Analytics Vidhya Jobathon Task

By- Prashant Singh

Problem statement:

Health Insurance Lead Prediction

Your Client FinMan is a financial services company that provides various financial services like loan, investment funds, insurance etc. to its customers. FinMan wishes to cross-sell health insurance to the existing customers who may or may not hold insurance policies with the company. The company recommend health insurance to its customers based on their profile once these customers land on the website. Customers might browse the recommended health insurance policy and consequently fill up a form to apply. **When these customers fill-up the form, their Response towards the policy is considered positive and they are classified as a lead.**

Once these leads are acquired, the sales advisors approach them to convert and thus the company can sell proposed health insurance to these leads in a more efficient manner.

Now the company needs your help in building a model to predict whether the person will be interested in their proposed Health plan/policy given the information about:

- Demographics (city, age, region etc.)
- Information regarding holding policies of the customer
- Recommended Policy Information

If you want to learn more about me, feel free to browse my website:
https://prashdash112.wixsite.com/prashdash112

Train Data

Variable	Definition
ID	Unique Identifier for a row
City_Code	Code for the City of the customers
Region_Code	Code for the Region of the customers
Accomodation_Type	Customer Owns or Rents the house
Reco_Insurance_Type	Joint or Individual type for the recommended insurance
Upper_Age	Maximum age of the customer
Lower _Age	Minimum age of the customer
Is_Spouse	If the customers are married to each other (in case of joint insurance)
Health_Indicator	Encoded values for health of the customer
Holding_Policy_Duration	Duration (in years) of holding policy (a policy that customer has already subscribed to with the company)
Holding_Policy_Type	Type of holding policy
Reco_Policy_Cat	Encoded value for recommended health insurance
Reco_Policy_Premium	Annual Premium (INR) for the recommended health insurance
Response (Target)	0 : Customer did not show interest in the recommended policy, 1 : Customer showed interest in the recommended policy

Test Data

Variable	Definition
ID	Unique Identifier for a row
City_Code	Code for the City of the customers
Region_Code	Code for the Region of the customers
Accomodation_Type	Customer Owns or Rents the house
Reco_Insurance_Type	Joint or Individual type for the recommended insurance
Upper_Age	Maximum age of the customer
Lower _Age	Minimum age of the customer
Is_Spouse	If the customers are married to each other (in case of joint insurance)
Health_Indicator	Encoded values for health of the customer
Holding_Policy_Duration	Duration (in years) of holding policy (a policy that customer has already subscribed to with the company)
Holding_Policy_Type	Type of holding policy
Reco_Policy_Cat	Encoded value for recommended health insurance
Reco_Policy_Premium	Annual Premium (INR) for the recommended health insurance

Sample Submission

This file contains the exact submission format for the predictions. Please submit CSV file only.

Variable	Definition
ID	Unique Identifier for a row
Response	(Target) Probability of Customer showing interest (class 1)

Approach towards different aspects of the project's lifecycle

First things first, problem statement analysis, statistics & Eda!!!!

Before jumping to machine learning, neural nets etc. I always encourage myself to get a good grasp over the problem cause the better I'll understand it, the better I'll form the end goal hence better the result.

I started solving the problem by checking whether the dataset have any null values or not. Later on, I've started to impute the null values using iterative imputer instead of imputing with mean or median as statistic imputation is not much effective, simply because a mean or median is not a good representation of the distribution. Iterative imputer works pretty good as it iteratively fits the line to the data to minimize the error. Ordinal encoding is also done to the string features as a ML model can't process a string. I've choose ordinal over nominal because all string features have a specific order/rank for ex. City_code, health_indicator etc & it worked out pretty well for my model.

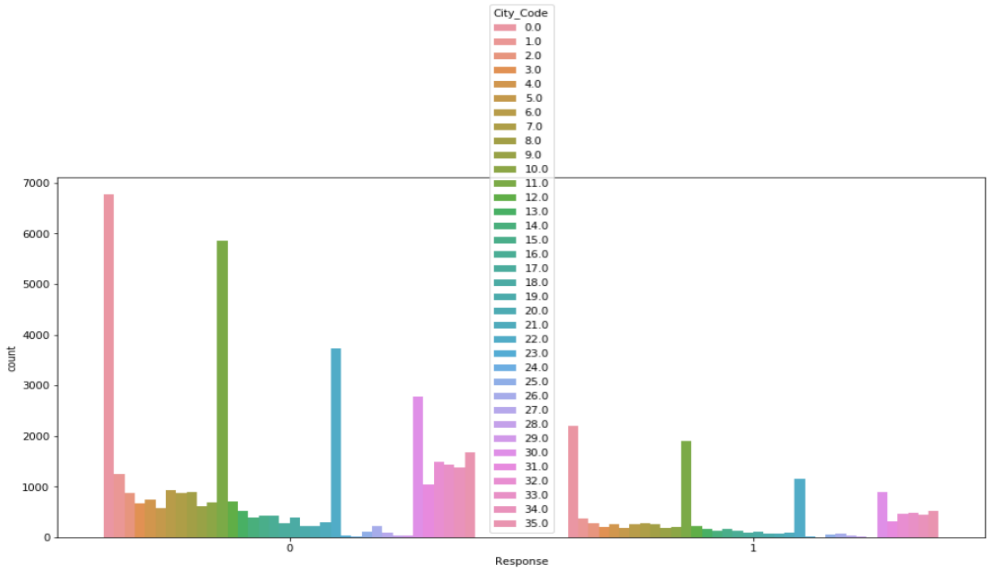
Now Comes the EDA into picture,

Doing EDA is always crucial for the end product as it gives huge insights into the data that naked eyes were unable to discover while looking at the tabular data.

