AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

DEPARTMENT OF COMPUTER SCIENCE ANDENGINEERING

Project Report On

HEALTH INSURANCE CROSS SELL PREDICTION

19AI716-CLOUD AND BIG DATA ANALYTICS

Course Instructor:- Dr.Karthi R
Associate Professor
Department of CSE

Submitted By: Medarametla Sravani
Roll No:-CB.EN.P2AID20029

INDEX

1. INTRODUCTION.....1

2. DATASET DESCRIPTION……1

3. BLOCK DIAGRAM……2

4. PYSPARK IMPLEMENTATION……..3

a. Preprocessing & EDA……3

b. Data Transformation & Scaling……..5

c. Feature Scaling & PCA……..6

d. Machine Learning Algorithms……7

e. Results……9

5. GCP IMPLEMENTATION……10

5.1 DATAPROC……10

1. Implementation……10

2. Results……...12

5.2 AUTOML TABLES……15

1. Implementation……15

2. Results…… ..17

5.3 DATA FUSION……20

1. Implementation……20

2. Results……..25

5.4 BIGQUERY……..26

1. Implementation……26

2. Results……...29

6. INFERENCE……42

7. REFERENCE……42

1. INTRODUCTION

Insurance company that has provided Health Insurance to its customers now they need your 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.

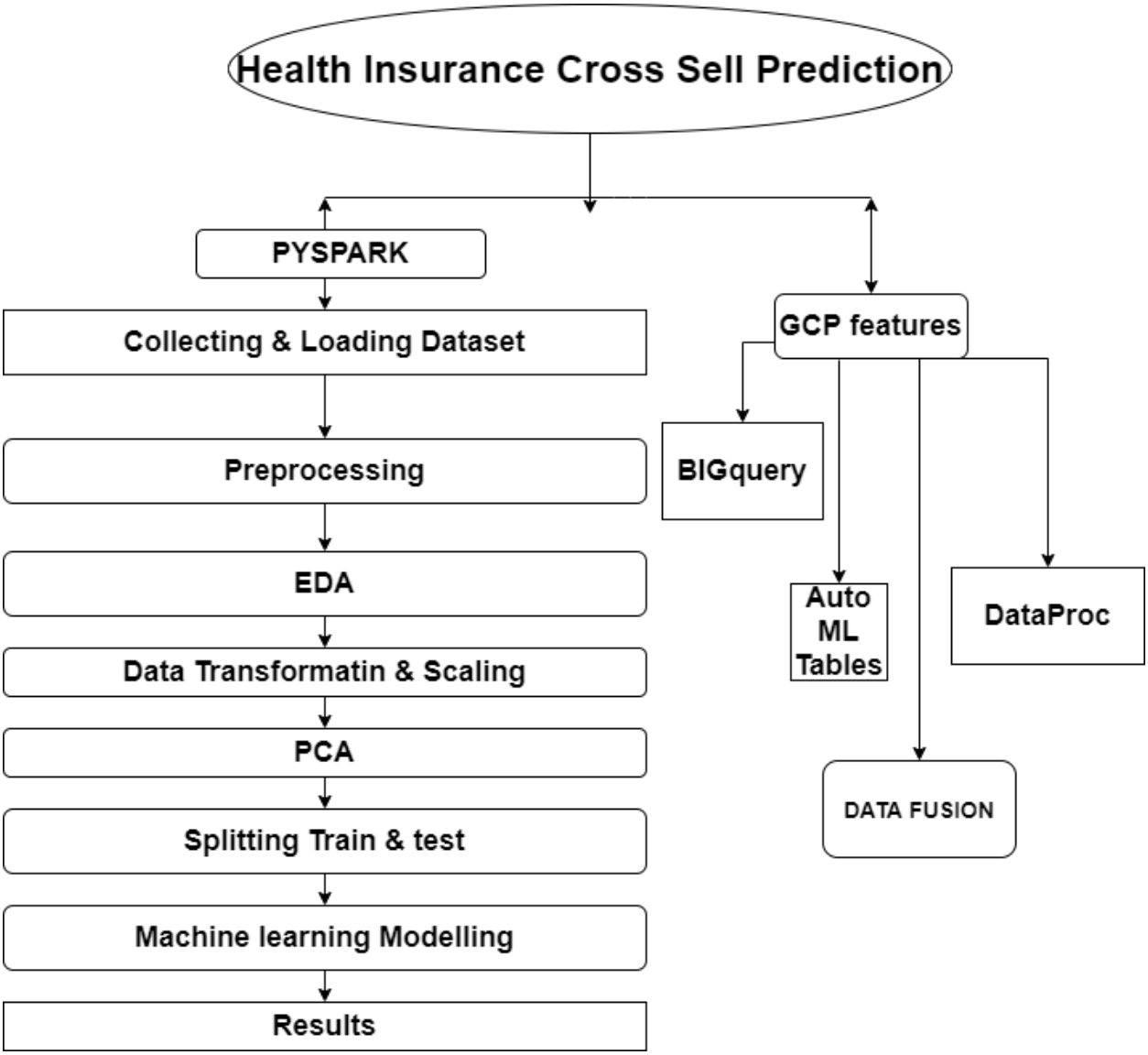
The main aim of this project is to Building a model to predict whether a customer would be interested in Vehicle Insurance.

2. DATASET DESCRIPTION

we used the data obtained from kaggle.com. it contains information about the 381,109 policyholders' demographics, vehicles, and policies in 12 columns without any missing data.

Variable	Definition
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
PolicySalesChannel	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

3.BLOCK DIAGRAM



4. PYSPARK IMPLEMENTATION

PySpark has been released in order to support the collaboration of Apache Spark and Python, it actually is a Python API for Spark. In addition, **PySpark**, helps you interface with Resilient Distributed Datasets (RDDs) in Apache Spark and Python programming language.

Loading Dataset

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
spark = SparkSession.builder.appName("LinearRegression").getOrCreate()
df = spark.read.csv('/content/drive/My Drive/healthinsurance.csv', header=True, inferSchema=True)
df.show(5)
```

```
+---+------+---+---------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
| id|Gender|Age|Driving_License|Region_Code|Previously_Insured|Vehicle_Age|Vehicle_Damage|Annual_Premium|Policy_Sales_Channel|Vintage|Response|
+---+------+---+---------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
|  1|  Male| 44|              1|       28.0|                0|  > 2 Years|           Yes|       40454.0|                26.0|    217|       1|
|  2|  Male| 76|              1|        3.0|                0|   1-2 Year|            No|       33536.0|                26.0|    183|       0|
|  3|  Male| 47|              1|       28.0|                0|  > 2 Years|           Yes|       38294.0|                26.0|     27|       1|
|  4|  Male| 21|              1|       11.0|                1|   < 1 Year|            No|       28619.0|               152.0|    203|       0|
|  5|Female| 29|              1|       41.0|                1|   < 1 Year|            No|       27496.0|               152.0|     39|       0|
+---+------+---+---------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
only showing top 5 rows
```

a. Preprocessing & EDA

Spark SQL: This feature was used for Exploratory data Analysis. **Exploratory Data Analysis** (**EDA**) is an approach to data analysis that attempts to maximize insight into data. This includes assessing the quality and structure of the data, calculating summary or descriptive statistics, and plotting appropriate graphs. It can uncover underlying structures and suggest how the data should be modeled.

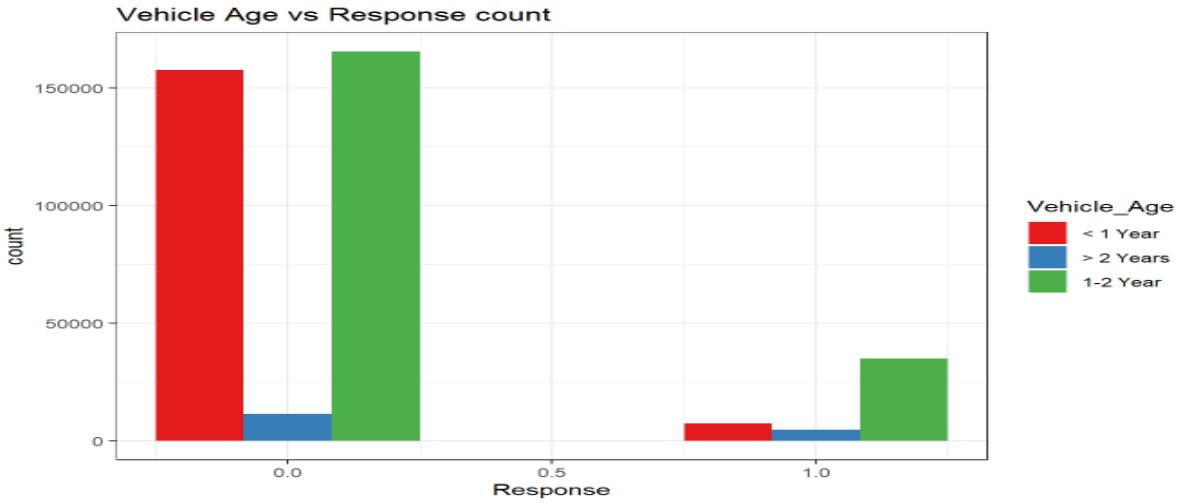
```
df.printSchema()
```

```
root
 |-- id: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Driving_License: integer (nullable = true)
 |-- Region_Code: double (nullable = true)
 |-- Previously_Insured: integer (nullable = true)
 |-- Vehicle_Age: string (nullable = true)
 |-- Vehicle_Damage: string (nullable = true)
 |-- Annual_Premium: double (nullable = true)
 |-- Policy_Sales_Channel: double (nullable = true)
 |-- Vintage: integer (nullable = true)
 |-- Response: integer (nullable = true)
```

```
# get the dimensions of the data
(df.count() , len(df.columns))
```

```
(381109, 12)
```

	Data Type
id	ID Column
Gender	Categorical
Age	Numeric
Driving_License	Categorical
Region_Code	Numeric
Previously_Insured	Categorical
Vehicle_Age	Categorical
Vehicle_Damage	Categorical
Annual_Premium	Numeric
Policy_Sales_Channel	Numeric
Vintage	Numeric
Response	Label




```
df.groupby('Response').count().show()
```

```
+--------+------+
|Response| count|
+--------+------+
|       1| 46710|
|       0|334399|
+--------+------+
```

```
df.describe().show()
```

```
+-------+------------------+------+------------------+--------------------+------------------+------------------+-----------+--------------+-----
|summary|                id|Gender|               Age|     Driving_License|       Region_Code| Previously_Insured|Vehicle_Age|Vehicle_Damage|    A
+-------+------------------+------+------------------+--------------------+------------------+------------------+-----------+--------------+-----
|  count|            381109|381109|            381109|              381109|            381109|            381109|     381109|        381109|
|   mean|          190555.0|  null|38.822583565331705|  0.9978693759528114|26.388807401557035| 0.4582101183650871|       null|          null|30564
| stddev|110016.83620776715|  null| 15.51161101809548|0.046109544207799495| 13.22988802578841|0.49825119888722824|       null|          null|17213
|    min|                 1|Female|                20|                   0|               0.0|                 0|   1-2 Year|            No|
|    max|            381109|  Male|                85|                   1|              52.0|                 1|  > 2 Years|           Yes|
+-------+------------------+------+------------------+--------------------+------------------+------------------+-----------+--------------+-----
```

```
from pyspark.sql import functions as F
df.select([F.count(F.when(F.isnull(c), c)).alias(c) for c in df.columns]).show()
```

```
+---+------+---+--------------+-----------+------------------+-----------+-------------+-------------+--------------------+-------+--------+
| id|Gender|Age|Driving_License|Region_Code|Previously_Insured|Vehicle_Age|Vehicle_Damage|Annual_Premium|Policy_Sales_Channel|Vintage|Response|
+---+------+---+--------------+-----------+------------------+-----------+-------------+-------------+--------------------+-------+--------+
|  0|     0|  0|             0|          0|                 0|          0|            0|            0|                   0|      0|       0|
+---+------+---+--------------+-----------+------------------+-----------+-------------+-------------+--------------------+-------+--------+
```

```
df.groupBy("Gender").agg(F.sum("Response")).show()
```

```
+------+-------------+
|Gender|sum(Response)|
+------+-------------+
|Female|        18185|
|  Male|        28525|
+------+-------------+
```

```
#Selecting few columns by their names
df.select(['Gender', 'Age', 'Policy_Sales_Channel']).show(5, False) # Method 1

df.select(df["Age"], df["Gender"], df["Policy_Sales_Channel"]).show(10, False) # Method 2

from pyspark.sql.functions import col # Method 3 - By import SQL function col
df.select(col("Age"), col("Gender"), col("Policy_Sales_Channel")).show(5, False)
```

```
+------+---+---------------------+
|Gender|Age|Policy_Sales_Channel|
+------+---+---------------------+
|Male  |44 |26.0                 |
|Male  |76 |26.0                 |
|Male  |47 |26.0                 |
|Male  |21 |152.0                |
|Female|29 |152.0                |
+------+---+---------------------+
only showing top 5 rows

+---+------+---------------------+
|Age|Gender|Policy_Sales_Channel|
+---+------+---------------------+
|44 |Male  |26.0                 |
|76 |Male  |26.0                 |
|47 |Male  |26.0                 |
```

```
#Creating a subset of dataframe using Filter
df.filter(df["Age"] > 30 ).filter(df["Age"] < 45).show(5) # Selecting subset using chain of Filter of option
df.filter((df["Age"] > 30) & (df["Age"] < 45 )).show(5, False) # Selcting subset by using AND between the criteria
```

```
+---+------+---+--------------+-----------+-----------------+----------+--------------+-------------+--------------------+-------+--------+
| id|Gender|Age|Driving_License|Region_Code|Previously_Insured|Vehicle_Age|Vehicle_Damage|Annual_Premium|Policy_Sales_Channel|Vintage|Response|
+---+------+---+--------------+-----------+-----------------+----------+--------------+-------------+--------------------+-------+--------+
|  1|  Male| 44|             1|       28.0|                0|  > 2 Years|           Yes|       40454.0|                26.0|    217|       1|
| 10|Female| 32|             1|        6.0|                1|   < 1 Year|            No|       28771.0|               152.0|     80|       0|
| 13|Female| 41|             1|       15.0|                1|  1-2 Year|            No|       31409.0|                14.0|    221|       0|
| 16|  Male| 37|             1|        6.0|                0|  1-2 Year|           Yes|        2630.0|               156.0|    147|       1|
| 19|  Male| 42|             1|       28.0|                0|  1-2 Year|           Yes|       33667.0|               124.0|    158|       0|
+---+------+---+--------------+-----------+-----------------+----------+--------------+-------------+--------------------+-------+--------+
only showing top 5 rows

+---+------+---+--------------+-----------+-----------------+----------+--------------+-------------+--------------------+-------+--------+
|id |Gender|Age|Driving_License|Region_Code|Previously_Insured|Vehicle_Age|Vehicle_Damage|Annual_Premium|Policy_Sales_Channel|Vintage|Response|
+---+------+---+--------------+-----------+-----------------+----------+--------------+-------------+--------------------+-------+--------+
|1  |Male  |44 |1             |28.0       |0                |> 2 Years |Yes           |40454.0       |26.0                |217    |1       |
|10 |Female|32 |1             |6.0        |1                |< 1 Year  |No            |28771.0       |152.0               |80     |0       |
|13 |Female|41 |1             |15.0       |1                |1-2 Year  |No            |31409.0       |14.0                |221    |0       |
|16 |Male  |37 |1             |6.0        |0                |1-2 Year  |Yes           |2630.0        |156.0               |147    |1       |
|19 |Male  |42 |1             |28.0       |0                |1-2 Year  |Yes           |33667.0       |124.0               |158    |0       |
+---+------+---+--------------+-----------+-----------------+----------+--------------+-------------+--------------------+-------+--------+
only showing top 5 rows
```

b.Data Transformation & scaling

1. **String Indexer:-** It converts string values to indexes for categorical features.

