AMEXPERT MACHINE LEARNING COMPETITION

- Penukonda Yeshwanth

MY APPROACH:

My approach included three main important steps:

1) **Filling up the missing values**
2) **Selecting the important features (feature engineering)**
3) **Choosing the best model for the data**

1) Filling up the missing values:

Some columns(features) of the data were missing such as ***'owns_car', 'no_of_children','no_of_days_employed','total_family_members','migrant_worker', 'yearly_debt_payments','credit_score'*** were missing.

In order to make the model learn the data, the data must not have any missing values.

For this, for the features with numerical data, I have filled up with the nearest integer to ***mean*** of the other feature values; and for the categorical features, I have filled up the missing values with the ***mode*** (largest frequency element). This results in a better accurate prediction and prevents much noise to enter into our data.

2) Selecting the important features (feature engineering)

Now, once our data is good to go with no missing values, we must extract only the important features which are important for predicting the required results. Unnecessary features would just add noise to our predictions and decrease our overall performance and efficiency of the model. So, in order to extract only the important features in the data, I have extracted the mutual information of all the

features with respect to the target variable. I have used the **mutual_info_classif** function from **sklearn.feature_engineering** library in order to extract the mutual information.

After getting the mutual information, I have considered only the features with relative importance, i.e. **features with mutual information > 0.001**. (Tried for different thresholds but this was the best out of them)

So, we have extracted only the important features required for the prediction.

3) Choosing the best model for the data

Once the important features are extracted, I have standardized them in order for equal weightage of each feature. Now, the task was choosing the best model for training the data.
After going through all the Machine Learning Models, I found that Random Forest and XGBoost would be the ideal models for the best performance on the data.

Using **cross_validation_score** for both the models by fitting the training data with extracted features, **XGBoost Classifier** performed the best by giving an **98%** accuracy w.r.t to validation data from training data.
(Also performed GridSearchCV for finding the best hyper-parameters for XGBoost. The initial default XGBoost parameters performed best on the testing data.)

So, I have chosen XGBoost Classifier for predicting the testing data results.
Then, the results obtained have given an **accuracy of 91.89%** approx. on the testing data.

IMPROVEMENTS WHICH CAN BE MADE:

1) ***More important features can be created for the data, which would increase the performance of models.***
2) ***Missing data filled with more accurate data for better performance.***
3) ***Selection and choosing over more better parameters and model which would have resulted in a much better performance.***