"X=df[['Weight']] ### independent features should be data frame or 2 dimesnionalarray y=df['Height'] ## this variiable can be in series or 1d array X\_series=df['Weight'] np.array(X\_series).shape np.array(y).shape" niyə bu kod belə yazılıb?

**ChatGPT said:**

Bu kod **Linear Regression** və digər ML modellərində **input (X) və output (y)** dəyişənlərinin **ölçü quruluşunu (shape)** düzgün təşkil etmək üçün yazılıb.

İndi müəllim kimi izah edim:

**🧠 Məqsəd:**

ML modelləri (məsələn, LinearRegression) **inputlar üçün 2 ölçülü** array (matrix), **nəticə üçün isə 1 ölçülü** array (vektor) gözləyir.

**🔍 Koda baxaq:**

python

CopyEdit

X = df[['Weight']] # Bu DataFrame-dir, yəni 2D

y = df['Height'] # Bu Series-dir, yəni 1D

X\_series = df['Weight']

np.array(X\_series).shape

np.array(y).shape

**📌 İndi tək-tək izah:**

**✅ X = df[['Weight']]**

* **Double square bracket** ilə yazılıb.
* df[['Weight']] → Pandas **DataFrame** qaytarır.
* shape → (n, 1) — yəni **2D** olur ✅

python

CopyEdit

>>> np.array(df[['Weight']]).shape

(100, 1) # 100 sətir, 1 sütun

**⚠️ X\_series = df['Weight']**

* **Single bracket** ilə yazılıb.
* df['Weight'] → Pandas **Series** qaytarır.
* shape → (n,) — yəni **1D** olur ❌

python

CopyEdit

>>> np.array(df['Weight']).shape

(100,) # 100 sətirlik bir ölçülü array

Bu 1D olduğu üçün ML modelləri **"expected 2D array, got 1D instead"** xətası verə bilər.

**✅ y = df['Height']**

* Burada isə y üçün 1D **tam uyğundur**, çünki LinearRegression target üçün 1D (vektor) gözləyir.

**🎯 Nəticə:**

| **Dəyişən** | **İstifadə tipi** | **Şəkli (shape)** | **Niyə belə olmalıdır?** |
| --- | --- | --- | --- |
| X = df[['Weight']] | Input | (n, 1) | 2D olmalıdır, çünki ML modelləri bunu gözləyir |
| y = df['Height'] | Target | (n,) | 1D olmalıdır, çünki bu cavab sütunudur |
| X\_series = df['Weight'] | Series versiyası | (n,) | 1D — ML üçün **uyğun deyil** |

**🧪 Bonus: Əgər Series-dirsə necə düzəldək?**

python

CopyEdit

X = df['Weight'].values.reshape(-1, 1) # Series → 2D array

İstəyirsənsə, bu mövzunu Python və numpy ilə **ML üçün data preprocessing dərsliyi** şəklində PowerPoint slaydlar ilə də hazırlaya bilərəm. Hazırlayım?

**You said:**

"from sklearn.model\_selection import train\_test\_split X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.25,random\_state=42)" bu bildiyim qədər heçnə qaytarmır. Hər hissəni izah et

**ChatGPT said:**

Çox gözəl sualdır! Bu sətir Machine Learning-də model qurmazdan əvvəl **məlumatların təlim və test hissələrinə bölünməsi** üçün istifadə olunur.

İndi sətir-sətir və anlayış-anlayış izah edim 👨‍🏫:

**📌 Kod:**

python

CopyEdit

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

**🔍 1. from sklearn.model\_selection import train\_test\_split**

Bu sətir scikit-learn kitabxanasından train\_test\_split funksiyasını **import edir**.

**Əsas funksiyası:**  
Dataseti **təlim (train)** və **test** hissələrinə **təsadüfi şəkildə bölmək**.

**🔍 2. train\_test\_split(X, y, ...)**

Bu funksiya:

* X – **Giriş xüsusiyyətləri** (məsələn: Weight)
* y – **Nəticə dəyişəni** (məsələn: Height)  
  götürüb dörd hissəyə ayırır:

| **Dəyişən** | **İzah** |
| --- | --- |
| X\_train | Modelin öyrənəcəyi girişlər |
| X\_test | Test üçün istifadə olunacaq girişlər |
| y\_train | Modelin öyrənəcəyi nəticələr |
| y\_test | Test üçün istifadə olunacaq nəticələr |

**🔍 3. test\_size=0.25**

Bu, **test hissəsinin nə qədər olacağını** göstərir.

* 0.25 = **datasetin 25%-i test**, **75%-i train** olacaq.

Məsələn, əgər X 100 sətrlikdirsə:

* X\_train: 75 sətir
* X\_test: 25 sətir

**🔍 4. random\_state=42**

Bu, **təkrar ediləbilən nəticə** almaq üçündür.

* train\_test\_split təsadüfi ayırma etdiyi üçün nəticə hər dəfə fərqli ola bilər.
* random\_state=42 qoyduqda bu bölmə **həmişə eyni olacaq**.

**Qeyd:** 42 sadəcə məşhur "default" dəyərdir, istənilən tam ədəd ola bilər.

**📌 Nəticədə kod:**

python

CopyEdit

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, # Giriş məlumatları (input features)

y, # Nəticə/səbəb dəyişəni (target)

test\_size=0.25, # 25% test

random\_state=42 # Sabitlik üçün

)

Bu funksiya **dörd dəyər qaytarır** — yəni None qaytarmır, əksinə **dəyişənlərə bölünmüş məlumatlar verir** ✅

**🎯 Niyə bu vacibdir?**

Çünki modelimizi **öyrəndiyi məlumatlarla test edə bilmərik** — bu halda model nəticəni "əzbərləyir".

Test datası: "Model yeni dataya necə reaksiya verir?" sualına cavab verir.

