PRIVACY PRESERVING HEALTH INSURANCE

TENKAMTE KENGNE Arsene Bolivar
S292635@studenti.polito.it

MAMMADLI Fidan
s293669@studenti.polito.it

Abstract: **Our concern during this project was to present a suitable solution to solve a prediction problem and in the meantime to build a Machine Learning Data Pipeline (from Data Preparation, to training, to inferencing) within a Trusted Execution Environment to enable data Privacy at all stages of the process. For this purpose, we use different approach study during our past experiences and also a great amount of new concepts that we discover through our own researches.**

I. CROSS SELL PREDICTION INSURANCE PROBLEM

A.PROBLEM OVERVIEW

a. Overview

In this report we proposed a model solution capable of predicting the interested of customers in vehicle insurance in base of their Health insurance through the exploration of a dataset of 381110 reviews. Each one of them is describe by 12 attributes: the id, Gender, Age, Driving License, Region Code, Previously Insured, Vehicle Age, Vehicle Damage, Annual Premium, PolicySalesChannel, Vintage and Response.

The Dataset is divide in two portion:

- A **development set**: containing 304888 reviews that will be used to construct our classification model.
- An **evaluation set:** containing 76222 reviews that will be used to computed our final result

b. Data analysis

Regarding the exploration of the dataset we can observe that between the 12 attributes we have 9 numerical attributes and we have 3 categorical attributes that need to be represented in a suitable way.

	Missing value, %	N unique value	dtype
id	0.0	381109	int64
Gender	0.0	2	object
Age	0.0	66	int64
Driving_License	0.0	2	int64
Region_Code	0.0	53	float64
Previously_Insured	0.0	2	int64
Vehicle_Age	0.0	3	object
Vehicle_Damage	0.0	2	object
Annual_Premium	0.0	48838	float64
Policy_Sales_Channel	0.0	155	float64
Vintage	0.0	290	int64
Response	0.0	2	int64
Gender_Code	0.0	2	int8
Vehicle_Age_code	0.0	3	int8
Vehicle_Damage_code	0.0	2	int8

Fig 1: evaluation of missing data and unique value present in each feature.

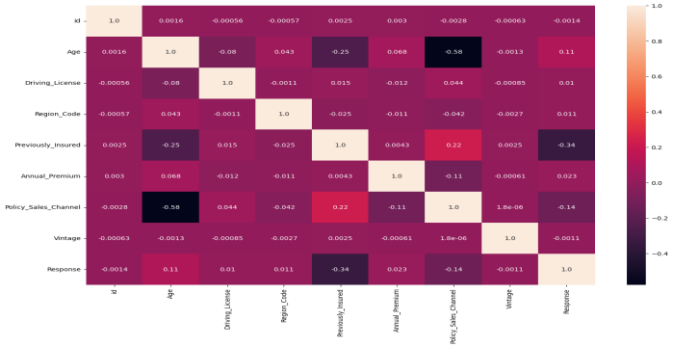

Fig 2: correlation matrix.

The analysis execute on the dataset shows there is no missing values on the different features. And also there are many features with very few unique values which is and information that we will use during the preprocessing step. Then we plot the *correlation matrix* of the different numerical attributes in order to verify if there's a relation between some of them. The figure 2 shows that the correlation is not that strong as the different values on the matrix are lower than 0.4. But we do find some interesting relation as show on the figure 3 between the age, the annual premium and driving license respect to the response. There is and interval of age and also interval amount of annual premium in which customers a more willing to subscribe to a vehicle insurance and also the majority of the customers that apply to vehicle insurance have a driving license.

