HERIOT WATT UNIVERSITY

B39DA – Applied Machine Learning

GA – MA (Hons) IT Management for Business

Petros A. Pasatas, July 2022

Introduction

Businesses and organisations servicing large number of clients are likely expected to possess valuable data in relation to their offering, profitability, and customer insight. The company in scope for this report has been selling health insurance to individuals over an undefined period. The company has supplied a single year's customer data for investigation that will support their expansion to a new product: vehicular insurance. The company leverages the relevant probability of pay-out, charging an annual premium for exchange of guaranteeing the hospitalization costs of policyholders.

Problem Statement

Incorporating technologies that leverage historic information, an asset owned by the company, identifying the part of their customer demographic is in scope of outreach purposed to develop revenue through the vehicle insurance product.

Importance

Deploying a Machine Learning model provides the company with direct benefits; allowing for optimization of resources and operations to realize the intended benefits of launching this new product. The respective unit responsible with developing business can plan for a strategy of maximizing overall operating profit, thus spending time in developing a communication strategy relevant to the population in scope for cross-sale.

Informed business decisions and strategic priorities are enabled by the utility of a technological solution, automating the extraction of information from raw data available to the company. The solution directly benefits sales teams tasked with the distribution of the new product. Instead of investing manual effort to identify cross-sale opportunities or expanding development efforts to all current clients, a machine learning model will exclude records that are not within scope, while highlighting the predictability of response, based on the available demographics and past behaviors.

Import Libraries

```
#Import Libraries
```

```python
#Data Pre-processing variables
import pandas as pd
import numpy as np

#Data Vizualisation & Plotting
import matplotlib
import matplotlib.pyplot as plt

#Data predictive analysis sklearn & scaler.
import sklearn
from sklearn.preprocessing import StandardScaler

#Sklearn categorical encoding variable & pipeline crossing validation
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline


#Data transformers & metrics.
from sklearn.compose import ColumnTransformer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, ConfusionMatrixDisplay

#Statistical vizualisation
import seaborn as sns

# Line plot
%matplotlib inline

# Python warnings suppression
import warnings
warnings.filterwarnings('ignore')
```

Figure 1: Libraries

Datasets

Two datasets have been sourced from the health insurance company, available on Kaggle. 2 .CSV files have been imported for processing and analysis. Most records are found within the file to be used to train the model, while a smaller population is to be used to test the solution's accuracy. The files are 'train.csv' & 'test.csv'. Datasets include information on the following variables, with response rate being limited to the train dataset. Total rows in train.csv = 381,109. Total in test.csv = 127,037. (Kaggle 2020)

- ID = Unique customer ID
- Gender = Customer gender
- Age = Customer age
- Driving License = Customer driving license [0/1]
- Region Code = Customer region
- Previous Insurance = 1 has vehicle insurance, 0 has no vehicle insurance.
- Vehicle Age = Age of vehicle
- Vehicle Damage = Vehicle damaged in the past?
- Annual Premium = Amount customer needs to pay annually.
- Sales Channel = Code reflecting client outreach channel
- Vintage = Days associated with company
- Response Rate = Interest or not interested.

```
 #   Column
---  ------
 0   id
 1   Gender
 2   Age
 3   Driving_License
 4   Region_Code
 5   Previously_Insured
 6   Vehicle_Age
 7   Vehicle_Damage
 8   Annual_Premium
 9   Policy_Sales_Channel
 10  Vintage
```

Figure 2: Data columns

Data Pre-Processing

Using the Pandas library, the datasets are imported on Python where pre-processing steps are applied to ensure that irrelevant variables are removed or otherwise refined to be integrated to the model. As the first step in pre-processing the data to ensure adequate quality, the dataset is investigated for empty rows that be removed or be statistically estimated to retain information. (Brownlee 2020)

```
In [6]: # checking for null or empty column in train.csv
        health_train.isnull().sum()

Out[6]: id                      0
        Gender                  0
        Age                     0
        Driving_License         0
        Region_Code             0
        Previously_Insured      0
        Vehicle_Age             0
        Vehicle_Damage          0
        Annual_Premium          0
        Policy_Sales_Channel    0
        Vintage                 0
        Response                0
        dtype: int64
```

```
In [7]: # checking for null or empty cell or column in test.csv
        health_test.isnull().sum()

Out[7]: id                      0
        Gender                  0
        Age                     0
        Driving_License         0
        Region_Code             0
        Previously_Insured      0
        Vehicle_Age             0
        Vehicle_Damage          0
        Annual_Premium          0
        Policy_Sales_Channel    0
        Vintage                 0
        dtype: int64
```

