1. **Problem to be solved.**

   What is the prediction problem to be solved?

   Who are the real (or hypothetical) users / beneficiaries of a solution?

   Answer: Insurance Lead Prediction

   - ✓ The Problem:
     - Insurance Lead Prediction: It is a binary classification problem where a positive outcome is classified as "1" (Lead) and any other outcomes will be classified as "0". When a person visits an insurance company website, a policy will be offered and if the person accepts the request and applies online for an insurance policy, then it is treated as a positive outcome.
     - The data consists of parameters of the persons who accepted the policy (ex: upper age, lower age) and already holding policy duration, policy type. There are other geographical parameters like city code, region code, accommodation type etc. It also has insurance details like reco policy category, reco policy premium.
     - Using the above-mentioned data, we should train a model to understand what kind of parameters of a person are involved in getting a positive outcome (policy accepted) or a negative outcome and should be able to classify the data.
   - ✓ Intended beneficiaries/users are:
     - The primary user of this information is the insurance company in the region/country. It enables them with information that can help them in providing related insurance policy offerings to the new customers in the future.
     - The intended beneficiaries can also be the insurance company as it helps them for predicting the future and at the same time, it is beneficiary for the people who want to take an insurance policy as it will give idea about what kind of parameters they should focus on before/after taking an insurance policy.

2. **Data set properties.**

   What is the source of your data set? Include a citation that specifies at least the author(s) and URL.

   Data set profile: number of items, class distribution, type of features, min/max/mean/mean or distribution for each feature, etc.

   Answer:

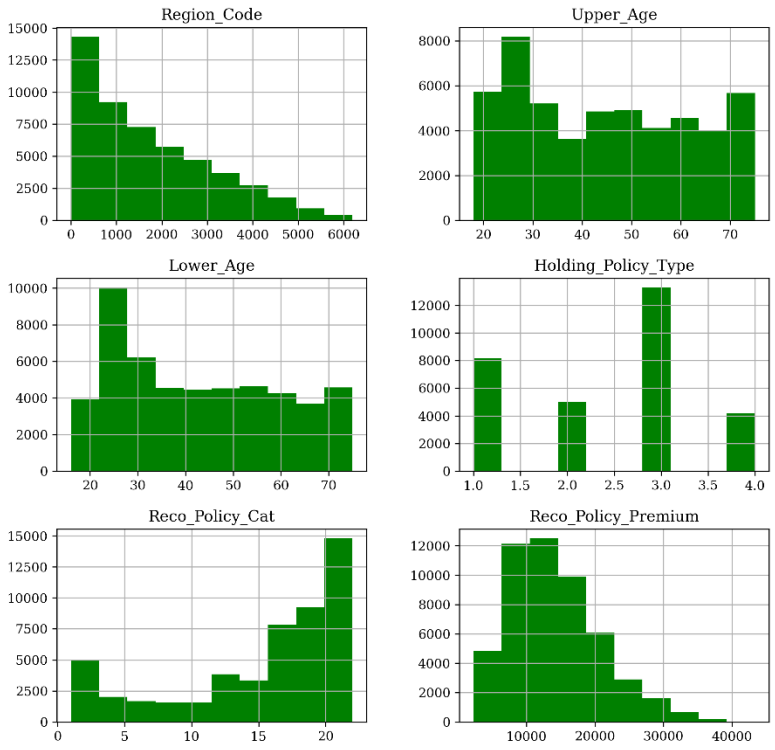
   - ✓ Source of the data:
     - Author: Owais Ahmad
     - Kaggle – URL is provided below. https://www.kaggle.com/datasets/owaiskhan9654/health-insurance-lead-prediction-raw-data
   - ✓ Initial data set profile:
     - Number of items: 50,882
     - Number of columns: 14 → 1 ID column + 12 Features + 1 target label
     - The numerical features in the dataset are:
       - Upper_Age, Lower_Age, and Reco_Policy_Premium