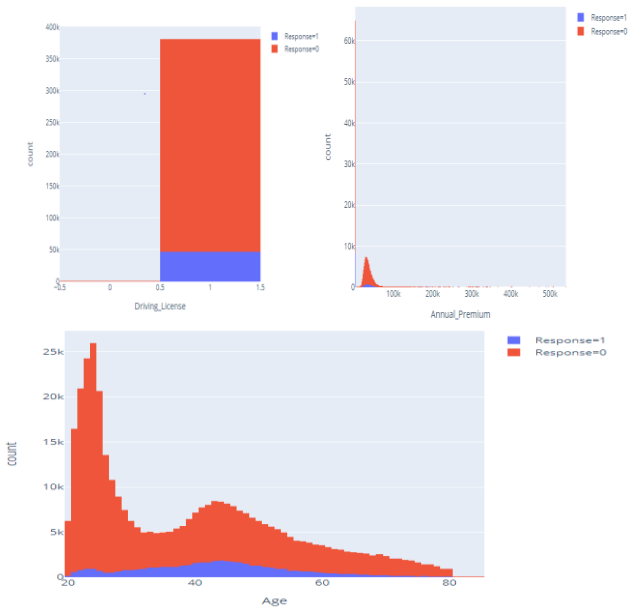

Fig 3: *relation between different features and the response.*

Regarding the categorical features in some cases did notice useful relation between some features and the target value. The figure 4 show the majority of the customer that did apply to the vehicle insurance are those who had a vehicle damage. On the figure 5 we can visualize that most of the vehicles that have been damage are those between 1-2 year. And at last we also noticed how the gender where pretty much equally distributed among the majority of the different attributes such as the ages, previously insured or target value to.

This comprehension will also be useful in order to understand how our models will understand and interpret the dataset. Finally, it's important to view (figure 6) how the target value is highly unbalance. Actually we have a more important amount of 0 than 1 and we will also have to take care of it during the following steps.

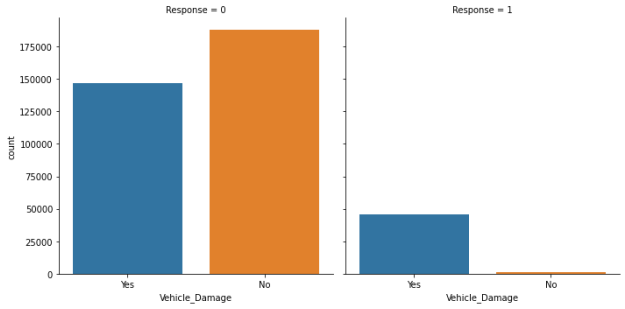

Fig 4: *relation between the vehicle damage and the target.*

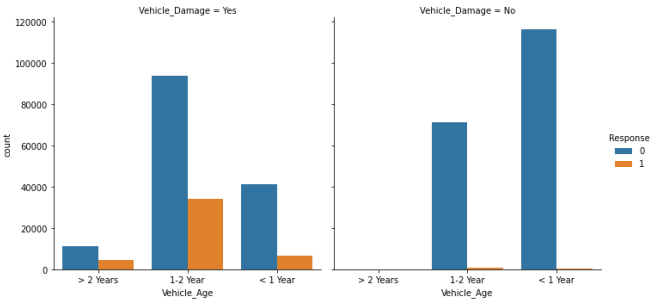

Fig 5: *relation between the vehicle damage, vehicle age and the target.*

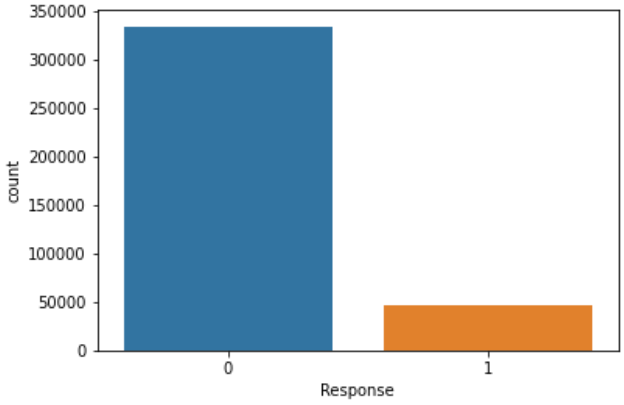

Fig 6: *response feature.*

B.PROPOSED APPROACH

a. Preprocessing

Relating to data cleaning we firstly drop the id features because of it's non relevance for the algorithm. In order to have the best performance we use to implement the *one-hot encoder* on the vehicle_Age feature to convert its different values into column and assigned a 1 or 0 (true or false in base of the presence on the row). It's a suitable approach because of the fact the vehicle age doesn't have many unique value. We decided to use the label on the other categorical features to transform them into numerical features.