Figure 3: Null control

A request to identify null values through Pandas across the 2 datasets displays a total 0 empty columns. To validate these results, information of the data frame is investigated, providing assurance on missing values and information on data types for each column.

```
In [8]: # Validation of null values and data type in train.csv
        health_train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381109 entries, 0 to 381108
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   id                    381109 non-null  int64
 1   Gender                381109 non-null  object
 2   Age                   381109 non-null  int64
 3   Driving_License       381109 non-null  int64
 4   Region_Code           381109 non-null  float64
 5   Previously_Insured    381109 non-null  int64
 6   Vehicle_Age           381109 non-null  object
 7   Vehicle_Damage        381109 non-null  object
 8   Annual_Premium        381109 non-null  float64
 9   Policy_Sales_Channel  381109 non-null  float64
 10  Vintage               381109 non-null  int64
 11  Response              381109 non-null  int64
dtypes: float64(3), int64(6), object(3)
memory usage: 34.9+ MB
```

```
In [9]: # Validation of null values and data type in test.csv
        health_test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 127037 entries, 0 to 127036
Data columns (total 11 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   id                    127037 non-null  int64
 1   Gender                127037 non-null  object
 2   Age                   127037 non-null  int64
 3   Driving_License       127037 non-null  int64
 4   Region_Code           127037 non-null  float64
 5   Previously_Insured    127037 non-null  int64
 6   Vehicle_Age           127037 non-null  object
 7   Vehicle_Damage        127037 non-null  object
 8   Annual_Premium        127037 non-null  float64
 9   Policy_Sales_Channel  127037 non-null  float64
 10  Vintage               127037 non-null  int64
dtypes: float64(3), int64(5), object(3)
```

Figure 4: Null validation

The data is visualized in its normal form in, followed by a statistical summary in preparation for exploratory analysis, visualizing the correlation between object and numerical variables across the two datasets, and selecting the response rate as the correlation target, observed to range between 0 and 1.

In [10]:
```
# Train Descriptive statistical summary
health_train.describe()
```

Out[10]:

	id	Age	Driving_License	Region_Code	Previously_Insured	Annual_Premium	Policy_Sales_Channel	Vintage	Response
count	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000
mean	190555.000000	38.822584	0.997869	26.388807	0.458210	30564.389581	112.034295	154.347397	0.122563
std	110016.836208	15.511611	0.046110	13.229888	0.498251	17213.155057	54.203995	83.671304	0.327936
min	1.000000	20.000000	0.000000	0.000000	0.000000	2630.000000	1.000000	10.000000	0.000000
25%	95278.000000	25.000000	1.000000	15.000000	0.000000	24405.000000	29.000000	82.000000	0.000000
50%	190555.000000	36.000000	1.000000	28.000000	0.000000	31669.000000	133.000000	154.000000	0.000000
75%	285832.000000	49.000000	1.000000	35.000000	1.000000	39400.000000	152.000000	227.000000	0.000000
max	381109.000000	85.000000	1.000000	52.000000	1.000000	540165.000000	163.000000	299.000000	1.000000

In [11]:
```
# Test Descriptive statistical summary
health_test.describe()
```

Out[11]:

