¿Qué es “Machine Learning”?

Definición simple:

Machine Learning = Enseñar a las computadoras a **aprender patrones** de datos para hacer **predicciones**

Como funciona el cerebro humano:

- **Ves** muchos ejemplos
- **Detectas** patrones
- **Haces** predicciones sobre cosas nuevas

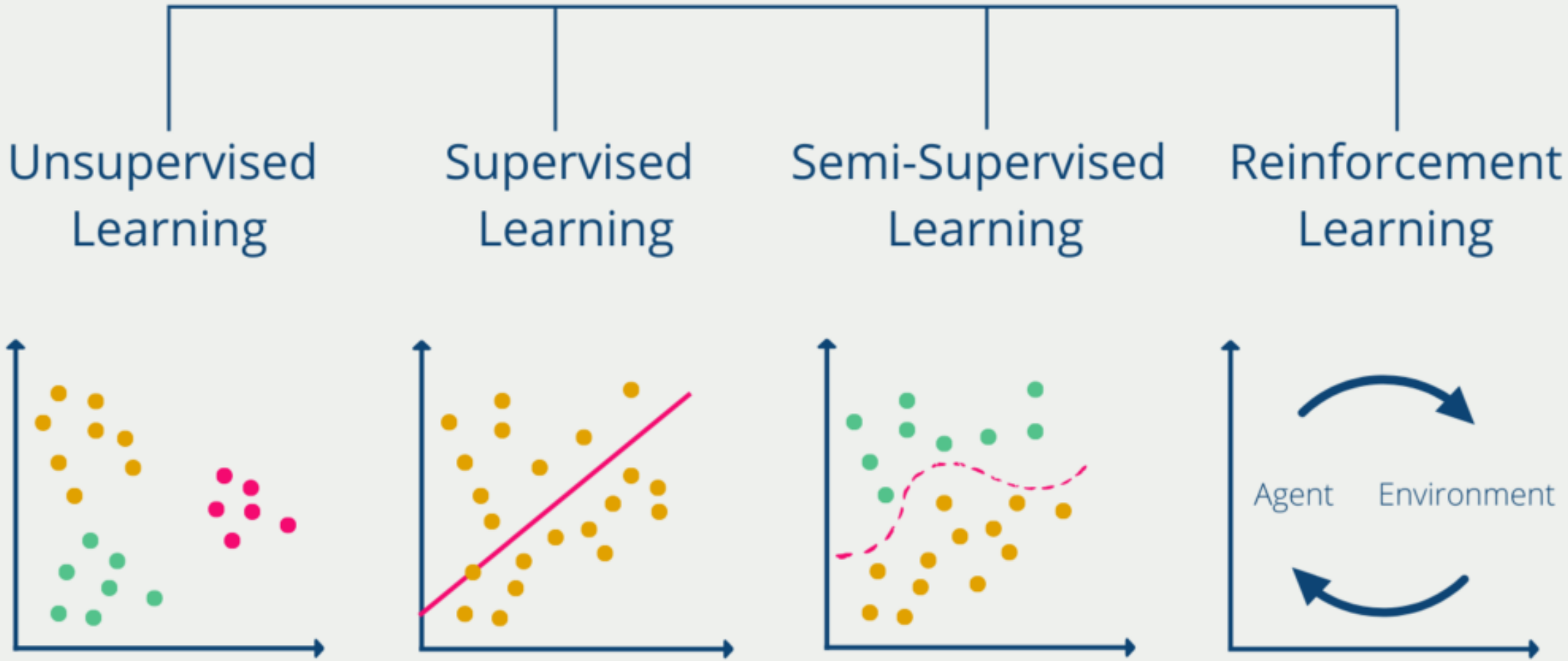
Ejemplo: Después de ver 100 gatos, reconoces un gato nuevo

Como funciona Machine Learning

1. Alimentas datos al algoritmo
2. Encuentra patrones matemáticos
3. Predice en datos nuevos

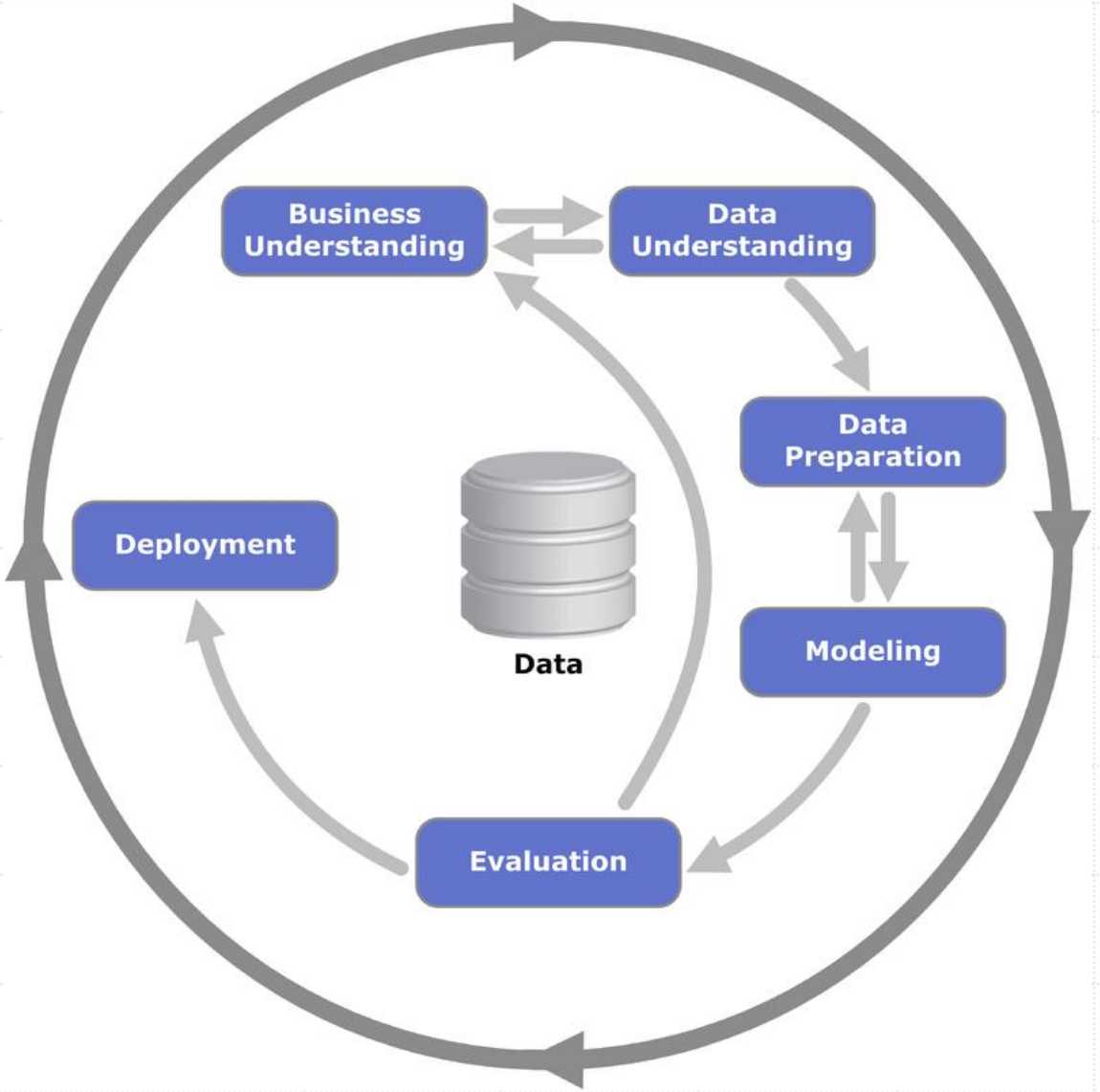
Ejemplo: Después de ver 1000 casas vendidas, predice precio de casa nueva

Machine Learning



Proceso CRISP-DM

- Comprensión del negocio
- Comprensión de Datos
- Preparación de datos
- Modelado
- Evaluación
- Despliegue



Entendimiento del negocio

El hundimiento del Titanic es uno de los naufragios más infames de la historia. El 15 de abril de 1912, durante su viaje inaugural, el RMS Titanic —considerado ampliamente como “insumergible”— se hundió tras chocar con un iceberg.