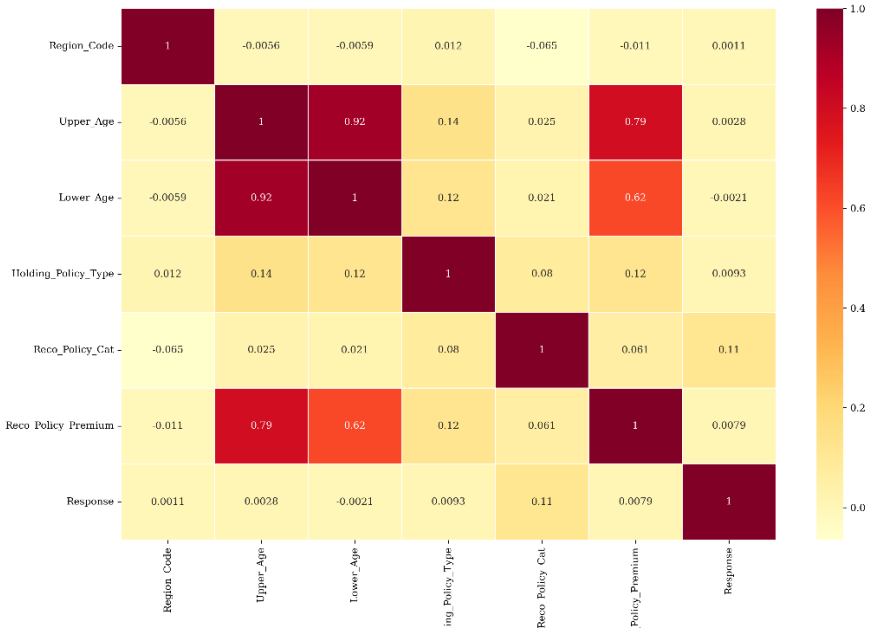
- Features like Region_Code, Holding_Policy_Type, Reco_Policy_Cat looks like numeric features, but these features are categorical features, and I will be handling these during my challenges & strategies.

- Shape, Datatype, Mean, Minimum, Median and Max Values from the python file print statements are as follows.

```
**********************************************************
The shape of the dataset is:  (50882, 14)
**********************************************************
ID                          int64
City_Code                  object
Region_Code                 int64
Accomodation_Type          object
Reco_Insurance_Type        object
Upper_Age                   int64
Lower_Age                   int64
Is_Spouse                  object
Health Indicator           object
Holding_Policy_Duration    object
Holding_Policy_Type       float64
Reco_Policy_Cat             int64
Reco_Policy_Premium       float64
Response                    int64
dtype: object
**********************************************************
                            mean     min    median      max
Upper_Age              44.856275    18.0     44.0      75.0
Lower_Age              42.738866    16.0     40.0      75.0
Reco_Policy_Premium  14183.950069  2280.0  13178.0  43350.4
**********************************************************
```

- <u>Histogram and Heatmap</u>





3. Machine learning model(s).

What type of model did you use? Why did you choose this model type?
What are some strengths and weaknesses of this model type?
What parameters must be specified, what values did you choose, and how did you choose them and/or what range of values did you explore for each parameter?

Answer:

- ✓ Model:
  - I have selected Random Forest as the machine learning model for my problem.
  - It is one of the popular supervised learning algorithms for classification problems. To overcome the overfitting issues, this algorithm uses randomness for improving accuracy. It creates decision trees based on random selection of data samples and through voting it selects the best solution. So, I have selected this model for my binary classification problem.

- ✓ Strengths and Weakness of the model:
  - Strengths:
    - Random forest is a significant and robust model when compared with most of the non-linear classifiers, I understand that this is because the model uses multiple decision trees before concluding to a result.
    - The classifier hardly overfits as it works on the approach of taking average of all the predictions which actually helps in reduction of bias as well.
    - Normalization of the data may not be required as the classifier uses rule-based approach instead of distance calculation.
  - Weakness:
    - It requires more computational power as it combines many trees unlike the decision tree classifier.

- It takes more training time as the classifier combines many decision trees and makes decision for determining the class label based on the majority votes.
- The model is very difficult to interpret as the functioning is similar to a black box algorithm where we have very little control over the internal process of the classifier.

✓ Hyper Parameters:

- Parameters Used:
  - N estimators: It is the number of trees in the forest. I used the default value 100 first and by hyper parameter tuning, I have set this value to 250.
  - Criterion: When the split happens, this parameter measures the quality of the split. I have used the default value "gini". Although "entropy" and "gini" work internally in similar fashion for measuring the quality of the split, I understand that "gini" has better computing power than "entropy". So, for this reason I have used the default value "gini" provided by the random forest classifier.
  - Random State: To generate consistent results when the algorithm is running multiple times, we should not have any shuffling of the data. Hence, I have set the random state to a fixed number and used the same random state when running the experiments multiple times.
  - Max Depth: I am using the default value as "None" which means the nodes will get expanded until all the leaves are pure or until all leaves contain less than min_samples_split samples.
- Hyper Parameter Tuning:
  - I have tried to tune the hyper parameters n_estimators and max_depth. After initial trials of hyper parameter tuning with challenge-1, I did the hyper parameter tuning at the last stage of my project (challenge-1+challenge-2+challenge-3). So, the logic is placed at the end of the code.
  - Once I found the best combination of hyper parameters, I have commented the hyper parameter tuning logic, and for running this logic, the specific code should be uncommented. I have made comments in the code to see this logic.
  - Using n_estimators = [100, 250] and max_depth = [None, 10, 20], I have tried the combination of these parameters to find the best combination.
  - Using the training set, I have used the cross-validation (RepeatedStratifiedKFold) and used the GridSearchCV approach to find the best combination of these parameters. The result is as follows.
  - Reference: sklearn.model_selection.GridSearchCV — scikit-learn 1.2.2 documentation

```
Best: 0.835873 using {'max_depth': None, 'n_estimators': 250}
0.834384 (0.006314) with: {'max_depth': None, 'n_estimators': 100}
0.835873 (0.005884) with: {'max_depth': None, 'n_estimators': 250}
0.786137 (0.008261) with: {'max_depth': 10, 'n_estimators': 100}
0.787374 (0.007982) with: {'max_depth': 10, 'n_estimators': 250}
0.832592 (0.006241) with: {'max_depth': 20, 'n_estimators': 100}
0.833574 (0.005775) with: {'max_depth': 20, 'n_estimators': 250}
```