İstəyirsənsə, bu mövzunu **vizual şəkillərlə, kod nümunələri və real datasetlər üzərində** PowerPoint slayd dərsliyi kimi də hazırlaya bilərəm. Hazırlayım?

**You said:**

25% və 75% olaraq bölünməsi nə qədər yaxşıdır

**ChatGPT said:**

Çox gözəl sualdır! 🔍 Modelin keyfiyyətli öyrənməsi və düzgün test olunması üçün **train-test bölünməsi** çox vacibdir.

**🎯 25%-75% bölünməsi nə qədər yaxşıdır?**

Bu **standart və geniş istifadə olunan bölünmədir**, çünki:

| **Bölmə** | **Məna** |
| --- | --- |
| 75% – **Train** | Model öyrənmək üçün kifayət qədər məlumat alır |
| 25% – **Test** | Modelin ümumiləşdirmə qabiliyyəti yaxşı yoxlanır |

Beləliklə, **ənənəvi və balanslı yanaşma** hesab olunur ✅

**💡 Amma hamı üçün ideal deyil — hansı hallarda dəyişmək olar?**

**🔸 Dataset çox böyükdürsə (məsələn, 100.000+ nümunə):**

* Hətta **10% test** belə yetərlidir.
* Məsələn: test\_size = 0.1, train\_size = 0.9

**🔸 Dataset çox kiçikdirsə (məsələn, 100 nümunə):**

* **25% test (25 nümunə)** bəzən az ola bilər
* Belə halda **Cross Validation** (CV) tövsiyə olunur

**📊 Nümunəvi test/train bölmələri:**

| **Train %** | **Test %** | **Nə vaxt istifadə olunur** |
| --- | --- | --- |
| 80% | 20% | Böyük datasetlərdə |
| 75% | 25% | **Ən çox istifadə olunan** |
| 70% | 30% | Kiçik datasetlərdə daha yaxşı test |
| 60% | 40% | Daha çox test ehtiyacı olduqda (təcrübə) |

**🧪 Alternativ: Cross-Validation**

Əgər dataset **kiçikdirsə**, daha doğru nəticə üçün KFold və ya cross\_val\_score istifadə olunur.

**✅ Yekun tövsiyə:**

| **Dataset ölçüsü** | **Təbii bölünmə** | **Əlavə tövsiyə** |
| --- | --- | --- |
| Kiçik (<1000) | 70/30 və ya CV | Cross Validation vacibdir |
| Orta (~10,000) | 75/25 | Normal halda kifayət edir |
| Böyük (>100k) | 90/10 və ya 80/20 | Sürət və balans üçün yetərlidir |

İstəyirsənsə, **"Train/Test bölünməsi və CV strategiyaları"** mövzusunda slayd dərsliyi də hazırlaya bilərəm. Nümunə datalar və qrafiklər ilə. Hazırlayım?

**You said:**

"from sklearn.preprocessing import StandardScaler scaler=StandardScaler() X\_train=scaler.fit\_transform(X\_train) X\_test=scaler.transform(X\_test) X\_test" indi bunlari izah et

**ChatGPT said:**

Əla! Gəlin bu kodu **müəllim kimi** addım-addım izah edim 👨‍🏫 — bu sətirlər **datanı normallaşdırmaq (standardizasiya etmək)** üçündür. Bu, xüsusilə **Linear Regression, SVM, KNN, PCA** kimi modellər üçün çox vacibdir.

**📌 Kod:**

python

CopyEdit

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**🔍 Niyə istifadə edirik?**

Çünki machine learning modelləri **böyük və kiçik ölçülü dəyişənlərə fərqli reaksiya verə bilər**.  
Məsələn:

* X1: gəlir (minlərlə manat)
* X2: yaş (illərlə)

Bu zaman **X1 modelə dominantlıq edəcək**, bu da modelin balansını pozar.

**🎯 Məqsəd:**

**StandardScaler** ilə biz X\_train və X\_test dəyişənlərini elə transformasiya edirik ki:

Yeni X=X−μσ\text{Yeni X} = \frac{X - \mu}{\sigma}Yeni X=σX−μ​

Yəni:

* Orta (mean) → **0** olur
* Standart sapma (std) → **1** olur

**🧩 İndi hər sətiri izah edək:**

**✅ from sklearn.preprocessing import StandardScaler**

Bu sətir **scikit-learn** kitabxanasından StandardScaler klassını import edir.

**✅ scaler = StandardScaler()**

Burada scaler adlı obyekt yaradılır.  
Bu obyektin içində fit() və transform() metodları mövcuddur.

**✅ X\_train = scaler.fit\_transform(X\_train)**

Bu **ən önəmli sətirdir**:

* .fit() → X\_train üzərində **ortalama (mean)** və **std (standard deviation)** hesablayır
* .transform() → həmin ortalama və std ilə **X\_train datanı dəyişir**

❗ Yəni fit() yalnız X\_train-ə tətbiq olunur. Heç vaxt X\_test-ə fit() tətbiq etmə!

**✅ X\_test = scaler.transform(X\_test)**

Burada **YALNIZ .transform()** istifadə olunur. Çünki:

* X\_test üçün eyni **mean** və **std** istifadə edilməlidir
* Əks halda **data leakage (məlumat sızması)** olar ❌

**✅ X\_test**

Bu sonda sadəcə dəyişdirilmiş test datasını göstərir.

