

# Regresie liniară

Ce ne așteaptă?

1. Regresie liniară simplă
2. Regresie liniară multiplă
3. Regresie polinomială
4. Regularizare Lasso, Ridge și ElasticNet

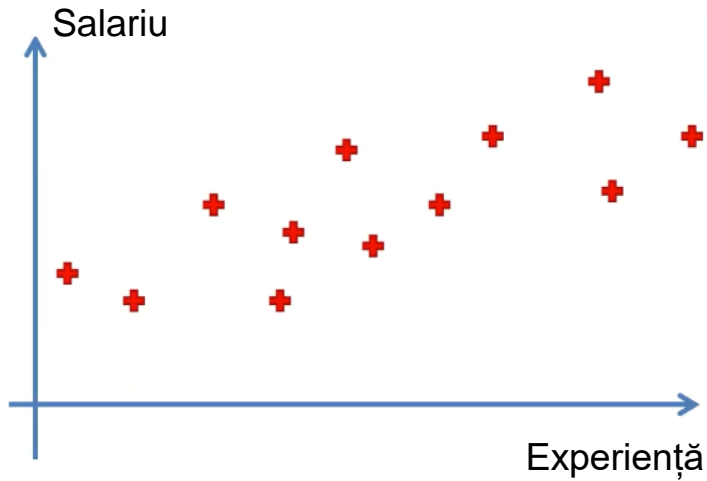
## 1. Regresie liniară simplă

- Datele dispun de o singura variabila independenta  $X$  și o variabila dependenta  $y$
- Ecuația de predicție a lui  $y$  pe baza lui  $x$ :

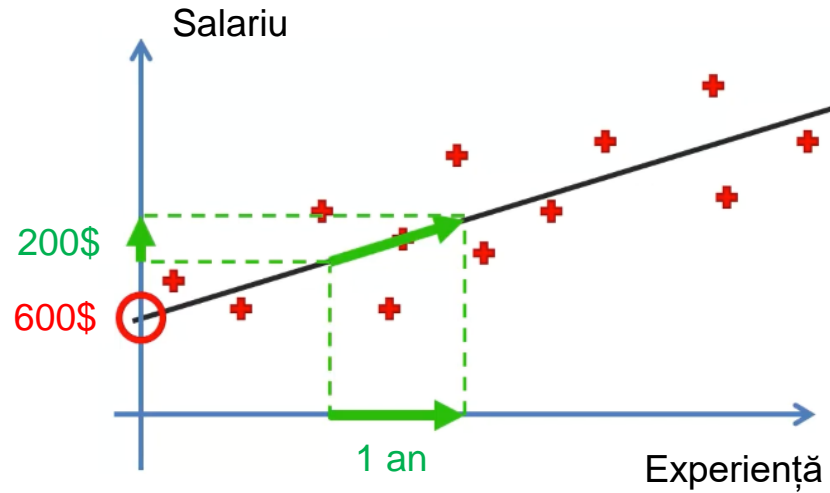
$$\hat{y} = \beta_0 + \beta_1 * x_1$$

- unde:
  - $\hat{y}$  - variabila dependenta prezisa
  - $x_1$  - variabila independenta de intrare
  - $\beta_1$  - coeficientul panta (slope) numit si coeficient al modelului
  - $\beta_0$  - coeficientul de intersecție (intercept) numit si constanta modelului