- From the above result, we can see the 'mean_test_score' – 0.835873 is the highest when the combination is n_estimators = 250 and max_depth = None. The value (0.005884) is the 'std_test_score'.
- So, for the overall code, I have used this best combination as my hyper parameters.

4. **Evaluation.**

What metrics did you use to measure performance?
What experimental methodology did you employ to construct training, testing, and evaluation data sets?
What baseline approach(es) did you compare your model to?

Answer:

- ✓ Metrics:
  - I am currently using accuracy as my metric to understand the performance of my model. I am also generating the confusion metric after completion of each challenge with the best strategy to understand the false negative and false positive impacts of the model. For each challenge+strategy, I am calculating the values of precision, recall and f-1 score. The values and the observations are shared in the results section.
- ✓ Methodology for construction of datasets:
  - I am using single train/test random split approach for splitting the train and test sets. The considered dataset is a large dataset, so I think this approach suits well and it is faster to compute. I am using the 80-20 approach for train-test split.
- ✓ Baseline Approach:
  - I am using the Dummy Classifier with strategy as "stratified" as the baseline classifier – Using the class probabilities from the training set as the baseline for the given dataset.

5. **Challenges.**

Describe at least three challenges you've found with this data set (outliers, missing values, range/scale/units for features, sampling bias, correlations that make the data not i.i.d., etc.) and/or the application area (deployment, maintenance, etc.).
For each challenge, describe three alternative strategies that you investigated.

Answer: The three challenges I have found with this dataset are as follows:

Handling Categorical Data:

- ✓ In real world dataset, we encounter data which is not numeric and is in text(alphabets/letters). In general, we can say that most of the machine learning models are mathematical models, so giving input in terms of numbers may help the model to understand the pattern in a better way.
- ✓ In the current dataset, features like City_Code, Accomodation_Type, Reco_Insurance_Type, Is_Spouse, Health Indicator are very clear alphanumeric/letters. These all can be considered as categorical data and need to be handled in a numeric way. Additionally, features like Region_Code, Holding_Policy_Type, Reco_Policy_Cat do look like numerical features, but these features also represent some information as a categorical

feature. So, based on the machine learning model these features as well need to be handled in a numeric way.

- ✓ I have handled the features City_Code, and Health Indicator by stripping the value of the alphabetic part and then handled the features as numeric.
- ✓ For the features Accomodation_Type, Reco_Insurance_Type, Is_Spouse I have handled using the strategies mentioned in this document below.

Missing Values:

- ✓ When we work with real world data, there will be instances where the data is not present in the dataset due to various reasons such as failure in collection of data, data not extracted or transferred correctly, missed data due to a valid reason etc. We call such absence of data as missing values. Technically the missing values can be attributed as MCAR, MAR, NMAR.
- ✓ In the given dataset, there are missing values in 3 of the features - Health Indicator, Holding_Policy_Duration, and Holding_Policy_Type. The number of missing values for these 3 features in my dataset out of 50, 882 rows(instances) is as follows.

```
Health Indicator          11691
Holding_Policy_Duration   20251
Holding_Policy_Type       20251
```

- ✓ By the observations of the features and its correlation, we can see that absence or presence of both "Holding_Policy_Duration" and "Holding_Policy_Type" at the same time does make sense as both are interlinked in the scenario where the customer has an existing policy. The Health Indicator feature may be considered as MAR since by manual observation of the data, we don't see any correlation between other features.