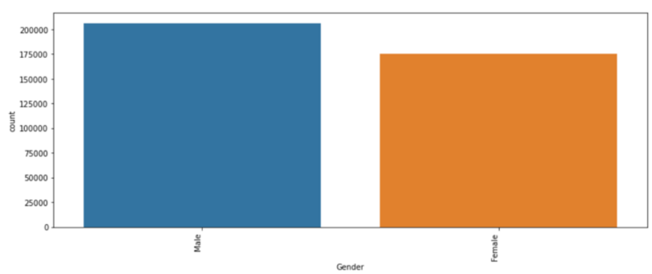
	id	Age	Driving_License	Region_Code	Previously_Insured	Annual_Premium	Policy_Sales_Channel	Vintage
count	127037.000000	127037.000000	127037.000000	127037.000000	127037.000000	127037.000000	127037.000000	127037.000000
mean	444628.000000	38.765903	0.998134	26.459866	0.460039	30524.643576	111.800468	154.318301
std	36672.567411	15.465814	0.043152	13.209916	0.498403	16945.297103	54.371765	83.661588
min	381110.000000	20.000000	0.000000	0.000000	0.000000	2630.000000	1.000000	10.000000
25%	412869.000000	25.000000	1.000000	15.000000	0.000000	24325.000000	26.000000	82.000000
50%	444628.000000	36.000000	1.000000	28.000000	0.000000	31642.000000	135.000000	154.000000
75%	476387.000000	49.000000	1.000000	35.000000	1.000000	39408.000000	152.000000	227.000000
max	508146.000000	85.000000	1.000000	52.000000	1.000000	472042.000000	163.000000	299.000000

Figure 5: Sumamry

In [53]:
```
# 'Object' Correlation analysis plot
```

In [15]:
```
def object_count_plot(health_train):
    """
    Description : Plot countplot for all categorical features present in the dataframe passed
    Argument : Dataframe
    """

    for var in health_train.columns:
        if health_train[var].dtype == 'object':
            print(health_train[var].value_counts())
            plt.figure(figsize=(12,5))
            g = sns.countplot(x=var,data=health_train)
            g.set_xticklabels(g.get_xticklabels(), rotation=90, ha="right")
            plt.tight_layout()
            plt.savefig('figure.png')
            plt.show()
object_count_plot(health_train)
```

```
Male      206089
Female    175020
Name: Gender, dtype: int64
```




```
1-2 Year     200316
< 1 Year     164786
> 2 Years     16007
Name: Vehicle_Age, dtype: int64
```

In [ ]:
```
# Numerical correlation analysis visualised plot
```

In [16]:
```
def numeric_distribution_plot(health_train):
    """
    Description : Gives distribution plot for all the numeric features in the dataframe passed
    Argument : Dataframe
    """
    for col in health_train.columns:
        if health_train[col].dtype != 'object':
            print(health_train[col].describe())
            plt.figure(figsize=(12,5))
            plt.title("Distribution of "+col)
            ax = sns.distplot(health_train[col].dropna())
            plt.tight_layout()
            plt.show()
numeric_distribution_plot(health_train)
```

```
count    381109.000000
mean     190555.000000
std      110016.836208
min           1.000000
25%       95278.000000
50%      190555.000000
75%      285832.000000
max      381109.000000
Name: id, dtype: float64
```

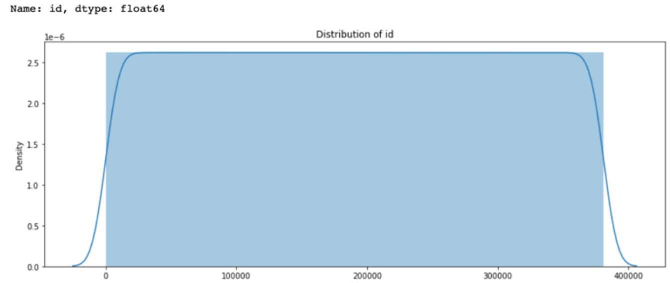


Figure 6: Analysis graphs

Conducting a multi-varied analysis and visually presenting the correlation of dataset variables with the response rate and plotting the age and previously insured correlation, along with the distribution density of values, providing insightful output components based on the correlation of the features. (Stackoverflow 2020)

Outliers

Data quality is refined through the removal of outliers that would otherwise impact the desired model's accuracy. To identify the possibility of outlier values in the dataset, numerical features are visualized using a box plot-graph that demonstrates which features appear as highly skewered. (Zheng Li 2020)

```
In [22]: # Features boxplots for each of the numerical columns
         sns.set_style('darkgrid')
         fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (20, 15))
         fig.suptitle('Box plots showing outliers', y= 0.93, fontsize = 20)

         for ax, data, name in zip(axes.flatten(), health_train, ["Vintage", "Policy_Sales_Channel", "Annual_Premium", "Age"]):
           sns.boxplot(health_train[name], ax = ax)
```

