunu bulabilirsiniz:

**Kütüphaneler**

1. #1. kutuphaneler
2. import numpy as np
3. import matplotlib.pyplot as plt
4. import pandas as pd
5. from sklearn.metrics import r2\_score
6. import statsmodels.api as sm

numpy ve pandas kütüphanelerini veriyi işlemek ve hafızada yönlendirmek için kullanıyoruz (data frame gibi sınıflar için)

**Veri Yükleme**

1. # veri yukleme
2. veriler = pd.read\_csv('maaslar\_yeni.csv')
3. x = veriler.iloc[:,2:5]
4. y = veriler.iloc[:,5:]
5. X = x.values
6. Y = y.values

Veri yükleme aşamasında, verinin yükleneceği dosyanın yanında, bağımlı ve bağımsız değişkenleri içeren iki ayrı kolon oluşturulmalıdır. Bağımsız değişkenlerin tamamının x isminde bir dataframe içerisinde ve tek bir kolondan oluşan bağımlı değişkenleri ise y ismine bir data firma içerisinde duracağı kabulü yapılmıştır. Ayrıca X ve Y değişkenleri de numpy dizisi olarak bu dataframe'lerden .values özelliğini alır.

**Korelasyon Matrisi**

print(veriler.corr())

Veriler üzerinde karar verirken, kullanılacak önemli ön işleme aşamalarından birsi de korelasyon matrisidir ve bu matris ile kolonların birbiri ile olan ilişkisi görülebilir.

**Linear Regression**

1. from sklearn.linear\_model import LinearRegression
2. lin\_reg = LinearRegression()
3. lin\_reg.fit(X,Y)
4. model = sm.OLS(lin\_reg.predict(X),X)
5. print(model.fit().summary())
6. print("Linear R2 degeri:")
7. print(r2\_score(Y, lin\_reg.predict((X))))

Sci-Kit Learn kütüphanesinin genel bir özelliği, fit() fonksiyonu ile eğitmesi ve predict fonksiyonu ile tahminde bulunmasıdır. Buradaki örnekte doğrusal regresyon (linear regression) üzerinden X ve Y dizileri verilerek bir makine öğrenmesi algoritması eğitilmiş ve oluşan model daha sonra OLS ve r2\_score fonksiyonları ile ölçülmüştür. Ölçüm sırasında gerçek verileri tutan Y değişkeni (numpy dizisi) ile lin\_reg objesinin predict fonksiyonun döndürdüğü tahmin değerleri (yani modelin tahmin etttiği değerler) karşılaştırılmış dolayısıyla modelin ne kadar başarılı tahmin yaptığı ölçülmüştür.

**Polynomial Regression**

1. from sklearn.preprocessing import PolynomialFeatures
2. poly\_reg = PolynomialFeatures(degree = 4)
3. x\_poly = poly\_reg.fit\_transform(X)
4. print(x\_poly)
5. lin\_reg2 = LinearRegression()
6. lin\_reg2.fit(x\_poly,y)
7. model2 = sm.OLS(lin\_reg2.predict(poly\_reg.fit\_transform(X)),X)
8. print(r2\_score(Y, lin\_reg2.predict(poly\_reg.fit\_transform(X)) ))

Polynomial regression yöntemi aslında doğrusal regresyondan farklı değildir. Hatta aynı nesne ve fonksiyonların kullanıldığı söylenebilir. Buradaki hile, verilerin doğrusal regresyona verilmeden önce bir polinom öznitelik fonksiyonuna verilmesidir. bu işlem yukarıdaki kodda da gösterildiği üzere, PolynomialFeattures nesnesi üzerinden yapılmaktadır.

**Destek Vektör Regresyonu ve Ölçekleme (Support Vector Regression , Scaling)**

1. from sklearn.preprocessing import StandardScaler
2. sc1 = StandardScaler()
3. x\_olcekli = sc1.fit\_transform(X)
4. sc2 = StandardScaler()
5. y\_olcekli = sc2.fit\_transform(Y)
6. from sklearn.svm import SVR
7. svr\_reg = SVR(kernel = 'rbf')
8. svr\_reg.fit(x\_olcekli,y\_olcekli)
9. model3 = sm.OLS(svr\_reg.predict(x\_olcekli),x\_olcekli)
10. print(model3.fit().summary())
11. print(r2\_score(y\_olcekli, svr\_reg.predict(x\_olcekli)) )

Destek vektör regresyonunun en önemli özelliği, marjinal verilere karşı hassas olmasıdır. Bu yüzden ve verilerin daha iyi tahminini sağlamak için, öncelikle standartlaştırma yapılması gerekir. Yukarıdaki kodun ilk parçasında, StandardScaler sınıfından türettilmiş sc1 ve sc2 nesneleri sayesinde hem X hem de Y dizileri ölçeklenmektedir. Ardından SVR sınıfından rbf çekirdeği (kernel) ile üretilen svr\_reg isimli nesne ile tahmin modeli oluşturulmakta, bunun için de sklearn klasiği olan fit metodu kullanılmaktadır.

**Karar Ağacı ile Tahmin (Decision Tree)**

1. from sklearn.tree import DecisionTreeRegressor
2. r\_dt = DecisionTreeRegressor(random\_state=0)
3. r\_dt.fit(X,Y)
4. print('dt ols')
5. model4 = sm.OLS(r\_dt.predict(X),X)
6. print(model4.fit().summary())
7. print("Decision Tree R2 degeri:")
8. print(r2\_score(Y, r\_dt.predict(X)) )

Yapı olarak buraya kadar kullanılan sınıflandırma algoritmalarından pek de farklı olmayan karar ağacı sınıflandırması, DecisionTreeRegressor sınıfından türetilmiş ve yine X ve Y dizileri üzerinde fit() metodu ile bir model inşası için kullanılmıştır.

**Rassal Orman (Random Forest)**

1. from sklearn.ensemble import RandomForestRegressor
2. rf\_reg = RandomForestRegressor(n\_estimators = 10, random\_state=0)
3. rf\_reg.fit(X,Y)
4. print('dt ols')
5. model5 = sm.OLS(rf\_reg.predict(X),X)
6. print(model5.fit().summary())
7. print("Random Forest R2 degeri:")
8. print(r2\_score(Y, rf\_reg.predict(X)) )

Alt yapısında karar ağacı kullanan rassal ormanlar (random forest), şimdiye kadar elde ettiğimiz yapıya çok benzer şekilde, sklearn kütüphanesi içerisinden bir sınıf olarak RandomForestRegressor eklemiş (import) ve bu sınıftan da ürettiğimiz nesne ile fit() ve predict() metotlarını kullanarak makine öğrenmesini gerçekleştirmiştir.

**Neler Öğrendik:**

1. Sci-Kit Learn kütüphanesinin genel akışını ve fit(), predict() gibi metotların genel kullanımını

2. Doğrusal regresyon, Polinomla regresyon, destek vektör regresyonu, karar ağacı regresyonu ve rassal orman regresyonu metotlarının kullanımını

3. Korrelasyon matrisini ve önemini

4. r2\_score ve OLS metotları ile yöntemler arasında karşılaştırma yapmayı