Lamentablemente, no había suficientes botes salvavidas para todos a bordo, lo que resultó en la muerte de 1502 de los 2224 pasajeros y tripulantes.

Si bien hubo un componente de suerte en la supervivencia, parece que algunos grupos de personas tenían más probabilidades de sobrevivir que otros.

EDA

```
[ ] train.head(3)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

EDA

```
[ ] train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

EDA

```
[ ]  train.describe(include='all').T
```

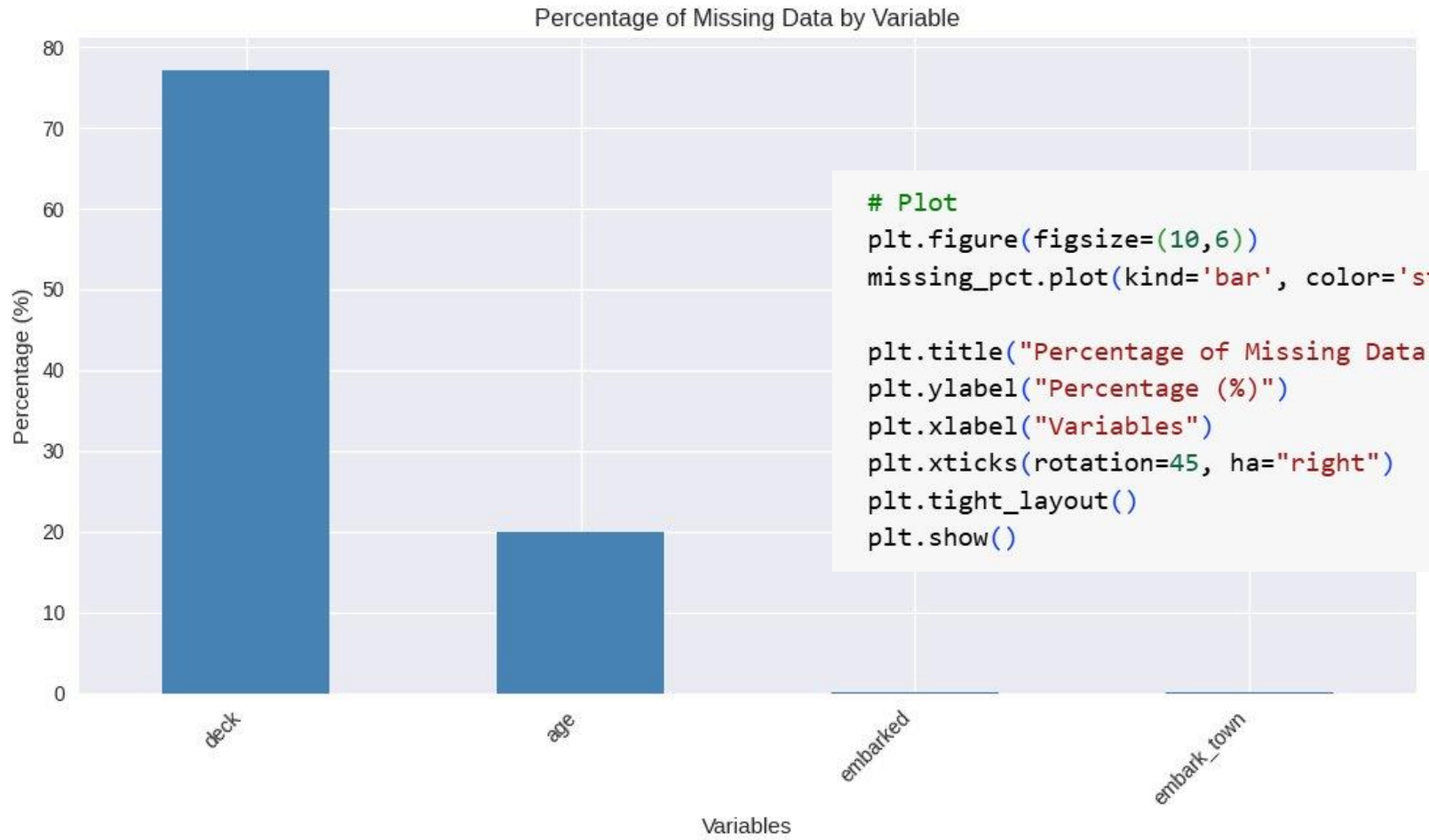
	count	unique	top	freq	mean	std	min	25%	50%	75%	max
PassengerId	891.0	NaN	NaN	NaN	446.0	257.353842	1.0	223.5	446.0	668.5	891.0
Survived	891.0	NaN	NaN	NaN	0.383838	0.486592	0.0	0.0	0.0	1.0	1.0
Pclass	891.0	NaN	NaN	NaN	2.308642	0.836071	1.0	2.0	3.0	3.0	3.0
Name	891	891	Dooley, Mr. Patrick	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Sex	891	2	male	577	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Age	714.0	NaN	NaN	NaN	29.699118	14.526497	0.42	20.125	28.0	38.0	80.0
SibSp	891.0	NaN	NaN	NaN	0.523008	1.102743	0.0	0.0	0.0	1.0	8.0
Parch	891.0	NaN	NaN	NaN	0.381594	0.806057	0.0	0.0	0.0	0.0	6.0
Ticket	891	681	347082	7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Fare	891.0	NaN	NaN	NaN	32.204208	49.693429	0.0	7.9104	14.4542	31.0	512.3292
Cabin	204	147	G6	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Embarked	889	3	S	644	NaN	NaN	NaN	NaN	NaN	NaN	NaN

EDA

```
[ ] train.isna().sum().sort_values(ascending=False)
```

	0
Cabin	687
Age	177
Embarked	2
PassengerId	0
Name	0
Pclass	0
Survived	0
Sex	0
Parch	0
SibSp	0
Fare	0
Ticket	0

dtype: int64




```
# Plot
plt.figure(figsize=(10,6))
missing_pct.plot(kind='bar', color='steelblue')

plt.title("Percentage of Missing Data by Variable")
plt.ylabel("Percentage (%)")
plt.xlabel("Variables")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```

EDA