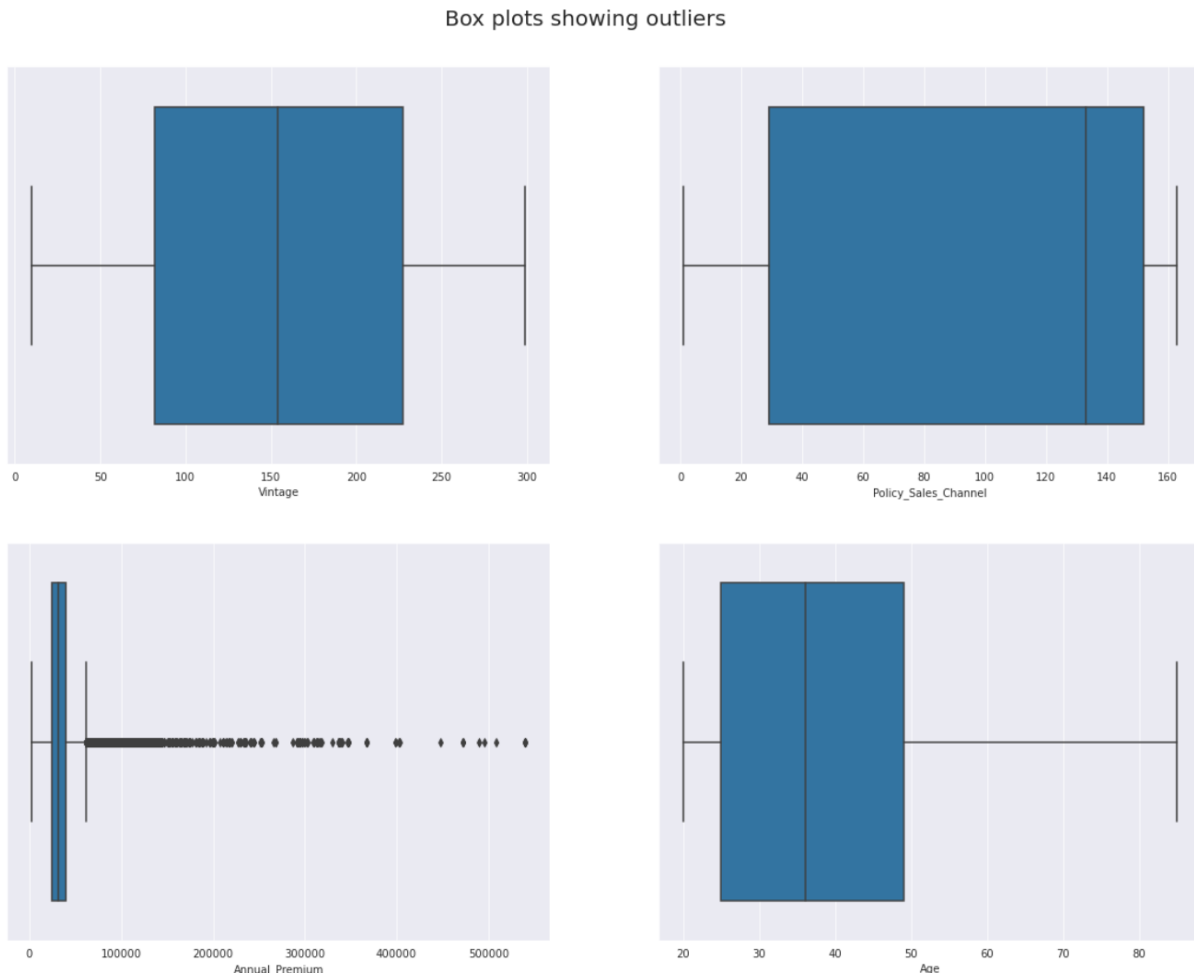