	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel	Vintage
count	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000	381109.000000
mean	0.540761	38.822584	0.997869	26.388807	0.458210	0.504877	30564.389581	112.034295	154.347397
std	0.498336	15.511611	0.046110	13.229888	0.498251	0.499977	17213.155057	54.203995	83.671304
min	0.000000	20.000000	0.000000	0.000000	0.000000	0.000000	2630.000000	1.000000	10.000000
25%	0.000000	25.000000	1.000000	15.000000	0.000000	0.000000	24405.000000	29.000000	82.000000
50%	1.000000	36.000000	1.000000	28.000000	0.000000	1.000000	31669.000000	133.000000	154.000000
75%	1.000000	49.000000	1.000000	35.000000	1.000000	1.000000	39400.000000	152.000000	227.000000
max	1.000000	85.000000	1.000000	52.000000	1.000000	1.000000	540165.000000	163.000000	299.000000

Fig 6: *Description of a portion of the dataset before the normalization.*

After that a quick look on the description of our dataset shows on figure 7 allow us to view a great unsteadiness on the values range of the different features.

And so to prevent and inaccurate, slow and inefficient solution we chose to normalize the dataset using the standardscaler algorithm.

	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel	Vintage
count	3.811090e+05	3.811090e+05	3.811090e+05	3.811090e+05	3.811090e+05	3.811090e+05	3.811090e+05	3.811090e+05	3.811090e+05
mean	8.502237e-16	-9.258629e-16	-2.971048e-15	-6.254366e-16	-2.781409e-15	1.470378e-15	-5.018194e-16	-4.104990e-15	-6.921441e-17
std	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00	1.000001e+00
min	-1.085134e+00	-1.213453e+00	-2.164130e+01	-1.994638e+00	-9.196380e-01	-1.009801e+00	-1.622853e+00	-2.048455e+00	-1.725174e+00
25%	-1.085134e+00	-8.911132e-01	4.620794e-02	-8.608404e-01	-9.196380e-01	-1.009801e+00	-3.578308e-01	-1.531887e+00	-8.646631e-01
50%	9.215448e-01	-1.819661e-01	4.620794e-02	1.217845e-01	-9.196380e-01	9.902940e-01	6.417254e-02	3.867931e-01	-4.151927e-03
75%	9.215448e-01	6.561169e-01	4.620794e-02	6.508902e-01	1.087384e+00	9.902940e-01	5.133064e-01	7.373213e-01	8.683108e-01
max	9.215448e-01	2.976962e+00	4.620794e-02	1.935861e+00	1.087384e+00	9.902940e-01	2.960534e+01	9.402586e-01	1.728822e+00

Fig 6: *Description of a portion of the dataset after the normalization.*

we decided to split the dataset into a training part (80%) and a test part (20%) using a stratify parameter activate in order to have the same percentage of different target value on the two parts.

As we have unbalance target value that can have a negative influence on algorithms like Logistic Regression we decided after utilizing pure random sample to obtain training data and a test data set to balance through the SMOTE concepts the training data (in terms of response) using oversampling and undersampling to achieve a 50/50 split. we received four data files to utilize in the construction and analysis of our models.

We used the SMOTE-ENN method to balance the dataset. SMOTE ability to generate synthetic examples for minority class and ENN (Edited Nearest Neighbor) ability to delete some observations from both classes that are identified as having different class between the observation's class and its K-nearest neighbor majority class. It's basically compute in 8 step (with repetition) which are the following:

1. (**Start of SMOTE**) Choose of a random data from the minority Class
2. Calculating the distance between the random data
3. Multiply the difference with a random number between 0 and 1, then add the result to the minority class as a synthetic sample.
4. Repeat step number 2–3 until the desired proportion of minority class is met. **(End of SMOTE)**
5. **(Start of ENN)** Determine K, as the number of nearest neighbors. If not determined, then K=3.
6. Find the K-nearest neighbor of the observation among the other observations in the dataset, then return the majority class from the K-nearest neighbor.
7. If the class of the observation and the majority class from the observation's K-nearest neighbor is different, then the observation and its K-nearest neighbor are deleted from the dataset.
8. Repeat step 2 and 3 until the desired proportion of each class is fulfilled. **(End of ENN)**

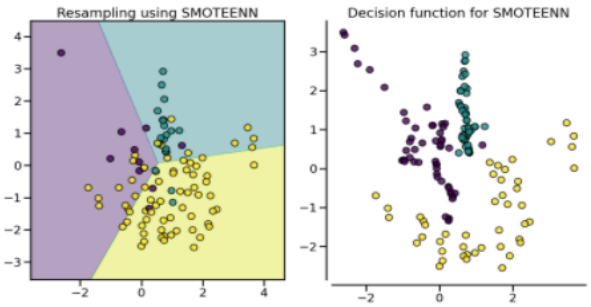

Fig 7: SMOTE-ENN.