```
[ ]  train['Survived'].value_counts(normalize=True)
```

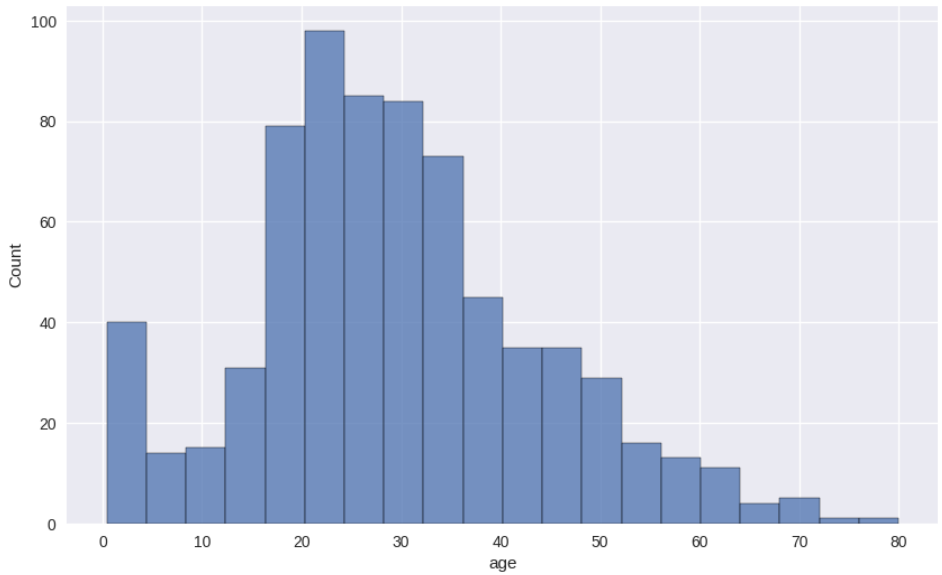
Survived	proportion
0	0.616162
1	0.383838

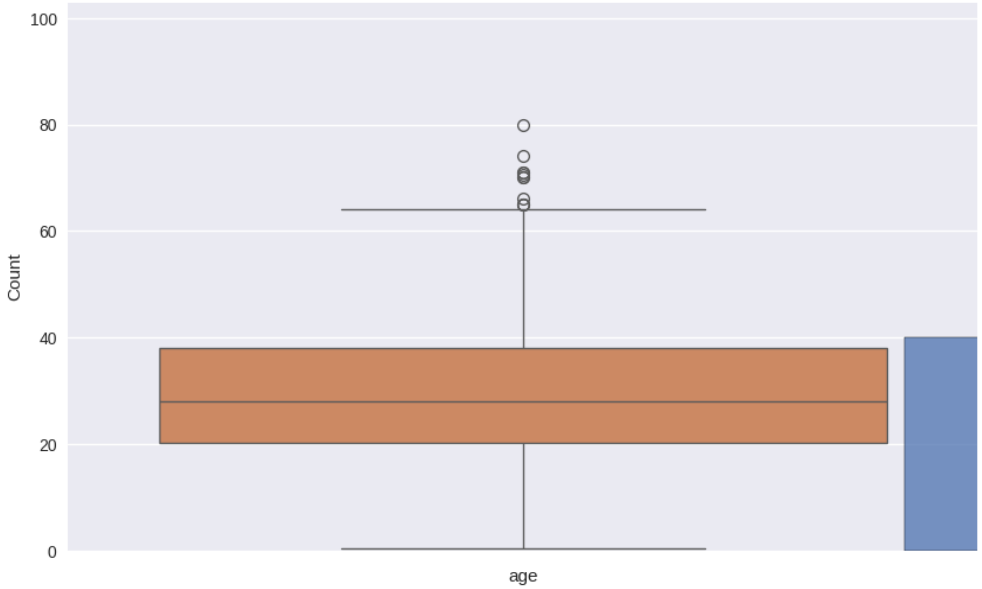
dtype: float64

EDA Visualizaciones

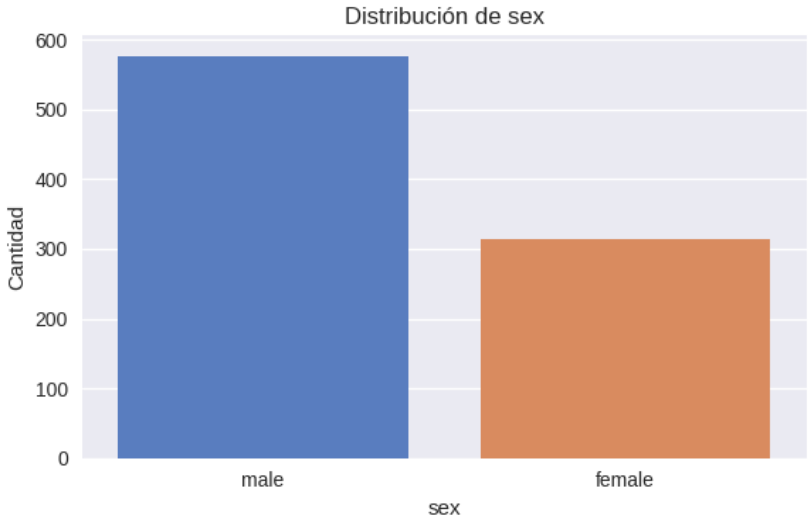
```python
# Representaciones enriquecidas
sns.histplot(df['age'])  # Distribución completa
sns.boxplot(df['age'])   # Detectar outliers
df['age'].describe()     # + Resumen numérico
```

	age
count	714.000000
mean	29.699118
std	14.526497
min	0.420000
25%	20.125000
50%	28.000000
75%	38.000000
max	80.000000





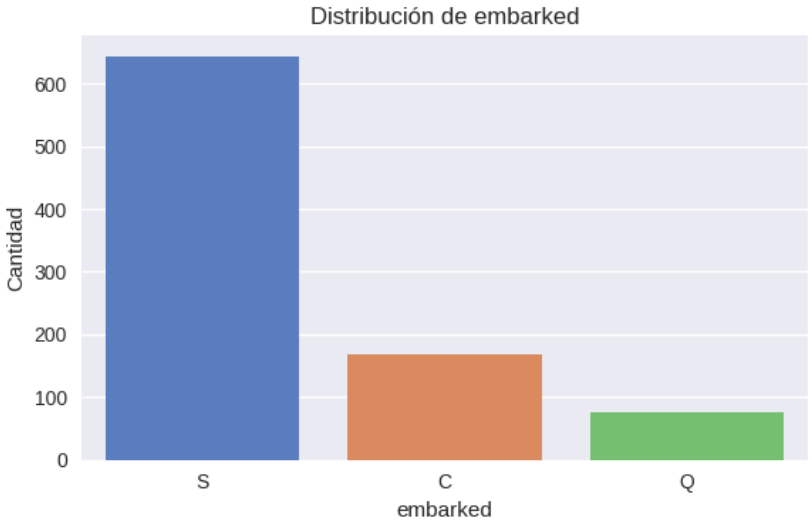
EDA Visualizaciones

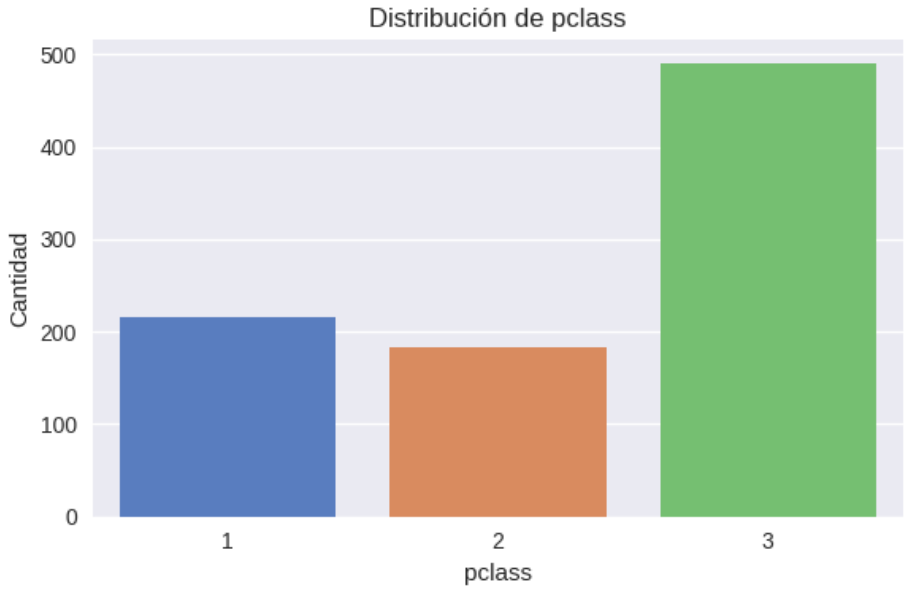



```python
import matplotlib.pyplot as plt
import seaborn as sns

categoricals = ['sex', 'pclass', 'embarked']

for col in categoricals:
    plt.figure(figsize=(6,4))
    sns.countplot(x=col, data=df, palette="muted")
    plt.title(f"Distribución de {col}")
    plt.ylabel("Cantidad")
    plt.tight_layout()
    plt.show()
```





EDA Visualizaciones


```
plt.figure(figsize=(6,4))
sns.countplot(x="pclass", hue="sex", data=df, palette="Set2")
plt.title("Distribución combinada: Clase vs Sexo")
plt.tight_layout()
plt.show()
```

Limpieza

```
# 🚫 PASO 1: Manejar valores faltantes (imputación)
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])  # Valor más común
df['Fare'] = df['Fare'].fillna(df['Fare'].median())               # Mediana
df['Age'] = df['Age'].fillna(df.groupby(['Sex','Pclass'])['Age'].transform('median'))
```

Feature Engineering

```
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1
df['IsAlone'] = (df['FamilySize'] == 1).astype(int)

df['Title'] = df['Name'].str.extract(',\s*([^\.]+)\.')
rare_titles = df['Title'].value_counts()[df['Title'].value_counts() < 10].index
df['Title'] = df['Title'].replace(rare_titles, 'Rare')
```

Features finales