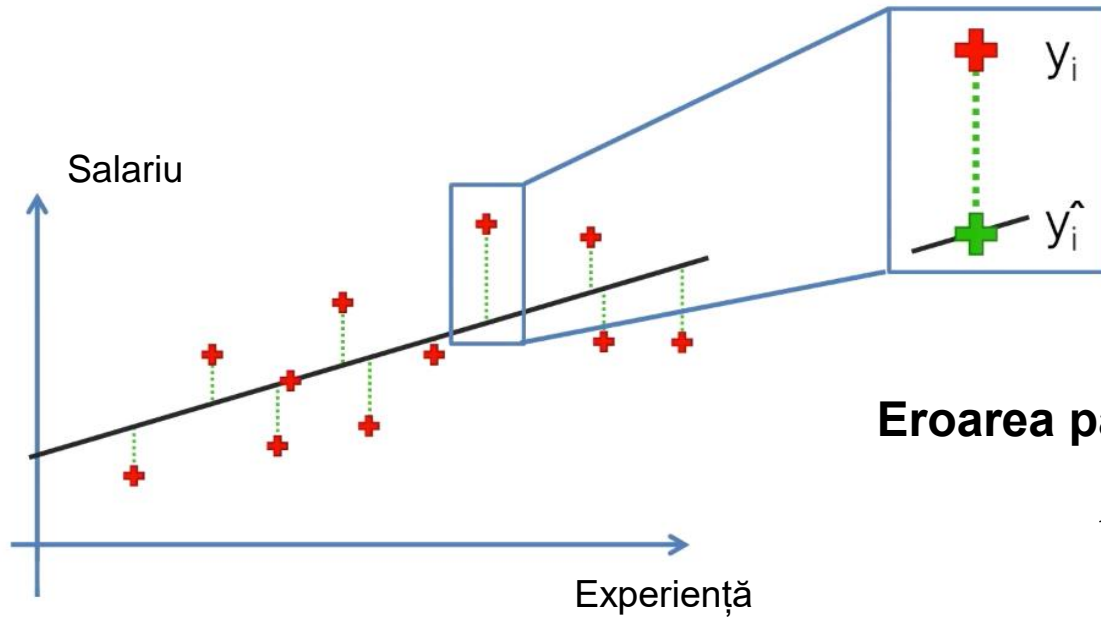
## Datele istorice



## Linia de predicție



$$\hat{y} = \beta_0 + \beta_1 * x_1 \quad \longrightarrow \quad \text{salariu} = \beta_0 + \beta_1 * \text{experienta}$$



**Eroarea pătratică (EP) a predicției  $i$**

$$EP = (y_i - \hat{y}_i)^2$$

- **Valorile coeficienților sunt determinate de valoarea minimă a sumei erorilor pătratice (SEP):**

$$SEP = \sum_i (y_i - \hat{y}_i)^2 \rightarrow \min$$

## 2. Regresie liniară multiplă

- Datele dispun de mai multe variabile independente  $X$  și o variabila dependentă  $y$
- Ecuația de predicție a lui  $y$  pe baza valorilor lui  $X$ :

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \dots + \beta_n * x_n$$

- unde:
  - $\hat{y}$  - variabila dependentă prezisă
  - $x_1, x_2, \dots, x_n$  - variabile independente de intrare
  - $\beta_1, \beta_2, \dots, \beta_n$  - coeficienții modelului
  - $\beta_0$  - constanta modelului

## Instrumente Scikit-Learn pentru regresia liniară

- Se importa clasa modelului LinearRegression  
`from sklearn.linear_model import LinearRegression`
- Se creează modelul ca o instanță a acestei clase cu parametri impliciți  
`model = LinearRegression()`
- Se realizează trainingul modelului pe datele de training  
`model.fit(X_train, y_train)`
- Se vizualizează coeficienți modelului  
`model.coef_`
- Se vizualizează constanta modelului  
`model.intercept_`
- Se realizează predicția pe datele de test  
`y_pred = model.predict(X_test)`
- Se determina eroare de predicție prin compararea valorii reale cu cea prezisă  
`er= y_test - y_pred`