```
from pyspark.ml.feature import StringIndexer
# create object of StringIndexer class and specify input and output column
indexer1 = StringIndexer(inputCol='Vehicle_Damage',outputCol='Vehicle_Damage_Index')
indexer2 = StringIndexer(inputCol='Vehicle_Age',outputCol='Vehicle_Age_Index')
# transform the data
df = indexer1.fit(df).transform(df)
df = indexer2.fit(df).transform(df)
# view the transformed data
df.select('Vehicle_Damage','Vehicle_Damage_Index','Vehicle_Age','Vehicle_Age_Index').show(10)
```

```
+--------------+---------------------+-----------+------------------+
|Vehicle_Damage|Vehicle_Damage_Index|Vehicle_Age|Vehicle_Age_Index|
+--------------+---------------------+-----------+------------------+
|           Yes|                  0.0|  > 2 Years|               2.0|
|            No|                  1.0|   1-2 Year|               0.0|
|           Yes|                  0.0|  > 2 Years|               2.0|
|            No|                  1.0|   < 1 Year|               1.0|
|            No|                  1.0|   < 1 Year|               1.0|
|           Yes|                  0.0|   < 1 Year|               1.0|
|           Yes|                  0.0|   < 1 Year|               1.0|
|           Yes|                  0.0|   1-2 Year|               0.0|
|            No|                  1.0|   < 1 Year|               1.0|
|            No|                  1.0|   < 1 Year|               1.0|
+--------------+---------------------+-----------+------------------+
only showing top 10 rows
```

2. OneHotEncoder:- It converts Categorical Features To Continuous Features

```
from pyspark.ml.feature import OneHotEncoder
OHE = OneHotEncoder(inputCols=['Vehicle_Damage_Index', 'Vehicle_Age_Index'],outputCols=['Vehicle_Damage_OHE', 'Vehicle_Age_OHE'])
# transform the data
df= OHE.fit(df).transform(df)
# view and transform the data
df.select('Vehicle_Damage', 'Vehicle_Damage_Index', 'Vehicle_Damage_OHE', 'Vehicle_Age', 'Vehicle_Age_Index', 'Vehicle_Age_OHE').show(5)
```

```
+-------------+-------------------+------------------+-----------+-----------------+---------------+
|Vehicle_Damage|Vehicle_Damage_Index|Vehicle_Damage_OHE|Vehicle_Age|Vehicle_Age_Index|Vehicle_Age_OHE|
+-------------+-------------------+------------------+-----------+-----------------+---------------+
|          Yes|                0.0|     (1,[0],[1.0])|  > 2 Years|              2.0|      (2,[],[])|
|           No|                1.0|         (1,[],[])|   1-2 Year|              0.0|  (2,[0],[1.0])|
|          Yes|                0.0|     (1,[0],[1.0])|  > 2 Years|              2.0|      (2,[],[])|
|           No|                1.0|         (1,[],[])|   < 1 Year|              1.0|  (2,[1],[1.0])|
|           No|                1.0|         (1,[],[])|   < 1 Year|              1.0|  (2,[1],[1.0])|
+-------------+-------------------+------------------+-----------+-----------------+---------------+
only showing top 5 rows
```

3. VectorAssembler:- It combines Categorical Features into single vector.

```
from pyspark.ml.feature import VectorAssembler
inputCols=['Age','Driving_License','Region_Code','Previously_Insured','Vehicle_Age_Index','Vehicle_Damage_Index','Annual_Premium','Policy_Sales_(
print(inputCols)
vectorAssembler = VectorAssembler(inputCols = inputCols, outputCol = 'features')
output = vectorAssembler.transform(df)
#display features and the label column
output.select("features", "Response").show(truncate=False)
```

```
['Age', 'Driving_License', 'Region_Code', 'Previously_Insured', 'Vehicle_Age_Index', 'Vehicle_Damage_Index', 'Annual_Premium', 'Policy_Sales_Chanr
+---------------------------------------------+--------+
|features                                     |Response|
+---------------------------------------------+--------+
|[44.0,1.0,28.0,0.0,2.0,0.0,40454.0,26.0,217.0] |1       |
|[76.0,1.0,3.0,0.0,0.0,1.0,33536.0,26.0,183.0]  |0       |
|[47.0,1.0,28.0,0.0,2.0,0.0,38294.0,26.0,27.0]  |1       |
|[21.0,1.0,11.0,1.0,1.0,1.0,28619.0,152.0,203.0]|0       |
|[29.0,1.0,41.0,1.0,1.0,1.0,27496.0,152.0,39.0] |0       |
|[24.0,1.0,33.0,0.0,1.0,0.0,2630.0,160.0,176.0] |0       |
|[23.0,1.0,11.0,0.0,1.0,0.0,23367.0,152.0,249.0]|0       |
|[56.0,1.0,28.0,0.0,0.0,0.0,32031.0,26.0,72.0]  |1       |
|[24.0,1.0,3.0,1.0,1.0,1.0,27619.0,152.0,28.0]  |0       |
|[32.0,1.0,6.0,1.0,1.0,1.0,28771.0,152.0,80.0]  |0       |
|[47.0,1.0,35.0,0.0,0.0,0.0,47576.0,124.0,46.0] |1       |
|[24.0,1.0,50.0,1.0,1.0,1.0,48699.0,152.0,289.0]|0       |
|[41.0,1.0,15.0,1.0,0.0,1.0,31409.0,14.0,221.0] |0       |
|[76.0,1.0,28.0,0.0,0.0,0.0,36770.0,13.0,15.0]  |0       |
```

c. Feature Scaling &PCA

1. Feature Scaling(Stand scalar)

```
from pyspark.ml.feature import StandardScaler
scaler = StandardScaler(inputCol='features', outputCol="scaledFeatures", withStd=False, withMean=True)
scalerModel = scaler.fit(output)
scaledData = scalerModel.transform(output)
scaledData.select(['scaledFeatures']).show(5, truncate = False)
```

```
+--------------------------------------------------------------------------------------------------------------
|scaledFeatures
+--------------------------------------------------------------------------------------------------------------
|[5.177416434667499,0.0021306240471885562,1.6111925984424893,-0.4582101183650871,1.4836122998932133,-0.4951234423747537,9889.610418541386,-86.03
|[37.1774164346675,0.0021306240471885562,-23.38880740155751,-0.4582101183650871,-0.5163877001067866,0.5048765576252463,2971.610418541386,-86.034
|[8.1774164346675,0.0021306240471885562,1.6111925984424893,-0.4582101183650871,1.4836122998932133,-0.4951234423747537,7729.610418541386,-86.0342
|[-17.8225835653325,0.0021306240471885562,-15.38880740155751,0.5417898816349129,0.4836122998932134,0.5048765576252463,-1945.389581458614,39.9657
|[-9.8225835653325,0.0021306240471885562,14.61119259844249,0.5417898816349129,0.4836122998932134,0.5048765576252463,-3068.389581458614,39.965705
+--------------------------------------------------------------------------------------------------------------
only showing top 5 rows
```

```
from pyspark.ml.feature import PCA
pca = PCA(k=2, inputCol = scaler.getOutputCol(), outputCol="pcaFeatures")
model = pca.fit(scaledData)
transformed_feature = model.transform(scaledData)
transformed_feature.select(['pcaFeatures']).show(5, truncate = False)
```

```
+-----------------------------------------+
|pcaFeatures                              |
+-----------------------------------------+
|[-9889.640570062405,-62.68362679820971]  |
|[-2971.6432718595656,-28.666926921856515]|
|[-7729.641455922532,127.32342306780409]  |
|[1945.4048090866763,-48.656157152989714] |
```

d. Machine Learning Algorithms

1. Logistic Regression:-logistic Regression Aims to Measure the Relationship between a Categorical Dependent Variable and One or More Independent Variables (Usually Continuous) By Plotting The Dependent Variables' Probability Scores.

```
[37] from pyspark.ml.classification import LogisticRegression
     lr = LogisticRegression(labelCol="Response", featuresCol="features")
     #Training algo
     lrModel = lr.fit(trainingData)
     lr_prediction = lrModel.transform(testData)
     lr_prediction.select("prediction", "Response", "features").show(5)
     evaluator = MulticlassClassificationEvaluator(labelCol="Response", predictionCol="prediction", metricName="accuracy")
     print('Logistic Regression Accuracy:', multi_evaluator.evaluate(lr_prediction))
     evaluator = BinaryClassificationEvaluator(labelCol='Response', metricName='areaUnderROC')
     roc_LR=evaluator.evaluate(gb_predictions)
     print('LR_ROC SCORE:',roc_LR )
```

```
+----------+--------+--------------------+
|prediction|Response|            features|
+----------+--------+--------------------+
|       0.0|       1|[47.0,1.0,28.0,0....|
|       0.0|       1|[56.0,1.0,28.0,0....|
|       0.0|       0|[76.0,1.0,28.0,0....|
|       0.0|       0|[25.0,1.0,35.0,1....|
|       0.0|       1|[51.0,1.0,28.0,0....|
+----------+--------+--------------------+
only showing top 5 rows

Logistic Regression Accuracy: 0.8786968031662545
LR_ROC SCORE: 0.8534113932238853
```

2. Factorization machines classifier:- A Combination Of Linear Regression And Matrix Factorization, The Cool Idea Behind This Type Of Algorithm Is It Aims Model Interactions Between Features Using Factorized Parameters.

```
from pyspark.ml.classification import FMClassifier
fm = FMClassifier(labelCol="Response", featuresCol="features", stepSize=0.001)
fm_model = fm.fit(trainingData)
fm_prediction = fm_model.transform(testData)
fm_prediction.select("prediction", "Response", "features").show(5)
fm_accuracy = evaluator.evaluate(fm_prediction)
print('Accuracy of Factorization machines classifier is:', multi_evaluator.evaluate(svm_prediction))
evaluator = BinaryClassificationEvaluator(labelCol='Response', metricName='areaUnderROC')
roc_FM=evaluator.evaluate(gb_predictions)
print('FMClassifier_ROC SCORE:',roc_FM )
```

```
+----------+--------+--------------------+
|prediction|Response|            features|
+----------+--------+--------------------+
|       0.0|       1|[47.0,1.0,28.0,0....|
|       0.0|       1|[56.0,1.0,28.0,0....|
|       1.0|       0|[76.0,1.0,28.0,0....|
|       0.0|       0|[25.0,1.0,35.0,1....|
|       0.0|       1|[51.0,1.0,28.0,0....|
+----------+--------+--------------------+
only showing top 5 rows

Accuracy of Factorization machines classifier is: 0.8786968031662545
FMClassifier_ROC SCORE: 0.8534113932238853
```

3. Linear Support Vector Machine:- Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyper plane which separates the data into classes.

```
from pyspark.ml.classification import LinearSVC
svm = LinearSVC(labelCol="Response", featuresCol="features")
svm_model = svm.fit(trainingData)
svm_prediction = svm_model.transform(testData)
svm_prediction.select("prediction", "Response", "features").show(5)
svm_accuracy = evaluator.evaluate(svm_prediction)
print('Accuracy of Support Vector Machine is:', multi_evaluator.evaluate(svm_prediction))
evaluator = BinaryClassificationEvaluator(labelCol='Response', metricName='areaUnderROC')
roc_LSVC=evaluator.evaluate(gb_predictions)
print('LinearSVC_ROC SCORE:',roc_LSVC )
```

```
+----------+--------+--------------------+
|prediction|Response|            features|
+----------+--------+--------------------+
|       0.0|       1|[47.0,1.0,28.0,0....|
|       0.0|       1|[56.0,1.0,28.0,0....|
|       0.0|       0|[76.0,1.0,28.0,0....|
|       0.0|       0|[25.0,1.0,35.0,1....|
|       0.0|       1|[51.0,1.0,28.0,0....|
+----------+--------+--------------------+
only showing top 5 rows

Accuracy of Support Vector Machine is: 0.8786968031662545
LinearSVC_ROC SCORE: 0.8534113932238853
```

4.Gradient Boosted Tree Classifier:- Gradient boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

```
from pyspark.ml.classification import GBTClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.evaluation import BinaryClassificationEvaluator
gbt = GBTClassifier(labelCol="Response", featuresCol="features")
gbModel = gbt.fit(trainingData)
gb_predictions = gbModel.transform(testData)
print(gb_predictions)
gb_predictions.select("prediction", "Response", "features").show(5)
multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'Response', metricName = 'accuracy')
print('Gradient-boosted Trees Accuracy:', multi_evaluator.evaluate(gb_predictions))
evaluator = BinaryClassificationEvaluator(labelCol='Response', metricName='areaUnderROC')
roc_gbt=evaluator.evaluate(gb_predictions)
print('GBT_ROC SCORE:',roc_gbt )
```

```
DataFrame[id: int, Gender: string, Age: int, Driving_License: int, Region_Code: double, Previously_Insured: int, V
+----------+--------+--------------------+
|prediction|Response|            features|
+----------+--------+--------------------+
|       0.0|       1|[47.0,1.0,28.0,0....|
|       0.0|       1|[56.0,1.0,28.0,0....|
|       0.0|       0|[76.0,1.0,28.0,0....|
|       0.0|       0|[25.0,1.0,35.0,1....|
|       0.0|       1|[51.0,1.0,28.0,0....|
+----------+--------+--------------------+
only showing top 5 rows

Gradient-boosted Trees Accuracy: 0.8786968031662545
GBT_ROC SCORE: 0.8528597869299667
```

4.Random Forest Classifier:- The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

```
from pyspark.ml.classification import RandomForestClassifier

rf = RandomForestClassifier(labelCol='Response',
                            featuresCol='features',
                            maxDepth=5)
model = rf.fit(trainingData)
rf_predictions = model.transform(testData)
print(rf_predictions)
rf_predictions.select("prediction", "Response", "features").show(5)
multi_evaluator = MulticlassClassificationEvaluator(labelCol = 'Response', metricName = 'accuracy')
print('Random Forest classifier Accuracy:', multi_evaluator.evaluate(rf_predictions))
evaluator = BinaryClassificationEvaluator(labelCol='Response', metricName='areaUnderROC')
roc_rf=evaluator.evaluate(gb_predictions)
print('RF_ROC SCORE:',roc_rf )
```

