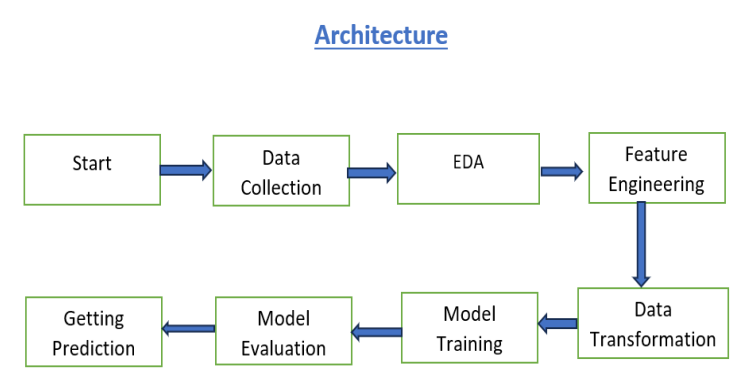
**Architecture Design**

*Fraud Transaction Detection*



**1. Data Description:**

Initially raw data is stored as csv file for the project which was provided by iNeuron

Columns:

▪ Time: Time elapsed in seconds since the first transaction.

▪ V1, V2, ..., V28: Anonymized features resulting from a PCA transformation to

protect user identities.

▪ Amount: Transaction amount.

▪ Class: Binary variable indicating whether the transaction is fraudulent (1) or

not (0).

**2. EDA**

• **Understand Data Structure:**

* Load the dataset and inspect the structure using functions like head(), info(), and describe().
* Check for data types, missing values, and basic statistics.

• **Visualize Data:**

* Use histograms, box plots, and density plots to understand the distribution of numerical features.
* Use bar plots and pie charts for categorical features.
* Employ pair plots or scatter plots to identify relationships between features.

• **Identify Correlations:**

* Generate a correlation matrix to identify potential relationships between features.
* Use heatmaps to visualize correlations.

**3. Feature Engineering**

* **Handle Missing Values:**
  + Impute missing values using mean, median, mode, or other imputation techniques.
  + Consider removing features or rows with a high percentage of missing values.
* **Encode Categorical Variables:**
  + Apply one-hot encoding for nominal categorical features.
  + Use label encoding or ordinal encoding for ordinal features.
* **Create New Features:**
  + Generate new features based on existing ones (e.g., interaction terms, polynomial features).
  + Extract datetime features (e.g., day, month, year, hour) from timestamp data.
* **Scale and Normalize:**
  + Apply scaling techniques such as standardization (Z-score normalization) or min-max scaling.
  + Normalize features if necessary, especially for algorithms sensitive to feature scaling.

**4. Data Transformation:**

* Implement data transformation techniques to enhance feature representation and prepare data for model training.
* Handle data normalization, scaling, and feature engineering.

**5. Model Training:**

• **Split Data:**

* Divide the dataset into training and testing sets (e.g., 80/20 split) using functions like train\_test\_split.
* Consider using stratified sampling for imbalanced datasets.

• **Select and Train Models:**

* Choose appropriate machine learning algorithms based on the problem type (e.g., regression, classification).
* Train models using training data (e.g., linear regression, decision trees, SVM, neural networks).

**6. Model Validation**

• **Model Performance Metrics:**

* For classification: accuracy, precision, recall, F1-score, ROC-AUC.
* For regression: mean absolute error (MAE), mean squared error (MSE), R-squared.

• **Cross-Validation:**

* Implement k-fold cross-validation to ensure model robustness and avoid overfitting.
* Average performance metrics across folds to get a reliable estimate of model performance.

• **Confusion Matrix:**

* For classification problems, use a confusion matrix to visualize true positives, false positives, true negatives, and false negatives.

**7. Getting Prediction**

• **Load Model:**

* Load the trained model and necessary preprocessing steps (e.g., encoders, scalers).

• **Preprocess Input Data:**

* Apply the same preprocessing steps to new input data as used during training.
* Ensure consistency in feature formats and encodings.

• **Generate Predictions:**

* Use the trained model to make predictions on new, unseen data.
* Output predictions in the desired format (e.g., probability scores, class labels, continuous values).