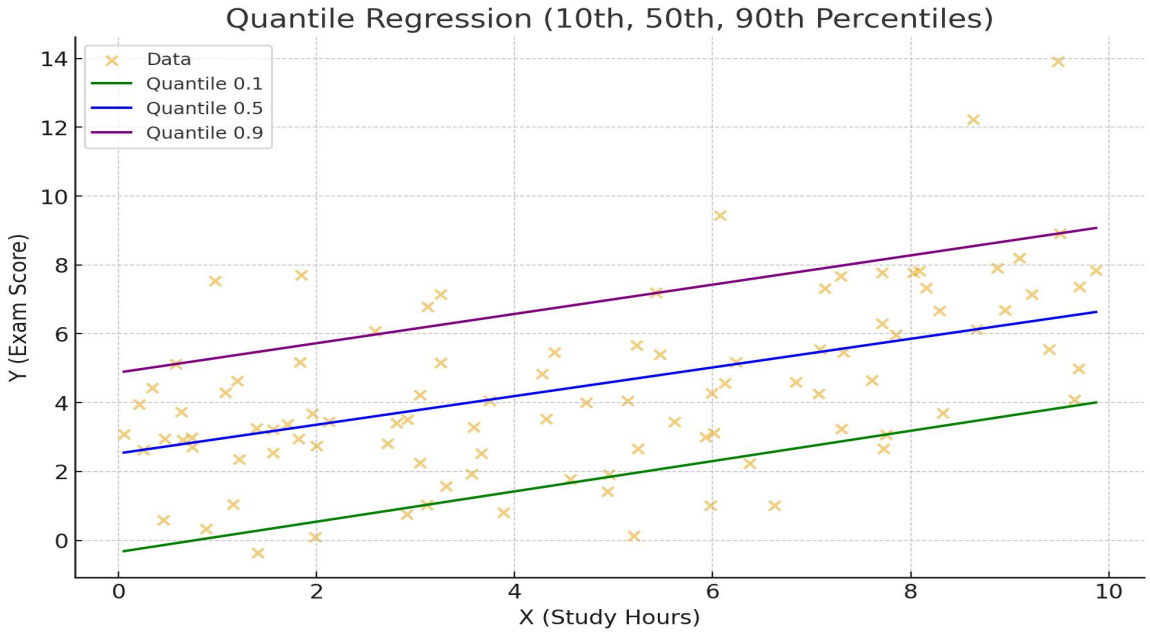
Types of Regression in ML

Quantile Regression:

Quantile Regression is a statistical technique that estimates the **conditional quantiles** (like the 10th, 25th, 50th/median, 75th, 90th percentile) of a response variable YYY given predictors XXX.

Instead of predicting just the **center** of the data (like the mean), quantile regression allows us to understand how predictors influence the **entire distribution** of outcomes.



Here's the visualization of **Quantile Regression** lines (10th, 50th, and 90th percentiles) on the simulated dataset:

Green line (10th percentile) → effect of study hours on low scorers.

Blue line (50th percentile, median) → effect on typical/average student.

Purple line (90th percentile) → effect on top scorers.

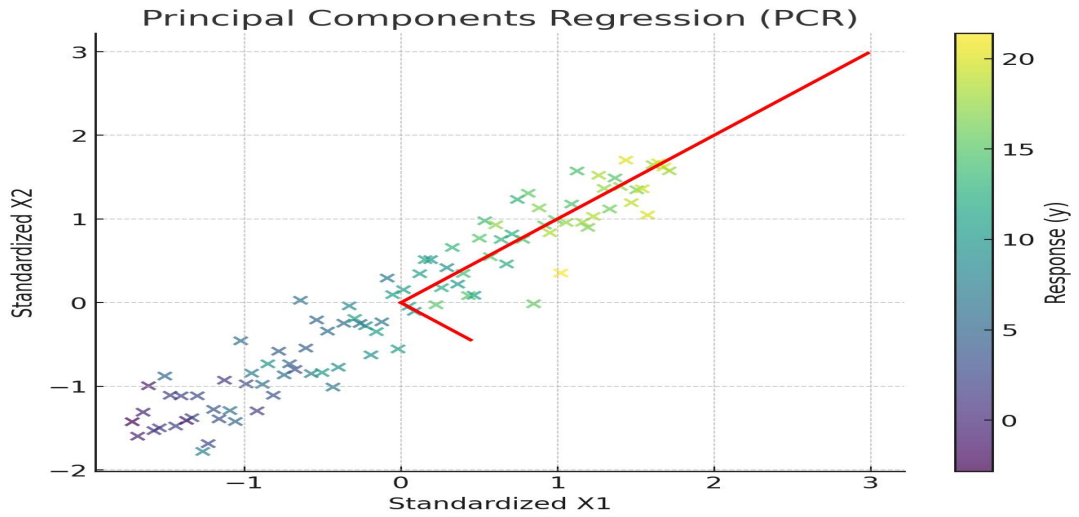
Code:

```
from sklearn.linear_model import QuantileRegressor

    qr = QuantileRegressor(quantile=q, alpha=0)  # alpha=0 → no regularization

    y_pred = qr.fit(X, y).predict(X)

    predictions[q] = y_pred
```