**📊 Nəticə:**

| **Addım** | **Əməliyyat** | **Məqsəd** |
| --- | --- | --- |
| fit() | X\_train üçün ortalama və std tapır | Ölçüləri müəyyənləşdirir |
| transform() | Datanı həmin ölçülərlə dəyişir | Standart miqyasda qurur |
| fit\_transform() | İkisini bir yerdə edir | Daha qısa yazılış |

**📦 Vizual izah (məsələn):**

Əvvəl:

| **Weight (kg)** |
| --- |
| 45 |
| 80 |
| 100 |

Standardize etdikdən sonra:

| **Std Weight** |
| --- |
| -1.2 |
| 0.3 |
| 0.9 |

**✅ Yekun:**

StandardScaler:

* Bütün X\_train üçün **fit\_transform** olunur
* X\_test yalnız **transform** olunur
* Yalnız **giriş xüsusiyyətləri (X)** üçün tətbiq olunur, y üçün yox!

İstəyirsənsə, bu prosesi real dataset və qrafiklərlə PowerPoint dərs şəklində hazırlaya bilərəm. Vizual şəkildə + koda əsasən. Hazırlayım?

**You said:**

\*\*StandardScaler\*\* ilə biz `X\_train` və `X\_test` dəyişənlərini elə transformasiya edirik ki: Yeni X\=X−μσ\\text{Yeni X} = \\frac{X - \\mu}{\\sigma}Yeni X\=σX−μ​ Yəni: \* Orta (mean) → \*\*0\*\* olur \* Standart sapma (std) → \*\*1\*\* olur

bu hissəni anlamadım

**ChatGPT said:**

Çox gözəl, gəlin bu hissəni **sadə və aydın** şəkildə izah edim. Müəllim tərzində başa salıram 👇

**🎯 Məqsəd: StandardScaler nə edir?**

**StandardScaler** funksiyası **məlumatın miqyasını dəyişir** ki, **bütün sütunlar oxşar ölçüdə olsun.**

Sadə desək: "Bütün dəyərləri orta ətrafında yerləşdirir və onları oxşar aralıqda sıxır."

**📐 Rəqəmsal olaraq bu necə olur?**

StandardScaler aşağıdakı **formulu** tətbiq edir:

Yeni dəyər=Dəyər−OrtaStandart Sapma\text{Yeni dəyər} = \frac{\text{Dəyər} - \text{Orta}}{\text{Standart Sapma}}Yeni dəyər=Standart SapmaDəyər−Orta​

**Misal:**

Tutaq ki, sən Weight dəyişəninə baxırsan:

* 1-ci insanın çəkisi: **80 kg**
* Weight sütununun ortalaması (**μ**) = 70
* Standart sapma (**σ**) = 10

Bu insana aid dəyəri belə dəyişirik:

Yeni Weight=80−7010=1.0\text{Yeni Weight} = \frac{80 - 70}{10} = 1.0Yeni Weight=1080−70​=1.0

Bu deməkdir ki, bu insanın çəkisi **ortalamadan 1 standart sapma yuxarıdır.**

**🔍 Nəticə nə olur?**

Əgər StandardScaler tətbiq etsən:

* **Ortalama (mean)** bütün X\_train üçün **0 olur**
* **Standart sapma (std)** isə **1 olur**

**Bu niyə vacibdir?**

Bəzi ML alqoritmləri (məsələn: Linear Regression, KNN, SVM) fərqli ölçülü sütunlardan təsirlənir.

Əgər bir sütun 0–1 arasında, o biri isə 0–10.000 arasında olsa, model **balansı pozar.**

**💡 Məsələn:**

**Əvvəl:**

| **Weight** | **Age** |
| --- | --- |
| 80 | 18 |
| 95 | 32 |
| 60 | 40 |

Bunların miqyası fərqlidir (Weight böyükdür, Age kiçik).

**StandardScaler tətbiq ediləndən sonra:**

| **Weight (scaled)** | **Age (scaled)** |
| --- | --- |
| 0.5 | -1.2 |
| 1.1 | 0.0 |
| -1.6 | 1.2 |

İndi **bütün dəyərlər eyni aralıqda və balanslıdır.**

Əgər istəsən, bu prosesi **Excel** faylı ilə və ya **PowerPoint qrafiklərlə** göstərərək daha vizual şəkildə izah edə bilərəm. Hazırlayım?

**You said:**

"from sklearn.linear\_model import LinearRegression regression=LinearRegression(n\_jobs=-1) regression.fit(X\_train,y\_train) print("Coefficient or slope:",regression.coef\_) print("Intercept:",regression.intercept\_)" bunu izah et

**ChatGPT said:**

Çox gözəl! Bu sətirlərdə biz artıq **Linear Regression modelini qurur və onu öyrədirik**. İndi bunu müəllim kimi **addım-addım və sadə** şəkildə izah edim 👨‍🏫:

**📌 Kod:**

python

CopyEdit

from sklearn.linear\_model import LinearRegression

regression = LinearRegression(n\_jobs=-1)

regression.fit(X\_train, y\_train)

print("Coefficient or slope:", regression.coef\_)

print("Intercept:", regression.intercept\_)

**🔍 İndi hər sətrin mənasını başa düşək:**

**✅ from sklearn.linear\_model import LinearRegression**

Bu sətir scikit-learn kitabxanasından **Linear Regression** modelini gətirir.

**✅ regression = LinearRegression(n\_jobs=-1)**

Bu sətirdə **model obyektini yaradırsan**.

* n\_jobs=-1 yazılması:
  + Modeli **çoxnüvəli prosessorlarda paralel** işlətməyə imkan verir.
  + -1 deməkdir: “mümkün olan bütün CPU nüvələrindən istifadə et”.
  + Bu, **böyük verilənlərdə** daha sürətli olur.

**✅ regression.fit(X\_train, y\_train)**

Bu çox vacib hissədir:

Bu sətir **modeli öyrədir**.

* X\_train — giriş xüsusiyyətləri (məsələn, çəki)
* y\_train — nəticə (məsələn, boy)

Model öyrənir ki, **girişlə nəticə arasındakı əlaqəni** tapıb xətti qurmaq üçün **ən uyğun əmsalları tapsın**.