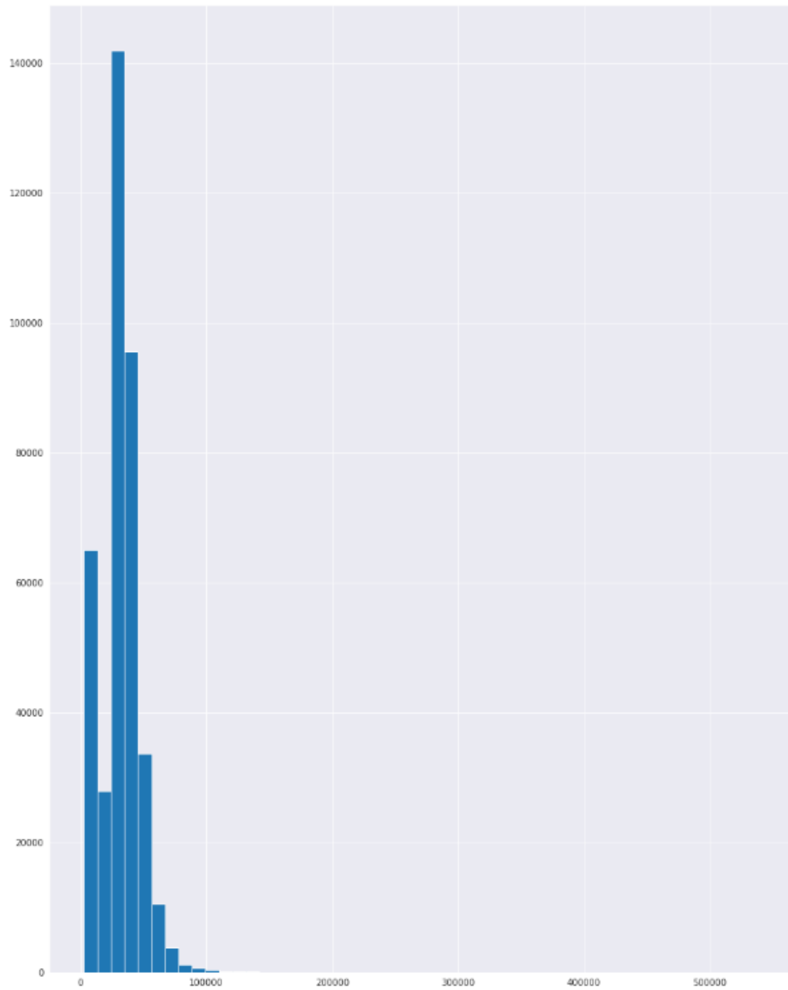