```
# 🔄 PASO 3: Preparar datos para el modelo
features = ['Pclass','Sex','Age','Fare','Embarked','FamilySize','IsAlone','Title','SibSp','Parch']
X = df[features].copy()
y = df['Survived']

X.shape, y.shape
```

```
((891, 10), (891,))
```

Train/Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

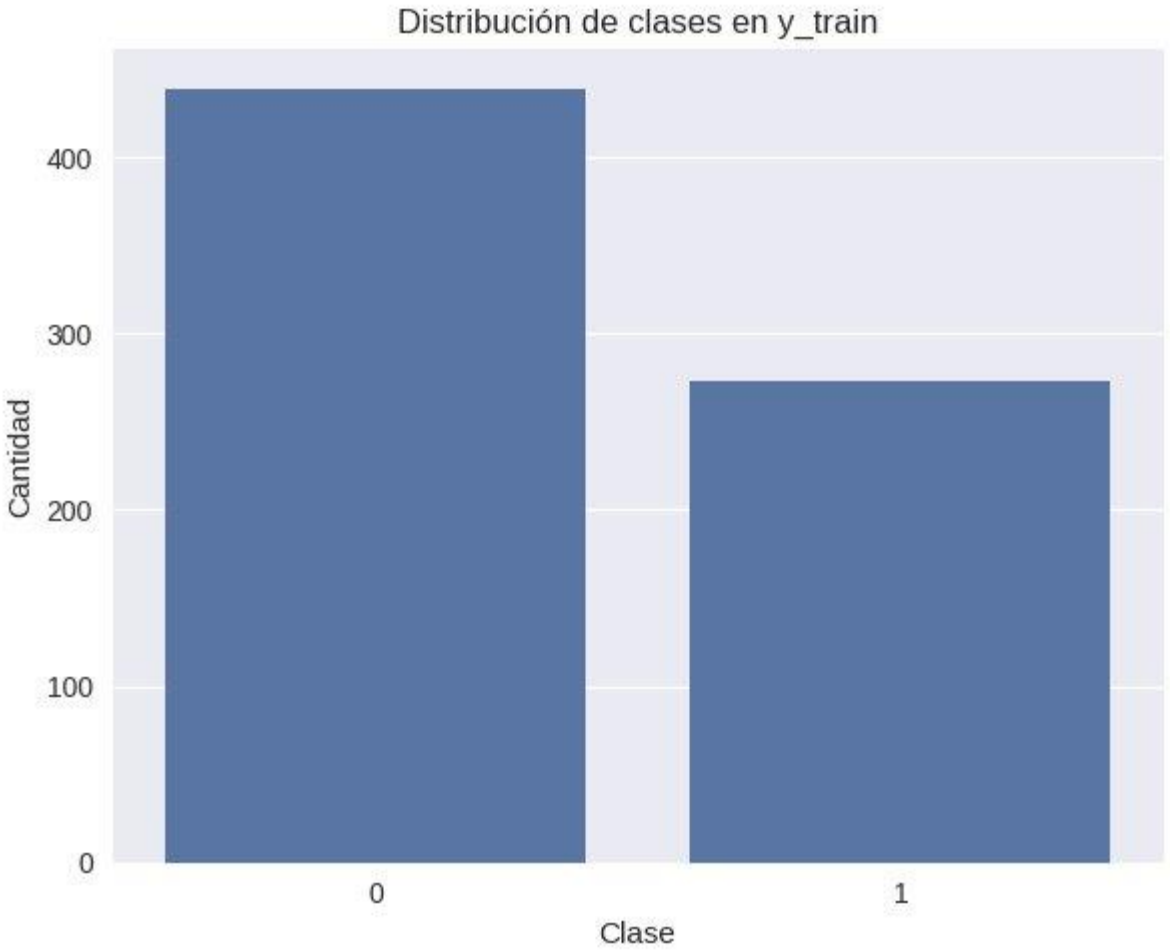
- **Train test split: Funcion que en base a un X, y inicial lo divide en dos (train y test), tomando en cuenta los parametros:**
- **X = Features ; y = variable objetivo/salida**
- **test size = porcentaje del dataset de test**
- **random state = semilla aleatoria**
- **stratify = forma de reordenamiento aleatorio en base a la salida.**

Train/Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

- **891 datos iniciales en X,y**
- **Train queda con: 712**
- **Test queeda con: 179**

Train/Test Split



Metricas

```
print('Baseline acc:', accuracy_score(y_test, baseline_pred))
print('LogReg acc  :', accuracy_score(y_test, pred))

print('\nClassification report (LogReg):')
print(classification_report(y_test, pred))

print('\nConfusion matrix (LogReg):')
print(confusion_matrix(y_test, pred))
```

```
Baseline acc: 0.6145251396648045
LogReg acc  : 0.8156424581005587

Classification report (LogReg):
              precision    recall  f1-score   support

           0       0.82      0.89      0.86       110
           1       0.80      0.70      0.74        69

    accuracy                           0.82       179
   macro avg       0.81      0.79      0.80       179
weighted avg       0.81      0.82      0.81       179


Confusion matrix (LogReg):
[[98 12]
 [21 48]]
```

Metricas – Clasificacion - Accuracy

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{correct predictions} + \text{incorrect predictions}}$$

$$\text{Accuracy} = \frac{40}{40 + 10} = 80\% \qquad \text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- TP es la cantidad de verdaderos positivos (predicciones correctas).
- TN es la cantidad de verdaderos negativos (predicciones correctas).
- FP es la cantidad de falsos positivos (predicciones incorrectas).
- FN es la cantidad de falsos negativos (predicciones incorrectas).

https://developers.google.com/machine-learning/glossary

Metricas - Clasificacion -Precision

- Cuando el modelo predijo la clase positiva, ¿qué porcentaje de las predicciones fueron correctas?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{Precision} = \frac{150}{150 + 50} = 0.75$$

https://developers.google.com/machine-learning/glossary

Metricas - Clasificacion -Recall

- Cuando la verdad fundamental era la clase positiva, ¿qué porcentaje de predicciones identificó correctamente el modelo como la clase positiva?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\text{Recall} = \frac{180}{180 + 20} = 0.9$$

https://developers.google.com/machine-learning/glossary

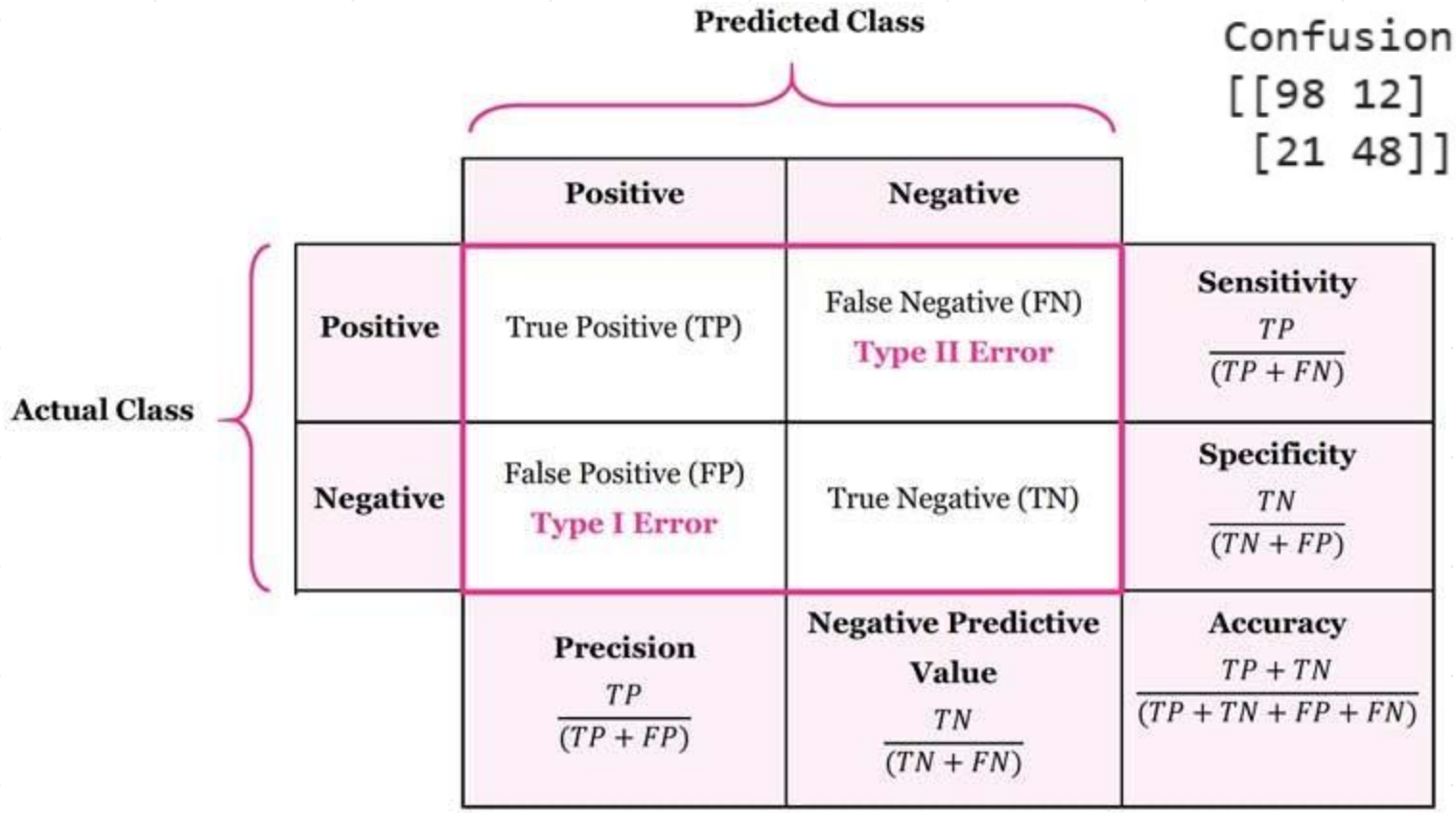
Metricas – Clasificacion -F1 Score

$$F_1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

https://developers.google.com/machine-learning/glossary

Metricas – Matriz de confusion