```
DataFrame[id: int, Gender: string, Age: int, Driving_License: int, Region_Code: double, Previously_Insu
+----------+--------+--------------------+
|prediction|Response|            features|
+----------+--------+--------------------+
|       0.0|       1|[47.0,1.0,28.0,0....|
|       0.0|       1|[56.0,1.0,28.0,0....|
|       0.0|       0|[76.0,1.0,28.0,0....|
|       0.0|       0|[25.0,1.0,35.0,1....|
|       0.0|       1|[51.0,1.0,28.0,0....|
+----------+--------+--------------------+
only showing top 5 rows

Random Forest classifier Accuracy: 0.8786968031662545
RF_ROC SCORE: 0.8528597869299667
```

e. RESULTS:-

Accuracy scores for all algorithms

```
Logistic Regression Accuracy: 0.8786968031662545
Accuracy of Factorization machines classifier is: 0.8786968031662545
Accuracy of Support Vector Machine is: 0.8786968031662545
Gradient-boosted Trees Accuracy: 0.8786968031662545
Random Forest classifier Accuracy: 0.8786968031662545
```

Roc score for all algorithms

```
LR_ROC SCORE: 0.8534113932238853
FMClassifier_ROC SCORE: 0.8534113932238853
LinearSVC_ROC SCORE: 0.8534113932238853
RF_ROC SCORE: 0.8534113932238853
GBT_ROC SCORE: 0.8534113932238853
```

5. GCP IMPLEMENTATION

5.1DATAPROC

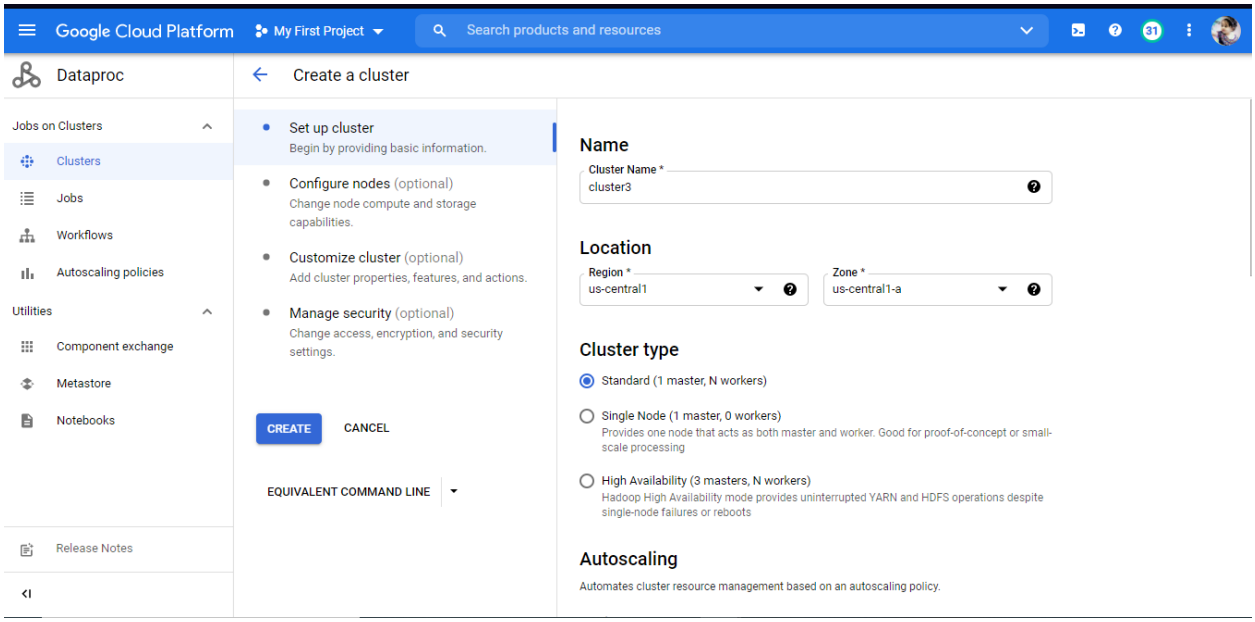
1.Implementation

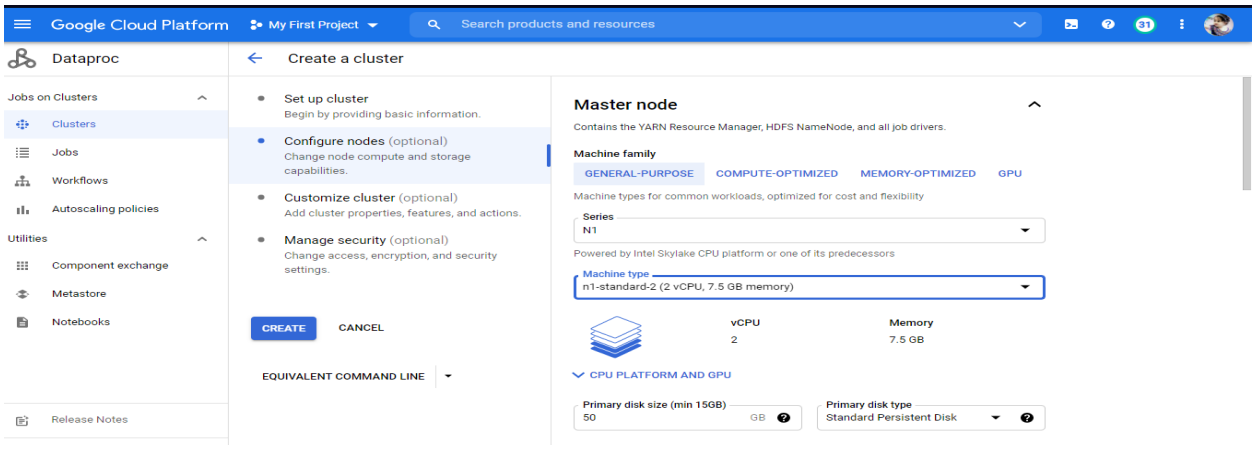
Dataproc is a managed Spark and Hadoop service that lets you take advantage of open source data tools for batch processing, querying, streaming, and machine learning. Dataproc automation helps you create clusters quickly, manage them easily, and save money by turning clusters off when you don't need them. With less time and money spent on administration, you can focus on your jobs and your data.

When compared to traditional, on-premises products and competing cloud services, Dataproc has a number of unique advantages for clusters of three to hundreds of nodes:

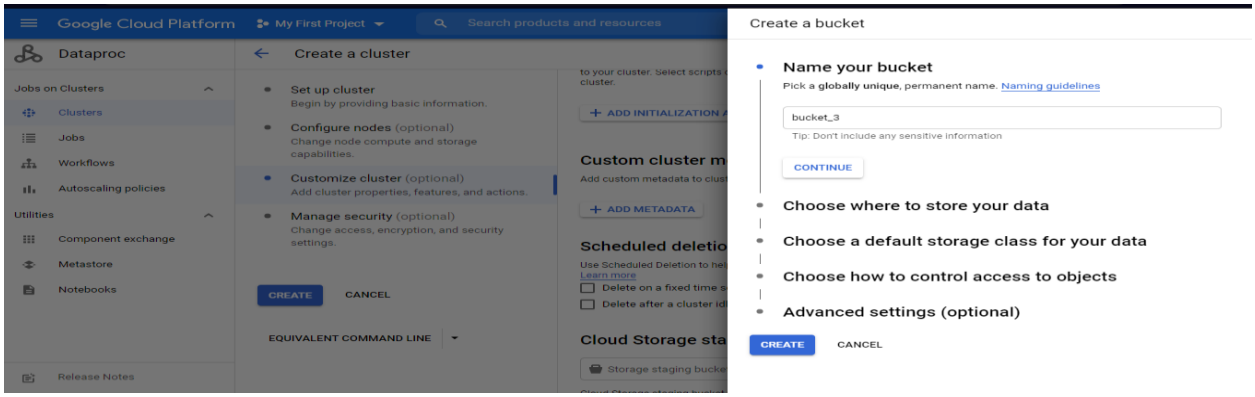
1. Low cost
2. Super fast
3. Integrated
4. Managed
5. Similar and familiar

STEP-1:- Create a cluster (Clustername, configure nodes -series –N1, n1 standard 2)

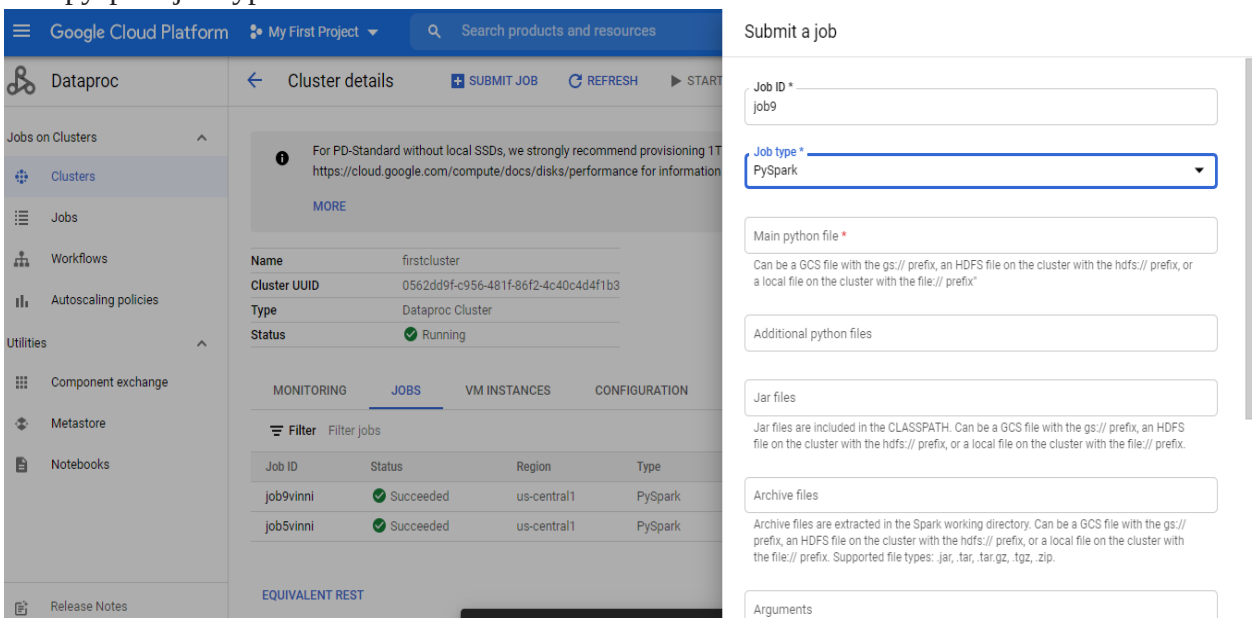




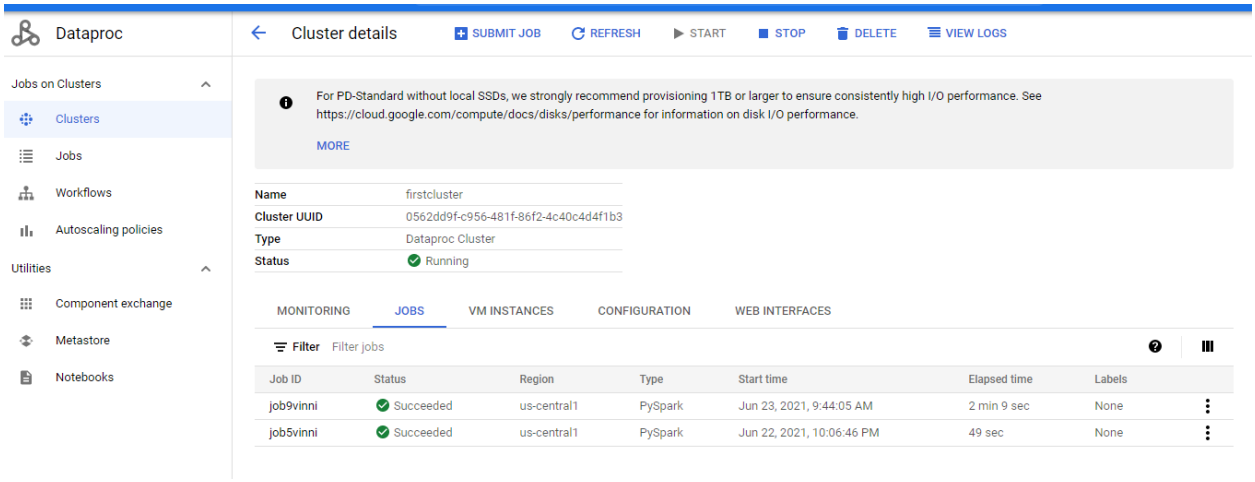
STEP-2:- create a bucket in cloud storage – to browse data files.



Step-3:- After uploading the data files, submit the job in cluster using cloud input file with pyspark job type.



Finally, Job was submitted successfully, Results will be runned,



2.RESULTS

Preprocessing

```
root
 |-- id: integer (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Driving_License: integer (nullable = true)
 |-- Region_Code: double (nullable = true)
 |-- Previously_Insured: integer (nullable = true)
 |-- Vehicle_Age: string (nullable = true)
 |-- Vehicle_Damage: string (nullable = true)
 |-- Annual_Premium: double (nullable = true)
 |-- Policy_Sales_Channel: double (nullable = true)
 |-- Vintage: integer (nullable = true)
 |-- Response: integer (nullable = true)
```

```
+--------+------+
|Response| count|
+--------+------+
|       1| 46710|
|       0|334399|
+--------+------+
```

```
+------+---+--------------------+
|Gender|Age|Policy_Sales_Channel|
+------+---+--------------------+
|Male  |44 |26.0                |
|Male  |76 |26.0                |
|Male  |47 |26.0                |
|Male  |21 |152.0               |
|Female|29 |152.0               |
+------+---+--------------------+
only showing top 5 rows
```