### 3. Regresia polinomială

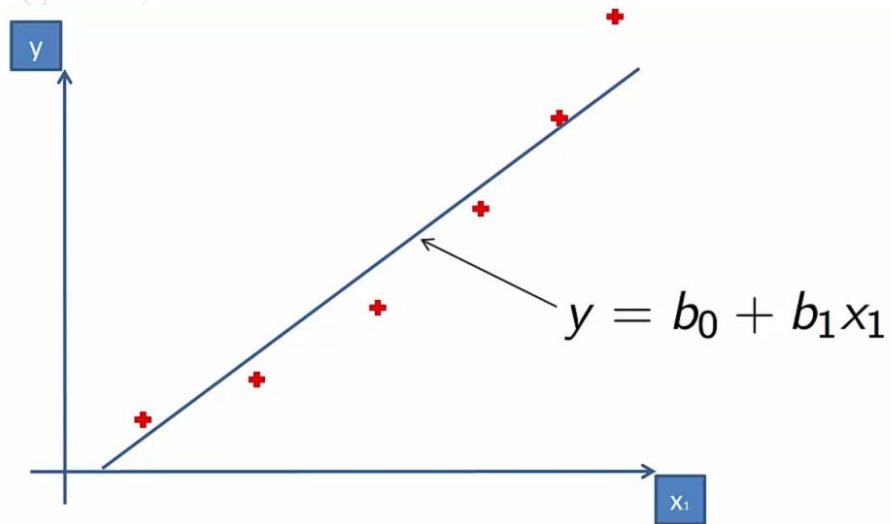
- Regresia polinomială permite:
  - Introducerea nelinialității pentru linia de predicția
  - Evaluarea interacțiunii variabilelor independente
- Ecuația generală de predicție a lui  $y$  pe baza unei singure valori  $x$ :

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_1^2 + \dots + \beta_d * x_1^d$$

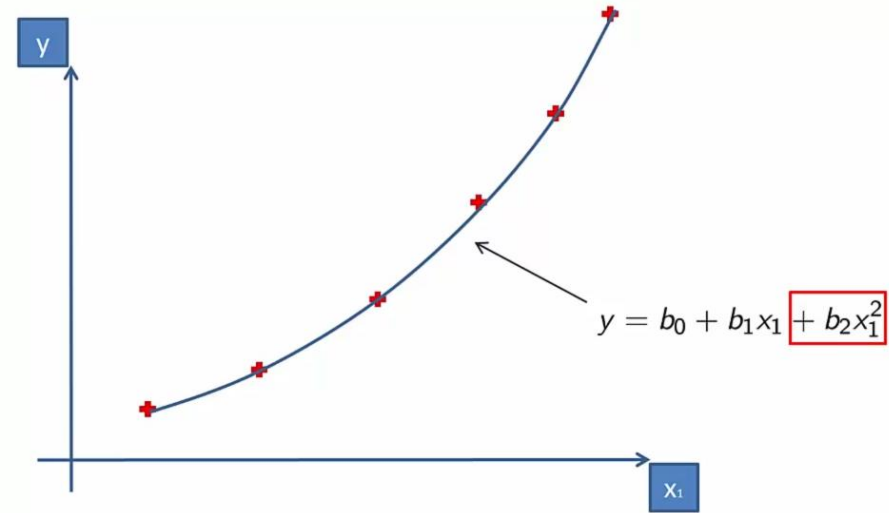
- unde:
  - $\hat{y}$  - variabila dependentă prezisă
  - $x_1$  - variabila independentă de intrare
  - $\beta_1, \beta_2, \dots, \beta_n$  - coeficienții modelului
  - $\beta_0$  - constanta modelului
  - $d$  - ordinul polinomului



## Regresie liniară simplă



## Regresie polinomială



- În cazul prezenței mai multor variabile independente se va realiza înmulțirea între acestea.
- Exemplu ecuație de predicție pentru 2 variabile independente  $x_1$  și  $x_2$  cu polinom de gradul  $d=2$

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_1^2 + \beta_4 * x_1 * x_2 + \beta_5 * x_2^2$$

- Variabilelor independente existente li se vor mai adăuga variabile polinomiale obținute prin combinarea acestora
- Variabile polinomiale adăugate se vor considera noi variabile independente și vor avea propriul coeficient