Principal Components Regression (PCR)

PCR is a regression technique which is widely used when you have many independent variables OR multicollinearity exist in your data. It is divided into 2 steps:

1. Getting the Principal components
2. Run regression analysis on principal components



Here's a **diagram of Principal Components Regression (PCR):**

The scatter shows your **two correlated predictors** (standardized).

The **red arrows** are the **principal component directions**:

The long arrow (PC1) captures most of the variance.

The short arrow (PC2) captures the leftover variance.

PCR keeps **PC1** (the dominant axis) and fits the regression in that direction instead of directly using X1 and X2 .

This way, PCR handles **multicollinearity** and reduces noise by discarding weaker components.

Code:

```
pcr = Pipeline([ ("scaler", StandardScaler()), ("pca", PCA(n_components=2)),
    ("regression", LinearRegression())
])
pcr.fit(X_train, y_train)
```

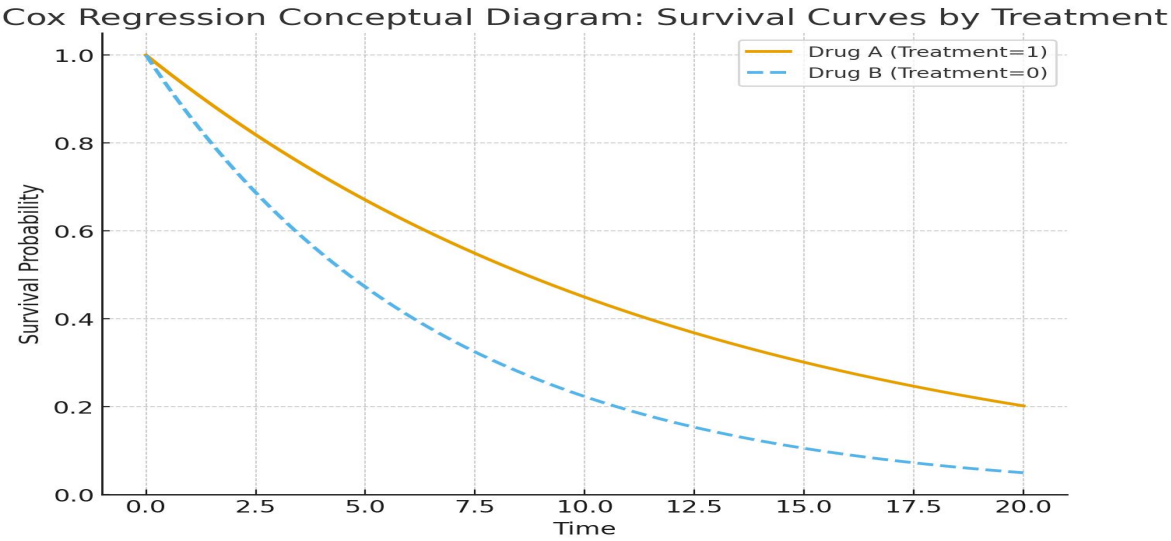
COX Regression:

Cox Regression is suitable for time-to-event data. See the examples below -

1. Time from customer opened the account until attrition.
2. Time after cancer treatment until death.
3. Time from first heart attack to the second.

Logistic regression uses a binary dependent variable but ignores the timing of events.

As well as estimating the time it takes to reach a certain event, survival analysis can also be used to compare time-to-event for multiple groups.



Here's a **conceptual diagram of Cox Regression** survival curves:

Drug A (blue solid line): Slower decline → patients survive longer.

Drug B (orange dashed line): Faster decline → higher risk, shorter survival.

This illustrates the **hazard ratio** idea: if HR < 1 for Drug A, its hazard is lower at **all times**, so its survival curve stays above Drug B's.