Imbalance in the dataset:

- ✓ When we work with any real-world data, we always observe that the sample data in most cases will not be balanced. There can be a certain class label which has more rows in comparison to other class labels. We can term such a scenario as Imbalance in the dataset.
- ✓ In most cases, we can't find or need exactly 50%, and 50% of data when there are 2 labels but any ratio like 70-30, 80-20 etc. can be treated as imbalanced dataset and should be addressed properly to have the right balance.
- ✓ Out of 50,882 instances, there are 38,673 instances with class label response as "0". This is almost equal to 76% of the data. Surely, this is an example of imbalance in the dataset, and I will be working on handling this imbalance in the data.

Challenge 1: Handling Categorical Data

- ✓ *Plan 1: Label encoding.*
  - For the categorical data, we can replace the categorical value with a numeric value. For example, if a particular feature has 4 distinct classes, then we can map each distinct class to a particular numeric value. Based on the importance of the class, we can assign the numeric values to the classes in ascending or descending order, or a particular value based on the context. Usually, the possible qualitative kind of data

of categorical data are 'Nominal' – Variables with no inherent order or ranking sequence. Ex: Gender(Male/Female) and 'Ordinal' – Variables with an ordered series. Ex: Blood Group, Flag(Yes/No), Degree(Bachelor of Science/Master of Science).

- In my current dataset, I initially thought my categorical data has both nominal and ordinal. So, I mentioned that if it is ordinal type, then based on the importance of class we can assign values like for Degree(Bachelor of Science/Master of Science) – 0 and 1 where 1 is considered to be more important. Since I mentioned above about importance of each value, I explained this about ordinal data. But my current dataset has only nominal type of categorical data, so each categorical value is assigned with an integer value based on alphabetical order.
- This label encoding can be performed either by using the scikit-learn labelencoder library or by using category codes. I have used scikit-learn labelencoder.

✓ *Plan 2: One hot encoding.*

- For a given categorical feature, we will have different categories. In this method, we will create a new column with binary encoding using 0, 1 to represent if for a particular instance(row) belongs to this category or not. This can be done using scikit-learn library or using pandas library. I have used the scikit-learn library.
- This is a technique which works well when categorical data has a smaller number of unique categories for a feature. For my dataset, this works well, and I have applied the one hot encoding for the categorical features 'Accomodation_Type', 'Reco_Insurance_Type', and 'Is_Spouse'.

✓ *Plan 3: Count/Frequency encoding.*

- In this approach, I can replace the categorical data with the count(frequency) of the particular category value in that feature(column). Instead of frequency, I can also use percentage. I have used the count here. I have used the count/frequency from the train dataset and applied the same for train dataset and test dataset as well.
- This may not give good results when two types of categories in a feature have same frequency and we may not understand the importance or difference of these categories in the feature.

Challenge 2: Missing Values

✓ *Plan 1: Delete the observations(rows) which have missing values.*

- The idea is to delete the instances (rows) from the dataset which have missing values. This may or may not be the efficient way to do it but definitely one of the easiest techniques to experiment with and compare with other ways. Out of 50,882 observations, a total of 27,334 observations will be deleted and the remaining observations are equal to 23,548.

✓ *Plan 2: Delete the features(columns) which have missing values.*

- I can also delete the features (columns) having the missing values. If the features with missing values are of no such importance, then this approach may work well for the model. The features which can be deleted are 3 - Health Indicator,

Holding_Policy_Duration, and Holding_Policy_Type. So, we will be left with 9 features.

- ✓ *Plan 3: Replace the missing values with the most frequent value.*
  - We can fill the missing value with some value and one of the possible ways is to fill the value with the "most frequent" value. I have used the scikit-learn SimpleImputer with strategy as "most_frequent".

Challenge 3: Imbalance in the dataset

- ✓ *Plan 1: Under Sampling – Down Sampling the majority class.*
  - In this method, we will randomly remove instances (rows) of the majority class to reduce the count and make it equal with the other class. It is also called down sampling. The initial sample size is Class Label: 0 – 14,264 and Class Label: 1 – 4574 and the dataset size after random under sampling is Class Label: 0 – 4574 and Class Label: 1 – 4574.
  - Drawback of this approach is that we might lose important information from the instances we delete from the dataset. I can use scikit-learn resample method or imblearn under sampling. I have used the imblearn under sampling technique.
- ✓ *Plan 2: Over Sampling – Up Sampling the minority class.*
  - Over sampling is also called up sampling where we will increase the count of the instances(rows) of the minority class to reduce the imbalance in the dataset. I have used the random over sampling approach. I have used the imblearn over sampling technique. The initial sample size is Class Label: 0 – 14,264 and Class Label: 1 – 4574 and the dataset size after random over sampling is Class Label: 0 – 14,264 and Class Label: 1 – 14,264.
  - One potential risk with this approach can be the increment in the duplicate data in the dataset which may have impact on the fit of the model.
- ✓ *Plan 3: SMOTE(Synthetic Minority Over Sampling Technique) – Create synthetic samples.*
  - This is improved version of up sampling. In this technique instead of generating duplicate data in the dataset, we will vary the attributes for the creation of fresh synthetic samples. I have used the imblearn SMOTE technique. The initial sample size is Class Label: 0 – 14,264 and Class Label: 1 – 4574 and the dataset size after random over sampling is Class Label: 0 – 14,264 and Class Label: 1 – 14,264.
  - Reference for SMOTE and other under/over sampling techniques is as below.
    SMOTE — Version 0.11.0.dev0 (imbalanced-learn.org)