```
Confusion matrix (LogReg):
[[98 12]
 [21 48]]
```



https://encord.com/glossary/confusion-matrix/

Metricas – Regresiones

- Suma de los errores al cuadrado $\mathrm{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2$

- Root mean square error $\mathrm{RSE} = \sqrt{\frac{1}{n-2}\mathrm{RSS}}$

- Coeficiente R^2: indica la proporción de la varianza que es explicada por el modelo (0: no 1: máximo)

$$R^2 = \frac{\mathrm{TSS} - \mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}}$$

$$\mathrm{TSS} = \sum (y_i - \bar{y})^2$$

- Error absoluto medio (MAE)

$$MAE = \frac{1}{n} \Sigma \left| y - \hat{y} \right|$$

Divide by the total number of data points

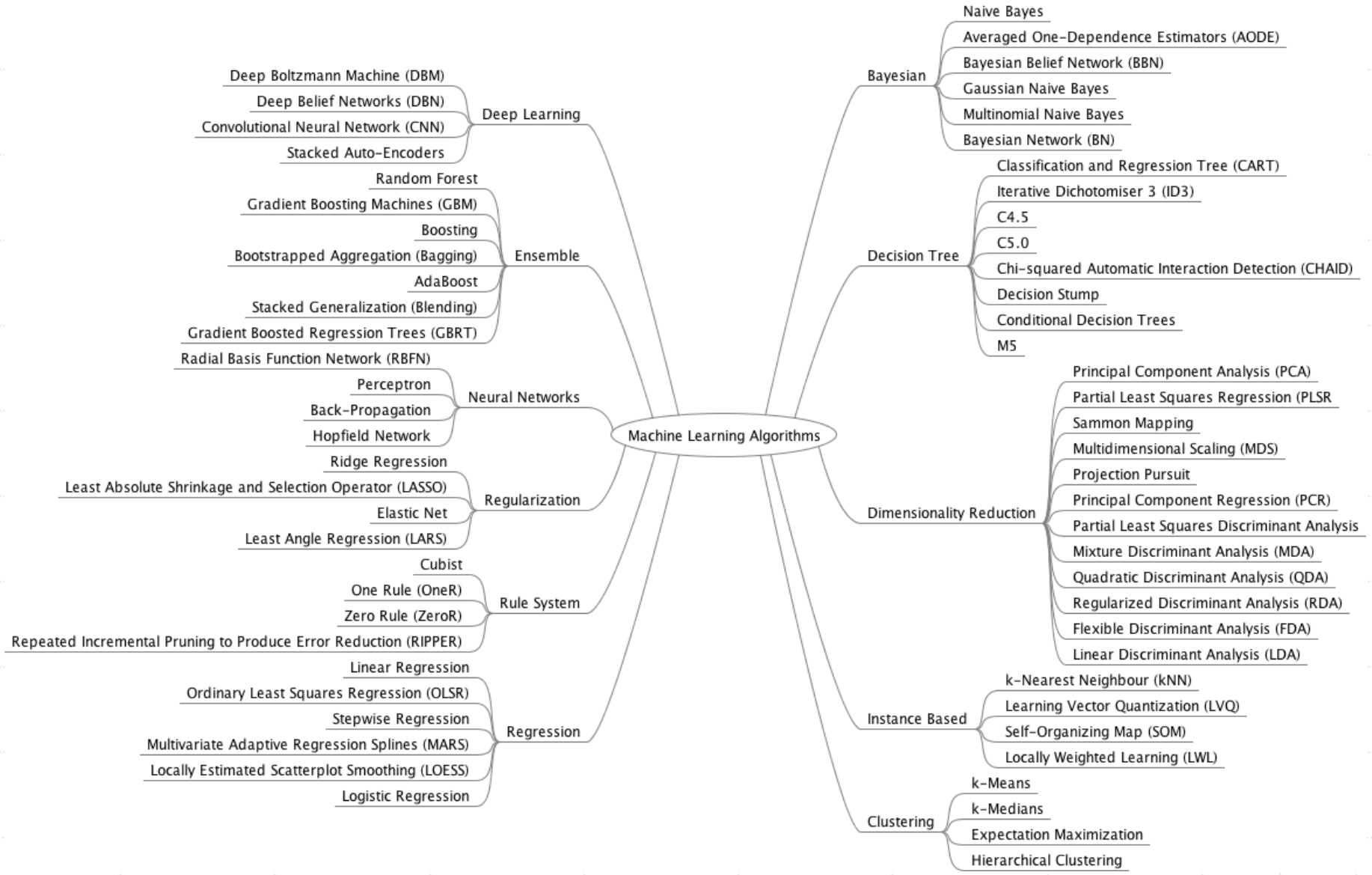
Predicted output value

Actual output value

Sum of

The absolute value of the residual

Modelos

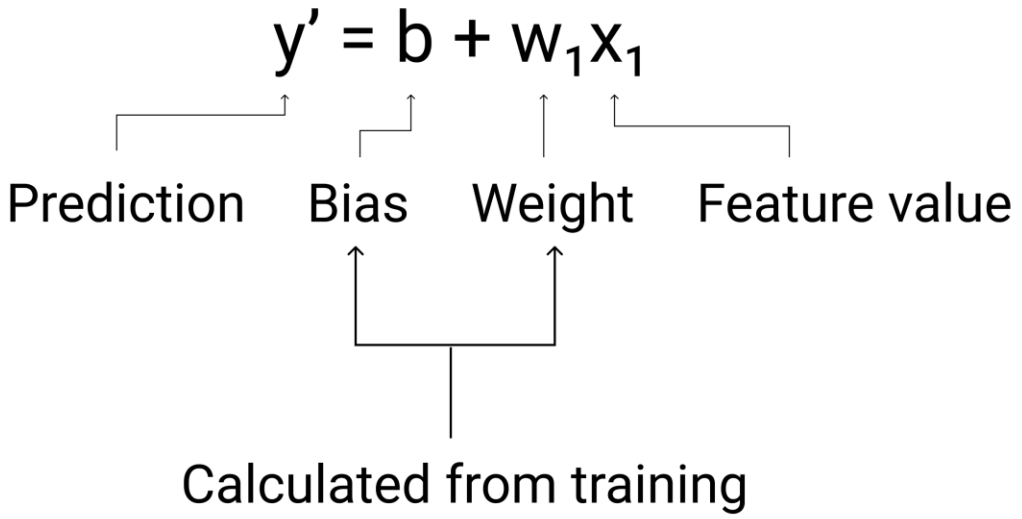

Machine Learning Algorithms
Deep Learning
Deep Boltzmann Machine (DBM)
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders
Ensemble
Random Forest
Gradient Boosting Machines (GBM)
Boosting
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (Blending)
Gradient Boosted Regression Trees (GBRT)
Neural Networks
Radial Basis Function Network (RBFN)
Perceptron
Back-Propagation
Hopfield Network
Regularization
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)
