LEAD CONVERSION PROCESS

PROBLEM STATEMENT

- Company - X Education

The company markets its courses on several websites and search engines like Google. Once these people land on the website, they might browse the courses or fill up a form for the course or watch some videos. When these people fill up a form providing their email address or phone number, they are classified to be a lead. Moreover, the company also gets leads through past referrals. Once these leads are acquired, employees from the sales team start making calls, writing emails, etc. Through this process, some of the leads get converted while most do not. The typical lead conversion rate at X education is around 30%.

The company requires building a model wherein you need to assign a lead score to each of the leads such that the customers with higher lead score have a higher conversion chance and the customers with lower lead score have a lower conversion chance.

ANALYSIS

- Data Cleanup
  - Removal of unnecessary columns
    - The unnecessary columns like - Asymmetrique Activity Score, Asymmetrique Profile Score are removed.
  - Grouping of less frequent categorical variables as a new common category
    - The variables in Country and City have few value counts for some of the values.

    These has been replaced by “Other”
  - Removal/Imputation of the missing values
    - The variable total_visits contains the null values present. These rows are dropped.

Conversion to numerical values

- The Yes/No categorical variables are mapped to 0/1.
- The other categorical variable are mapped by creating dummies for the original categorical variable and the other variables.

```
varlist = [
                'dont_email',
                'dont_call',
                'search',
                'magazine',
                'newspaper_article',
                'x_edu_forums',
                'news_paper',
                'digital_advertisement',
                'recommndations',
                'receive_updates',
                'supply_chain_content',
                'dm_content',
                'cheque_payment',
                'free_copy'
          ]

def binary_map(x):
    return x.map({'Yes': 1, "No": 0})

data[varlist] = data[varlist].apply(binary_map)
```

```
#/*

try:
    m2 = pd.get_dummies(data['lead_source'], prefix='lead_source')
    data = pd.concat([data,m2], axis=1)
    data = data.drop(['lead_source', 'lead_source_Google'], axis = 1)
except Exception as e:
    print(e)


try:
    m3 = pd.get_dummies(data['lead_profile'], prefix='lead_profile')
    data = pd.concat([data,m3], axis=1)
    data = data.drop(['lead_profile', 'lead_profile_Potential Lead'], axis = 1)
except Exception as e:
    print(e)


try:
    m4 = pd.get_dummies(data['city'], prefix='city')
    data = pd.concat([data,m4], axis=1)
    data = data.drop(['city', 'city_Mumbai'], axis = 1)
except Exception as e:
    print(e)


try:
    m5 = pd.get_dummies(data['aai'], prefix='aai')
    data = pd.concat([data,m5], axis=1)
    data = data.drop(['aai', 'aai_02.Medium'], axis = 1)
except Exception as e:
    print(e)
```

SCALER TRANSFORMATION

- The normalisation of the values is done by using standard max scaler.
- The fit_transform() function is used to transform the variables.

```
scaler = StandardScaler()
X_train[numerical_columns] = scaler.fit_transform(X_train[numerical_columns])
X_train.head()
```

MODEL

Since, this is a classification problem, the logistic regression algorithm is used for fitting the model.

In [285]:
```
# Logistic regression model
logm1 = sm.GLM(y_train,(sm.add_constant(X_train)), family = sm.families.Binomial())
logm1.fit().summary()
```

In [ ]:
```
```

In [287]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [288]:
```
from sklearn.feature_selection import RFE
rfe = RFE(logreg, 15)             # running RFE with 13 variables as output
rfe = rfe.fit(X_train, y_train)
```

Using RFE

We use RFE with 15 variables and remove the other unnecessary variables

In [ ]:

In [287]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [288]:
```
from sklearn.feature_selection import RFE
rfe = RFE(logreg, 15)             # running RFE with 13 variables as output
rfe = rfe.fit(X_train, y_train)
```

Predicting Output

In [316]:
```
y_train_pred_final['predicted'] = y_train_pred_final.converted_prob.map(lambda x: 1 if x > 0.5 else 0)

y_train_pred_final.head()
```

Out[316]:

	converted	converted_prob	ID	predicted
0	0	0.061995	7962	0
1	0	0.241041	5520	0
2	0	0.013858	1962	0
3	1	0.865110	1566	1
4	0	0.006990	9170	0

Metrics

The confusion metrics is plotted.

In [317]:
```
from sklearn import metrics
```

In [319]:
```
confusion = metrics.confusion_matrix(y_train_pred_final.converted, y_train_pred_final.predicted )
print(confusion)
```

```
[[3801  152]
 [ 506 1913]]
```

VIF

Any variable with VIF value greater than 5 should be eliminated.

In [323]:
```
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train[col].columns
vif['VIF'] = [variance_inflation_factor(X_train[col].values, i) for i in range(X_train[col].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[323]:

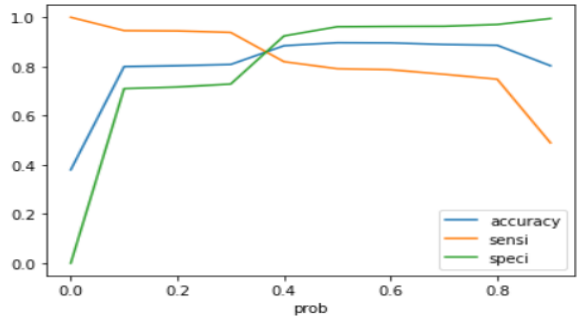
	Features	VIF
8	tags_Closed by Horizzon	1.41
0	lead_source_Welingak Website	1.36
7	tags_Busy	1.05
2	aai_03.Low	1.04
9	tags_Lost to EINS	1.04
4	last_activity_Others	1.02
3	lead_origin_Lead Add Form	0.67
6	occupation_Working Professional	0.55
12	lead_quality_High in Relevance	0.53
13	lead_quality_Low in Relevance	0.35
1	lead_profile_Other Leads	0.25
11	last_notable_activity_Modified	0.13
5	last_activity_SMS Sent	0.09
10	tags_Will revert after reading the email	0.06

Finding Optimal Cut-Off

- The optimal cutoff is found by plotting the graph of accuracy, sensitivity and specificity.

In [ ]:

In [329]:
```
# Let's plot accuracy sensitivity and specificity for various probabilities.
cutoff_df.plot.line(x='prob', y=['accuracy','sensi','speci'])
plt.show()
```



In [ ]:
```
### Cutoff 0.36
```

In [331]:
```
y_train_pred_final['final_predicted'] = y_train_pred_final.converted_prob.map( lambda x: 1 if x > 0.36 else 0)

y_train_pred_final.head()
```

Out[331]:

	converted	converted_prob	ID	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	final_predicted
0	0	0.061995	7962	0	1	0	0	0	0	0	0	0	0	0	0
1	0	0.241041	5520	0	1	1	1	0	0	0	0	0	0	0	0
2	0	0.013858	1962	0	1	0	0	0	0	0	0	0	0	0	0

Making Prediction on Test Set

In [ ]:

In [350]:
```
y_pred_final['final_predicted'] = y_pred_final.converted_prob.map(lambda x: 1 if x > 0.36 else 0)
```

In [ ]:

In [351]:
```
y_pred_final.head()
```

Out[351]:

	converted	ID	converted_prob	final_predicted
0	0	3504	0.006990	0
1	1	4050	0.999113	1
2	0	7201	0.061995	0
3	0	1196	0.006990	0
4	1	8219	0.993994	1