Figure 7: Outliers graphs

By observing the visual output of the box plot, outlines within the numerical features are only appearing in the annual premiums graph, displaying as being right-skewed. (Stackoverflow 2021)

```
In [23]: # Annual Premium  histogram
         health_train['Annual_Premium'].hist(figsize=(15, 20), bins=50)

Out[23]: <AxesSubplot:>
```



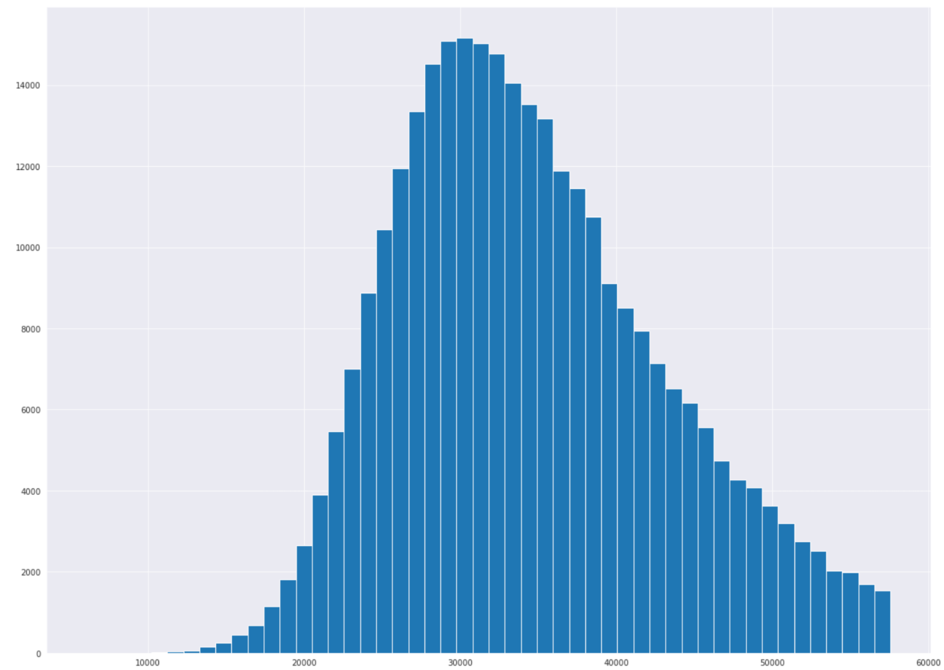
Outliers are then removed by applying the quantile range, removing values from the lowest 3% and highest 4% to avoid skewing.

```
In [25]: # Selecting the quantile in the range of 4% to 96%.

         min_threshold_1, max_threshold_1 = health_train['Annual_Premium'].quantile([0.04, 0.96])
         min_threshold_1, max_threshold_1

Out[25]: (2630.0, 57564.67999999999)

In [26]: # Quantile range for annual premium in train.

         health_train_1 = health_train[(health_train.Annual_Premium >min_threshold_1) & (health_train.Annual_Premium  < max_thre
```

Following the transformation process of annual premiums, displaying the graph again shows that the distribution of values has become more symmetric and ideal for handling.

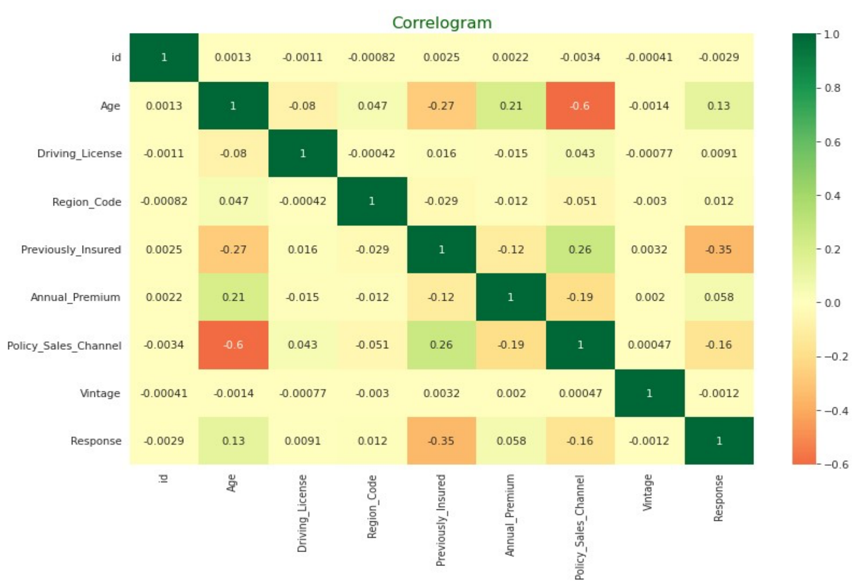
```
health_train_1['Annual_Premium'].hist(figsize=(20, 15), bins=50)
```

```
Out[29]: <AxesSubplot:>
```



Correlogram

To understand the correlation between features in the dataset, the correlogram is used to visualise the relationship between the selected target feature (customer response rate) to the remaining variables. (Stackoverflow 2021)




```
Out[32]: Response                1.000000
         Age                     0.125045
         Annual_Premium          0.058259
         Region_Code             0.011812
         Driving_License         0.009136
         Vintage                -0.001234
         id                     -0.002889
         Policy_Sales_Channel   -0.161982
         Previously_Insured     -0.352340
         Name: Response, dtype: float64
```