```
+---+------+--------------------+
|Age|Gender|Policy_Sales_Channel|
+---+------+--------------------+
|44 |Male  |26.0                |
|76 |Male  |26.0                |
|47 |Male  |26.0                |
|21 |Male  |152.0               |
|29 |Female|152.0               |
|24 |Female|160.0               |
|23 |Male  |152.0               |
|56 |Female|26.0                |
|24 |Female|152.0               |
|32 |Female|152.0               |
+---+------+--------------------+
only showing top 10 rows
```

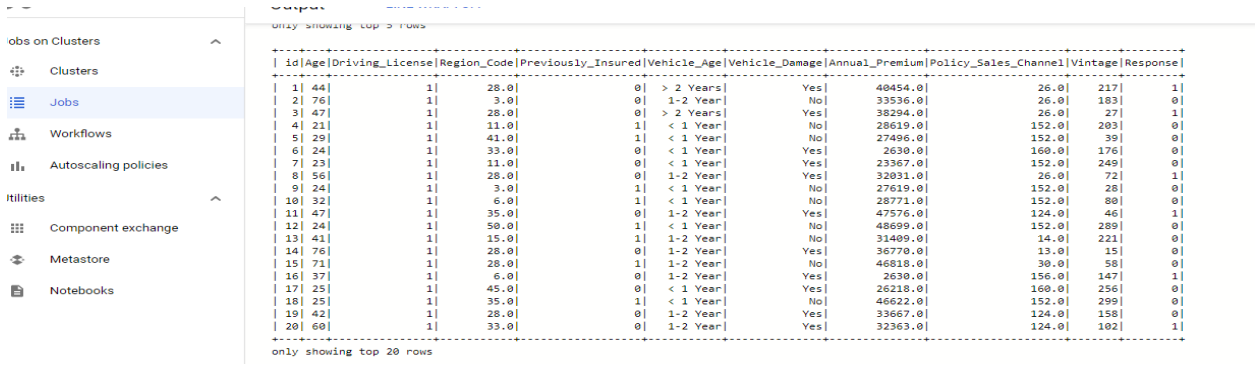
```
+---+------+--------------------+
|Age|Gender|Policy_Sales_Channel|
+---+------+--------------------+
|44 |Male  |26.0                |
|76 |Male  |26.0                |
|47 |Male  |26.0                |
|21 |Male  |152.0               |
|29 |Female|152.0               |
+---+------+--------------------+
only showing top 5 rows
```

```
+---+------+---+--------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
| id|Gender|Age|Driving_License|Region_Code|Previously_Insured|Vehicle_Age|Vehicle_Damage|Annual_Premium|Policy_Sales_Channel|Vintage|Response|
+---+------+---+--------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
|  1|  Male| 44|             1|       28.0|                0|  > 2 Years|           Yes|       40454.0|                26.0|    217|       1|
| 10|Female| 32|             1|        6.0|                1|   < 1 Year|            No|       28771.0|               152.0|     80|       0|
| 13|Female| 41|             1|       15.0|                1|   1-2 Year|            No|       31409.0|                14.0|    221|       0|
| 16|  Male| 37|             1|        6.0|                0|   1-2 Year|           Yes|        2630.0|               156.0|    147|       1|
| 19|  Male| 42|             1|       28.0|                0|   1-2 Year|           Yes|       33667.0|               124.0|    158|       0|
+---+------+---+--------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
only showing top 5 rows
```

```
+-------+------------------+------+------------------+------------------+------------------+------------------+-----------+--------------+------------------+-----
|summary|                id|Gender|               Age|    Driving_License|       Region_Code| Previously_Insured|Vehicle_Age|Vehicle_Damage|    Annual_Premium|Polic
+-------+------------------+------+------------------+------------------+------------------+------------------+-----------+--------------+------------------+-----
|  count|            381109|381109|            381109|            381109|            381109|            381109|     381109|        381109|            381109|
|   mean|          190555.0|  null|38.822583565331705|0.9978693759528114|26.388807401557035|0.4582101183650871|       null|          null|30564.389581458323|  112
| stddev|110016.83620776715|  null|15.511611018095289|0.04610954420779939|13.229888025788491|0.4982511988722647|       null|          null| 17213.15505698001|   54
|    min|                 1|Female|                20|                 0|               0.0|                 0| 1-2 Year|            No|            2630.0|
|    max|            381109|  Male|                85|                 1|              52.0|                 1| > 2 Years|           Yes|          540165.0|
+-------+------------------+------+------------------+------------------+------------------+------------------+-----------+--------------+------------------+-----
```

```
+---+------+---+--------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
| id|Gender|Age|Driving_License|Region_Code|Previously_Insured|Vehicle_Age|Vehicle_Damage|Annual_Premium|Policy_Sales_Channel|Vintage|Response|
+---+------+---+--------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
|  0|     0|  0|             0|          0|                0|          0|             0|             0|                   0|      0|       0|
+---+------+---+--------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
```

```
+------+-------------+
|Gender|sum(Response)|
+------+-------------+
|Female|        18185|
|  Male|        28525|
+------+-------------+
```




```
+---+---+--------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
| id|Age|Driving_License|Region_Code|Previously_Insured|Vehicle_Age|Vehicle_Damage|Annual_Premium|Policy_Sales_Channel|Vintage|Response|
+---+---+--------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
|  1| 44|             1|       28.0|                0|  > 2 Years|           Yes|       40454.0|                26.0|    217|       1|
|  2| 76|             1|        3.0|                0|   1-2 Year|            No|       33536.0|                26.0|    183|       0|
|  3| 47|             1|       28.0|                0|  > 2 Years|           Yes|       38294.0|                26.0|     27|       1|
|  4| 21|             1|       11.0|                1|   < 1 Year|            No|       28619.0|               152.0|    203|       0|
|  5| 29|             1|       41.0|                1|   < 1 Year|            No|       27496.0|               152.0|     39|       0|
|  6| 24|             1|       33.0|                0|   < 1 Year|           Yes|        2630.0|               160.0|    176|       0|
|  7| 23|             1|       11.0|                0|   < 1 Year|           Yes|       23367.0|               152.0|    249|       0|
|  8| 56|             1|       28.0|                0|   1-2 Year|           Yes|       32031.0|                26.0|     72|       1|
|  9| 24|             1|        3.0|                1|   < 1 Year|            No|       27619.0|               152.0|     28|       0|
| 10| 32|             1|        6.0|                1|   < 1 Year|            No|       28771.0|               152.0|     80|       0|
| 11| 47|             1|       35.0|                0|   1-2 Year|           Yes|       47576.0|               124.0|     46|       1|
| 12| 24|             1|       50.0|                1|   < 1 Year|            No|       48699.0|               152.0|    289|       0|
| 13| 41|             1|       15.0|                1|   1-2 Year|            No|       31409.0|                14.0|    221|       0|
| 14| 76|             1|       28.0|                0|   1-2 Year|           Yes|       36770.0|                13.0|     15|       0|
| 15| 71|             1|       28.0|                1|   1-2 Year|            No|       46818.0|                30.0|     58|       0|
| 16| 37|             1|        6.0|                0|   1-2 Year|           Yes|        2630.0|               156.0|    147|       1|
| 17| 25|             1|       45.0|                0|   < 1 Year|           Yes|       26218.0|               160.0|    256|       0|
| 18| 25|             1|       35.0|                1|   < 1 Year|            No|       46622.0|               152.0|    299|       0|
| 19| 42|             1|       28.0|                0|   1-2 Year|           Yes|       33667.0|               124.0|    158|       0|
| 20| 60|             1|       33.0|                0|   1-2 Year|           Yes|       32363.0|               124.0|    102|       1|
+---+---+--------------+-----------+-----------------+-----------+--------------+--------------+--------------------+-------+--------+
only showing top 20 rows
```

```
+--------------+--------------------+-----------+-----------------+
|Vehicle_Damage|Vehicle_Damage_Index|Vehicle_Age|Vehicle_Age_Index|
+--------------+--------------------+-----------+-----------------+
|           Yes|                 0.0|  > 2 Years|              2.0|
|            No|                 1.0|   1-2 Year|              0.0|
|           Yes|                 0.0|  > 2 Years|              2.0|
|            No|                 1.0|   < 1 Year|              1.0|
|            No|                 1.0|   < 1 Year|              1.0|
|           Yes|                 0.0|   < 1 Year|              1.0|
|           Yes|                 0.0|   < 1 Year|              1.0|
|           Yes|                 0.0|   1-2 Year|              0.0|
|            No|                 1.0|   < 1 Year|              1.0|
|            No|                 1.0|   < 1 Year|              1.0|
+--------------+--------------------+-----------+-----------------+
only showing top 10 rows
```

```
+--------------+--------------------+-----------------+-----------+-----------------+---------------+
|Vehicle_Damage|Vehicle_Damage_Index|Vehicle_Damage_OHE|Vehicle_Age|Vehicle_Age_Index|Vehicle_Age_OHE|
+--------------+--------------------+-----------------+-----------+-----------------+---------------+
|           Yes|                 0.0|    (1,[0],[1.0])|  > 2 Years|              2.0|      (2,[],[])|
|            No|                 1.0|        (1,[],[])|   1-2 Year|              0.0| (2,[0],[1.0])|
|           Yes|                 0.0|    (1,[0],[1.0])|  > 2 Years|              2.0|      (2,[],[])|
|            No|                 1.0|        (1,[],[])|   < 1 Year|              1.0| (2,[1],[1.0])|
|            No|                 1.0|        (1,[],[])|   < 1 Year|              1.0| (2,[1],[1.0])|
+--------------+--------------------+-----------------+-----------+-----------------+---------------+
only showing top 5 rows
```

Gradient Boosted Tree

```
DataFrame[id: int, Gender: string, Age: int, Driving_License: int, Region_Code: double, Previously_Insured: int, Vehicle_Age: string, Vehicle_Damage: string, Annual
+----------+--------+--------------------+
|prediction|Response|            features|
+----------+--------+--------------------+
|       0.0|       1|[47.0,1.0,28.0,0....|
|       0.0|       1|[56.0,1.0,28.0,0....|
|       0.0|       0|[76.0,1.0,28.0,0....|
|       0.0|       0|[25.0,1.0,35.0,1....|
|       0.0|       1|[51.0,1.0,28.0,0....|
+----------+--------+--------------------+
only showing top 5 rows

Gradient-boosted Trees Accuracy: 0.8776009364244658
DataFrame[id: int, Gender: string, Age: int, Driving_License: int, Region_Code: double, Previously_Insured: int, Vehicle_Age: string, Vehicle_Damage: string, Annual
```

Random Forest Classifier

```
Gradient-boosted Trees Accuracy: 0.8776009364244658
DataFrame[id: int, Gender: string, Age: int, Driving_License: int, Region_Code: double, Previously_Insured: int, Vehicle_Age: string, Vehicle_Damage: string, Annual
+----------+--------+--------------------+
|prediction|Response|            features|
+----------+--------+--------------------+
|       0.0|       1|[47.0,1.0,28.0,0....|
|       0.0|       1|[56.0,1.0,28.0,0....|
|       0.0|       0|[76.0,1.0,28.0,0....|
|       0.0|       0|[25.0,1.0,35.0,1....|
|       0.0|       1|[51.0,1.0,28.0,0....|
+----------+--------+--------------------+
only showing top 5 rows

Random Forest classifier Accuracy: 0.8776009364244658
```

Linear Regression

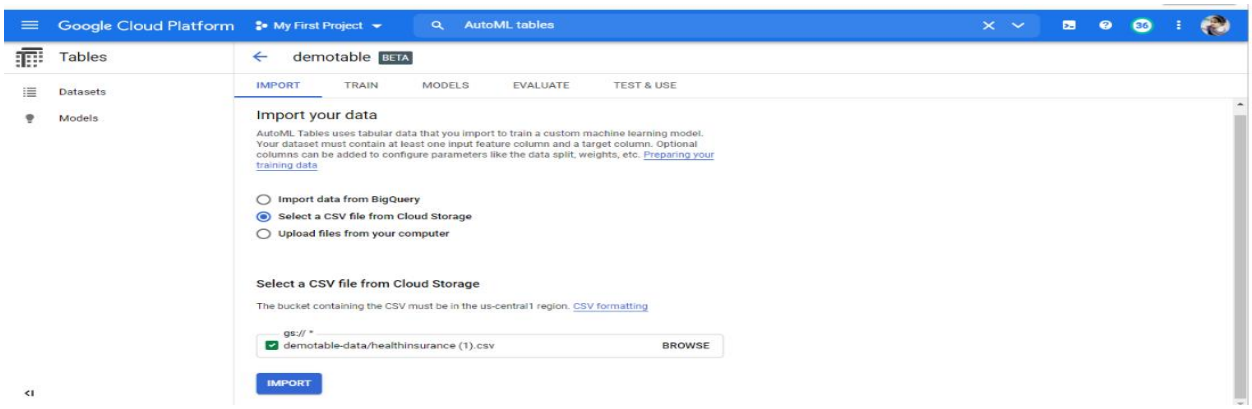
```
Coefficients: [0.01902751283606541]
Intercept: 25.65648188890467
RMSE: 13.238974
r2: 0.001313
DataFrame[id: int, Gender: string, Age: int, Driving_License: int, Region_Code: double, Previously_Insured: int, Vehicle_Age: string, Vehicle_Damage: string, Annual_
+------------------+-----------+--------+
|        prediction|Region_Code|features|
+------------------+-----------+--------+
|26.493692453691548|       28.0|  [44.0]|
| 26.55077499219974|       28.0|  [47.0]|
|26.113142196970237|       33.0|  [24.0]|
| 26.26536229965876|        6.0|  [32.0]|
| 26.55077499219974|       35.0|  [47.0]|
+------------------+-----------+--------+
only showing top 5 rows