b. *Model Selection***:**

We will now present the following models that have been implement for this project:

- **Decision Tree:** it's adopts a greedy approach. The Best attribute for the split is selected locally at each step, it is not a global optimum. To make that "Best" split it define some Measure of impurity like the Gini index or the Entropy
- **Logistic Regression**: is a predictive analysis used to predict a data value based on prior observations of a data set. The analysis is more accurate to conduct when the dependent variable is binary.
- **Random Forest Regressor**: used to solve regression and classification problems. Random forests is consist of many decision trees and predictions from all trees are pooled to make the final prediction. Increasing the number of trees increases the precision of the outcome
- **KNN Classifier**: is a data classification algorithm that attempts to determine what group a data point is in by looking at the data points around it. It's require The set of stored records and a distance metric to compute distance between records.
- **LGBM classifier**: is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks. it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise.

Artificial Neural Network (Ann)

ANN are networks that emulate a biological neural network and they use a reduced set of concepts from biological neural systems. The basic idea is to break the big task of learning and inference into a number of micro-tasks. These micro-tasks are not independent but interdependent.

In the neural network each layer is consists of a number of neurons that are connected from the input layer via the hidden layer to the output layer.

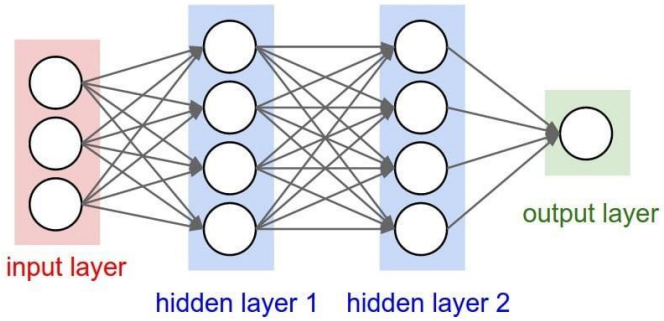

Fig 8: *Artificial Neural Network.*

- **Multilayer Perceptron (MLP):** This algorithm has input and output layers, and one or more **hidden layers** with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function with also Backpropagation mechanism.
- **Keras Sequential model:** Sequential model is a building models as a linear stack of layers. First, you instantiate your Sequential model object and then, you add layers to it one by one using the add() method.

c. Implementation tuning

Regarding this section, we first of all split the development dataset in training set (75%) and test set (25%). We use the training set to train our different models using also the default configuration for them in order to find which perform better than the others. we compute the **Accuracy**, F_1 **score, Precision, and ROC AUC** of those different model and test them with the test set. The result is given on the tables below with the unbalance and balance sample.

	Accuracy	F1 Score	Precision	Recall	ROC AUC
Decision Tree	0.823183	0.825646	0.828223	0.823183	0.603431
SGD	0.877437	0.820156	0.769896	0.877437	0.000000
Random Forest	0.867370	0.836485	0.824716	0.867370	0.833563
Logistic Regression	0.877437	0.820173	0.831183	0.877437	0.838770
KNN	0.855912	0.835990	0.823660	0.855912	0.756851
XGBoost	0.877437	0.820156	0.769896	0.877437	0.856251
CatBoost	0.876615	0.827808	0.828976	0.876615	0.857620
LGBM	0.877183	0.820669	0.814445	0.877183	0.858634

Table 1: Score of the models by default with unbalance data

	Accuracy	F1 Score	Precision	Recall	ROC AUC
Decision Tree	0.897279	0.897375	0.897521	0.897279	0.894858
SGD	0.823874	0.825209	0.833943	0.823874	0.000000
Random Forest	0.915910	0.916021	0.916240	0.915910	0.978852
Logistic Regression	0.837176	0.838326	0.844970	0.837176	0.899075
KNN	0.883190	0.883599	0.884857	0.883190	0.948852
XGBoost	0.894150	0.894831	0.899750	0.894150	0.969736
CatBoost	0.918982	0.919002	0.919025	0.918982	0.981938
LGBM	0.910524	0.910732	0.911290	0.910524	0.978346

Table 2: Score of the models by default with balance data

The ROC_AUC using the MLP **Multilayer Layer Perceptron (MLP)** and the **Keras sequential Model** give us respectively 0.95209840 and 0.95248197.