Elastic Net
Least Angle Regression (LARS)
Rule System
Cubist
One Rule (OneR)
Zero Rule (ZeroR)
Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
Regression
Linear Regression
Ordinary Least Squares Regression (OLSR)
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)
Logistic Regression
Bayesian
Naive Bayes
Averaged One-Dependence Estimators (AODE)
Bayesian Belief Network (BBN)
Gaussian Naive Bayes
Multinomial Naive Bayes
Bayesian Network (BN)
Decision Tree
Classification and Regression Tree (CART)
Iterative Dichotomiser 3 (ID3)
C4.5
C5.0
Chi-squared Automatic Interaction Detection (CHAID)
Decision Stump
Conditional Decision Trees
M5
Dimensionality Reduction
Principal Component Analysis (PCA)
Partial Least Squares Regression (PLSR)
Sammon Mapping
Multidimensional Scaling (MDS)
Projection Pursuit
Principal Component Regression (PCR)
Partial Least Squares Discriminant Analysis
Mixture Discriminant Analysis (MDA)
Quadratic Discriminant Analysis (QDA)
Regularized Discriminant Analysis (RDA)
Flexible Discriminant Analysis (FDA)
Linear Discriminant Analysis (LDA)
Instance Based
k-Nearest Neighbour (kNN)
Learning Vector Quantization (LVQ)
Self-Organizing Map (SOM)
Locally Weighted Learning (LWL)
Clustering
k-Means
k-Medians
Expectation Maximization