6. **Results.**

For each challenge:

Show your experimental results (probably a table comparing the strategies for this challenge with one or more metrics).

Discuss which strategy(ies) were most effective for this challenge.

Answer:

- ✓ Experimental Results:
  - Baseline: I have used the dummy classifier with strategy "stratified" as my baseline.

```
Test Accuray with baseline classifier is: 63.18
```

- Challenge 1 (Handling Categorical Data – Representation Challenge):
  - In my dataset, the challenges of categorical data and missing values are interlinked. To handle my first challenge categorical data, I also need to address missing values for the time being. So, firstly I have filled all the missing values with "zeroes" and then handled the categorical data challenge. The overall accuracy table for all the three strategies for challenge-1 is as follows:

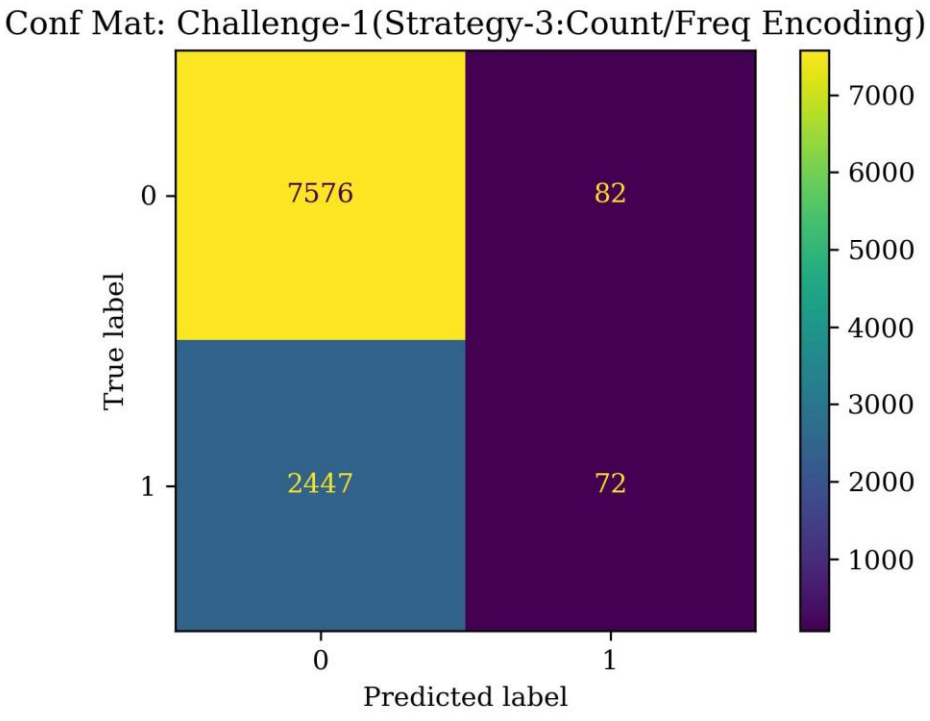
Challenge-1(Categorical Data(Representation Challenge))	Baseline	Strategy-1:Label Encoding	Strategy-2:One Hot Encoding	Strategy-3:Count/Frequency Encoding
Accuracy	63.18	75.13	75.06	75.15

- Precision: It's the ability not to label a negative sample as positive. Precision = True Positive / (True Positive + False Positive)
- Recall: It's the ability to find all the positive samples. Recall = True Positive / (True Positive + False Negative)
- F-1 Score: Harmonic mean of the precision and recall. F-1 score = (2 x Precision x Recall) / (Precision + Recall)
- Support: Number of occurrences of each class in "Y_true" of test set.