**✅ print("Coefficient or slope:", regression.coef\_)**

Bu sətir **slope** (meyl, əmsal) çap edir.

y=mx+by = mx + by=mx+b

Burada:

* m → **coefficient (regression.coef\_)**
  + Girişdəki 1 birim artım, nəticəyə necə təsir edir?

**✅ print("Intercept:", regression.intercept\_)**

Bu sətir **intercept** çap edir, yəni:

* b → xəttin y oxunu kəsdiyi nöqtə.

**🔢 Məsələn:**

Əgər çap etdikdə görsən:

yaml

CopyEdit

Coefficient or slope: [1.45]

Intercept: 62.3

Deməkdir ki:

Height=1.45⋅Weight+62.3\text{Height} = 1.45 \cdot \text{Weight} + 62.3Height=1.45⋅Weight+62.3

Yəni:

* 1 kg artım, boyda **1.45 sm artıma** səbəb olur.

**📌 Yekun:**

| **Əməliyyat** | **Məna** |
| --- | --- |
| .fit() | Modeli öyrədir (X\_train və y\_train ilə) |
| .coef\_ | Hər bir xüsusiyyətin təsir əmsalı (slope) |
| .intercept\_ | Başlanğıc dəyər (modelin sabiti) |
| n\_jobs=-1 | Hesablama üçün bütün prosessorlardan istifadə |

İstəyirsənsə, bu hissəni **nümunəli PowerPoint** dərsi kimi və ya **vizual xətti qrafiklərlə** göstərim. Xəttin necə çəkildiyini və coef\_, intercept\_-in qrafikdə necə göründüyünü də göstərə bilərəm. Hazırlayım?

**You said:**

"## plot Training data plot best fit line plt.scatter(X\_train,y\_train) plt.plot(X\_train,regression.predict(X\_train))" bunu izah et

**ChatGPT said:**

Əla! Bu kod parçası **Linear Regression modelini vizual göstərmək** üçündür. Gəlin müəllim kimi **addım-addım və çox sadə** şəkildə izah edim 👨‍🏫

**🎯 Məqsəd:**

Bu 3 sətir kod:

* X\_train, y\_train təlim datasını **nöqtələr kimi göstərir** (scatter)
* Regression modelin tapdığı **ən uyğun xətti çəkir** (plot)

**📌 Kod:**

python

CopyEdit

## plot Training data plot best fit line

plt.scatter(X\_train, y\_train)

plt.plot(X\_train, regression.predict(X\_train))

**✅ plt.scatter(X\_train, y\_train)**

Bu sətir **təlim datasını nöqtə kimi çəkir**.

* X\_train – X oxundakı dəyərlər (məsələn, çəki)
* y\_train – Y oxundakı dəyərlər (məsələn, boy)

➡️ **Gerçək datanı** (müşahidələri) göstərir.

**✅ plt.plot(X\_train, regression.predict(X\_train))**

Bu sətir **ən uyğun xətti çəkir** (best fit line).

* regression.predict(X\_train) → Linear Regression modelinin X\_train üçün **təxmin etdiyi y dəyərləridir**.

➡️ Bu xətt, "model bu dataya ən yaxşı uyğun olan düz xətt budur" deməkdir.

**📊 Nəticə olaraq nə görəcəyik?**

* Nöqtələr — təlim datası (çəki vs boy)
* Xətt — Linear Regression modelin tapdığı **ən yaxşı xətti**

**📉 Vizual olaraq:**

markdown

CopyEdit

Y oxu ↑

● ● ● <- təlim datası

● ●

——————————————— <- best fit xətti (y = mx + b)

X oxu →

**🔍 Niyə bu vacibdir?**

Çünki:

* Bu qrafik sayəsində **modelin necə uyğunlaşdığını** görmək olur
* Əgər nöqtələr xəttə yaxınsa → **yaxşı uyğunlaşıb**
* Əgər nöqtələr çox uzaqdadırsa → **underfitting və ya overfitting ola bilər**

**✅ Tövsiyə:**

Vizualı daha aydın etmək üçün rəng və etiketlər əlavə etmək olar:

python

CopyEdit

plt.scatter(X\_train, y\_train, color='blue', label='Training Data')

plt.plot(X\_train, regression.predict(X\_train), color='red', label='Best Fit Line')

plt.xlabel("Weight")

plt.ylabel("Height")

plt.legend()

plt.title("Linear Regression: Training Data vs Best Fit Line")

plt.show()

İstəyirsənsə, bu qrafiki **real misal və data ilə bir PowerPoint** və ya **Jupyter Notebook** dərs formasında hazırlaya bilərəm. Hazırlayım?

**You said:**

plt.plot(X\_train, regression.predict(X\_train)) bu dusturda X\_train yerine niye regression.predict(y\_train) yazilmadi?

**ChatGPT said:**

Çox gözəl sualdır! Bu sualın cavabı **Linear Regression modelinin necə işlədiyini başa düşmək** üçün vacibdir. Müəllim kimi sadə və dərin izah edim 👨‍🏫

**✅ Qısa cavab:**

python

CopyEdit

regression.predict(X\_train)

→ **Doğrudur**, çünki **model girişə (X)** əsasən **çıxış (y)**-ı proqnozlaşdırır.