Hierarchical Clustering

Modelos – Regresión Lineal

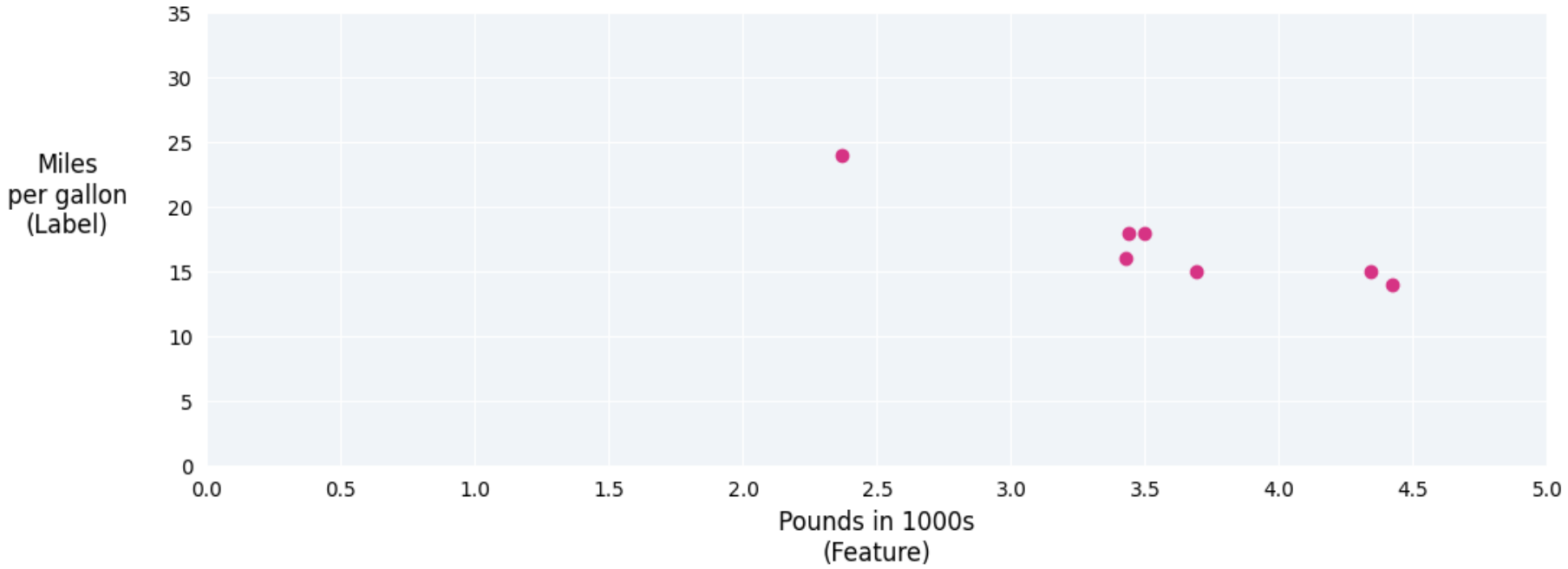
- El modelo es un modelo lineal.
- La predicción es un valor de punto flotante.

$$y = mx + n$$

- Y es la salida
- X es la entrada
- M y n (b y w1) son los pesos a entrenar

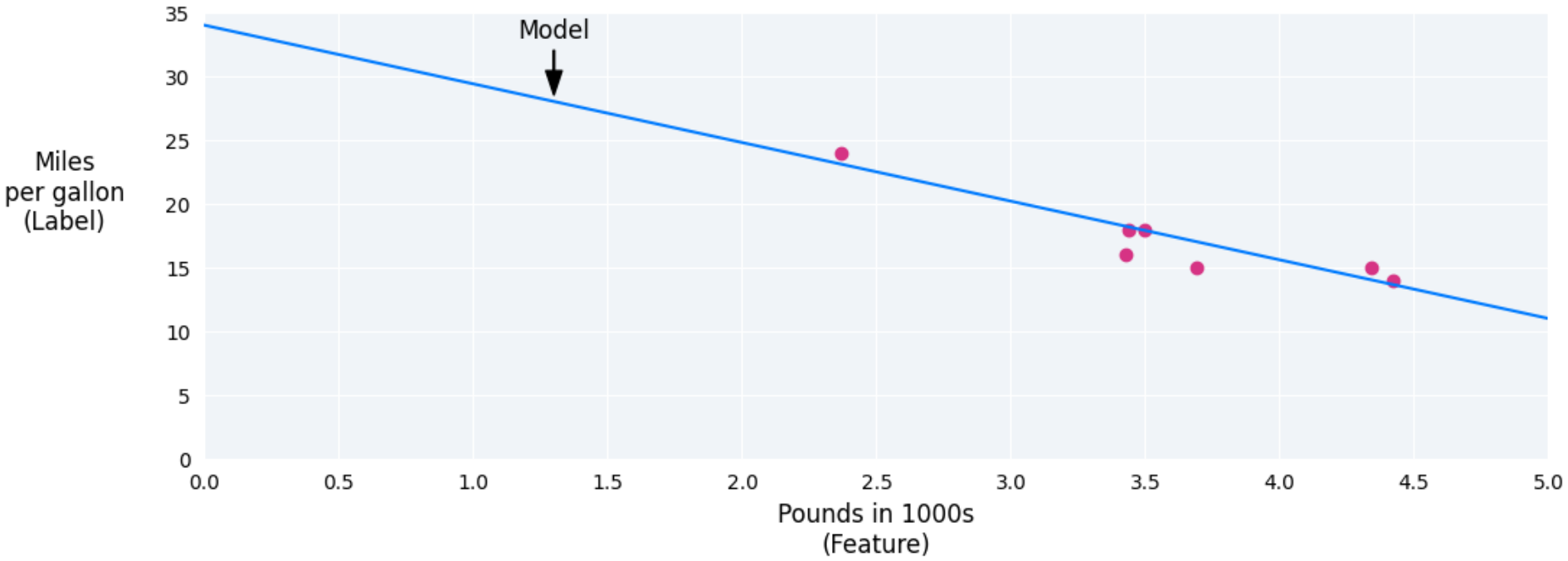


Modelos – Regresión Lineal



https://developers.google.com/machine-learning/crash-course/linear-regression

Modelos – Regresión Lineal



https://developers.google.com/machine-learning/crash-course/linear-regression

Modelos – Regresión Lineal

$$y' = b + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5$$

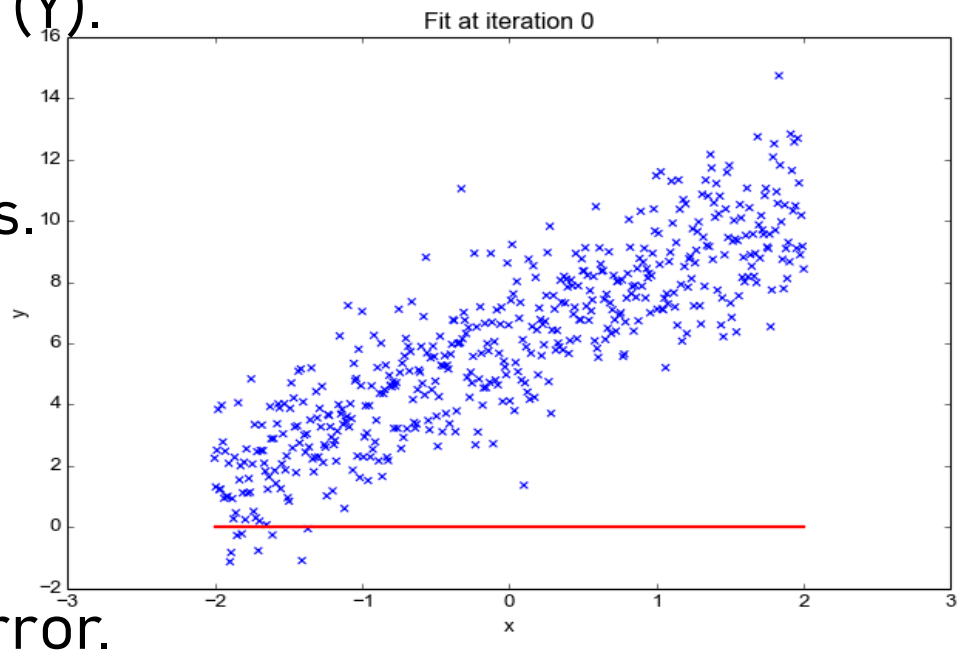
Pounds Displacement Acceleration Number of cylinders Horsepower

https://developers.google.com/machine-learning/crash-course/linear-regression

Modelos – Regresión Lineal

Entrenamiento:

1. **Datos** → ejemplos con entradas (X) y salidas (Y).
2. **Modelo** → fórmula con pesos
3. **Predice** → calcula valores con pesos iniciales.
4. **Error** → compara predicción vs. valor real.
5. **Ajusta** → corrige pesos según el error.
6. **Repite** → predecir → medir → ajustar.
7. **Final** → pesos “aprendidos” que minimizan error.

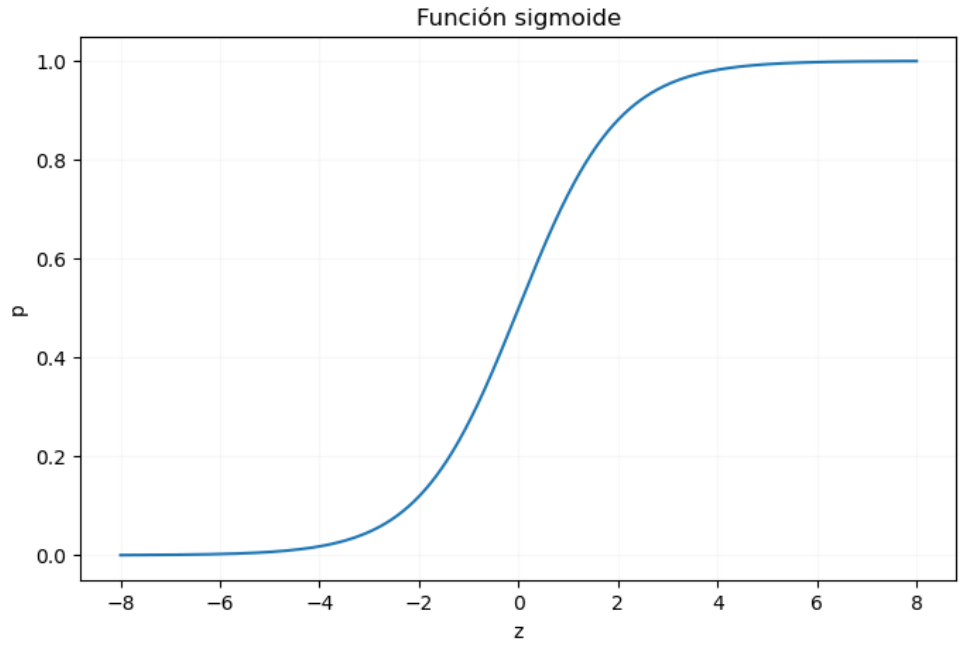


Modelos – Regresión Logistica

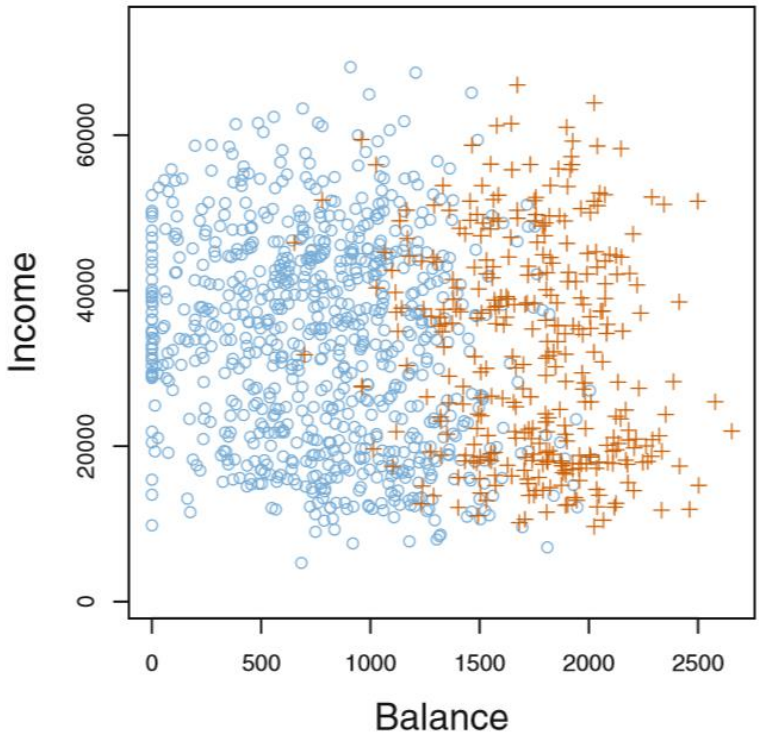
