Summary Report

Introduction

This report summarizes the process & learnings gained towards building a ML logistic regression model for predicting the probability of a lead (potential customer) converting to a paid customer.

Data Cleaning

Check for duplicates values and drop duplicate rows, if any.

Drop obvious columns with no analysis values; such as "Prospect ID" and "Lead Number".

Replace all "Select" (i.e. no value selected) with "NaN"

Check for "Null" percentages of all columns. Drop all columns with > 40% null values.

Several columns have massive imbalance, i.e., 99%+ values are the same. The near-zero variance results in almost zero analysis value of such columns. Hence, we drop columns like:

- 'Country', 'City', 'What matters most to you in choosing a course'
- And about 12 other additional columns similarly dropped

For Categorical columns, replace NaN values in most columns to either "UnSpecified", "Unknown" or mode value. For Numerical columns, replace NaN values with median value.

Values in several categorical columns are consolidated into a single value. Example: In 'Tags' column, several values with very low frequency are grouped as "Others".

Exploratory Data Analysis

Analyze count plots of categorical columns against converted-vs-nonConverted users. This provides insight on which areas to focus on for improvement of conversion rates.

Learnings:

- When it comes to a Lead's origin, landing page submission and API origins are largest in volume and also result in substantial number of conversions. Hence, focus on increasing the number of conversions through these methods.
- "Lead Add Form", even though lower in volume results in maximum percentage of conversions. We should focus on increasing the volume of "Lead Add Form".

For Numerical variables, we remove outliers (values below 1% and above 99% percentiles) and chart box-plots of these variables for converted & non-converted sub-categories.

Learnings: Total time spent on a website by the user seems to have strong correlation with the conversion rate.

Check if numerical variables have any linear relationships amongst themselves and with "converted" value using a heat-map (i.e., of a correlation matrix).

Learnings: 'TotalViews' and 'Page Views per visit' are strongly correlated and one of these columns can be eliminated to avoid effects of multicolinearity.

Build Logistic Regression Model.

We first prepare our data to be usable for a logistic regression model.

- For binary (Yes/No) columns, we map it to 1/0 values.
- For other categorical columns, create dummy variables (dropping one column)

Split data into training and test (holdout) data sets: using 70/30 percent split.

Scale the numerical columns using a standard scalar.

Use RFE to filter which features (variables) are most relevant for logistic regression.

We now build a logistic regression model by iteratively:

- Train the model using statsmodel's GLM module
- Check results of GLM regression. Features with high P-value (> 0.05) are candidates for dropping. Check VIF values. Features with VIF > 3.0 are candidates for dropping.

We are finally left with a model with relevant features only.

Fine-Tuning Model's Hyper-parameter (probability threshold)

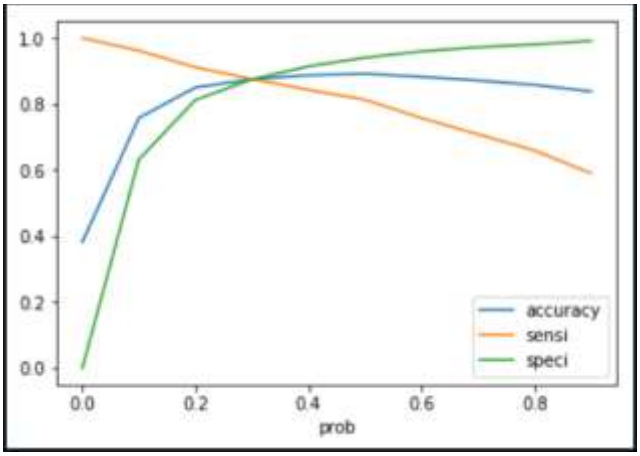
We use the training data to predict the "probability" of the conversion outcome for each training sample. Typically, a probability threshold (i.e., a cutoff) is fixed to something like 0.5 and if a lead's probability is above this cut-off, then we predict the lead will convert. The best cut-off will be the one which leads to the minimum error between (previously known) "converted" values and the new "predicted" values.

Hence, we evaluate our model for multiple values of thresholds ranging from 0, 0.1, 0.2 ... 1.0.

For each of these probability cut-off values, we can:

- Compute the confusion matrix & calculate various performance parameters like accuracy, sensitivity, specificity, TPR, FPR,
- Plot accuracy, sensitivity and specificity curves for the different probabilities.

This will show the optimal probability threshold (0.3 in this case) to consider where the curves intersect.



Prediction on Test Set & Evaluate Model Performance.

Using the final model above, use it on test data to make predictions.

Check accuracy and other performance parameters of how the model performed on this test data. We use the **confusion matrix** (and associated parameters like accuracy, specificity etc.) and **ROC curve** to evaluate the model's performance.

RESULT: The model's performance in predicting the conversion rates on the test data is just as good as on the training data. Most parameters are in the high 80's (percentage) indicating the model is pretty good and usable for production.