**Problem Statement:**

Smart Agent Recruitment Challenge

FinMan is a Financial Distribution company. Over the last 10 years, they have created an offline distribution channel across India. They sell Financial products to consumers by hiring agents in their network. These agents are freelancers and get commission when they make a product sale.

Overview of FinMan On-boarding process

The Managers at FinMan are primarily responsible for recruiting agents. Once a manager has identified a potential applicant, the would explain the business opportunity to the agent. Once the agent provides the consent, an application is made to FinMan to become an agent. This date is known as application\_receipt\_date.

In the next 3 months, this potential agent has to undergo a 7 day training at the FinMan branch (about Sales processes and various products) and clear a subsequent examination in order to become a FinMan agent.

The problem - Who are the best agents?

As is obvious in the above process, there is a significant investment which FinMan makes in identifying, training and recruiting these agents.

**Solution Methodology :**

* **Data Cleaning and data manipulation.**

1. **Check and Handle duplicate data**

**Approach :** No duplicate data was observed.

1. **Check and Handle *NAN*(Null) values.**

**Approach :** Replaced Null values of the columns depending on the type of values it contains, like for example all the date columns were replaced with forward fill/backward fill functions (ffill()/bfill()). The Gender columns missing values were replaced with ‘mode’ of the data. Business columns missing values were replaced with mean of the corresponding manager. Some columns like Manager’ Designations, Status or the Grade columns missing values were replaced by category ‘Others’.

* **Feature Engineering & Exploratory Data Analysis**

1. **Feature Engineering**

* **We ended up creating features like :**
* **Manager\_Promotion –** Using features Manager\_Current\_Designation & Manager\_Joining\_Designation.
* **PinDifference –** Using features Office\_PIN & Applicant\_City\_PIN
* **Manager\_Num\_Products\_new -** Using features Manager\_Num\_Products & Manager\_Num\_Products2
* **applicantAge -** Using features Application\_Receipt\_Date & Applicant\_BirthDate
* **managerExperience -** Using features Application\_Receipt\_Date & Manager\_DOJ
* **managerAge -** Using featuresApplication\_Receipt\_Date & Manager\_DoB
* **DateRank -** Using feature Application\_Receipt\_Date

1. **Univariate data analysis :**
   * **Univariate analysis of some categorical variables :**

Chart, bar chart

Description automatically generated

**Observation/Approach :**

-> Business Not Sourced bar is almost twice as Business Sourced.

-> Count of Male members is much more than female members for both the features.

* + **Univariate analysis of Numerical variables variables :**

Shape, histogram

Description automatically generated

**Observation/Approach :**

* We can clearly observe that **-** we could apply some transformation like `**Log Transformation**` to some features like **Manager\_Business** or **Manager\_Num\_products2** to make them **Normally distributed** as they are ***right skewed***, but we can't apply as they contain some values which are 0 or close to zero. Then their **log\_transformation** can lead to infinite or undefined values.

1. **Outlier analysis :**

* **Outlier analysis of Numerical variables :**

Graphical user interface

Description automatically generated

**Observation/Approach :**

* We can see outliers in these all features. So, we will try to remove those with *upper capping method*, as the outliers are lying in the right side of the boxplots, with the respective percentile values of **98%** and **97%**, depending on the features and the outlier values. We will not cap the values further as it might lead to information loss.

1. **Bivariate/Multivariate data analysis :**

* **Bivariate analysis of some categorical variables :**
* **Then went ahead with countplots of different categorical columns with Target variable.**
* **Then with the pairplot and Heatmap to check the relation and correlation coefficients among different variables.**

1. **Class Imbalance Analysis :**

**Approach:**

Text

Description automatically generated

* So, we can see the ratio of Business Not Sourced by the applicant to Business Sourced by the applicant is around 2.

But in real world scenario, the data holds true values, as the **Business Not Sourced by applicant will always be greater than Business Sourced by the applicant**. So, performing any sampling techniques could lead to in inaccurate results. For e.g. if we go ahead with undersampling, there might be the issue of data constrictions(data loss), whereas if go ahead with oversampling technique, then no. of 'Yes' will be given equal weightage as no. of 'No'. So, this should never be the case.

1. **Label Encoding/OneHotEncoding the categorical variables :**

* Label Encoding/OneHotEncoding(get\_dummies()) was applied, depending on the features in both, Train and Test dataframe.

1. **Feature Scaling :**

* **After the Train & Test split, StandardScaler(), scaling method was used.**
* **Classification Modelling Technique :**
* I have used four models, which are:
  + 1. Logistic Regression.
    2. Random Forest Classifier.
    3. Extreme gradient boosting(XGBoost).
* Evaluated these models on the basis of roc\_auc score which is the best evaluation metric when Probability has to be predicted.

1. Logistic Regression :
   * Used simple logistic regression model with the params like class\_weight= ‘balanced’, which can take care of the class imbalance issue.
   * Achieved a roc\_auc score of 87.70% in Train and 87.84% Test data.
2. Random Forest Classifier :
   * Used a Random forest classifier.
   * Achieved a roc\_auc score of 100% on training data and 88.43% on the test data respectively.
   * Cleary this is the case of overfitting model.
   * Went ahead with some Hyperparameter Tuning technique like GridSearchCV to find the best parameters for RandomForestClassifier().
   * When the new parameters were used the roc\_auc score changed to 95.77% and 88.59% in Train and Test data respectively, which is a great improvement in the model.
3. Extreme Gradient Boosting(XGBoost) Classifier :
   * Used a XGBoost Classifier(as this works well with the data where there is class imbalance issue) with class\_weight = ‘balanced’.
   * Achieved a roc\_auc score of 99.84% and 87.53% on Train and Test data.
   * For further improvement in the model, applied some hyperparameter tuning using GridSearchCV, to get the best parameters for this classifier.
   * After fitting the model again with new parameters, achieved a roc\_auc score of 90.79% and 88.37% on Train and Test data respectively, which is a quite decent score.

* **Final Submission :**
* Submitting the final model with **XGBoost** which is **hyperparameter tuned**. As, it has descent **roc\_auc** score on the Train and the Test data, along with good **Recall score**.