R Squared (R2) on test data = 0.00149068
```

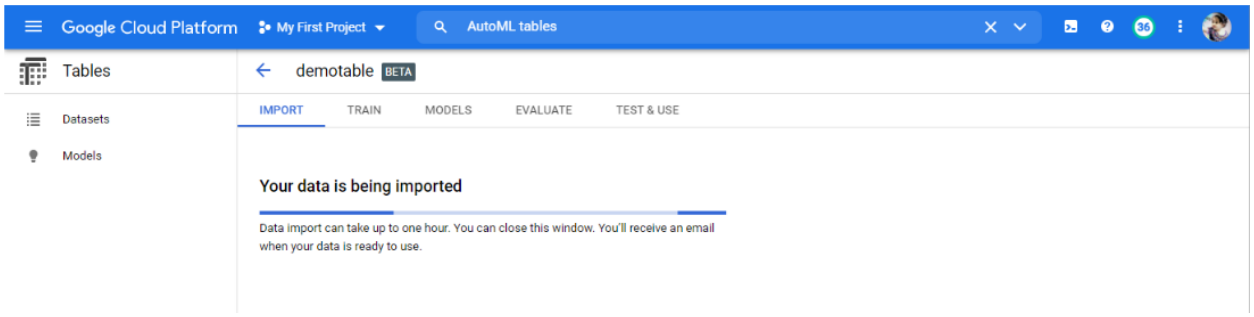
5.2 AUTOML TABLES

1. Implementation

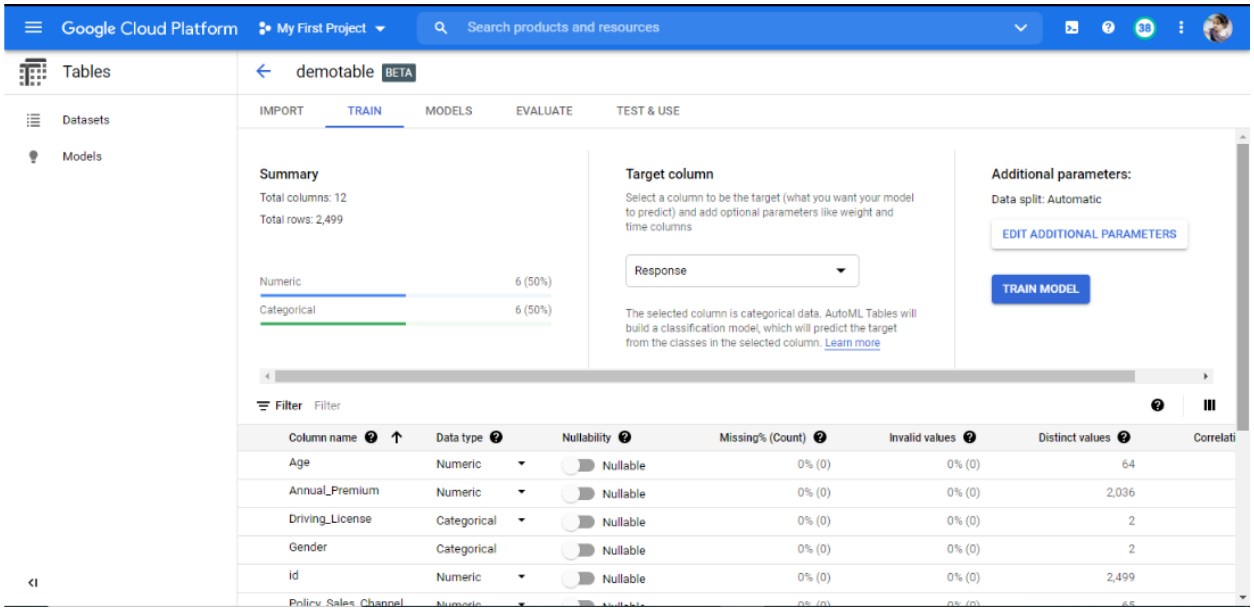
AutoML Tables enables your entire team to automatically build and deploy state-of-the-art machine learning models on structured data at massively increased speed and scale. ... **AutoML Tables** is now available in the new, unified Vertex AI. **It** is a supervised learning service. This means that you train a machine learning model with example data. **AutoML Tables** uses tabular (structured) data to train a machine learning model to make predictions on new data.

STEP-1:- Create a dataset and import a csv file from cloud storage.

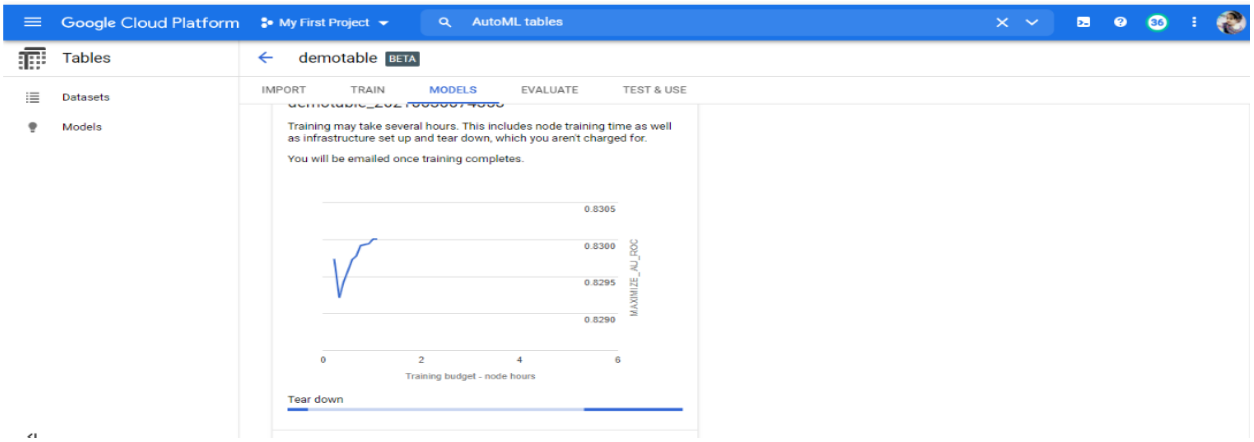


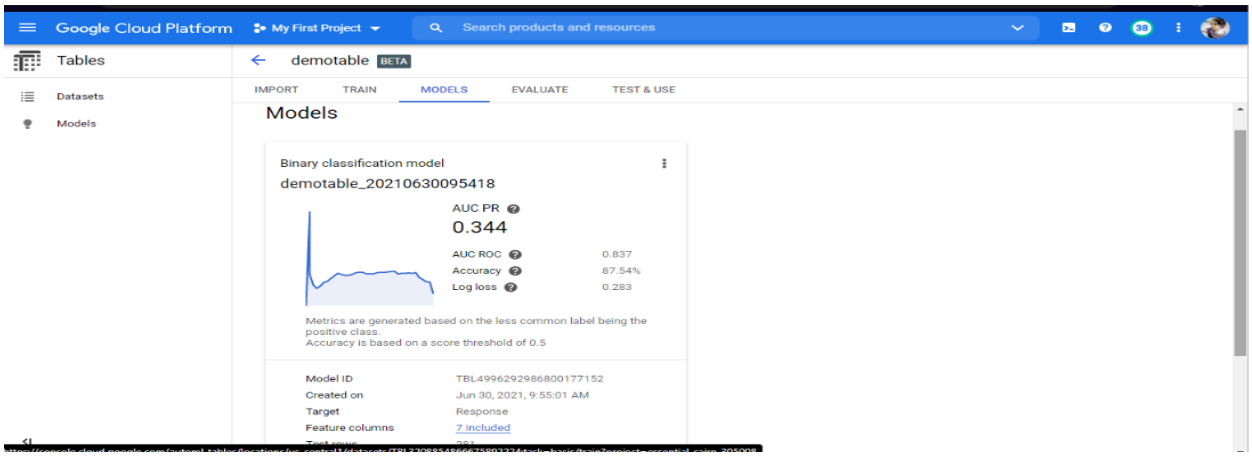


STEP-2:- Give the Target Column & Train the model

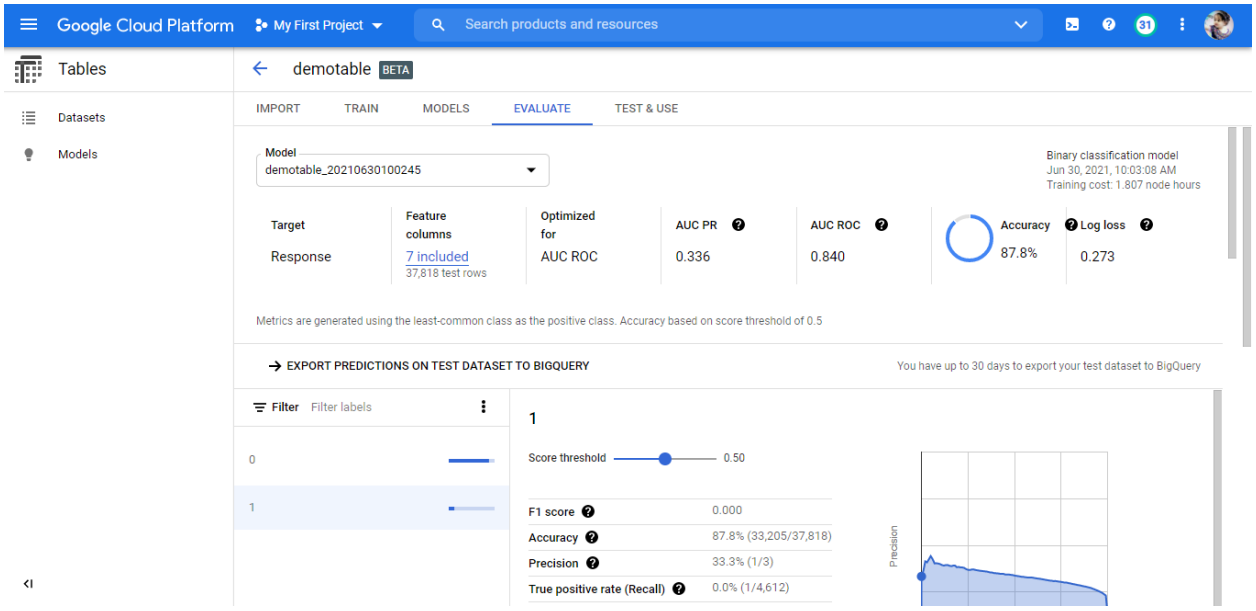


Model is training

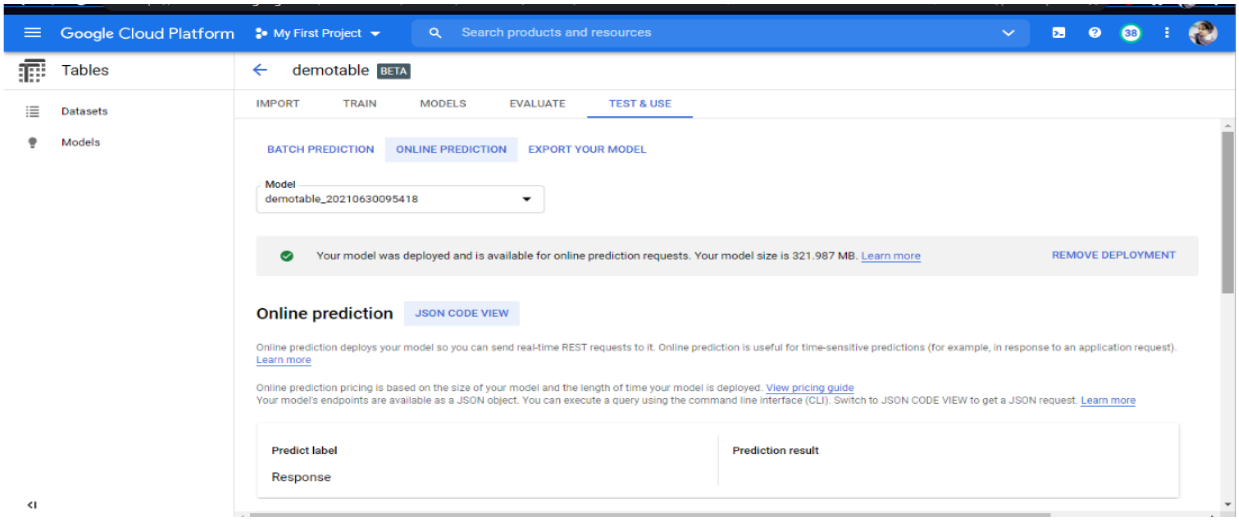




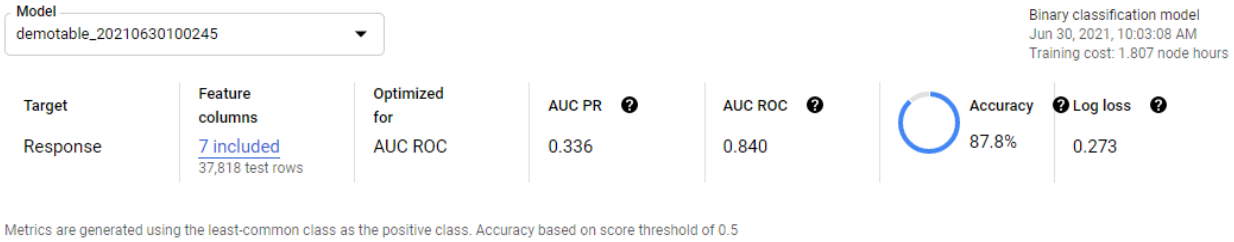
After model is trained, Evaluate the results

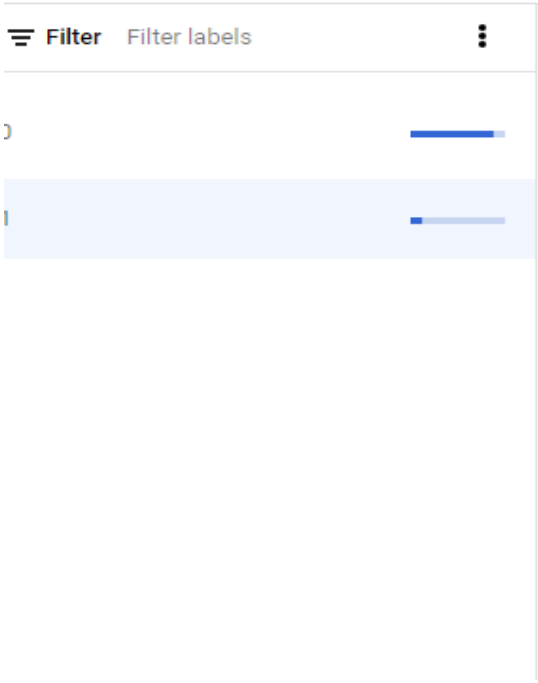


STEP-3:- In Test & use Deploy the model in online prediction.



2.Results



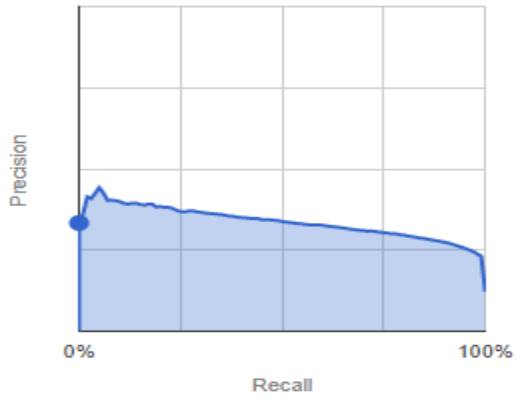


1

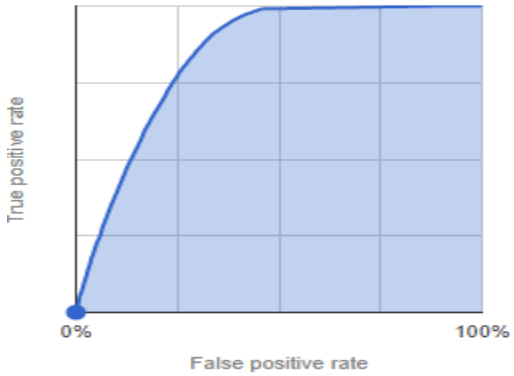
Score threshold 0.50

F1 score	0.000
Accuracy	87.8% (33,205/37,818)
Precision	33.3% (1/3)
True positive rate (Recall)	0.0% (1/4,612)
False positive rate	0.000 (2/33,206)

The score threshold determines the minimum level of confidence needed to make a prediction positive. Learn more about model evaluation



AUC: 0.336 PRC



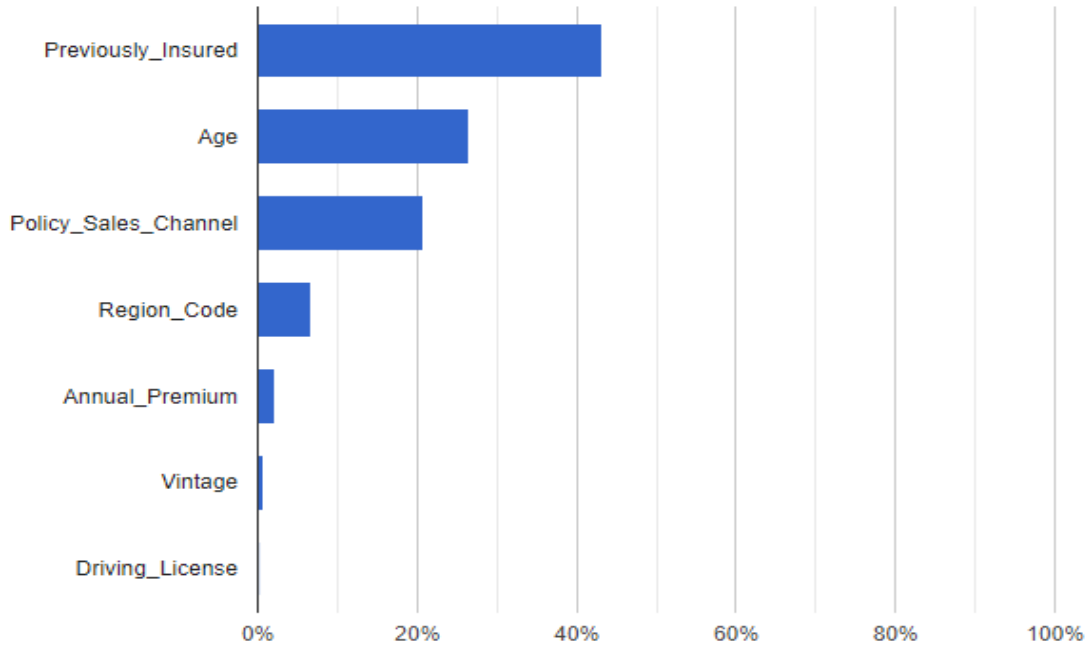
AUC: 0.84 ROC