## Instrumente Scikit-Learn pentru regresie polinomiala

- Se importa clasa convertorului ce va forma variabilele polinomiale  
`from sklearn.preprocessing import PolynomialFeatures`
- Se creează convertorul cu specificare ordinului (degree) și prezența constantei (include\_bias)  
`polynomial_converter = PolynomialFeatures(degree=2, include_bias=False)`
- Se determina parametrii convertorului și apoi acesta se aplica asupra datelor  
`poly_features = polynomial_converter.fit_transform(X)`
- Se vizualizează forma noilor date  
`poly_features.shape`
- Se formează setul de training și setul de test conform datelor convertite  
`X_train, X_test, y_train, y_test = train_test_split(poly_features, y, test_size=0.3, random_state=9)`
- Se urmează pașii creării modelului ca în cazul regresiei liniare cu datele convertite  
`from sklearn.linear_model import LinearRegression`  
`model = LinearRegression()`  
`model.fit(X_train, y_train)`  
`model.coef_`  
`y_pred = model.predict(X_test)`

## 4. Regularizare Lasso, Ridge și ElasticNet

- Regularizarea presupune includerea unei penalități în suma erorilor pătratice (SEP) ce determina valorile coeficienților  $\beta_j$
- Regularizarea permite excluderea efectului de overfitting a modelului
- Regularizarea L1 numită și regularizarea Lasso presupune includerea unei penalități determinată de valorile absolute ale coeficienților  $\beta_j$

$$\sum_i (y_i - \hat{y}_i)^2 + alpha * \sum_j |\beta_j|$$

- $alpha$  – hiperparametru ce determina gradul de includere a penalității ( $alpha = 0 \div +\infty$ )
- Modelul Lasso permite selecția celor mai importante caracteristici atribuind un coeficient  $\beta_j = 0$  pentru caracteristicile neesențiale

- Regularizarea L2 numită și regularizarea Ridge presupune includerea unei penalități determinată de valorile pătratice ale coeficienților  $\beta_j$

$$\sum_i (y_i - \hat{y}_i)^2 + \alpha * \sum_j \beta_j^2$$

- $\alpha$  – hiperparametru ce determina gradul de includere a penalității ( $\alpha = 0 \div +\infty$ )
- Regularizarea ElasticNet include atât regularizarea Lasso cat și Ridge

$$\sum_i (y_i - \hat{y}_i)^2 + \alpha * \left( \frac{1 - l_1\_ratio}{2} \sum_j \beta_j^2 + l_1\_ratio * \sum_j |\beta_j| \right)$$

- $\alpha$  – hiperparametru ce determina gradul de includere a penalității ( $\alpha = 0 \div +\infty$ )
- $l_1\_ratio$  – hiperparametru ce determina gradul de includere a regularizării L1

## Instrumente Scikit-Learn pentru regresie cu regularizare

- Se adauga caracteristicile polinomiale

```
from sklearn.preprocessing import PolynomialFeatures  
polynomial_converter = PolynomialFeatures(degree=3,  
include_bias=False)  
poly_features = polynomial_converter.fit_transform(X)
```

- Se formeaza setul de training si de test

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test =  
train_test_split(poly_features, y, test_size=0.3, random_state=9)
```

- Se scaleaza caracteristicile datelor de training si de test

```
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
X_train = scaler.fit_transform(X_train)  
X_test = scaler.transform(X_test)
```

## ■ Regularizare Lasso cu parametrul alpha fix

- Se importa clasa algoritmului Lasso

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

- Se creează un model cu fixarea valorii parametrului alpha

```
lasso_model = Lasso(alpha=1)
```

- Se realizează trainingul modelului pe datele de training

```
lasso_model.fit(X_train, y_train)
```

- Se vizualizează coeficienții modelului