Figure 8: Correlation analysis graph

Data Enriching & Normalisation

In the correlation output, it is observed that previously insured and policy sales channel are not correlated to the target function and can be dropped from the set, enriching the quality of the data. The data is normalized using the StandardScaler library, creating 2 pipelines for numerical and categorical data where the data is fitted in the scaler array.

```
In [46]: # Appending columns that are fit for numerical operation and classification operations.
         # Appending them to a list.

         num_column = []
         cat_column = []
         for column in health_train_3.columns:
             if health_train_3[column].dtype != "object":
                 num_column.append(column)
             else:
                 cat_column.append(column)
         print(cat_column)
         print(num_column)

         ['Gender', 'Vehicle_Age', 'Vehicle_Damage']
         ['id', 'Age', 'Driving_License', 'Region_Code', 'Annual_Premium', 'Vintage']
```

```
In [47]: # Creating a dataframe and converting it to a list so they can work in the pipeline effectively.

         train_data_num = list(health_train_3[num_column])
         train_data_object = list(health_train_3[cat_column])
```

```
In [48]: # Encoding variables and pipeline


         num_pipeline = Pipeline([
             ("Standard scaler", StandardScaler())
         ])

         cat_pipeline = Pipeline([
             ("cat_encoder", OneHotEncoder(sparse=False)),
         ])
```

```
In [49]: # Combining the two pipelines so they can work on a dataset generally.

         full_pipeline = ColumnTransformer([
             ("num", num_pipeline, train_data_num),
             ("objects", OneHotEncoder(), train_data_object),
         ])
```

```
In [50]: # Fitting the train set to the pipeline.

         health_train_prepared = full_pipeline.fit_transform(health_train_3)
```

```
In [51]: # Transforming the validation set also.

         health_validation_prepared = full_pipeline.transform(health_validation)
```

```
In [52]: # Then transforming the test set.

         health_test_prepared = full_pipeline.transform(health_test)
```

Solution

The prepared data is then fit into different predictive learning models through open-source libraries that predicts classification of the target function of the algorithm. The accuracy measures introduced to measure the effectiveness of the model, along with the receiver operating characteristic curve statistical representation of the probability score. (Edalati 2021)

Random Forest Classifier Model

In [58]:
```
from sklearn.tree import DecisionTreeClassifier as dtc

tree_clf = dtc(max_depth=10)
tree_clf.fit(health_train_prepared, health_train_label)
health_pred = tree_clf.predict(health_validation_prepared)
print("Accuracy ", accuracy_score(health_validation_label, health_pred))
health_prob = tree_clf.predict_proba(health_validation_prepared)
print("Roc_AUC_Score ", roc_auc_score(health_validation_label, health_prob[:,1]))
```

```
Accuracy  0.8793149274062261
Roc_AUC_Score  0.8452719145713353
```

Decision Tree Classifier Model

DecisionTreeClassifier

In [58]:
```
from sklearn.tree import DecisionTreeClassifier as dtc

tree_clf = dtc(max_depth=10)
tree_clf.fit(health_train_prepared, health_train_label)
health_pred = tree_clf.predict(health_validation_prepared)
print("Accuracy ", accuracy_score(health_validation_label, health_pred))
health_prob = tree_clf.predict_proba(health_validation_prepared)
print("Roc_AUC_Score ", roc_auc_score(health_validation_label, health_prob[:,1]))
```

```
Accuracy  0.8793149274062261
Roc_AUC_Score  0.8452719145713353
```

XG Boost Classifier Model

XGBoostClassifier

In [59]:
```
import xgboost

from xgboost.sklearn import XGBClassifier
xgboost_clf = XGBClassifier(random_state=42, eval_metric='mlogloss')
xgboost_clf.fit(health_train_prepared, health_train_label)
health_pred = xgboost_clf.predict(health_validation_prepared)
print("Accuracy ", accuracy_score(health_validation_label, health_pred))
health_prob = xgboost_clf.predict_proba(health_validation_prepared)
print("Roc_AUC_Score ", roc_auc_score(health_validation_label, health_prob[:,1]))
```

```
Accuracy  0.8806272633642314
Roc_AUC_Score  0.85015514657063
```

