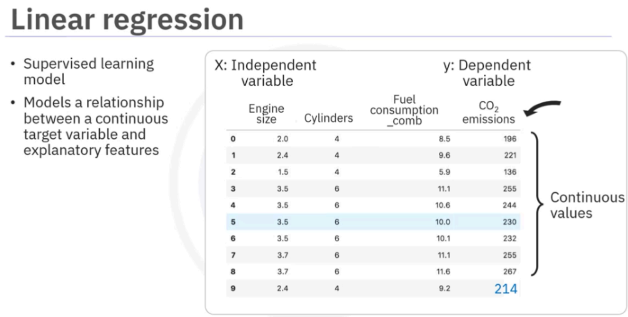
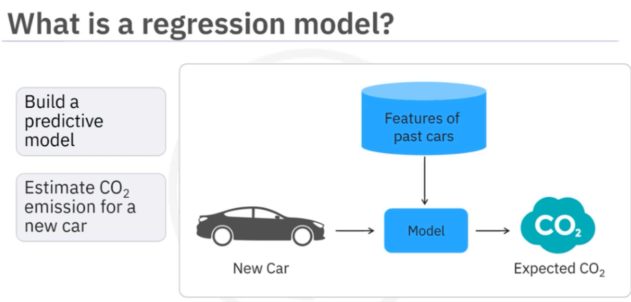
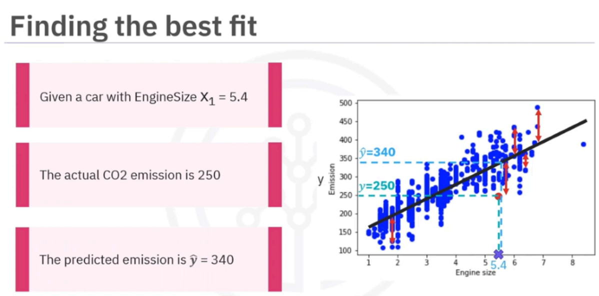
Module 2

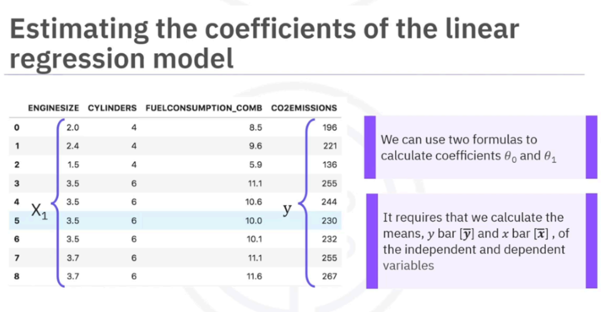
Thursday, July 03, 2025 12:24 AM



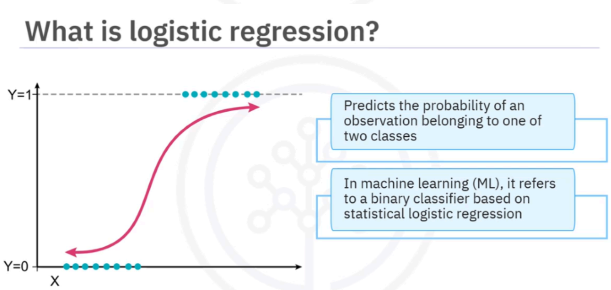
	Engine size	Cylinders	Fuel consumption_comb	CO_2 emissions
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	214