The visualization of the *classification_report* give us a quick look of the precision and recall of the result and we notice that models didn't perform well with unbalance data because we have some score of 0.38 and 0.24 for the precision and recall of the 1. On the other side the models perform much better well with balance data. We have a score of 0.80 and 0.93 for the precision and the recall.

	precision	recall	f1-score	support
0	0.90	0.95	0.92	100320
1	0.38	0.24	0.30	14013
accuracy			0.86	114333
macro avg	0.64	0.59	0.61	114333
weighted avg	0.84	0.86	0.85	114333

Table 3: Classification_report score of the model (Catboost) by default with balance data

	precision	recall	f1-score	support
0	0.94	0.84	0.89	62157
1	0.80	0.93	0.86	43313
accuracy			0.88	105470
macro avg	0.87	0.89	0.88	105470
weighted avg	0.89	0.88	0.88	105470

Table 4: Classification_report score of the model (Catboost) by default with balance data.

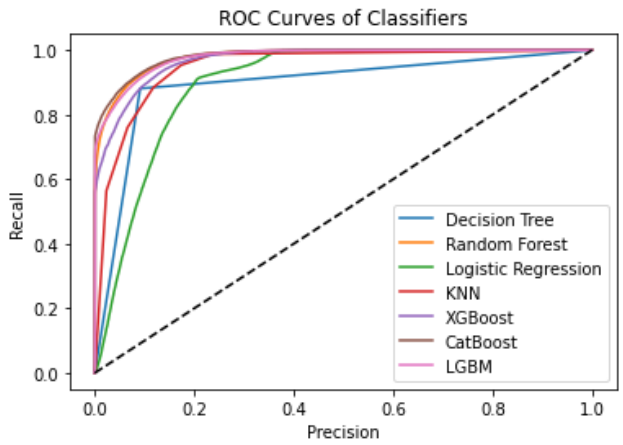

Fig 11: ROC curves of Classifers.

The hyperparameter has been defined for the **Catboost**, **Random forest classifier**, **Multilayer Layer Perceptron** and **Keras Sequential model** which where the best performing models by default.

Model	Parameter configuration
Random Forest	*max_features* = ['auto', 'sqrt'] *criterion* = [‘mse’] *bootstrap* = [True, False] *n_estimators* = [100, 300, 500, 1000]
CatBoost	*n_estimators* = [100, 200, 500, 1000] *depth* = [3,1,2,6,4,5,7,8,9,10] *learning_rate* =[0.03,0.001,0.01,0.1,0.2,0.3] *border_count* =[32,5,10,20,50,100,200] *ctr_border_count*=[50,5,10,20,100,200] *thread_count* = 4
Keras sequential	*epoch=[20,25,50,100]* InputLayer=[255,8]
MLP	*Hidden_layer_sizes* = [(50,50,50), (50,100,50), (100,)] *Activation* = ['tanh', 'relu'] *Solver* =['sgd', 'adam'] *Alpha* = [0.0001, 0.05] *learning_rate* = ['constant','adaptive']

Table 5: set of parameter of the different model

C. RESULT:

The table 6 show the *ROC AUC score* of the execution on the test set of the different models with their best hyper parameter. Through the execution of the GridSearch we had the opportunity to retrieve the best configuration of our different model. As we can visualize on the Table 6 the score of the CatBoost is higher than all the others with a value of *0.981938* follow by the Random Forest, then the Multilayer Perceptron and Keras Sequential. Impressively the gap between the differents score is not really big and also having a quick look on their classification_score gave us also the same interpretation regarding the Precision and the Recall. Also the computation of the different models with test set gave us the same order of score. As the models was performing with a very high score by default there was a possibility that the models where being overfitting but the fact of using the GridSearch with number of cross-validation = 3 give us the opportunity to bypass the training data and check for the score on testing data (validation data) allow us to negate that hypothesis.

Model	Parameter configuration	ROC AUC
Random Forest	*max_features* = 'auto' *criterion* = ‘mse’ *bootstrap* = False *n_estimators* = 200 *min_samples_split* = 2 *min_samples_leaf*' = 2	*0.977124522 8843219*
CatBoost	*n_estimators* = 300 *depth* = 9 *max_depth* = *5* *learning_rate* = 0.01 *border_count* = 50 *ctr_border_count* = 10	*0.981938*
Keras sequential	*epoch=[100]* InputLayer=[64,32,16,8]	*0.957599893 7222543*
MLP	*Hidden_layer_sizes* = (50,50,50) *Activation* = ’relu' *Solver* = 'adam', *Alpha* = 0.0001 *learning_rate* = 'adaptive'	*0.963419350 9552655*

Table 6: best parameter of models with test score.