**🚫 Amma:**

python

CopyEdit

regression.predict(y\_train)

→ **Səhvdir**, çünki:

* y\_train — nəticə dəyərləridir (hədəf)
* Model **çıxışdan çıxışı proqnozlaşdıra bilməz**
* Model **giriş (X)** görməlidir ki, proqnoz versin

**📊 Sadə misal:**

Tutaq ki, sən Weight → Height modelini qurursan:

| **Weight (X)** | **Height (y)** |
| --- | --- |
| 50 | 160 |
| 60 | 170 |
| 70 | 180 |

Model öyrənir:

Height=m⋅Weight+b\text{Height} = m \cdot \text{Weight} + bHeight=m⋅Weight+b

Yəni:

y=f(x)=m⋅x+by = f(x) = m \cdot x + by=f(x)=m⋅x+b

**🔁 İndi proqnoz zamanı:**

* Sən modelə **çəki (X)** verirsən
* Model sənə **boy (y)** təxmin edir

Yəni bu doğrudur:

python

CopyEdit

regression.predict(X\_train)

Çünki bu, belə işləyir:

predict(x)→y\text{predict}(x) \rightarrow ypredict(x)→y

**🚫 Niyə regression.predict(y\_train) səhvdir?**

Çünki:

* Sən modelə **çıxış verirsən**
* Amma model **çıxışı deyil, girişə görə çıxışı hesablamaq** üçün qurulub

Bu, belə olardı ki:

Mənə cavabı ver, mən sənə sualı deyim! — Bu, mümkün deyil 😄

**🧠 Analogi ilə:**

Təsəvvür et:

* **X\_train** = yaş
* **y\_train** = maaş

Model öyrənib: yaşa görə maaş təxmin etsin.

İndi sən qrafik çəkmək istəyirsənsə:

python

CopyEdit

plt.plot(X\_train, regression.predict(X\_train)) ✅

Yəni: **Hansı yaşda hansı maaşı proqnozlaşdırır?**

**✅ Yekun:**

| **İfadə** | **İzah** | **Doğru?** |
| --- | --- | --- |
| regression.predict(X\_train) | Modelə giriş verirsən → çıxış alırsan | ✅ |
| regression.predict(y\_train) | Modelə çıxış verirsən → bu, mümkün deyil | ❌ |

Əgər istəsən bu prosesi **animasiya və ya qrafiklərlə PowerPoint** dərsliyinə çevirə bilərəm. Hazırlayım?

**You said:**

"### prediction of test data 1. predicted height output= intercept +coef\_(Weights) 2. y\_pred\_test =156.470 + 17.29(X\_test) y\_pred=regression.predict(X\_test)

**ChatGPT said:**

Əla! Gəlin bu hissəni **müəllim kimi sadə və aydın** şəkildə izah edim 👨‍🏫

**🎯 Məqsəd:**

Sən **Linear Regression** modelini artıq öyrətmisən və indi onu **test datası ilə sınayırsan.**

Yəni:

Modeli yoxlayırıq: "Verilmiş çəkiyə (X\_test) görə model boyu (Height) düzgün təxmin edir?"

**📌 1. Riyazi düsturla təxmin (manual):**

**Deyilir ki:**

text

CopyEdit

Predicted Height = Intercept + Coefficient × Weight

Tutaq ki:

* **Intercept (qəbuledici sabit)** = 156.470
* **Coefficient (slope / əmsal)** = 17.29
* X\_test-də çəki dəyərləri var

**Məsələn:**

Əgər bir insanın çəkisi (X\_test) = 60 kq-dırsa:

Height=156.470+17.29×60=159.47+1037.4=1193.87\text{Height} = 156.470 + 17.29 \times 60 = 159.47 + 1037.4 = 1193.87Height=156.470+17.29×60=159.47+1037.4=1193.87

Bu, modelin həmin çəkiyə görə **proqnozlaşdırdığı boy**dur.

**📌 2. Kod ilə təxmin (avtomatik):**

python

CopyEdit

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

Bu kod sətiri **X\_test** verilənlərinə görə y\_pred adlı **təxmin edilmiş boy siyahısı** çıxarır.

Bu, **yuxarıdakı düsturla hər bir nümunəyə tətbiq olunur** – amma avtomatik.

**✅ Nəticə nə olur?**

y\_pred — **təxmin edilmiş boylar**, X\_test-ə görə.  
Bunları y\_test (gerçək boylar) ilə müqayisə edib **nə qədər düzgün işlədiyini** qiymətləndirə bilərsən.

**🔍 Genişləndirilmiş şəkildə bu belədir:**

python

CopyEdit

# Modelin proqnozları

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

# Gerçək və proqnoz müqayisəsi

for actual, predicted in zip(y\_test, y\_pred):

print(f"Actual: {actual:.1f}, Predicted: {predicted:.1f}")

**📊 Əlavə Vizual Təklif**

Bunu qrafiklə də göstərə bilərsən:

python

CopyEdit

plt.scatter(X\_test, y\_test, color='blue', label='Actual')

plt.plot(X\_test, y\_pred, color='red', label='Predicted')

plt.legend()

plt.title("Actual vs Predicted Heights")

plt.xlabel("Weight")

plt.ylabel("Height")

plt.show()

**🔚 Nəticə:**

| **Addım** | **İzah** |
| --- | --- |
| Riyazi düstur | y = intercept + coef × x |
| regression.predict(X\_test) | Avtomatik bu düsturu tətbiq edir |
| y\_pred | Modelin təxmin etdiyi Height dəyərləri |
| Sonrakı mərhələ | y\_pred ilə y\_test-i müqayisə etmək (performans ölçümü üçün) |

İstəyirsənsə, bu mərhələləri əyani şəkildə izah edən **PowerPoint dərsliyi və ya Jupyter Notebook template** hazırlaya bilərəm. Hazırlayım?

1/2

**You said:**

"from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error mse=mean\_squared\_error(y\_test,y\_pred) mae=mean\_absolute\_error(y\_test,y\_pred) rmse=np.sqrt(mse) print(mse) print(mae) print(rmse) output: 114.84069295228699 9.66512588679501 10.716374991212605"

**ChatGPT said:**

Gəlin bu kod parçasını **müəllim kimi** addım-addım və çox sadə izah edim 👨‍🏫  
Bu hissə Linear Regression modelinin **performansını ölçmək** üçündür.