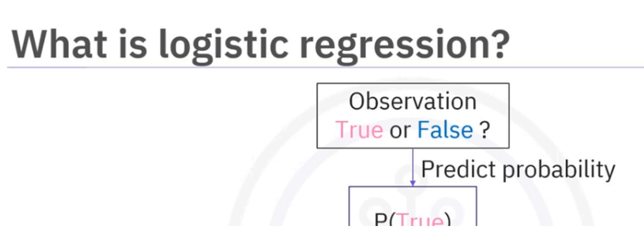


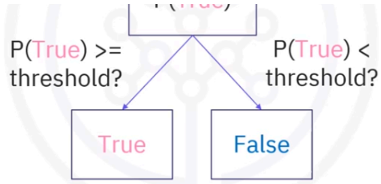




	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267

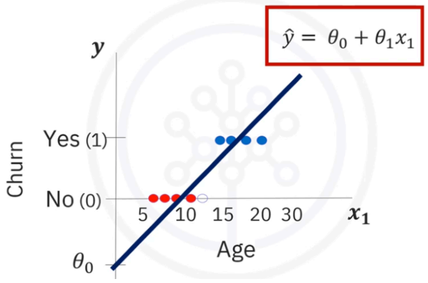






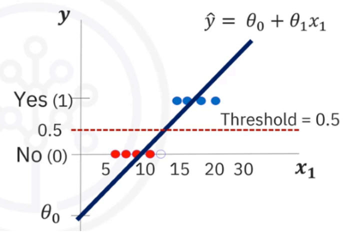
Binary classification

Predicting churn using linear regression



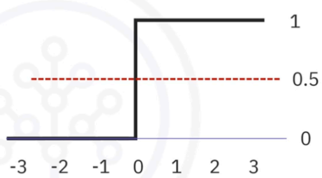
Predicting churn using linear regression

$$\hat{y} \rightarrow \begin{cases} 0 & \text{if } \hat{y} < 0.5 \\ 1 & \text{if } \hat{y} \geq 0.5 \end{cases}$$



Challenges of linear regression

$$\hat{y} \rightarrow \begin{cases} 0 & \text{if } \hat{y} < 0.5 \\ 1 & \text{if } \hat{y} \geq 0.5 \end{cases}$$

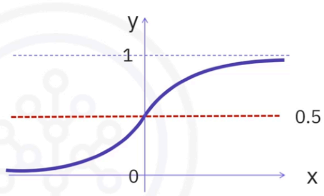


Step function

Towards probabilities

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

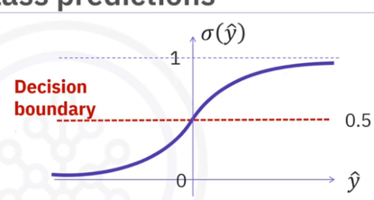
Sigmoid function



Probabilities to class predictions

$$\widehat{p} = \sigma(\hat{y}) = \frac{1}{1 + e^{-\hat{y}}}$$

Probability that class is 1



$$\sigma(\hat{y}) \rightarrow \begin{cases} 0 & \text{if } \sigma(\hat{y}) < 0.5 \\ 1 & \text{if } \sigma(\hat{y}) \geq 0.5 \end{cases}$$

Recap

- ML logistic regression: A binary classifier based on statistical logistic regression, a probability predictor logistic regression, a probability predictor

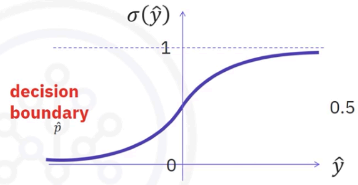
- Logistic regression is a good choice for a binary target, probabilistic results, and understanding feature impact
- Logistic regression is a probability predictor and a binary classifier
- Goal: Build a model to predict class by considering the predicted probability

Optimal logistic regression

$\hat{y} = \theta_0 + \theta_1 x_1$

$\hat{p} = \sigma(\hat{y}) = \dfrac{1}{1 + e^{-\hat{y}}}$

Probability that class of $\hat{y}$ is 1



$$\sigma(\hat{y}) \rightarrow \begin{cases} 0 & \text{if } \sigma(\hat{y}) < 0.5 \\ 1 & \text{if } \sigma(\hat{y}) \geq 0.5 \end{cases}$$

Understanding log-loss

$$\text{Log–loss} = -\frac{1}{N}\sum_{i-1}^{N} y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)$$

Confident and correct: Predicted probability of class 1 is high and correct => log-loss is small

Confident and incorrect: Predicted probability of class 0 is high and incorrect => log-loss is large

Module 2 Summary and Highlights

Congratulations! You have completed this lesson. At this point in the course, you know:

- Regression models relationships between a continuous target variable and explanatory features, covering simple and multiple regression types.
- Simple regression uses a single independent variable to estimate a dependent variable, while multiple regression involves more than one independent variable.
- Regression is widely applicable, from forecasting sales and estimating maintenance costs to predicting rainfall and disease spread.
- In simple linear regression, a best-fit line minimizes errors, measured by Mean Squared Error (MSE); this approach is known as Ordinary Least Squares (OLS).
- OLS regression is easy to interpret but sensitive to outliers, which can impact accuracy.
- Multiple linear regression extends simple linear regression by using multiple variables to predict outcomes and analyze variable relationships.
- Adding too many variables can lead to overfitting, so careful variable selection is necessary to build a balanced model.
- Nonlinear regression models complex relationships using polynomial, exponential, or logarithmic functions when data does not fit a straight line.
- Polynomial regression can fit data but mayoverfit by capturing random noise rather than underlying patterns.
- Logistic regression is a probability predictor and binary classifier, suitable for binary targets and assessing feature impact.
- Logistic regression minimizes errors using log-loss and optimizes with gradient descent or stochastic gradient descent for efficiency.
- Gradient descent is an iterative process to minimize the cost function, which is crucial for training logistic regression models.

Cheat Sheet: Linear and Logistic Regression

Comparing different regression types

Model Name	Description	Code Syntax

Simple linear regression	**Purpose:** To predict a dependent variable based on one independent variable.
	Pros: Easy to implement, interpret, and efficient for small datasets.
	Cons: Not suitable for complex relationships; prone to underfitting.
	Modeling equation: $y = b_0 + b_1x$
		`from sklearn.linear_model import LinearRegression`
		`model = LinearRegression()`
		`model.fit(X, y)`

Polynomial regression	**Purpose:** To capture nonlinear relationships between variables.
	Pros: Better at fitting nonlinear data compared to linear regression.
	Cons: Prone to overfitting with high-degree polynomials.
	Modeling equation: $y = b_0 + b_1x + b_2x^2 + ...$
		`from sklearn.preprocessing import PolynomialFeatures`
		`from sklearn.linear_model import LinearRegression`
		`poly = PolynomialFeatures(degree=2)`
		
		`X_poly = poly.fit_transform(X)`
		`model = LinearRegression().fit(X_poly, y)`

Multiple linear regression	**Purpose:** To predict a dependent variable based on multiple independent variables.
	Pros: Accounts for multiple factors influencing the outcome.
	Cons: Assumes a linear relationship between predictors and target.
	Modeling equation: $y = b_0 + b_1x_1 + b_2x_2 + ...$
		`from sklearn.linear_model import LinearRegression`
		`model = LinearRegression()`
		`model.fit(X, y)`

Logistic regression	**Purpose:** To predict probabilities of categorical outcomes.
	Pros: Efficient for binary classification problems.
	Cons: Assumes a linear relationship between independent variables and log-odds.
	Modeling equation: $\log(p/(1-p)) = b_0 + b_1x_1 + ...$
		`from sklearn.linear_model import LogisticRegression`
		`model = LogisticRegression()`
		`model.fit(X, y)`

Associated functions commonly used

Function/Method Name	**Brief Description**	**Code Syntax**
train_test_split	Splits the dataset into training and testing subsets to evaluate the model's performance.	`from sklearn.model_selection import train_test_split` `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)`

StandardScaler	Standardizes features by removing the mean and scaling to unit variance.	`from sklearn.preprocessing import StandardScaler`
		`scaler = StandardScaler()`
		`X_scaled = scaler.fit_transform(X)`

log_loss	Calculates the logarithmic loss, a performance metric for classification models.	`from sklearn.metrics import log_loss` `loss = log_loss(y_true, y_pred_proba)`
mean_absolute_error	Calculates the mean absolute error between actual and predicted values.	`from sklearn.metrics import mean_absolute_error` `mae = mean_absolute_error(y_true, y_pred)`
mean_squared_error	Computes the mean squared error between actual and predicted values.	`from sklearn.metrics import mean_squared_error` `mse = mean_squared_error(y_true, y_pred)`

root_mean_squared_error	Calculates the root mean squared error (RMSE), a commonly used metric for regression tasks.	`from sklearn.metrics import mean_squared_error`
		`import numpy as np`
		`rmse = np.sqrt(mean_squared_error(y_true, y_pred))`

r2_score	Computes the R-squared value, indicating how well the model explains the variability of the target variable.	`from sklearn.metrics import r2_score` `r2 = r2_score(y_true, y_pred)`