  - Classification Reports: Challenge-1: Strategies-1, 2, 3:

```
************************************************************
Classification Report for Challenge-1: Strategy-1(Label Encoding) is as below
************************************************************
              precision    recall  f1-score   support

           0       0.76      0.99      0.86      7658
           1       0.46      0.03      0.05      2519

    accuracy                           0.75     10177
   macro avg       0.61      0.51      0.45     10177
weighted avg       0.68      0.75      0.66     10177

************************************************************
************************************************************
Classification Report for Challenge-1: Strategy-2(One Hot Encoding) is as below
************************************************************
              precision    recall  f1-score   support

           0       0.75      0.99      0.86      7658
           1       0.43      0.02      0.04      2519

    accuracy                           0.75     10177
   macro avg       0.59      0.51      0.45     10177
weighted avg       0.67      0.75      0.65     10177

************************************************************
************************************************************
Classification Report for Challenge-1: Strategy-3(Count/Frequency Encoding) is as below
************************************************************
              precision    recall  f1-score   support

           0       0.76      0.99      0.86      7658
           1       0.47      0.03      0.05      2519

    accuracy                           0.75     10177
   macro avg       0.61      0.51      0.46     10177
weighted avg       0.68      0.75      0.66     10177

************************************************************
```

- Discussion of experimental results – Challenge 1:
  - Although there won't be any significant impact in the strategies for this scenario, I applied the strategies after splitting the train and test datasets just to make sure that there won't be any information leakage from test to train dataset. I understand that I can do the encoding before or after the train-test split. For my third strategy of count/frequency encoding, it is important to perform this encoding after train-test split as the count/frequency is based on the occurrence of a particular value in a feature in that dataset (test or train). So, I followed the encoding process for all the three strategies after splitting the train-test dataset once.
  - In my dataset, I have 5 features with categorical data, in which 2 features have alphanumeric representation where the values are in the form of C1, C2 etc. for one feature and X1, X2 etc. for another feature. For these 2 features, I have handled the data by stripping the alphabet of C, X which then be easily converted to numeric values. So, I have handled the remaining 3 categorical features as below.
  - For the representation (categorical data) challenge, I tried the label encoding, one hot encoding and count/frequency encoding strategies.
  - Firstly, I can confirm that all the three strategies with the random forest classifier produced better accuracy results than the baseline classifier.
  - In terms of accuracy, although there isn't any significant difference for the three strategies (considering the hyper parameter tuning as well), I see that ***strategy-3: count/frequency encoding*** performed (75.15%) slightly better than the other two strategies.
  - I have generated the confusion matrix for the best strategy(count/frequency encoding) of challenge-1 as below.



- When I am trying these strategies, I read and understood that although count/frequency encoding is a good technique, it works better when there are many categories in a particular feature. In my present dataset with categorical features, I have mostly 2 categories in each feature. Still, I see that this approach worked little better than the other approaches.

- Challenge-1(Count/Frequency Encoding) + Challenge-2 (Missing Values):
  - I have picked the best strategy (count/frequency encoding) from challenge-1 and mixed it with the missing values challenge with each of its three strategies. The overall accuracy table for all the three strategies for challenge-1 + challenge-2 is as follows:

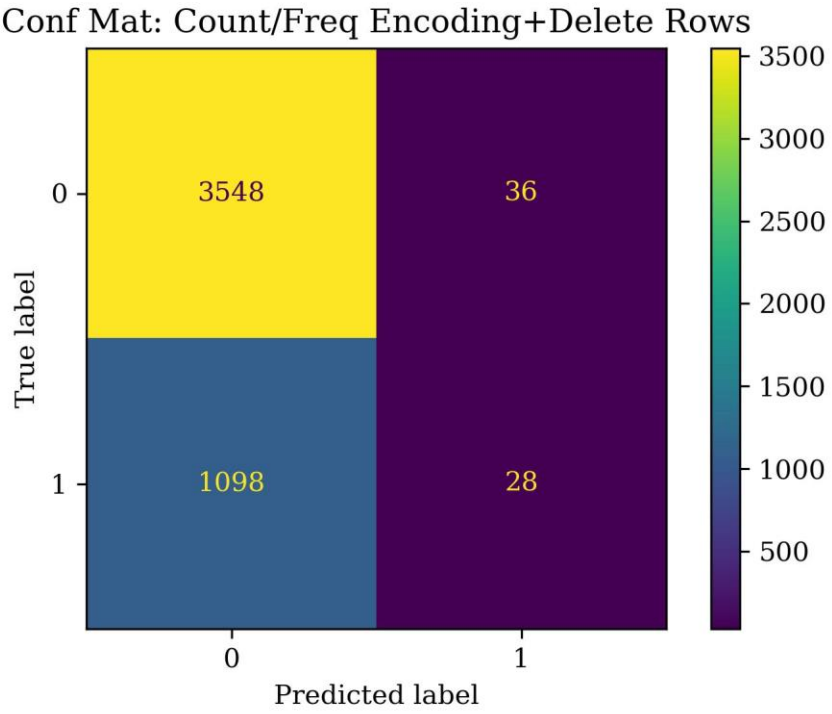
Count/Frequency Encoding + Challenge-2(Missing Values)	Baseline	Strategy-1:Delete Rows(Observations)	Strategy-2:Delete Columns(Features)	Strategy-3:Fill With Most Frequent Value
Accuracy	63.18	75.92	74.67	75