**📌 Məqsəd:**

Modelin təxmin etdiyi y\_pred dəyərləri ilə **gerçək** y\_test dəyərləri arasındakı **səhvi (error)** hesablamaq.

**📦 1. from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error**

Bu sətir, sklearn.metrics modulundan **səhv ölçü alətlərini** import edir:

* mean\_absolute\_error — MAE
* mean\_squared\_error — MSE

**📐 2. Səhv ölçüləri:**

python

CopyEdit

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

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

rmse = np.sqrt(mse)

**✅ mean\_squared\_error(y\_test, y\_pred)**

MSE=1n∑i=1n(yactual−ypredicted)2MSE = \frac{1}{n} \sum\_{i=1}^{n} (y\_{\text{actual}} - y\_{\text{predicted}})^2MSE=n1​i=1∑n​(yactual​−ypredicted​)2

* Kvadrat fərqlərin ortalamasıdır.
* Böyük səhvləri **daha çox cəzalandırır**.
* Bizdə çıxan nəticə:

ini

CopyEdit

mse = 114.84

**✅ mean\_absolute\_error(y\_test, y\_pred)**

MAE=1n∑i=1n∣yactual−ypredicted∣MAE = \frac{1}{n} \sum\_{i=1}^{n} |y\_{\text{actual}} - y\_{\text{predicted}}|MAE=n1​i=1∑n​∣yactual​−ypredicted​∣

* Fərqlərin **modulunun ortalamasıdır**
* Daha **real həyati** ölçüdür.
* Bizdə çıxan nəticə:

ini

CopyEdit

mae = 9.66

**✅ rmse = np.sqrt(mse)**

RMSE=MSERMSE = \sqrt{MSE}RMSE=MSE​

* MSE-nin **kvadrat kökü** alınır
* **Səhv vahidi** nəticənin vahidi ilə **eyni olur**
* Bizdə çıxan nəticə:

ini

CopyEdit

rmse = 10.71

**📤 print(...) nəticəsi:**

CopyEdit

114.84069295228699 → MSE

9.66512588679501 → MAE

10.716374991212605 → RMSE

**🧠 Hansı nə zaman istifadə olunur?**

| **Ölçü** | **Açıqlama** | **Nə vaxt yaxşıdır?** |
| --- | --- | --- |
| **MAE** | Real həyati orta səhv | Fərqlərin ölçüsünü anlamaq üçün |
| **MSE** | Kvadratlandırılmış səhv | Böyük səhvləri önəmli saymaq istəyəndə |
| **RMSE** | Səhv ilə ölçünün eyni vahiddə olması üçün | Ən yayılmış ölçüdür |

**📊 Əlavə: Vizual analiz edə bilərsən**

python

CopyEdit

import matplotlib.pyplot as plt

plt.scatter(y\_test, y\_pred, color='green')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], color='red', lw=2)

plt.xlabel("Actual Height")

plt.ylabel("Predicted Height")

plt.title("Actual vs Predicted")

plt.grid()

plt.show()

**✅ Nəticə olaraq:**

Sənin modelinin orta səhvi təxminən **9.66 sm**-dir və **kvadrat səhvin kökü** təxminən **10.71 sm**. Bu nəticəyə əsasən deyə bilərik:

Model pis deyil, amma hələ də təkmilləşdirilə bilər (məsələn, əlavə xüsusiyyətlərlə və ya daha çox datayla).

İstəyirsənsə, bu izahı və hesablama hissəsini vizuallaşdıran **PowerPoint slayd** və ya **Jupyter Notebook** dərsi hazırlaya bilərəm. Hazırlayım?

**You said:**

modelin pis və ya yaxşı olmasını hansı rəqəmlərə görə deyə bilərdik?

**ChatGPT said:**

Çox əla və vacib sualdır!  
**Linear Regression modelinin yaxşı və ya pis olduğunu** demək üçün **birtərəfli bir rəqəm yox**, **bir neçə metrik** və **domen biliyi** lazımdır. Müəllim kimi sadə və sistematik şəkildə izah edim 👨‍🏫

**📏 1. Əsas metriklər:**

**✅ Mean Absolute Error (MAE)**

* Orta səhv: |actual - predicted|
* **Aşağı olduqca daha yaxşıdır**
* MAE = 9.66 → "ortalama olaraq 9.66 sm səhv edirik"

**✅ Mean Squared Error (MSE)**

* Kvadratlaşdırılmış səhv: böyük səhvləri daha çox "cəzalandırır"
* MAE-dən daha həssasdır
* MSE = 114.84 (nə qədər kiçikdirsə, o qədər yaxşıdır)

**✅ Root Mean Squared Error (RMSE)**

* Səhv dəyəri ilə eyni vahiddə olur (məs: sm, kq)
* RMSE = 10.71 → bu o deməkdir ki, proqnozlar **orta hesabla 10.71 sm** yanılır

**✅ R² Score (R-squared / Determination Coefficient)**

**Model ümumi dəyişkənliyin neçə faizini izah edir?**

R2=1−SSRSSTR^2 = 1 - \frac{SSR}{SST}R2=1−SSTSSR​

* 1.0 → mükəmməl model
* 0.0 → heç nə izah etmir
* Müsbət çıxsa → model nəyisə izah edir
* Mənfi çıxsa → çox pis modeldir

python

CopyEdit

from sklearn.metrics import r2\_score

r2 = r2\_score(y\_test, y\_pred)

print("R² Score:", r2)

**📊 2. "Yaxşı" nə deməkdir?**

Bu **problemə görə dəyişir**. Məsələn:

| **Sahə** | **MAE** | **Yaxşı hesab edilirmi?** |
| --- | --- | --- |
| 🌡️ Hava proqnozu (°C) | 2-3 | Bəli |
| 📏 Boy təxmini (sm) | 10 | Bəli (təxmini üçün yetərlidir) |
| 💸 Qiymət təxmini ($) | 500 | Ola bilər, amma çox dəqiq deyil |
| 🏥 Tibbi diaqnoz (risk skoru) | 0.5 | Çox vacib – az olmalıdır |