```
lasso_model.coef_
```

- Se realizează predicția pe datele de test

```
y_pred = lasso_model.predict(X_test)
```

## ■ Regularizare Lasso cu parametrul alpha reglabil

- Se importa clasa algoritmului LassoCV

```
from sklearn.linear_model import LassoCV
```

- Se creează un model cu specificare diferitor valori a parametrului alpha

```
lasso_cv_model = LassoCV(eps=0.001, n_alphas=100, cv=5,  
max_iter=1000000)
```

- Se realizează trainingul modelului pe datele de training

```
lasso_cv_model.fit(X_train, y_train)
```

- Se vizualizează parametrul alpha selectat de model

```
lasso_cv_model.alpha_
```

- Se vizualizează coeficienții modelului

```
lasso_cv_model.coef_
```

- Se realizează predicția pe datele de test

```
y_pred = lasso_cv_model.predict(X_test)
```



## ■ Regularizare Ridge cu parametrul alpha fix

- Se importa clasa algoritmului Ridge

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

- Se creează un model cu fixarea valorii parametrului alpha

```
ridge_mode = Ridge(alpha=10)
```

- Se realizează trainingul modelului pe datele de training

```
ridge_mode.fit(X_train, y_train)
```

- Se vizualizează coeficienții modelului

```
ridge_mode.coef_
```

- Se realizează predicția pe datele de test

```
y_pred = ridge_mode.predict(X_test)
```

## ■ Regularizare Ridge cu parametrul alpha reglabil

- Se importa clasa algoritmului RidgeCV

```
from sklearn.linear_model import RidgeCV
```

- Se creează un model cu specificare diferitor valori a parametrului alpha

```
ridge_CV_model = RidgeCV(alphas=(0.1, 1.0, 10), scoring =  
'neg_mean_absolute_error')
```

- Se realizează trainingul modelului pe datele de training

```
ridge_CV_model.fit(X_train, y_train)
```

- Se vizualizează parametrul alpha selectat de model

```
ridge_CV_model.alpha_
```

- Se vizualizează coeficienții modelului

```
ridge_CV_model.coef_
```

- Se realizează predicția pe datele de test

```
y_pred = ridge_CV_model.predict(X_test)
```

■ **Regularizare ElasticNet cu parametri alpha și l1\_ratio ficși**

- Se importa clasa algoritmului ElasticNet

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

- Se creează un model cu fixarea valorilor parametrilor alpha și l1\_ratio

```
elasticnet_model = ElasticNet(alpha=1.0, l1_ratio=0.5)
```

- Se realizează trainingul modelului pe datele de training

```
elasticnet_model.fit(X_train, y_train)
```

- Se vizualizează coeficienții modelului

```
elasticnet_model.coef_
```

- Se realizează predicția pe datele de test

```
y_pred = elasticnet_model.predict(X_test)
```

## ■ Regularizare ElasticNet cu parametrii alpha și l1\_ratio reglabili

- Se importa clasa algoritmului ElasticNetCV

```
from sklearn.linear_model import ElasticNetCV
```

- Se creează un model cu specificare diferitor valori a parametrilor alpha și l1\_ratio

```
elasticnet_CV_model=ElasticNetCV(l1_ratio=[0.1, 0.5, 0.7, 0.9, 0.95, 0.99, 1],  
eps=0.001, n_alphas=100, cv=5, max_iter=1000000)
```

- Se realizează trainingul modelului pe datele de training

```
elasticnet_CV_model.fit(X_train, y_train)
```

- Se vizualizează parametrul alpha selectat de model

```
elasticnet_CV_model.alpha_
```

- Se vizualizează parametrul l1\_ratio selectat de model

```
elasticnet_CV_model.l1_ratio_
```

- Se vizualizează coeficienții modelului

```
elasticnet_CV_model.coef_
```

- Se realizează predicția pe datele de test

```
y_pred = elasticnet_CV_model.predict(X_test)
```