Code:

```
cph = CoxPHFitter()

cph.fit(data, duration_col="time", event_col="event")

cph.print_summary()
```

ROC AUC Curve

The ROC AUC curve is a performance evaluation metric for binary classification models, visualizing their ability to distinguish between two classes across various classification thresholds. The Area Under the Curve (AUC) summarizes the performance of the entire curve into a single value, offering a quick way to compare different models.

The ROC curve

A Receiver Operating Characteristic (ROC) curve plots two metrics against each other at all possible classification thresholds:

- True Positive Rate (TPR): Also known as sensitivity or recall, this is the percentage of actual positive cases that are correctly identified by the model.
- False Positive Rate (FPR): The percentage of actual negative cases that are incorrectly identified as positive.

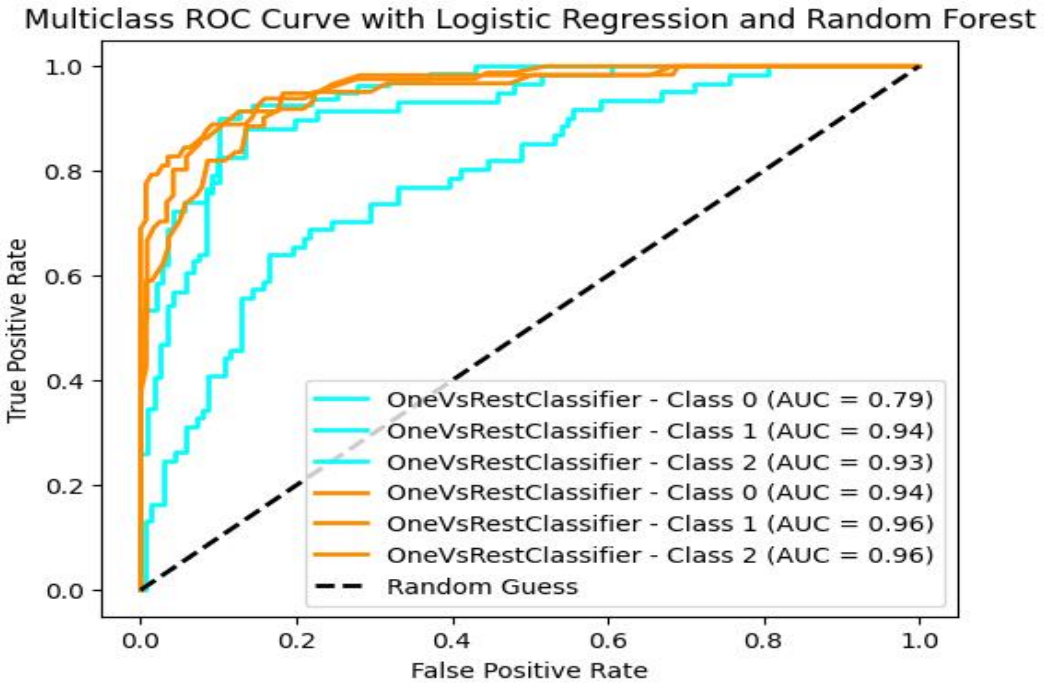
By calculating the TPR and FPR at different threshold settings and plotting them, the ROC curve shows the trade-off between the benefits (true positives) and the costs (false positives) of a model.

The AUC score

The Area Under the ROC Curve (AUC) is a single number representing the model's overall performance. It can be interpreted as the probability that the model will rank a randomly chosen positive example higher than a randomly chosen negative example.

Interpreting the AUC score:

- AUC = 1.0: A perfect model that correctly identifies all positive and negative cases. The curve hugs the top-left corner of the plot.
- AUC = 0.5: The model performs no better than random guessing. The curve follows the diagonal line from the bottom-left to the top-right.
- AUC < 0.5: The model performs worse than random guessing. Its predictions are consistently wrong, effectively inverting the results.
- 0.5 < AUC < 1.0: The model has some discriminatory power, and higher values indicate better performance.
- Good: 0.8–0.9
- Excellent: >0.9


Multiclass ROC Curve with Logistic Regression and Random Forest
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
OneVsRestClassifier - Class 0 (AUC = 0.79)
OneVsRestClassifier - Class 1 (AUC = 0.94)
OneVsRestClassifier - Class 2 (AUC = 0.93)
OneVsRestClassifier - Class 0 (AUC = 0.94)
OneVsRestClassifier - Class 1 (AUC = 0.96)
OneVsRestClassifier - Class 2 (AUC = 0.96)
Random Guess