**📌 3. Nəticəyə necə gəlirik?**

Sənin nəticələr:

* MAE = 9.66
* RMSE = 10.71

Əgər bu boy təxmini üçündürsə, o zaman:

✅ **Qənaətbəxşdir**, çünki:

* Boy ± 10 sm yanılma gündəlik işlərdə qəbul ediləndir
* Amma tibbi və ya təhlükəsizlik sahəsində bu **pis nəticə** ola bilər

**✅ Qısaca:**

| **Metrik** | **Nə qədər az olsa, o qədər yaxşıdır** |
| --- | --- |
| MAE | ✅ |
| MSE | ✅ |
| RMSE | ✅ |
| R² | Yüksək olsa, yaxşıdır (~1.0 ən idealı) ✅ |

**💡 Tövsiyə:**

Əgər bu qiymətlər sənə çox görünürsə:

* Daha çox data topla
* Yeni xüsusiyyətlər əlavə et (məs: yaş, cins, bədən tipi və s.)
* Daha kompleks model sınaya bilərsən (Random Forest, XGBoost və s.)

İstəsən bu metrikləri **nümunə data ilə müqayisəli cədvəl və PowerPoint** şəklində təqdim edə bilərəm. Hazırlayım?

**You said:**

"## R square Formula \*\*R^2 = 1 - SSR/SST\*\* R^2 = coefficient of determination SSR = sum of squares of residuals SST = total sum of squares from sklearn.metrics import r2\_score score=r2\_score(y\_test,y\_pred) print(score) output: 0.7360826717981276 \*\*Adjusted R2 = 1 – [(1-R2)\*(n-1)/(n-k-1)]\*\* where: R2: The R2 of the model n: The number of observations k: The number of predictor variables 1 - (1-score)\*(len(y\_test)-1)/(len(y\_test)-X\_test.shape[1]-1) output: 0.6701033397476595"

**ChatGPT said:**

Gəlin bu hissəni müəllim kimi tam izah edim 👨‍🏫 — həm R², həm də Adjusted R² nədir, necə hesablamaq olur, nəticələr necə şərh edilir.

**📘 1. R² (R-squared) — Determination Coefficient**

**📌 Formula:**

R2=1−SSRSSTR^2 = 1 - \frac{SSR}{SST}R2=1−SSTSSR​

* **SSR** – Sum of Squared Residuals (modelin təxmin etdiyi ilə faktiki nəticələr arasındakı fərq)
* **SST** – Total Sum of Squares (y dəyərlərinin ümumi dəyişkənliyi)

**✅ Məna:**

* **0.0** → Model heç bir şeyi izah etmir
* **1.0** → Model hər şeyi mükəmməl izah edir
* **0.73** → Yəni **model y\_test dəyişkənliyinin 73%-ni izah edir**

**🧪 Python kodu:**

python

CopyEdit

from sklearn.metrics import r2\_score

score = r2\_score(y\_test, y\_pred)

print(score) # 0.7360826717981276

🔎 Bu deməkdir ki, model yetərincə güclüdür — çünki çox hissəni izah edə bilir.

**📘 2. Adjusted R² (Düzəldilmiş R-squared)**

**📌 Formula:**

Adjusted R2=1−((1−R2)(n−1)n−k−1)\text{Adjusted } R^2 = 1 - \left( \frac{(1 - R^2)(n - 1)}{n - k - 1} \right)Adjusted R2=1−(n−k−1(1−R2)(n−1)​)

| **Simvol** | **Mənası** |
| --- | --- |
| n | Müşahidə sayı (datadakı sətir sayı) |
| k | Müstəqil dəyişənlərin sayı |
| R² | Əvvəlki nəticə |

**✅ Niyə lazımdır?**

* Modelə **daha çox dəyişən əlavə etdikcə** R² artmağa meyllidir, **istər faydalı olsun, istər olmasın**
* **Adjusted R² isə buna qarşı qorunur** – lazımsız dəyişənlər əlavə olunarsa, **azalır**

**🧪 Python hesablaması:**

python

CopyEdit

n = len(y\_test)

k = X\_test.shape[1]

adjusted\_r2 = 1 - (1 - score) \* (n - 1) / (n - k - 1)

print(adjusted\_r2) # 0.6701033397476595

🔎 Demək:

* Model 73% izah etsə də (R²),
* **Düzəldilmiş nəticə 67%** çıxır → bu, o deməkdir ki:

Bəzi dəyişənlər modelə tam töhfə vermir və ya data azdır.

**🔎 Hansı nə üçün istifadə olunur?**

| **Metrik** | **İstifadə yeri** | **Şərh** |
| --- | --- | --- |
| **R²** | Model nə qədər izah edir? | Sadə, amma çox dəyişən əlavə edəndə yanıltıcı ola bilər |
| **Adjusted R²** | Daha ədalətli müqayisə üçün | Lazımsız dəyişənlər əlavə olunarsa "cəzalandırır" |

**✅ Sənin nəticələrin**

| **Metrik** | **Dəyər** | **Təhlil** |
| --- | --- | --- |
| R² | 0.736 | Model yaxşıdır, 73.6% izah edir |
| Adjusted R² | 0.670 | Bəzi dəyişənlər çox faydalı olmaya bilər və ya data azdır |

**📌 Nəticə:**

* **R²** yüksək → modelin əsas hissəni izah etdiyini göstərir.
* **Adjusted R²** aşağı → ehtimal ki, data həcmi az və ya dəyişənlər az əhəmiyyətlidir.
* Daha çox data + faydalı dəyişən → hər iki dəyəri yüksəldər.