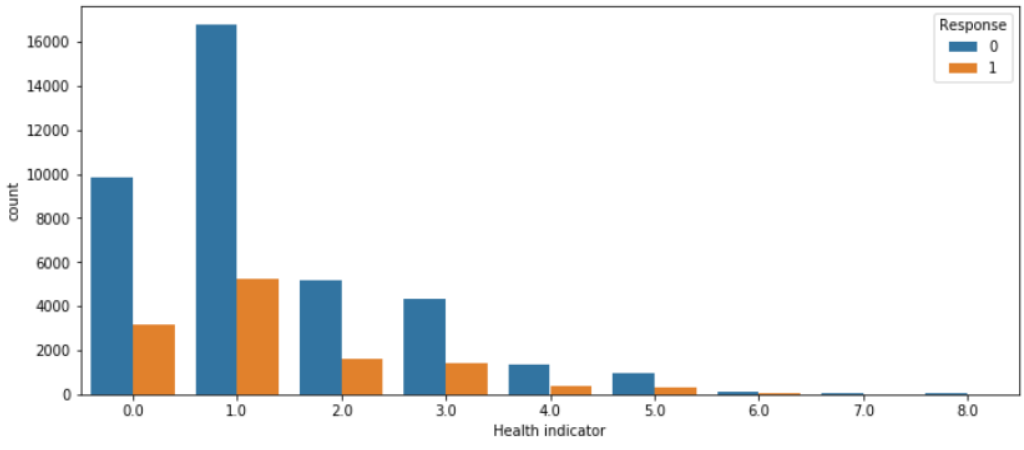
I've plotted several countplots, heatmap & a boxplot to count, measure, relate & most importantly visualize the data.

Some visualizations:

[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1fdbc4520f0>



[15]: Text(0.5, 0, 'Health indicator')



Some key observations:

- # There is a very high correlation (0.92) between upper age & lower age as for many cases only 1 induvidual is there instead of 2 or more related.

- # There's also a very strong (+ve) correlation between features upper age & Reco_policy_premium as more aged people are more prone to diseases, hence require premium policies.
- # There's also an average (+ve) correlation between is_spouse & Reco_policy_premium as it's always better to buy a premium policy if an individual have people who depend on them.
- # Majority of individuals from city codes 0-2, 11-12,30 showed no response for the recommended health policy.
- # Out of all positive responses towards the policy i.e 1, individuals from city code 0 are highest in number simply because of large sample size of people residing in that particular city & not because people are interested more in the policy.
- # People with holding policy type=1 are largest in number among all.
- # People with health indicator=1 showed the highest positive response to the recommended policy among all groups formed on the basis of health indicator.
- # Money spent over 32k is considered as an outlier as majority of people in distribution spends money less than 32k on their health policy.

Finally, model building & prototyping comes into play, I've Ideated, searched & used several model to get good predictive scores for this problem. As the data is of higher dimension i.e. significant no of features, I think it's better to use high variance & low bias models like random forest instead of low variance & high bias models like linear regression etc.

So In order to build a better predictive model, I think a good practice is to adopt the ensemble methods. So after using & tuning several models, the model that I found to work 2nd best for me is a voting regressor fitted with 3 estimators(Random forest, gradient boosting, ada boosting). Instead of killing time with tuning, I've used randomized search CV for finding best parameters for each of the mentioned 3 estimators & fitted these estimators in the voting classifier which yields the average of prediction values from each of them.

Voting regressor helps in making a high performance & a more generalized model.

```
'learning_rate': 0.1}
```

[24]:
```
# after finding the best set of tuned parameters for different estimators, here I used the best set of hyperparameters for making the final model

clf = RandomForestRegressor(n_estimators=200,
                            min_samples_split= 5,
                            min_samples_leaf= 8,
                            max_depth= 10,
                            criterion= 'mse'
                           )

gb = GradientBoostingRegressor(n_estimators= 200,
                               min_samples_split= 7,
                               min_samples_leaf= 8,
                               max_depth= 5,
                               learning_rate= 0.1,
                               criterion= 'friedman_mse'
                              )


ada = AdaBoostRegressor(random_state= 1,
                        n_estimators= 100,
                        loss= 'linear',
                        learning_rate= 0.1)
```

[25]:
```
# A voting regressor is an ensemble meta-estimator that fits base
#regressors each on the whole dataset. It, then, averages the individual
#predictions to form a final prediction.

# Inserting all tuned estimators in the voting regressor which yeilds the avg of results of individual models when fitted to data

voting_reg = VotingRegressor(estimators=[('clf', clf), ('gbrt', gb),('ada',ada)], n_jobs= -1)
voting_reg.fit(X, y)
```

The results are good but not so great!!!!!!

The best model I've tuned & deployed is lightBGM regressor. Again used the randomizedsearchCV class for finding the best hyper parameters for the lgbm regressor. Lgbm regressor constructs a gradient boosting model which is a type of machine learning **boosting**. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error.

This time results are really good

At last, saved the prediction results in the csv file.

You can also browse the code file here:

https://github.com/prashdash112/EDA-ML-DL-projects/blob/master/submission.ipynb

Thanks.

If you want to learn more about me, feel free to browse my website:
https://prashdash112.wixsite.com/prashdash112