- El modelo es un modelo lineal con función sigmoide.
- La predicción es una **probabilidad** entre 0 y 1.

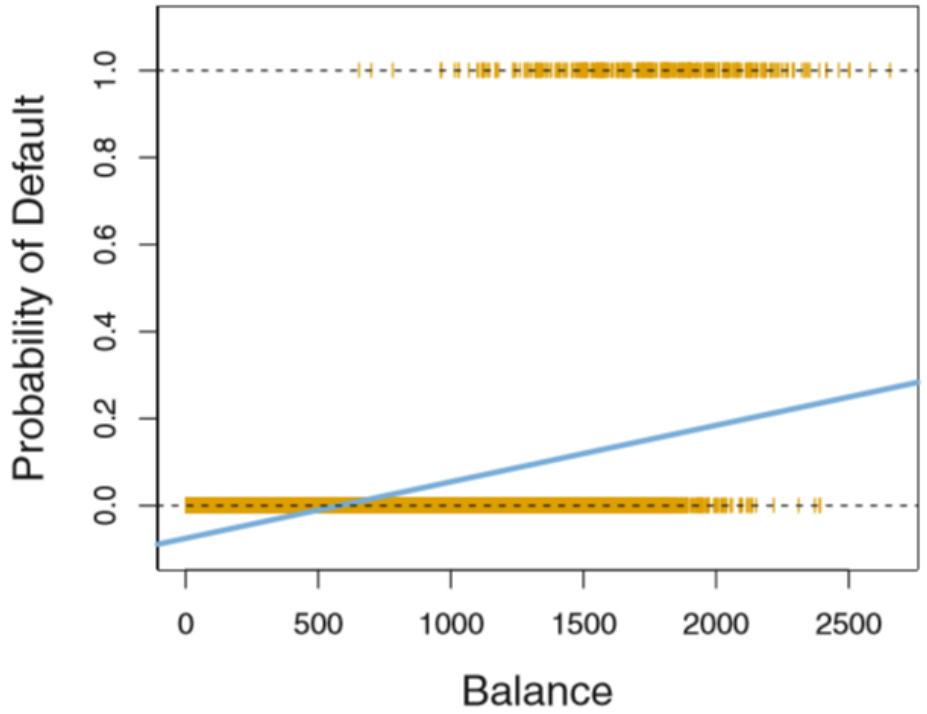
$$p = \frac{1}{1 + e^{-(mx+n)}}$$

- p es la probabilidad de que Y = 1.
- X es la entrada.
- m y n son los pesos a entrenar.

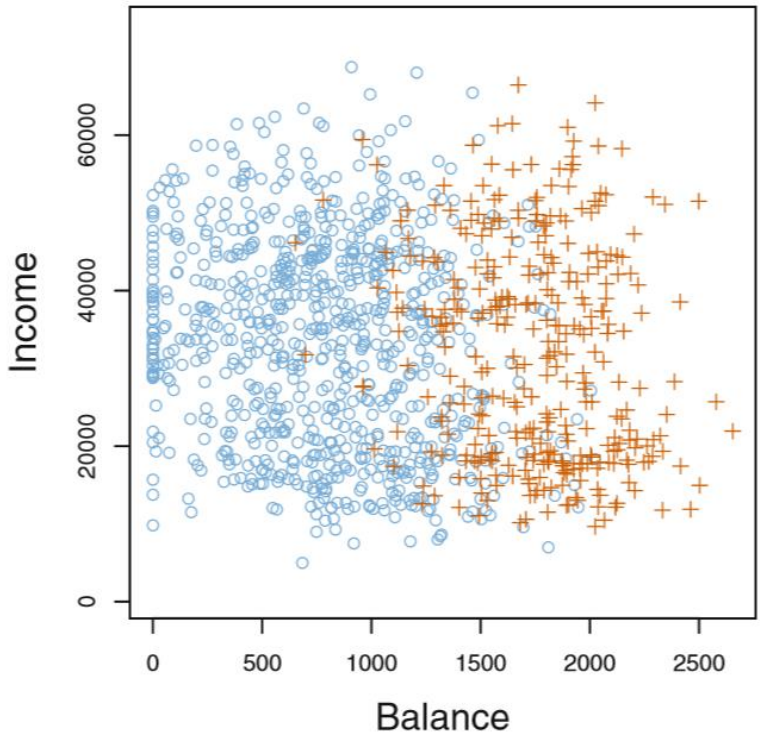


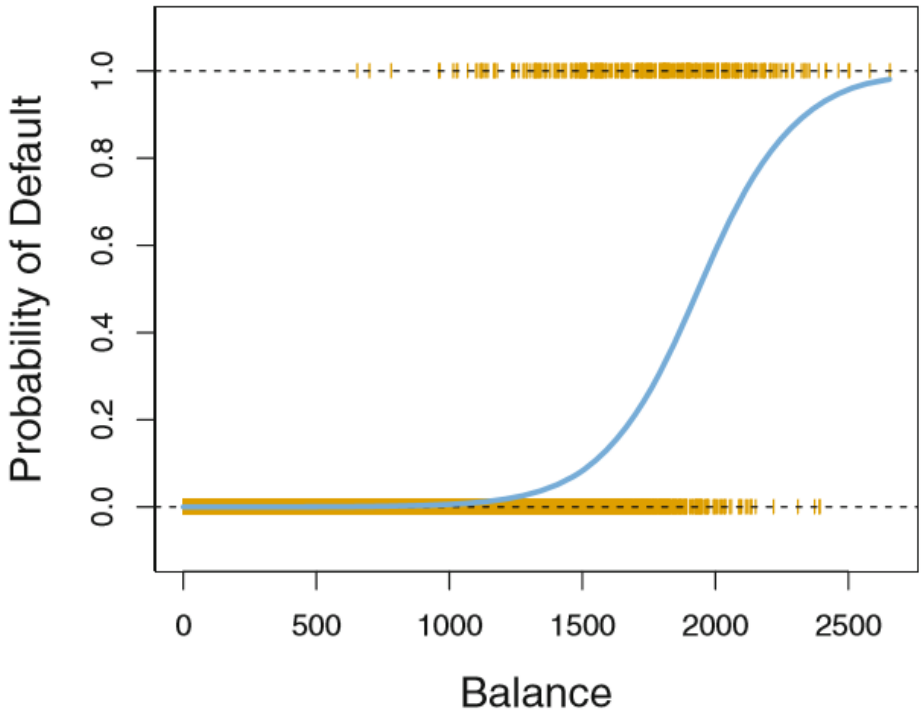
Modelos – Regresión Logistica





Modelos – Regresión Logistica





Modelos – Regresión Logistica

$$y' = b + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5$$

Pounds Displacement Acceleration Number of cylinders Horsepower

Modelos – Regresión Logistica

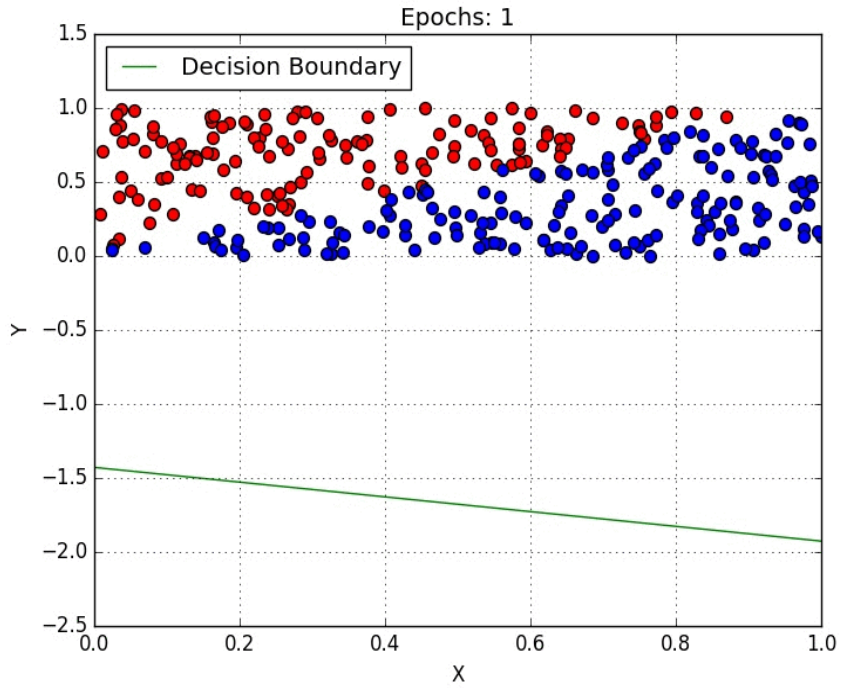
- La salida es una probabilidad (0–1).
- No sirve usar MSE → difícil de entrenar.
- Se usa Log-loss / Cross-entropy:

$$\text{Log Loss} = \sum_{(x,y)\in D} -y\log(y') - (1-y)\log(1-y')$$

Modelos – Regresión Logistica

Entrenamiento:

1. **Datos** → ejemplos con entradas (X) y salidas (Y).
2. **Modelo** → fórmula con pesos
3. **Predice** → calcula valores con pesos iniciales.
4. **Error** → compara predicción vs. valor real.
5. **Ajusta** → corrige pesos según el error.
6. **Repite** → predecir → medir → ajustar.
7. **Final** → pesos “aprendidos” que minimizan error.



TA4 – Regresion Lineal y Regresion Logistica