# Summary of Methodology

Started with two datasets, one was Customer details and second was credit card approval detail for every corresponding customer. Merged both data together and started with data preprocessing.

First checked for duplicate values and the null values. Imputed null values in categorical columns with **mode** and continuous columns with **mean**. Then created new feature using the available feature and removed the feature to remove multicollinearity and also dropped irrelevant features as well. After that visualized all the features to work with outliers present in data by plotting the univariate distribution and then used boxplot to visualize the outliers and finally used IQR method to remove the outliers. Then verified it again using the distribution plot and box-plot.

Next step was encoding the categorical variables. So I used ordinal encoding for ordinal column, one-hot encoding for rest to encode the columns. Then splits the dataset into training and testing data with 75:25 ratio. Then performed feature scaling using min-max scaler before training the model.

Now comes the model implementation part. Imported all the required models and metrices using scikit-learn and then initializes them. After that trained the models using training data and predicted the target variable using test data and evaluated the model performance using accuracy, precision, recall, f1, and roc\_auc\_score.

Next step was to optimize the model performance using hyperparameter tuning. So tuned the hyperparameter of all the models using grid search cv to get the best values and then again trained the models using these hyperparameter to improve the model performance and compared it with model performance before tuning.

After that analyses the impact of feature selection on model performance by reducing the number of feature to 10 using filter method. Retrained the model on reduced features and compared the performance with previous results.

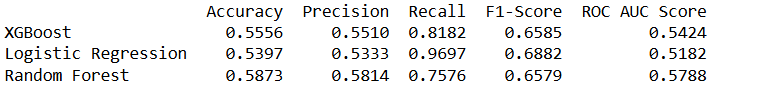
Next calculated the optimal number of features required for each model and finally plotted the feature importance plots for each model.

# Comparative analysis tables

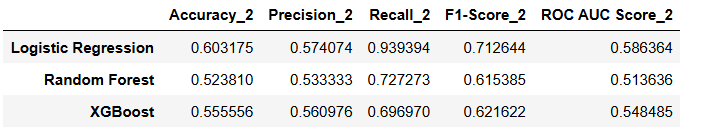
**Model Performance before Hyperparameter tuning:**

|  | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **ROC AUC Score** |
| --- | --- | --- | --- | --- | --- |
| **Logistic Regression** | 0.492063 | 0.510638 | 0.727273 | 0.600000 | 0.480303 |
| **Random Forest** | 0.603175 | 0.6000 | 0.727273 | 0.657534 | 0.596970 |
| **XGBoost** | 0.523810 | 0.536585 | 0.666667 | 0.594595 | 0.516667 |

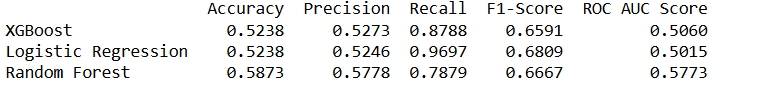
**Model Performance after Hyperparameter tuning:**



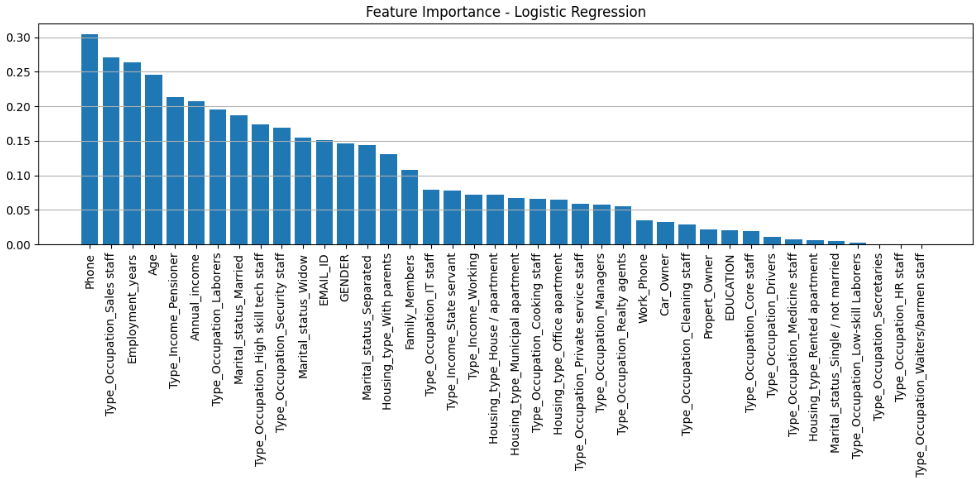
**Model Performance after feature reduction before hyperparameter tuning:**

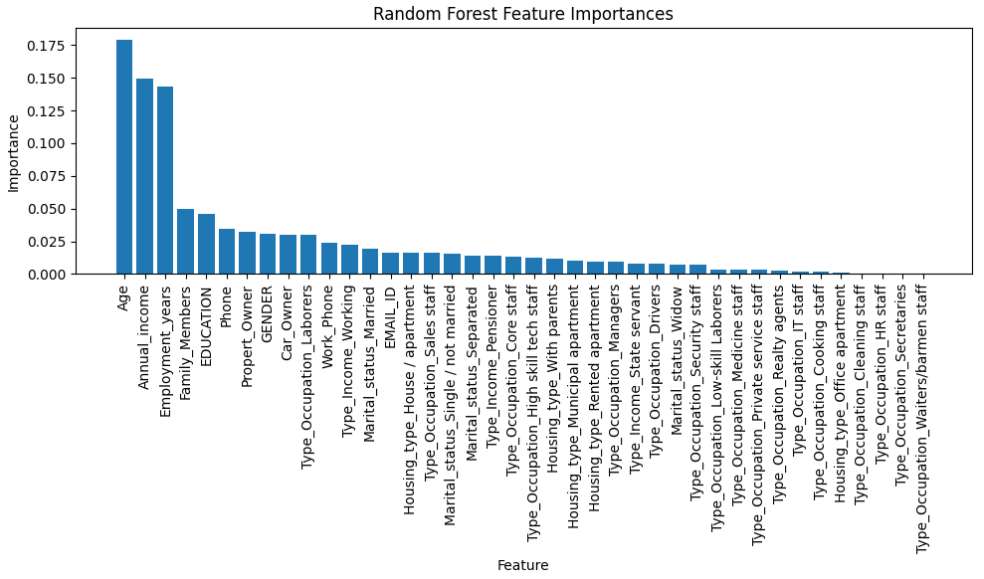
****

**Model Performance after feature reduction after hyperparameter tuning:**

****

**Feature Importance**

****

****

# Justification of model recommendation:

Random Forest achieved the best overall performance after tuning because of its ensemble bagging approach, which constructs multiple decision trees in parallel using random subsets of the data and features. This method reduces variance and helps prevent overfitting, resulting in consistently high accuracy and robust generalization across different metrics.