Introduction

An Insurance company which provides Health Insurance to its customers' needs our help in building a model to predict whether the policyholders (customers) from past year will also be interested in Vehicle Insurance provided by the company.

An insurance policy is an arrangement by which a company undertakes to provide a guarantee of compensation for specified loss, damage, illness, or death in return for the payment of a specified premium. A premium is a sum of money that the customer needs to pay regularly to an insurance company for this guarantee.

Just like medical insurance, there is vehicle insurance where every year customer needs to pay a premium of certain amount to insurance provider company so that in case of unfortunate accident by the vehicle, the insurance provider company will provide a compensation (called 'sum assured') to the customer.

Goal

Building a model to predict whether a customer would be interested in Vehicle Insurance is extremely helpful for the company because it can then accordingly plan its communication strategy to reach out to those customers and optimize its business model and revenue.

Data Definition

The list below represents meanings of all of the variables in the dataset:

id	Unique ID for the customer
Gender	Gender of the customer
Age	Age of the customer
Driving_License	0 : Customer does not have DL, 1 : Customer already has DL
Region_Code	Unique code for the region of the customer
Previously_Insured	1 : Customer already has Vehicle Insurance, 0 : Customer doesn't have Vehicle Insurance
Vehicle_Age	Age of the Vehicle
Vehicle_Damage	1 : Customer got his/her vehicle damaged in the past. 0 : Customer didn't get his/her vehicle damaged in the past.
Annual_Premium	The amount customer needs to pay as premium in the year

Policy*Sales*Channel	Anonymized Code for the channel of outreaching to the customer ie. Different Agents, Over Mail, Over Phone, In Person, etc.
Vintage	Number of Days, Customer has been associated with the company
Response	1 : Customer is interested, 0 : Customer is not interested

Data Encoding

In order to feed data to the models, variables should be in numeric values. In order to do so I decided to look every column closely and figure out in which way they need to be decoded.

Id column is not needed so I decided to drop. Gender and Vehicle Damage columns are binary so I decided to encode them as 1 and 0. If it comes to gender Female is 1 and Male is 0 and Vehicle Damage, while yes is 1 and no is 0. Since vehicle Age is some interval, I decided to use One hot encoding, to not cause some bias.

Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel	Vintage	Response	(1-2 Year,)	(< Year,
1	44	1	28	0	1	40454.0	26	217	1	0.0	0.(
1	76	1	3	0	0	33536.0	26	183	0	1.0	0.(
1	47	1	28	0	1	38294.0	26	27	1	0.0	0.(
1	21	1	11	1	0	28619.0	152	203	0	0.0	1.(
0	29	1	41	1	0	27496.0	152	39	0	0.0	1.(

Train & Test splitting

In order to train the Random Forest, I needed to divide the data to train and test and I decided to divide it with 0.8 training and 0.2 test proportion. And I set the random state in order to be able to reproduce the same results.

Benchmark Model (Random Forest)

Random Forest is an ensemble method that trains several decision trees in parallel with bootstrapping followed by aggregation, jointly referred as bagging. It's pretty straightforward and since I am going to use it as benchmark model I am not going to do any hyperparameter tuning.

Random Forest	Accuracy	AUC
Base Model	0.866	0.833
Top 4 feature	0.862	0.655

We can see while base Random Forest model did a good job using just top 4 features decrease the AUC significantly.

Main Model (XGBoost)

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

XGBoost expects the inputs as and I decided to use MinMaxScaler in order to do so. To not cause data leak I scale them after dividing to train and test.

With default parameters XGBoost perfrom with 0.877 accuracy and 0.843 auc score. Which is already better than benchmark model. But in order to achieve better result I decide to continue with Parameter tuning.

Optimizing the parameters

I decided to try 2 different parameter tuning methods and see which one will perform better. First one is setting up manually the parameters and continue in each turn with the best result. And the other one is using automatic optimizer which is Grid Search CV. In order to measure the accuracy for best parameter I divide the test data to test and validation data. I tune the parameters on validation dataset and take the final accuracy and AUC on the test dataset

Manually optimizing

Learning rate

I tried the following values [0.01, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1] and the best result was:

```
Learning rate:  0.5
Accuracy score (training):  0.8776891110476996
Accuracy score (validation):  0.8782241347642413
```

Max depth

The maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. Tune this parameter for best performance; the best value depends on the interaction of the input variables.

I tried the following values [1,2,3,4,5,6] and the best result was:

```
max_depth:  4
Accuracy score (training):  0.8781679769881956
Accuracy score (validation):  0.8785127653433392
```

Gamma

Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be.

I tried the following values [1,2,3,4,5,10,20] and the best result was:

```
Gamma:  3
Accuracy score (training):  0.87820733583262
Accuracy score (validation):  0.8785127653433392
```

Subsample

Subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. and this will prevent overfitting. Subsampling will occur once in every boosting iteration.

I tried the following values [0.25, 0.5, 0.6, 0.7, 0.8, 0.9, 1] and the best result was:

```
Subsample:  0.8
Accuracy score (training):  0.8775644747070226
Accuracy score (validation):  0.8783028521949043
```

Min child weight

I tried the following values [1,3,5,10] and the best result was:

```
min_child_weight:  1
Accuracy score (training):  0.8781843765067058
Accuracy score (validation):  0.8778305476109259
```

With Optimizing parameters manually XGBoost perfrom with 0.87714 accuracy and 0.8561 auc score.

Grid Search CV

Exhaustive search over specified parameter values for an estimator.

```
param_grid = {
    'learning_rate' : [0.01, 0.05, 0.1, 0.25, 0.5, 0.75],
    'min_child_weight': [1, 5, 10],
    'gamma': [0.5, 1, 1.5, 2, 5],
    'subsample': [0.6, 0.8, 1.0],
    'max_depth': [2, 3, 4],
    }
```

With over 1600 combinations and computing over an hour Grid Search CV found the best parameters as following:

```
{'gamma': 5,
 'learning_rate': 0.25,
 'max_depth': 4,
 'min_child_weight': 1,
 'subsample': 1.0}
```

With Grid Search CV parameters XGBoost perfrom with 0.8772 accuracy and 0.8579 auc score.

Conclusion

Random Forest	Accuracy	AUC
Base Model	0.866	0.833
Top 4 feature	0.862	0.655
XGBoost	Accuracy	AUC
Base	0.8773	0.8433
Manual optimization	0.8771	0.8561
Grid Search CV	0.8772	0.8579

It is possible to see that XGBoost performed better than Random Forest with 1% more accuracy and 2.5% more AUC. It is possible to say that optimizing the parameters of XGBoost increase the performance of the model with around 1.5% AUC.

If it comes to manual optimization and Grid Search CV there is not much performance difference between two methods. If one has enough computational power and doesn't have much time or experience to try then can clearly choose Grid Search CV. But from other hand manual optimization is giving more understanding of the model, more freedom and it does not require as much computational power as Grid Search CV.