# **📝 Insurance Enrollment Prediction Report**

## **📌 Problem Statement**

The objective is to **predict whether an employee will opt into a new voluntary insurance product**, using demographic and employment data. The dataset contains synthetic employee profiles including:

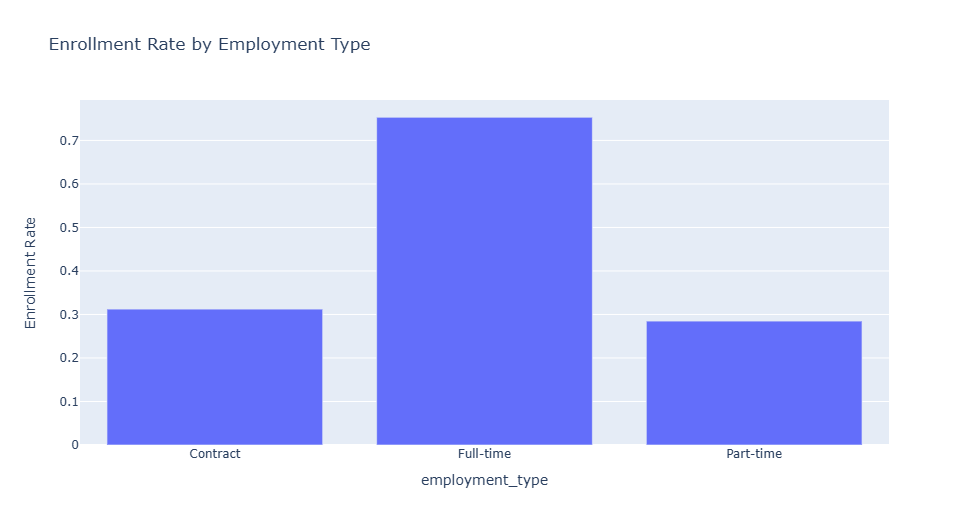
* Age
* Salary
* Gender
* Marital status
* Region
* Employment type
* Dependents
* Tenure

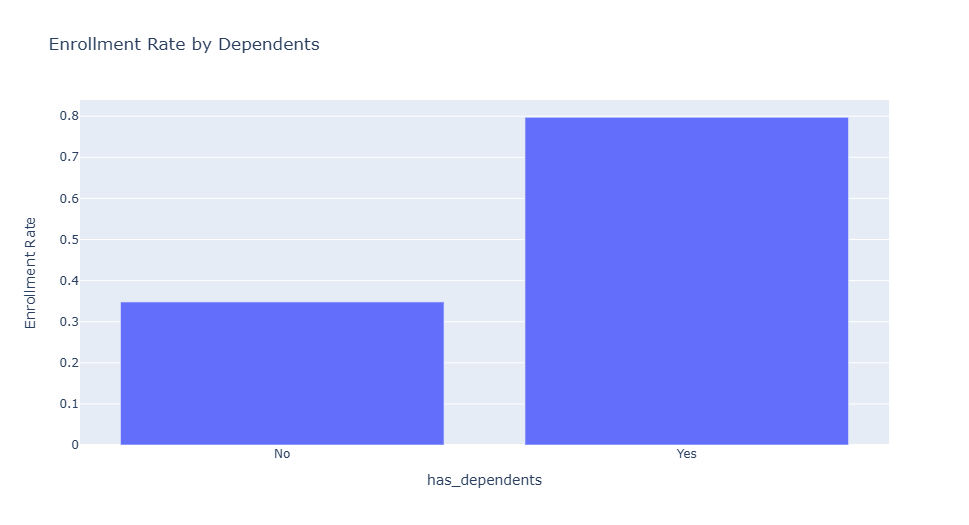
## **🔍 Exploratory Data Analysis (EDA)**