İstəyirsənsə, bununla bağlı **PowerPoint dərsliyi**, vizuallar və real nümunələrlə təqdimat hazırlaya bilərəm. Hazırlayım?

**You said:**

"import statsmodels.api as sm model=sm.OLS(y\_train,X\_train).fit() prediction=model.predict(X\_test) print(prediction) output: [ 5.79440897 5.79440897 -28.78711691 23.60913442 -7.82861638 34.08838469] print(model.summary()) output: OLS Regression Results ======================================================================================= Dep. Variable: Height R-squared (uncentered): 0.012 Model: OLS Adj. R-squared (uncentered): -0.050 Method: Least Squares F-statistic: 0.1953 Date: Mon, 01 Aug 2022 Prob (F-statistic): 0.664 Time: 19:51:00 Log-Likelihood: -110.03 No. Observations: 17 AIC: 222.1 Df Residuals: 16 BIC: 222.9 Df Model: 1 Covariance Type: nonrobust ============================================================================== coef std err t P>|t| [0.025 0.975] ------------------------------------------------------------------------------ x1 17.2982 39.138 0.442 0.664 -65.671 100.267 ============================================================================== Omnibus: 0.135 Durbin-Watson: 0.002 Prob(Omnibus): 0.935 Jarque-Bera (JB): 0.203 Skew: -0.166 Prob(JB): 0.904 Kurtosis: 2.581 Cond. No. 1.00 ============================================================================== Notes: [1] R² is computed without centering (uncentered) since the model does not contain a constant. [2] Standard Errors assume that the covariance matrix of the errors is correctly specified. C:\Users\win10\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=17 warnings.warn("kurtosistest only valid for n>=20 ... continuing "

**ChatGPT said:**

Bu çox əhəmiyyətli və professional bir nəticədir — gəlin statsmodels ilə qurulmuş **OLS (Ordinary Least Squares)** regresiya modelinin nəticələrini müəllim kimi **bənd-bənd** izah edək 👨‍🏫

**📘 1. sm.OLS(y\_train, X\_train).fit()**

Bu sətrlə biz statsmodels paketində OLS (Yəni *Adi Kiçik Kvadratlar*) regresiya modelini qururuq.

**Diqqət:** Sən X\_train-ə intercept (sabit termin) **əlavə etməmisən**, yəni:

python

CopyEdit

X\_train = sm.add\_constant(X\_train)

yoxdur. Bu səbəbdən model.summary() hissəsində qeyd olunur:

**Note [1]: R² is computed without centering (uncentered) since the model does not contain a constant.**

**📊 2. Prediction Output**

plaintext

CopyEdit

prediction = [ 5.79 5.79 -28.78 23.60 -7.82 34.08]

Model X\_test-ə əsasən Height (boy) təxminləri verir. Amma **nəticələrdə mənfi dəyər** var — məsələn -28.78, bu isə **fiziki olaraq mümkün deyil**, və modelin düzgün öyrənmədiyini göstərir.

**📈 3. Summary — Ən vacib hissə**

**✔️ Dep. Variable: Height**

* Asılı dəyişən

**✔️ R-squared (uncentered): 0.012**

* Yəni model yalnız **1.2%** dəyişkənliyi izah edə bilir → Çox zəifdir

**✔️ Adj. R-squared (uncentered): -0.050**

* **Düzəldilmiş R² mənfidir** → Model sadəcə pis deyil, **faydasız**dır

**📉 4. Coefficient Cədvəli**

| **Dəyişən** | **coef (meil)** | **std err** | **p-value** | **95% etibar aralığı** |
| --- | --- | --- | --- | --- |
| x1 (Weight) | 17.298 | 39.138 | 0.664 | [-65.67, 100.27] |

**Açıqlama**:

* Bu coef deyir ki, **1 kq çəki artanda, ortalama boy 17.29 sm artır**.
* Amma p-value = 0.664 → Bu **statistik olaraq əhəmiyyətli deyil** (çünki > 0.05).
* Etibar aralığı çox genişdir → Modelin güvənilə bilən nəticə vermədiyini göstərir.

**🧪 5. F-statistic: 0.1953, Prob(F): 0.664**

* Modeldəki ümumi uyğunluq **əla deyil** → Prob(F) > 0.05 olduğu üçün **model statistik əhəmiyyət daşımır**

**❗ 6. Nə üçün bu nəticələr zəifdir?**

**Səbəblər:**

* X\_train-ə intercept əlavə olunmayıb  
  → **sm.add\_constant(X)** istifadə edilməlidir
* Dataset çox azdır: n = 17
* Mümkün digər faktorlar daxil edilməyib (yaş, cins, bədən quruluşu və s.)
* Modelin öyrəndiyi məlumatlar real trendi tutmur

**✅ Düzgün formada necə yazmaq olar?**

python

CopyEdit

import statsmodels.api as sm

# Constant əlavə edirik

X\_train\_const = sm.add\_constant(X\_train)

X\_test\_const = sm.add\_constant(X\_test)

model = sm.OLS(y\_train, X\_train\_const).fit()

prediction = model.predict(X\_test\_const)

print(model.summary())

Bu zaman:

* Intercept olacaq
* R² düzgün hesablanacaq (centered)
* Daha sağlam nəticə alacaqsan

**📌 Qısaca:**

| **Faktor** | **Hal-hazırkı vəziyyət** | **Tövsiyə** |
| --- | --- | --- |
| R² | 0.012 (çox zəif) | Sabit əlavə et, data artır |
| p-value | 0.664 | Əhəmiyyətli deyil |
| Intercept | yoxdur | sm.add\_constant() istifadə et |
| Prediction | mənfi dəyərlər var | model zəifdir |

İstəsən, bu izahı PowerPoint slaydlar və vizual analizlə birlikdə sənə göndərə bilərəm. Hazırlayım?