LGBM Classifier Model

LGBMClassifier

In [60]:
```
from lightgbm import LGBMClassifier

lgb_clf = LGBMClassifier()
lgb_clf.fit(health_train_prepared, health_train_label)
health_pred = lgb_clf.predict(health_validation_prepared)
print("Accuracy ", accuracy_score(health_validation_label, health_pred))
health_prob = lgb_clf.predict_proba(health_validation_prepared)
print("Roc_AUC_Score ", roc_auc_score(health_validation_label, health_prob[:,1]))
```

```
Accuracy  0.8810259477059038
Roc_AUC_Score  0.8536414883600165
```

Gradient Boosting Classifier Model

GradientBoostingClassifier

In [61]:
```
from sklearn.ensemble import GradientBoostingClassifier

gbc_clf = GradientBoostingClassifier()
gbc_clf.fit(health_train_prepared, health_train_label)
health_pred = gbc_clf.predict(health_validation_prepared)
print("Accuracy ", accuracy_score(health_validation_label, health_pred))
health_prob = gbc_clf.predict_proba(health_validation_prepared)
print("Roc_AUC_Score ", roc_auc_score(health_validation_label, health_prob[:,1]))
```

```
Accuracy  0.8810591714010432
Roc_AUC_Score  0.853314628935259
```

Results

Observing the accuracy output from the different models that were applied on the dataset show a small score difference across the used algorithms. A higher accuracy score is preferred. The maximum accuracy parameter in the model sits at 88%, while the receiver operator curve score consistently sits near 85%.

Limitations

Although the model takes scientific approach in determining the population in scope for cross-sale opportunities based on the correlation of features following the noise reduction and identification of outlier, the effectiveness of this solution must be tested based on the performance information to understand its impact on the communicated problem statement. Correlation, even though at the heist accuracy standards, may prove to be of limited relevance of the customer interest in purchasing the offered product.

Conclusion

Comparing between the different models that were trialed in this solution, it is outlined that the performance rate is closely similar, with Gradient Boosting being the most accurate with the highest evaluation metrics of area under the characteristic operator. The model is trialed on the test dataset, storing the output in the Pandas data frame.

As the correlation between features is low, it is most effective to identify correlation with previous response rates – information available in the collated dataset. Optimizing the business strategy to include digital objectives may become useful in the future, permitting the handling of additional demographic data that will enhance the predictable classification of records or allow for alternative machine learning techniques to address the defined problem statement.

The company can use this model in enabling sales operational strategy to address individuals that are more likely to provide a higher rate of return when compared with blind efforts of engagement. The more data features are stored by the company, the model can be refined to be more accurate and effective as new information is introduced.

References

1. Li, Y. Zhao, N. Botta, C. Ionescu and X. Hu (2020), "COPOD: Copula-Based Outlier Detection," IEEE International Conference on Data Mining (ICDM), 2020, pp. 1118-1123.

2. Anmol Kumar (2020), Health Insurance Cross-Sale Prediction. Available at: https://www.kaggle.com/datasets/anmolkumar/health-insurance-cross-sell-prediction [Accessed 20/06/2022]

3. Brownlee, J (2020) "Data Preparation for Machine Learning", 2020, pp. 18 -76.

4. Trenton McKinney (2020) Data Analysis with python using numpy. Plot graph with Matplotlib. Available at: https://stackoverflow.com/questions/62964506/data-analysis-with-python-using-numpy-plot-graph-with-matplotlib. [Accessed 23/06/2022]

5. daniël.vandijk (2021), "How to remove outliers correctly and visualize right skewed data clearly white piping?". Available at:https://stackoverflow.com/questions/67092446/how-to-remove-outliers-correctly-and-visualize-right-skewed-data-clearly-while-p [Accessed 23/06/2022]

6. NicoH (2021) "Plot correlation matrix using Pandas". Available at: https://stackoverflow.com/questions/29432629/plot-correlation-matrix-using-pandas [Accessed 23/06/2022]

7. Edalati, M., Imran, A.S., Kastrati, Z. and Daudpota, S.M., 2021, September. The potential of machine learning algorithms for sentiment classification of students' feedback on MOOC. In Proceedings of SAI Intelligent Systems Conference (pp. 11-22)