Confusion matrix

A confusion matrix summarizes how successful a classification model's predictions were. The matrix rows are ground truth labels and the columns are labels predicted by the model. Scan each row to determine where misclassifications occur for a given label: the blue cell indicates accurate predictions for the label (true positives or negatives) while the grey cells indicate errors (false positives or negatives). Hover over the matrix cells to learn more.

True labels \ Predicted labels	0	1
0	100%	0%
1	100%	0%

Feature importance



Online prediction results

Feature column name	Column ID	Data type	Status	Value	Local feature importa
Age	1293419849183133696	Numeric	Required	47	
Annual_Premium	3022802106093404160	Numeric	Required	35116	
Driving_License	4175723610700251136	Categorical	Required	1	
Policy_Sales_Channel	7634488124520792064	Numeric	Required	61	
Previously_Insured	5905105867610521600	Categorical	Required	1	
Region_Code	1869880601486557184	Numeric	Required	28	

Predict label

Response

Prediction result

Baseline prediction value: 0.767

0 Confidence score: 0.976

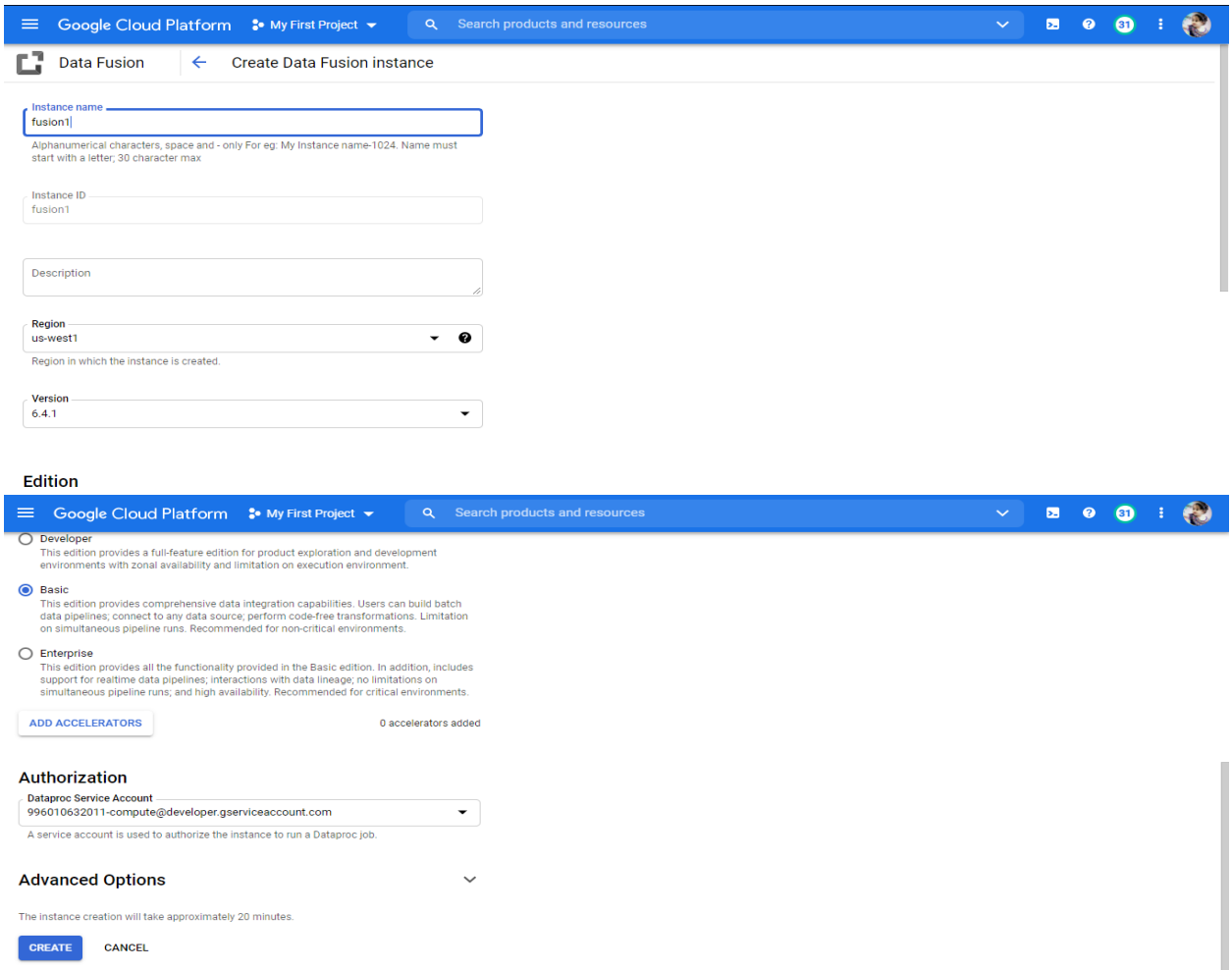
1 Confidence score: 0.024

5.3 DATAFUSION

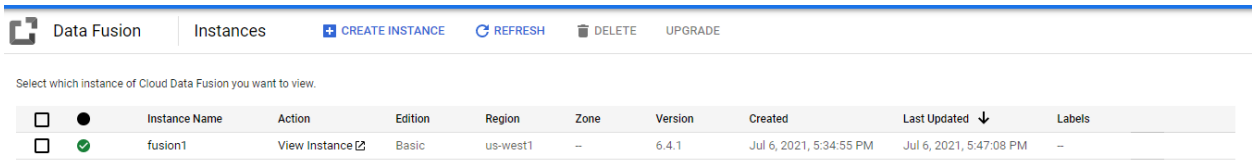
1.Implementation

Cloud **Data Fusion** is the brand new, fully-managed data engineering product from Google Cloud. It will help users to efficiently build and manage ETL/ELT data pipelines. ... Built on top of the open-source project CDAP, it leverages a convenient user interface for building data pipelines in a 'drag and drop' manner. It helps users build scalable, distributed data lakes on Google Cloud by integrating data from siloed on-premises platforms.