This final model compute will be use inside our Trusted environment execution that we are going to deploy.

II. TRUSTED EXECUTION ENVIRONMENT

A. INTRODUCTION TO THE TEE

A Trusted Execution Environment (TEE) is an environment for executing code, in which those executing the code can have high levels of trust in the asset management of that surrounding environment because it can ignore threats from the “*unknown*” rest of the device. nowadays to run AI workloads at scale, Enterprises are increasingly relying on the public cloud and in this context, data security and privacy can become key concerns. And even if the use of cryptography has been seen as the common solution to protecting data in the private cloud or in the public cloud in all states, this type of mechanism became more vulnerable to attacks and exploits because of

his accessed by more users. So it's a good resolution to rely on TEE that can be used on-premises, in the cloud or within embedded hardware platforms.

Trusted Execution Environment add a layer security to an existing solution and can be use with confidential computing technology, without any changes in the application itself. Some other advantages in deploying in a trusted execution environment include:

- **A Rapid deployment:** Advanced TEE solutions are quick to deploy
- **The Insurance of data confidentiality and integrity:** Ensuring the security of sensitive data in transit, in use, and at rest for critical applications and data.
- **The Secure deployment and execution of applications**
- **A Trusted Collaboration:** A TEE is safe environment for multiple parties to share and process data.
- **A Simplified Compliance:** Achieve compliance with key management and encryption through an easy-to-use cryptographic API.

B. GENERAL SCHEMATIC OF TH E DEPLOYMENT

a. Overview plan

In a General way let consider the following scenario: we have an AI pipeline schematic show by the figure 12. The AI model, the training data, and both the code for training and inference are provided by different stakeholders. In the meantime, all these stakeholders must have confidence in each other and the cloud service provider with their valuable data or intellectual property. Concerning our project, we have a Health Insurance company that wants to build a predictive model to determine if their Health insurance policyholders would be also interested in a Vehicle Insurance. And also the company would like to monetize that model with other insurance companies which could use the model to target new clients. In fact, the different companies involve need to have confidence in each other and a confidential computing is a key technology that can solve these fundamental.

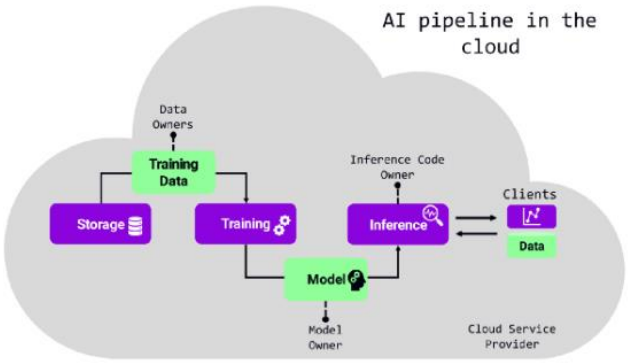

Figure 12: AI pipeline in the cloud

b. Functional Description.

We will firstly make a recapitulation and present the existing difficulties and the needs between the different companies involve through the table 7 that our TEE must respond concerning our project. As you can see the main preoccupation is the assurance of a high security and the respect too of the privacy. Regarding now The implementation, it has been done on an Azure virtual machine provided by to us by the Accenture company. And the following functional diagram (**figure 13**) represent the main operation of the collaboration between two companies in and Trusted Environment Execution.

We can distinguish two main steps concerning the running of our project.

- **First Step**: Preparation of the Model

- **The company A** upload the training data into the Database
- The deployment of the training data into the **Cross Insurance Machine Learning Algorithm**
  - Creation of the **model** and saving into de Database
- **Second Step**: Prediction step

- **The company B** upload his data inside the Application programming interface
- **The API** deploy the model from the Database and process with the inferencing of the data
- **Return** the predicted result.

DIFFICULTIES	NEEDS
The absence of confidence between the different companies involve on the project.	Creation of a secure enclave environment
The lack of privacy regarding the sharing of the data throw the could	Storage of the dataset on a Secure Database
The lack of security through the download and the processing of the model	Execution of the model in the enclave environment.

