LEADS SCORING ASSIGNMENT SUMMARY...

A Brief Overview of The Problem Statement: -

X Education, an Education company, sells online courses to Professionals. They have abundance of potential leads who visit their website enquiring about the course. But the problem lies in the fact that around 70% of those potential leads don't pan out (i.e., Don't enrol in their course).

The company has hired our team of Data Analysts to pinpoint the traits that a lead shows to ascertain, with a respectable accuracy, if the lead will turn into an actual customer or not. We are required to build a model that gives these leads a score between 0-100. This will help their sales team to focus on only the clients which have a high leads score.

Now, The Solution: -

We'll be highlighting each step on a point-by-point basis for better readability, and, if necessary, we'll be explaining why it had to be done...

- We imported the necessary libraries to make our task easier and efficient.
- Then, the dataset was imported, which has 37 different parameters on the basis of which their 9240 leads have performed.
- Then, we did the necessary Datatype checks, to make sure that every feature has a suitable datatype.
- We were informed that certain categorical columns had a "Select" variable, which is as good as a null value.
  So, we made a copy of the original dataset, so as to keep the data intact, and converted all "Select" values to null to get the true picture of the dataset provided.

- Now, we went on to inspect the percentage of null values that each column has, if it does so. And, we found that 17 of the 37 columns had missing values, and some columns had huge percentage of null values.
- Earlier data-checks suggested that many columns had only Binary values (yes/no). Hence, created a list (yn_vars) of their names. Visualized their skewness, (the tendency of a feature to be partial towards a particular value), using Bar charts, and it turns out that the majority of them, barring only one, were skewed. Hence, created another list (skewed_yn_vars), which contains all such columns with more than 85% skewness.
- Created another list to store the names of all the features with categorical features, and labelled the skewed categorical features (Skewness of more than 70% and unique values less than 5) and visualized their skewness and stored the names of skewed categorical features in another list.
- Merged the above two lists containing the names of columns of skewed features, and dropped them from our data. Turns out we got rid of more than 40% of our Total features.
- Mapped the Binary features' values to 0s and 1s to prepare them for modelling.
- Now, found out the categorical features which had missing values, and imputed them with a new category named, "Missing", so as to not create any biasness among the existing values.
- Initial visualizations of categorical features showed that there were some features that had huge percentage of total values. This will create a problem when we introduce One-Hot Encoding as we'll end up creating 100s of columns for niche categories.
  So, came up with a block of code to impute a categorical value as "rare_val", if it makes less than 5% of the total values. This will combine all the categories with petite volumes into one and will help our model to test the significance of these values.
- Stored the names of numerical columns other than "Id" in a new list. Visualized the numerical columns' distribution.
  Found out, which of these numerical features had missing values. Used a boxplot to analyse if these columns had outliers, and they did. Hence, imputed the missing values in numerical columns with medians.

- Done with missing value treatments and initial visualizations.
- Now, as we know this is a CLASSIFICATION problem. Hence, we need to figure out, if our data is evenly spread out in terms of the Binary outputs in the target variable i.e., "Converted".
  To judge this, made a 'Class Imbalance Pie Chart', and the ratio came out to about 3:2 of Negatives and Positives. Now, this isn't ideal but is manageable.
- Got rid of both ID columns as they won't help in modelling and performed bivariate analysis on numerical features using a Seaborn Pair-plot. Turns out, 2 of our 5 numerical features are having discrete levels.
- Performed a 'Correlation Heatmap' to visualize the correlations between different numerical features and our target variable. Our data isn't an inherently correlated one.
- Moving onto creating dummy variables for categorical features in the dataset, dropping the original features and merging the dummy dataset with our dataset.
- Did a train-test split of our Dataframe with 70% as train and 30% as testing set.
- Used standardization technique to scale the numerical features of our dataset to aid the algorithm and reap better results.
- Split our train dataset into X and Y, used scikit-learn to create a preliminary model out of all the features and used Recursive Feature Elimination to select Top 25 features from our dataset.
- Made several models and zeroed in on the final model i.e., "logreg6". Implemented the VIF technique to glance onto the multi-collinearity in our model.
  The multi-collinearity levels aren't ideal, but didn't remove the top correlated feature as our model recommends that the feature is pretty significant and VIF is still under 10%.
- Made a Predictions Dataframe with columns such as ID, target feature and the predicted probability of leads converting into actual customers.
- In this Dataframe, added all the possible probability thresholds and ran the Dataframe through some evaluation metrics and stored the results in a Metrics Dataframe.
  This Dataframe consists all the important metrics like: -

1. Accuracy – The number of total correct predictions divided by total predictions, to judge the overall picture.
2. Sensitivity – Of all the actual Yes', how many were correctly predicted by the model.
3. Specificity – Of all the actual No's did the model predict correctly.
4. Precision of both Classes – These measures the precision of both the classes. As in, of all the predicted values how many were accurately predicted.
5. Recall of both Classes – These measures how many correct values was the model able to recall. As in, how many of all the actual values were correctly predicted by the model.

- Visualized different metrics from the above Dataframe using separate line-plots and checked their validity. Accuracy, Sensitivity and Specificity suggest "0.3" as the optimal threshold value, while Precision and Recall suggest "0.4" as the answer.
- Made a classification report of all the top thresholds and picked "0.4" as all the metrics point towards it being the best choice.
- Made an ROC Curve (Receiver Operating Characteristic Curve), which yielded a 96% Area Under the Curve, which is spectacular.

- Now, moving onto preparing the test data to render it through the model and compare results. And, it too, yields very similar and equally impressive results, with every metric pointing towards a value above 88%, hence suggesting that our model is performing just fine.

- Finally, we perform the same manipulations to our original Dataframe copy and make predictions, multiply these with 100 and round them up to 0 decimal values.

And, VOILA! We have our Score metric, which was what we intended to reach to…