  - Classification Reports: Challenge-2: Strategies-1, 2, 3:

```
**********************************************************
Classification Report for Challenge-1(Count/Frequency Encoding) + Challenge-2: Strategy-1(Delete the rows) is as below
**********************************************************
              precision    recall  f1-score   support

           0       0.76      0.99      0.86      3584
           1       0.44      0.02      0.05      1126

    accuracy                           0.76      4710
   macro avg       0.60      0.51      0.45      4710
weighted avg       0.69      0.76      0.67      4710

**********************************************************
**********************************************************
Classification Report for Challenge-1(Count/Frequency Encoding) + Challenge-2: Strategy-2(Delete the columns) is as below
**********************************************************
              precision    recall  f1-score   support

           0       0.76      0.97      0.85      7658
           1       0.42      0.06      0.11      2519

    accuracy                           0.75     10177
   macro avg       0.59      0.52      0.48     10177
weighted avg       0.68      0.75      0.67     10177

**********************************************************
**********************************************************
Classification Report for Challenge-1(Count/Frequency Encoding) + Challenge-2: Strategy-3(Replace with most frequent) is as below
**********************************************************
              precision    recall  f1-score   support

           0       0.76      0.99      0.86      7658
           1       0.43      0.03      0.05      2519

    accuracy                           0.75     10177
   macro avg       0.59      0.51      0.46     10177
weighted avg       0.67      0.75      0.66     10177

**********************************************************
```

- Discussion of experimental results – Challenge 1 + Challenge 2:
  - First, I tried to delete the observations which are having missing values. I have deleted 27,334 observations from the dataset and then applied the best strategy of challenge-1 (count/frequency encoding).

- Secondly, I tried to delete the features which have missing values. Out of 12, 3 features Health Indicator, Holding_Policy_Duration, and Holding_Policy_Type have missing values and then applied the best strategy of challenge-1 (count/frequency encoding).
- Lastly, I applied the best strategy of challenge-1 and then I tried to impute the missing values with the most frequent value of the feature. I have used the simple imputer with "most_frequent" strategy.
- Overall, I can see that ***strategy-1: (delete the rows)*** has resulted in better accuracy(75.92%) than the other two strategies. Although, there is some loss of data, I see that I can still take this as the best strategy as the next challenge is to handle the imbalance in the dataset. Filling with most frequent value can also be a good strategy but deleting the feature may not be a good strategy as I think, by deleting the features we might lose some important information from these features if the deleted features have any important information in deciding the prediction label.
- I have generated the confusion matrix for the best strategy(count/frequency encoding) of challenge-1 + best strategy(delete the rows) of challenge-2 as below.



- Overall, the strategies of challenge-1 + challenge-2 together yielded accuracies better than the baseline accuracy (63.18%) and also each of the strategies of challenge-2 yielded better accuracies than the baseline accuracy.

- Challenge-1(Count/Frequency Encoding) + Challenge-2 (Delete the Rows) + Challenge-3 (Imbalance in the dataset):
  - I have picked the best strategy (count/frequency encoding) from challenge-1, best strategy(delete the rows) and mixed it with the imbalance in the dataset challenge with each of its three strategies. The overall accuracy table for all the three strategies for challenge-1 + challenge-2 + challenge-3 is as follows:

Count/Frequency Encoding + Delete Rows + Challenge-3(Imbalance in the dataset)	Baseline	Strategy-1:Random Under Sampling	Strategy-2:Random Over Sampling	Strategy-3:SMOTE(Over Sampling)
Accuracy	63.18	54.65	74.23	75.03