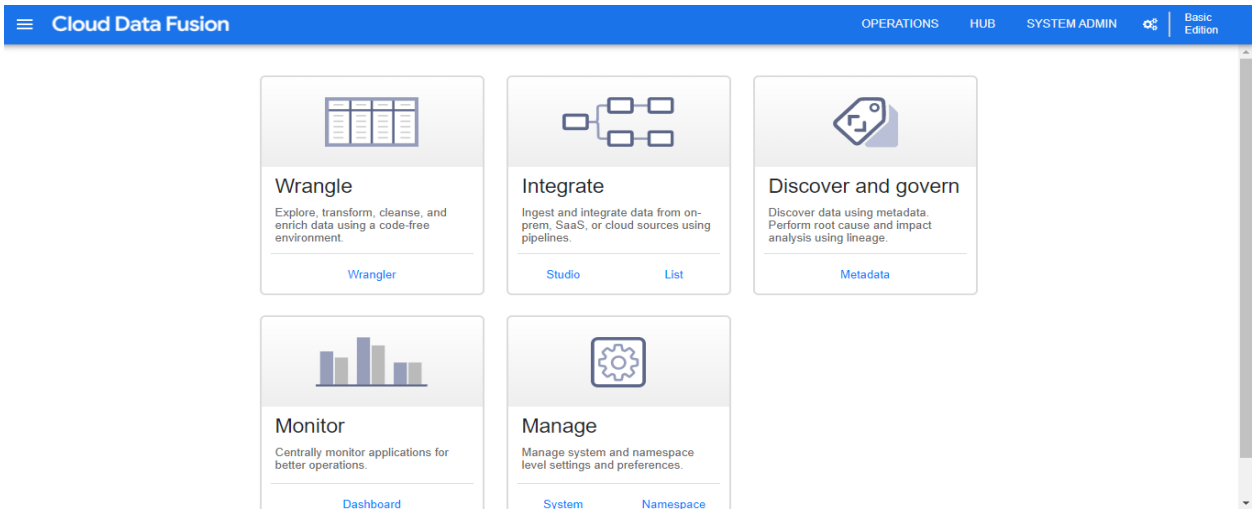
STEP-1:- Create a DataFusion Instance.



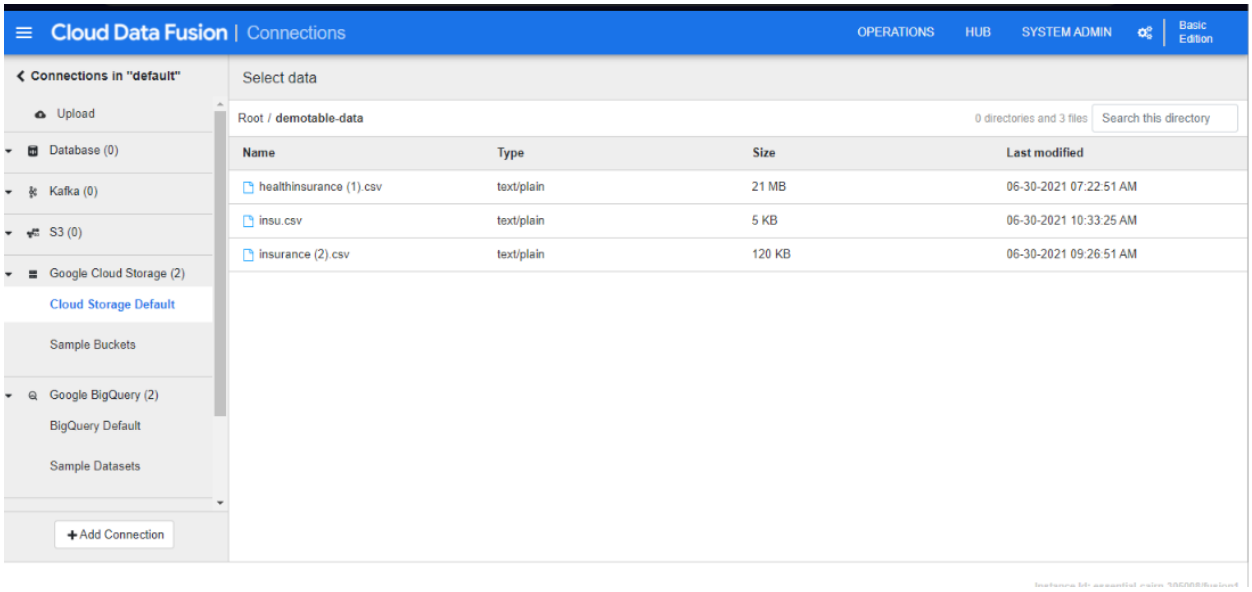
Step-2:- if, Instance is created.view the instance

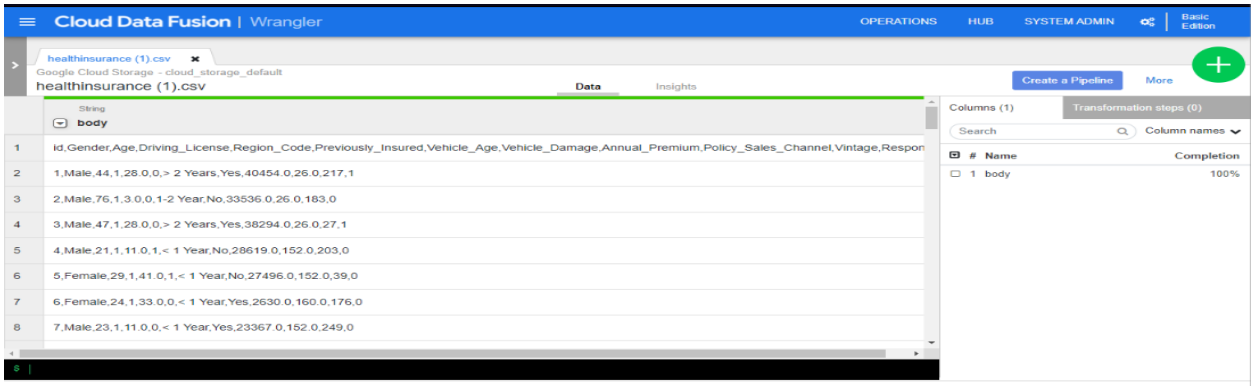


Step-3:- click the Wrangler

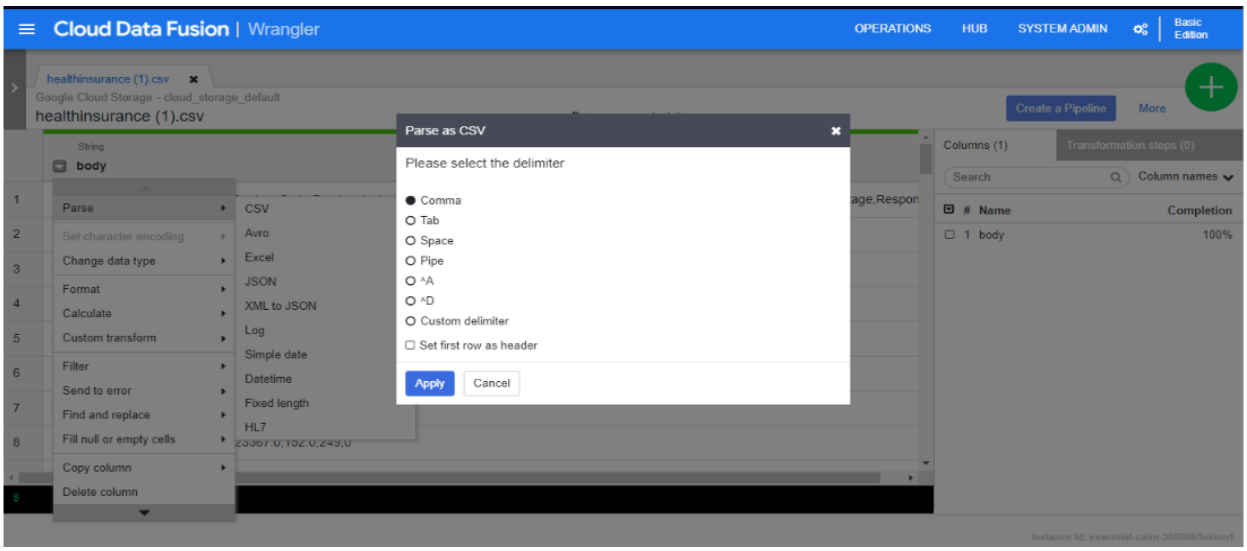


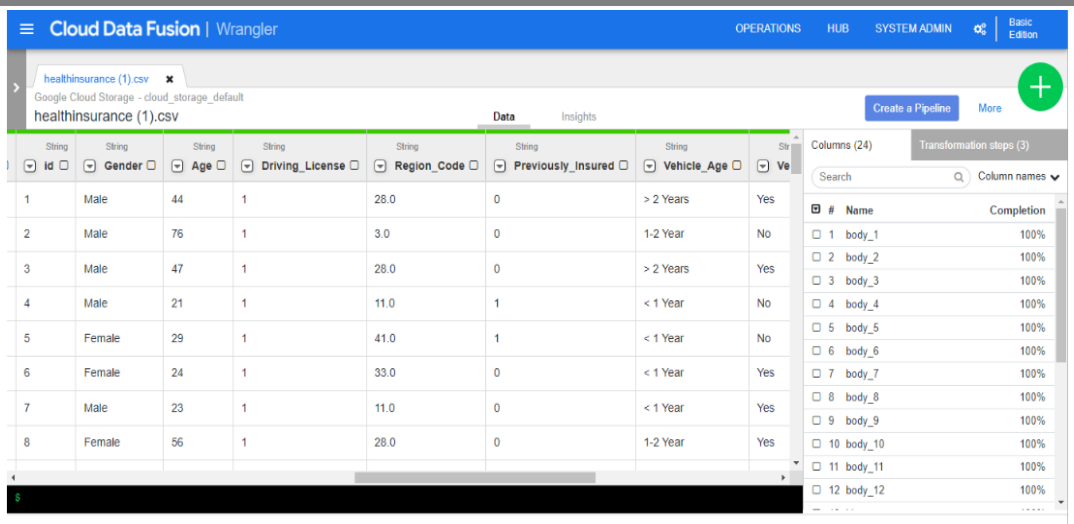
Step-4:- Upload the csv file from cloud storage or device.



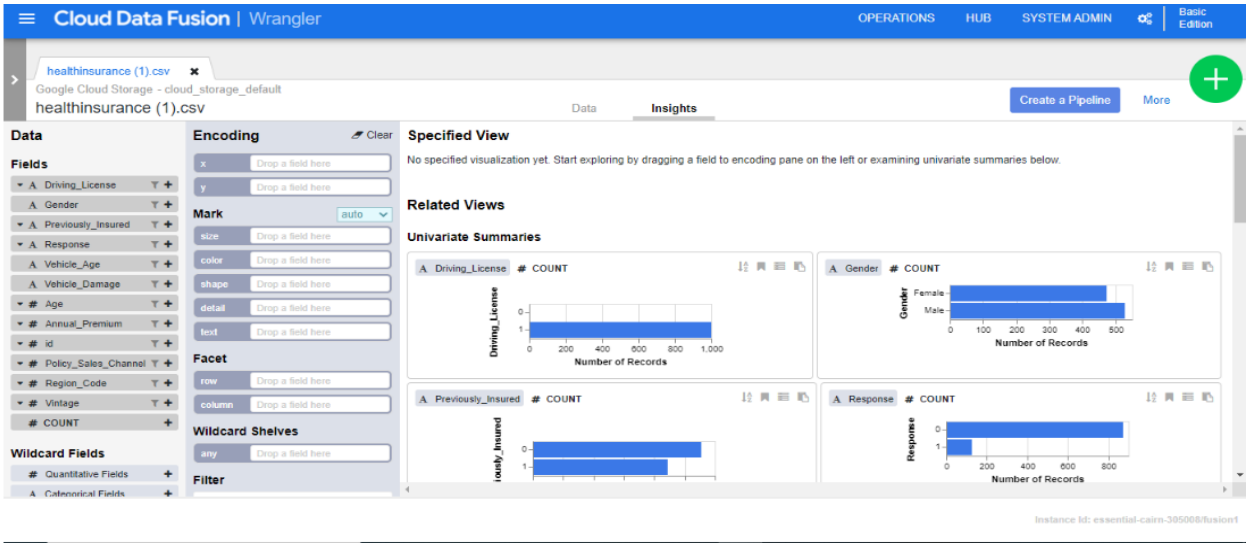


Parse the data into column names

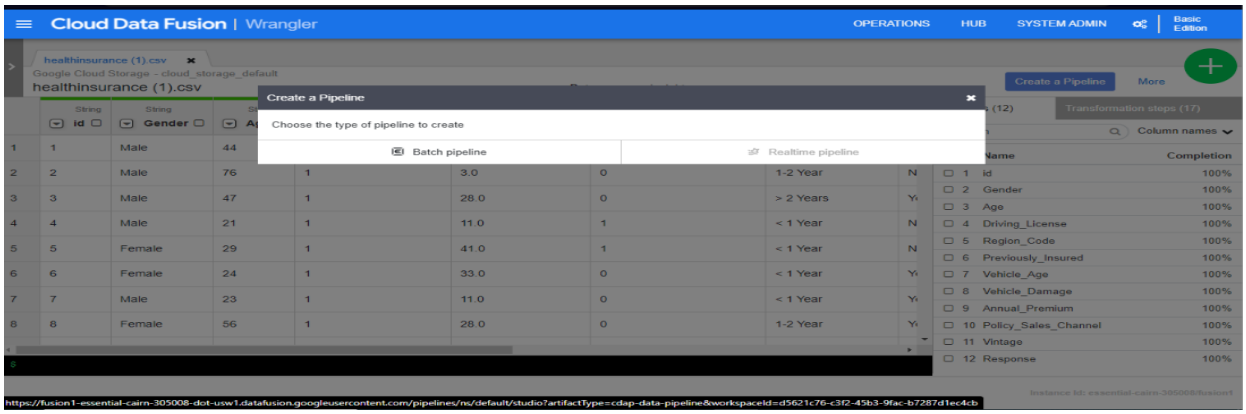




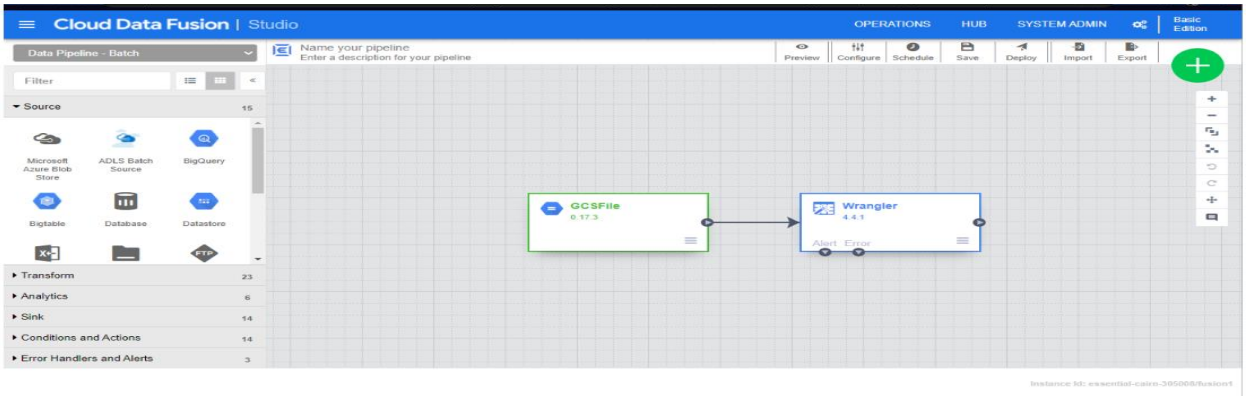
Step-5:- Click the insights & plot the different insights from dataset

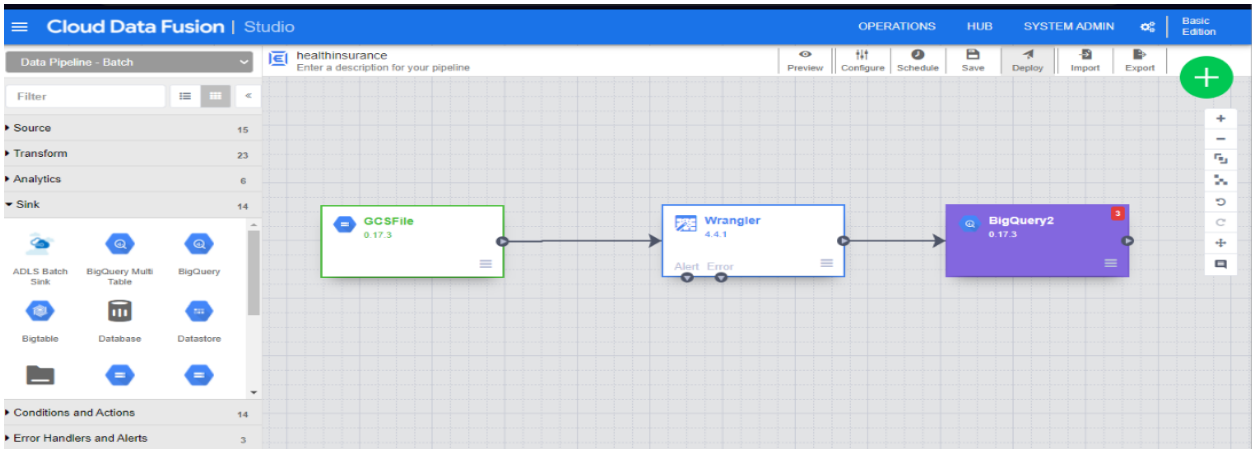


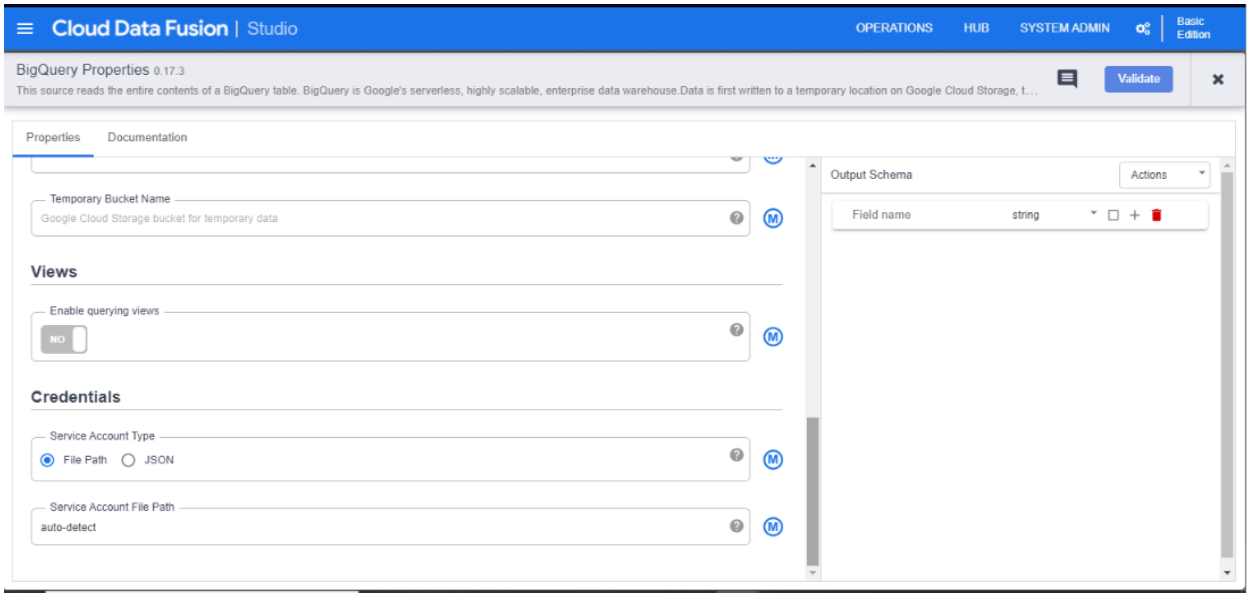
Step-6:- Create a pipeline with Batch pipeline.

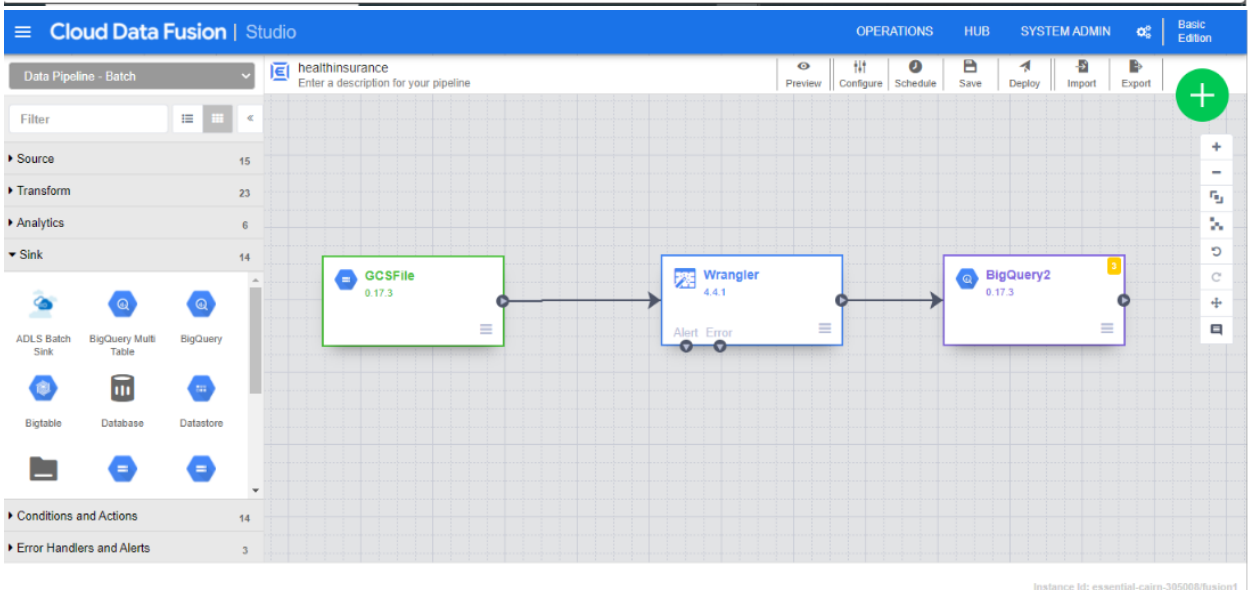


Step-7:- Sink the BigQuery pipeline to Wrangler and give the project details and properties.



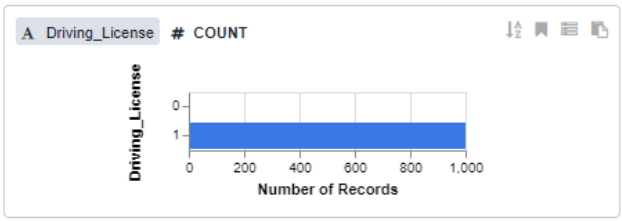

Cloud Data Fusion | Studio
OPERATIONS
HUB
SYSTEM ADMIN
Basic Edition
Data Pipeline - Batch
healthinsurance
Enter a description for your pipeline
Preview
Configure
Schedule
Save
Deploy
Import
Export
Filter
Source 15
Transform 23
Analytics 6
Sink 14
ADLS Batch Sink
BigQuery Multi Table
BigQuery
Bigtable
Database
Datastore
Conditions and Actions 14
Error Handlers and Alerts 3
GCSFile 0.17.3
Wrangler 4.4.1
Alert Error
BigQuery2 0.17.3

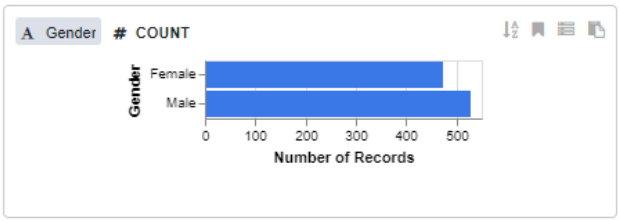

Cloud Data Fusion | Studio
OPERATIONS
HUB
SYSTEM ADMIN
Basic Edition
BigQuery Properties 0.17.3