The lack of privacy through the evaluation of the data	Creation a app that will be launch into the enclave

Table 7: Evaluation of the needs and difficulties

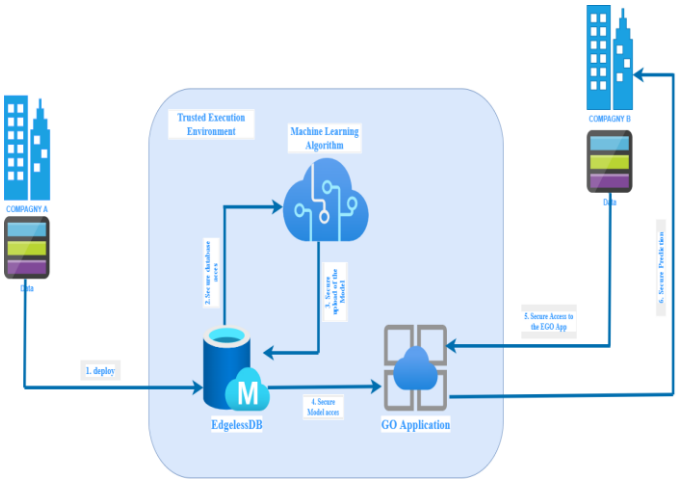

Figure 13: Functional Diagram

C. CONFIGURATION AND DEPLOYMENT

a. Configuration

In order to setup our TEE inside our Azure virtual machine we have install and use a set of resources and services.

- **Intel SGX driver**: is an Intel technology for application developers seeking to protect select code and data from disclosure or modification.
- **FSGBASE:** which is a feature in recent processors which allows direct access to the FS and GS segment base addresses
- **EdgelessRT:** SDK for Trusted Execution Environments (TEE) build on top of open enclave. It adds support for modern programming languages (in particular Go) and facilitates the porting of existing applications.

This are the principal tools that were install to create the enclave. Regarding now the different services that we have implemented we can enumerate:

- **EdgelessDB:** is an open-source MySQL-compatible database for confidential computing. Like a normal SQL database, EdgelessDB writes a transaction log to disk and also keeps cold data on disk. With EdgelessDB, all data on disk is strongly encrypted. Data is only ever decrypted inside SGX enclaves.
- **EGO:** is a framework for building *confidential apps* in Go. Confidential apps run in always-encrypted and verifiable enclaves on Intel SGX-enabled hardware.
- **GOLANG:** is an open source programming language used for general purpose. Most similarly modeled after C, Go is statically typed and explicit. We did use this language in order to code our machine learning Algorithm for our project. Also Basically some advantages it that Golang has a High performance in network tasks and multiprocessing, with and Excellent code readability and ease of use. One last is that he is a Static typing and performance.

b. Deployment

Relating to the implementation we first of all install the different tools to setup the enclave and we also install the different resources. Then we begin by launching the EdgelessDB using a support call the manifest to instantiate and grant specific access to the different actors with their proper authentication keys that will have access to the Database. I our case will have two actors who are the company A that will upload his data inside the database and the machine learning algorithm that will download the data for the training step and also save the model and download it from the database when necessary. Speaking now about the machine learning model, we code it in golang and we implemented a Neural Network model using the parameters similar as much as possible to the one that we implemented in python. Finally, we deploy and API in enclave that will essentially receive the request form the company B, then he will deploy the save model and do the inference service and send the result to client. A final precision is to say that all this process is run inside a Trusted Environment so it is an added value in terms of security and none of the stakeholder involved we be able to visualize the data that is none of his concern.

CONCLUSION

Concerning our cross insurance prediction model between all the models tested the most suitable one was the Catboost Classifier. But we can also notice that the order models retained at the end did also perform well because they all have a final roc auc score in a very close interval. As we did obtain a high score with the best parameter, we can be sure that our model is able to predict in a very efficient way the different customer that will be willing to take a Vehicle Insurance. On the order side we also succeed creating a Trusted Environment in which the company can monetize its model selling the inference job

to another in a Trusted and Secure way. To the question of why we use the Neural Network instead of the Catboost model, it's simply because as today, the catboost machine learning model is not available in Golang or it's not very much efficient. Unfortunately, as the Golang is a fairly young language and not widely used the ML domain is not that much develop. And despite that and also the lack of some parameters we were quite surprise to see how the model was still performing very well during our different test. Ultimately another extension that could have been done regarding the security and the privacy is the setup of a remote access control for the client in order for him to be sure that his data will be send in an enclave environment before uploading them.