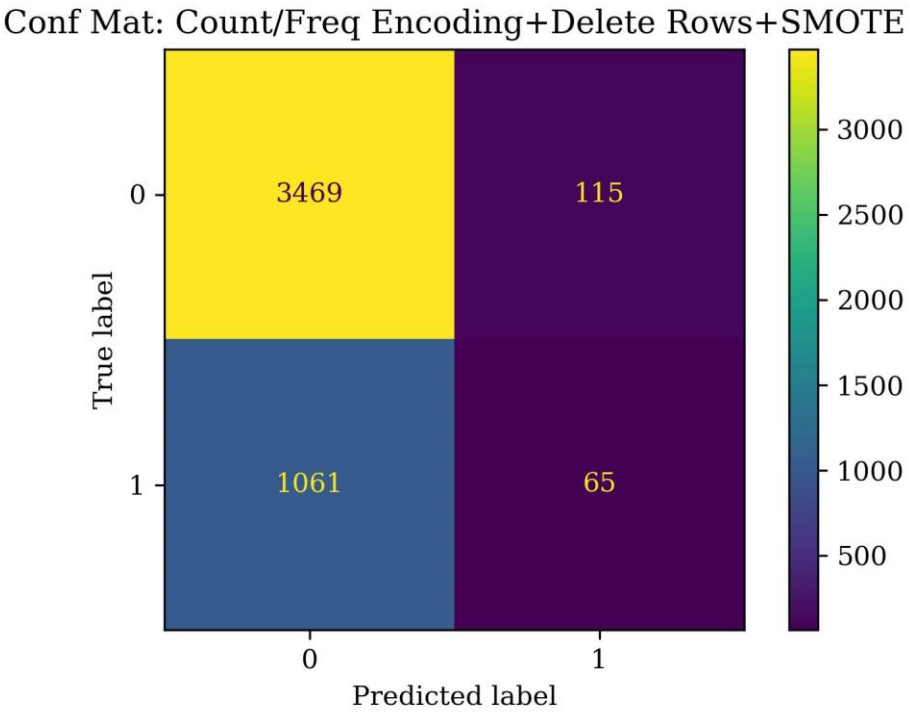
  - Classification Reports: Challenge-3: Strategies-1, 2, 3:

```
**********************************************************
Classification Report for Challenge-1(Count/Frequency Encoding) + Challenge-2(Delete the rows) + Challenge-3: Strategy-1(Under Sampling) is as below
**********************************************************
              precision    recall  f1-score   support

           0       0.81      0.53      0.64      3584
           1       0.29      0.60      0.39      1126

    accuracy                           0.55      4710
   macro avg       0.55      0.56      0.51      4710
weighted avg       0.68      0.55      0.58      4710

**********************************************************
**********************************************************
Classification Report for Challenge-1(Count/Frequency Encoding) + Challenge-2(Delete the rows) + Challenge-3: Strategy-2(Over Sampling) is as below
**********************************************************
              precision    recall  f1-score   support

           0       0.77      0.95      0.85      3584
           1       0.34      0.08      0.13      1126

    accuracy                           0.74      4710
   macro avg       0.55      0.52      0.49      4710
weighted avg       0.66      0.74      0.68      4710

**********************************************************
**********************************************************
Classification Report for Challenge-1(Count/Frequency Encoding) + Challenge-2(Delete the rows) + Challenge-3: Strategy-3(SMOTE - Over Sampling) is as below
**********************************************************
              precision    recall  f1-score   support

           0       0.77      0.97      0.86      3584
           1       0.36      0.06      0.10      1126

    accuracy                           0.75      4710
   macro avg       0.56      0.51      0.48      4710
weighted avg       0.67      0.75      0.67      4710

**********************************************************
```

- Discussion of experimental results – Challenge 1 + Challenge 2 + Challenge 3:
  - Challenge-1 best strategy (count/frequency encoding) + Challenge-2 best strategy (delete the rows) is performed first. Later challenge-3, imbalance in the dataset is handled.
  - First, random under sampling is performed where the majority class label (class label: 0) is brought to the same count as the minority class label (class label: 1). After the random under sampling, each class label has a count of 4574 observations.
  - Next, random over sampling is performed where the minority class label (class label: 1) is brought to the same count as the majority class label (class

label: 0). After the random over sampling, each class label has a count of 14,264 observations.

- Lastly, as a third strategy, tried the SMOTE technique, where synthetic data is created to over sample the minority class label (class label: 1). The overall count of each class label after applying SMOTE technique is 14,264.
- In comparison of the three strategies, in terms of accuracy we can see that the under-sampling technique performed poorly than the baseline classifier, so definitely it is not the best technique. When over sampling and SMOTE are compared, we can see that ***strategy-3: SMOTE*** accuracy(75.03%) is little better than the over sampling method. Also, we know that random over sampling will over sample the minority class randomly where there are chances of creating duplicate data where are in SMOTE technique, synthetic data is created. So, SMOTE is the best strategy for addressing this imbalance in the dataset.
- I have generated the confusion matrix for the best strategy(count/frequency encoding) of challenge-1 + best strategy(delete the rows) of challenge-2 + best strategy(SMOTE) of challenge-3 as below.



- Overall, the strategies of challenge-1 + challenge-2 + challenge-3 together yielded accuracies better than the baseline accuracy (63.18%) except for the under-sampling strategy and also the strategies of challenge-3 like over sampling, SMOTE yielded better accuracies than the baseline accuracy.