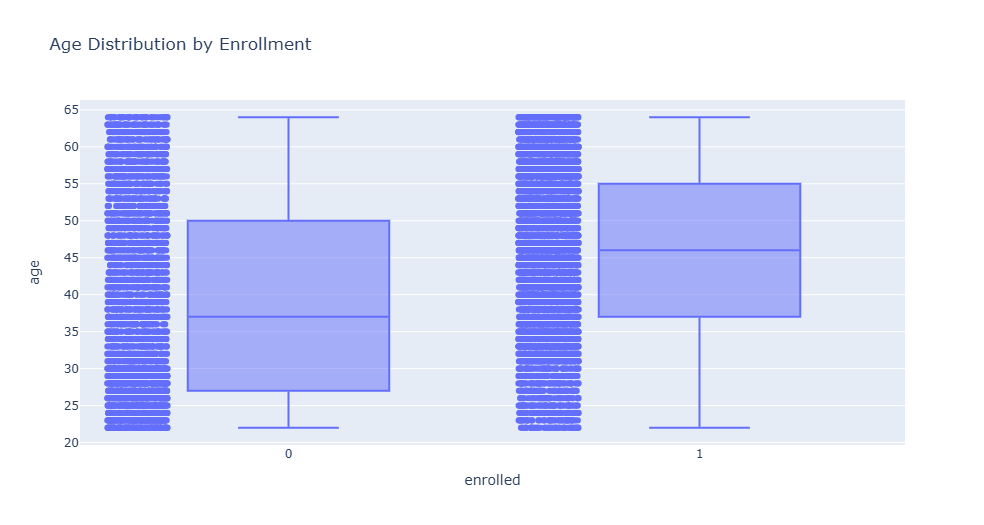
* ✅ **No missing data** or outliers were identified.
* ⚖️ The dataset is **slightly imbalanced**, but not severely enough to require resampling in the baseline phase.

**Key enrollment trends:**

* Employees with **dependents** are significantly more likely to enroll.
* **Full-time employees** show much higher enrollment than part-time or contract workers.
* **Higher salary** and **older age** are positively correlated with enrollment likelihood.







## **🧹 Preprocessing Pipeline**

A ColumnTransformer was used to streamline preprocessing:

* 🔢 **Numeric features** (age, salary, tenure) → StandardScaler
* 🔣 **Categorical features** (gender, marital\_status, employment\_type, region, dependents) → OneHotEncoder

## **🧠 Model Selection**

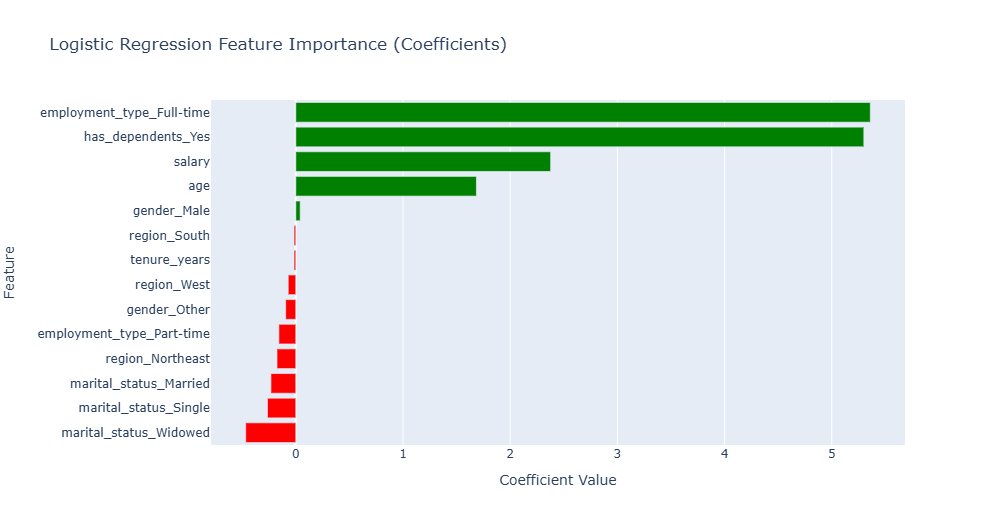
Two models were chosen to balance interpretability and predictive power:

* **Logistic Regression** – quick, interpretable baseline
* **Random Forest** – non-linear model with better generalization capacity

## **📊 Feature Importance (Logistic Regression)**

A Plotly bar chart visualized the model’s coefficients:

* ✅ **Positive Influence (Green):**
  + Has Dependents = Yes
  + Salary
  + Age
  + Full-Time Employment
* ❌ **Negative Influence (Red):**
  + Contract/Part-Time Employment
  + No Dependents



## **📈 Results Analysis**

* **Logistic Regression Accuracy**: ~90%
* **Random Forest Accuracy**: 100% (Possible overfitting; needs validation but not disregarded)

## **🚀 Next Steps**

* ✅ Implement **k-Fold cross-validation** for more robust evaluation.
* 🔍 Introduce a **separate validation set** (not just train-test split) for tuning.
* 🧪 Explore **SMOTE sampling** to address minor class imbalance.
* 🛠️ Deploy a **REST API using FastAPI** for scalable inference.