This source reads the entire contents of a BigQuery table. BigQuery is Google's serverless, highly scalable, enterprise data warehouse.Data is first written to a temporary location on Google Cloud Storage, t...
Validate
Properties
Documentation
Temporary Bucket Name
Google Cloud Storage bucket for temporary data
Views
Enable querying views
NO
Credentials
Service Account Type
File Path
JSON
Service Account File Path
auto-detect
Output Schema
Actions
Field name
string

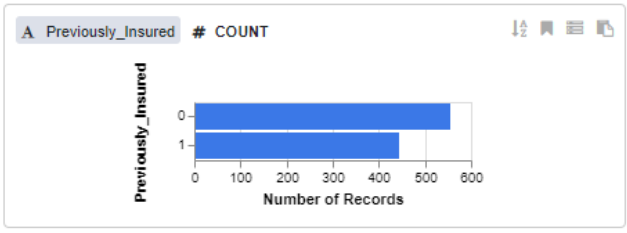

Cloud Data Fusion | Studio
OPERATIONS
HUB
SYSTEM ADMIN
Basic Edition
Data Pipeline - Batch
healthinsurance
Enter a description for your pipeline
Preview
Configure
Schedule
Save
Deploy
Import
Export
Filter
Source 15
Transform 23
Analytics 6
Sink 14
ADLS Batch Sink
BigQuery Multi Table
BigQuery
Bigtable
Database
Datastore
Conditions and Actions 14
Error Handlers and Alerts 3
GCSFile 0.17.3
Wrangler 4.4.1
Alert Error
BigQuery2 0.17.3

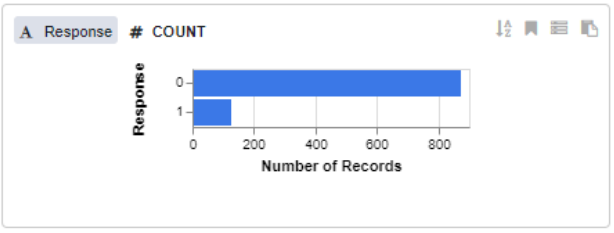
2. RESULTS

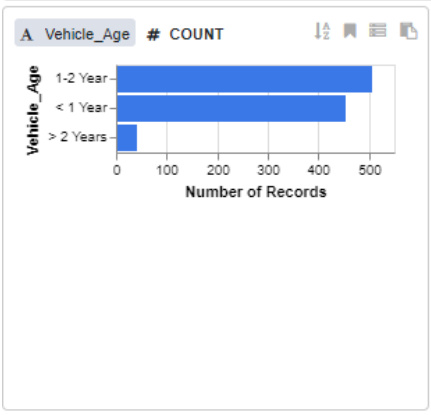
String id	String Gender	String Age	String Driving_License	Float Region_Code	String Previously_Insured	String Vehicle_Age	
1	Male	44	1	28.0	0	> 2 Years	Y
2	Male	76	1	3.0	0	1-2 Year	N
3	Male	47	1	28.0	0	> 2 Years	Y
4	Male	21	1	11.0	1	< 1 Year	N
5	Female	29	1	41.0	1	< 1 Year	N
6	Female	24	1	33.0	0	< 1 Year	Y
7	Male	23	1	11.0	0	< 1 Year	Y
8	Female	56	1	28.0	0	1-2 Year	Y

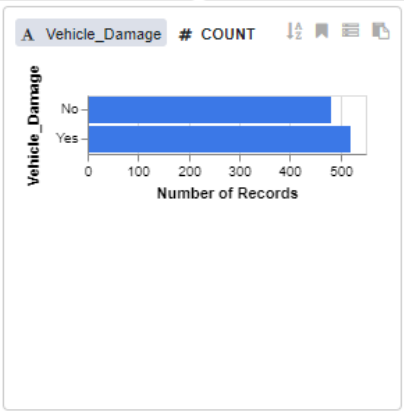


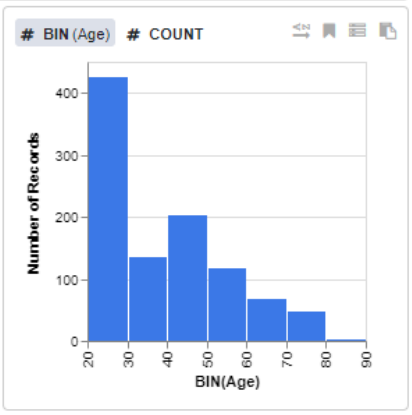


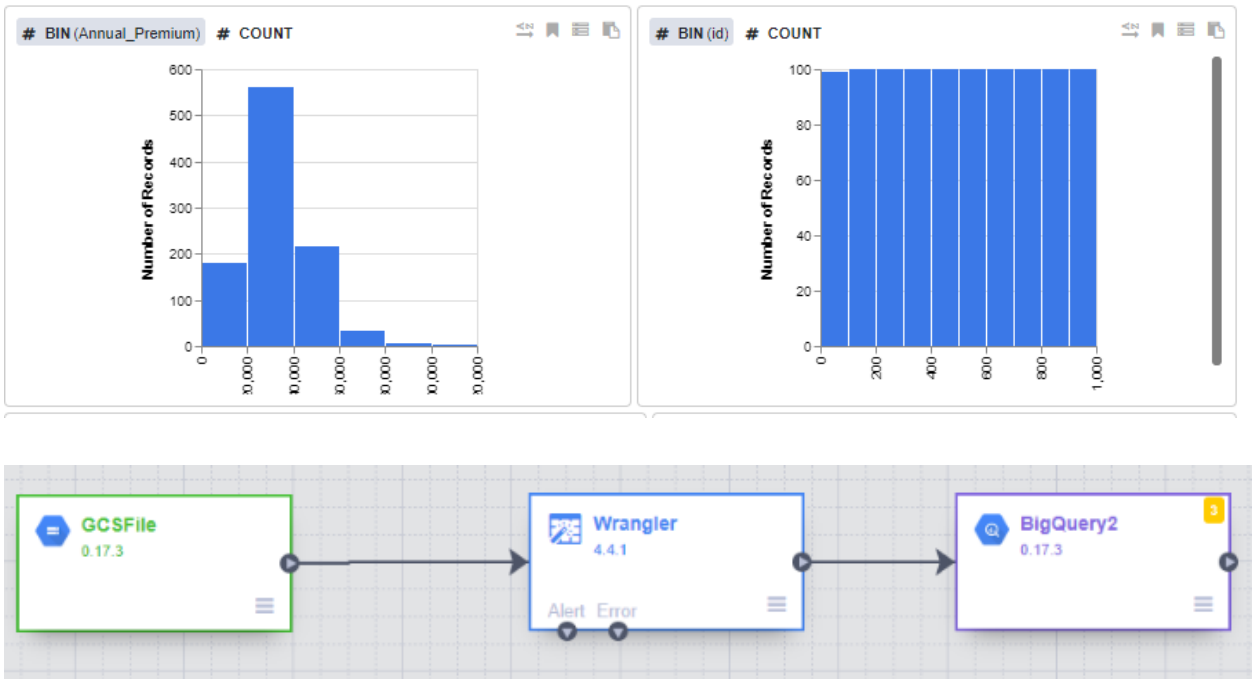








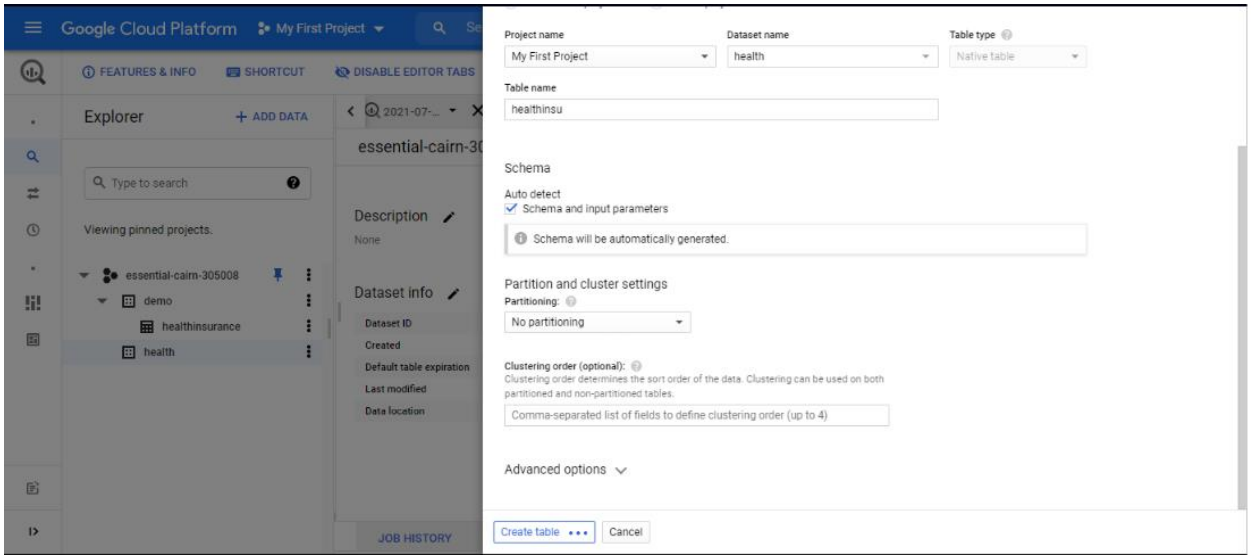


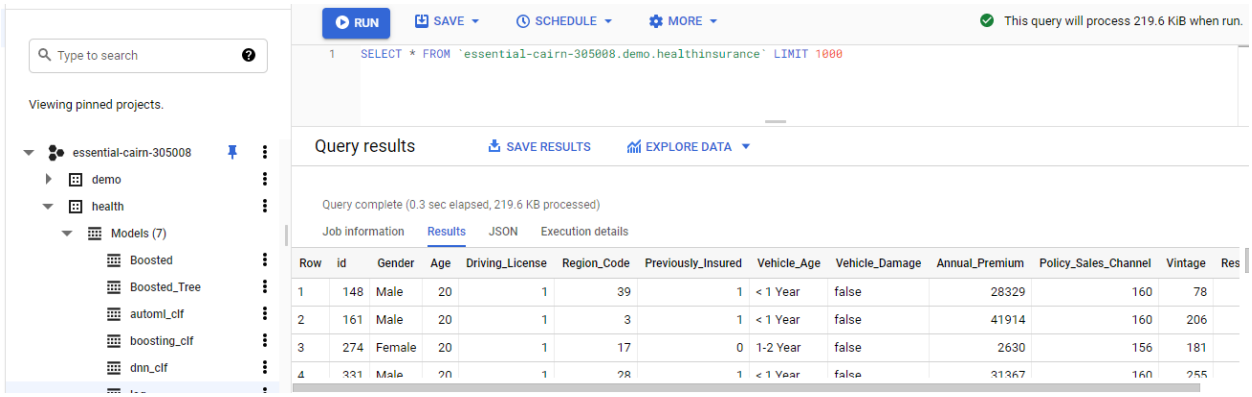


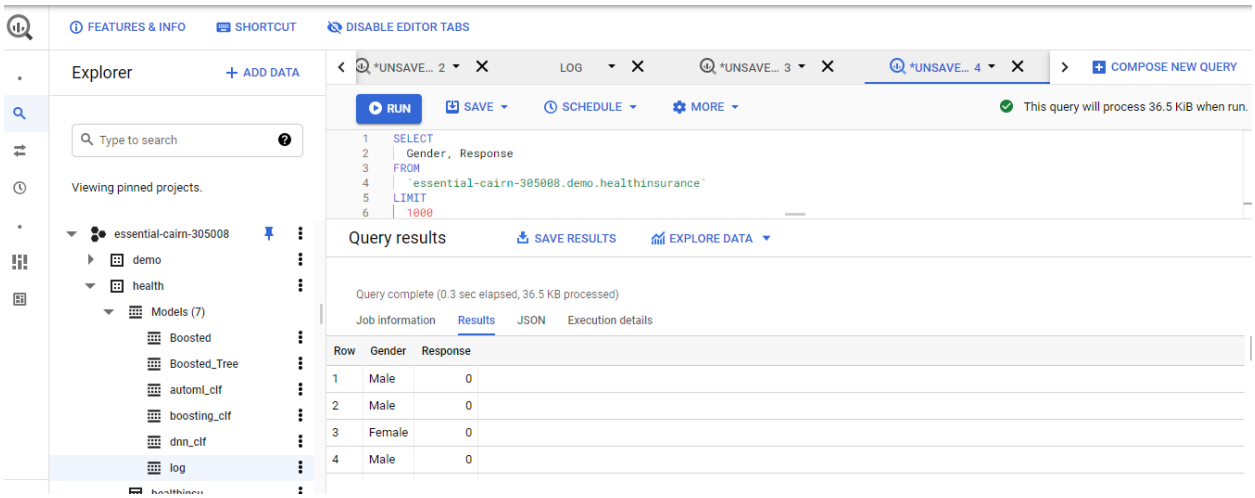
5.4 BIGQUERY

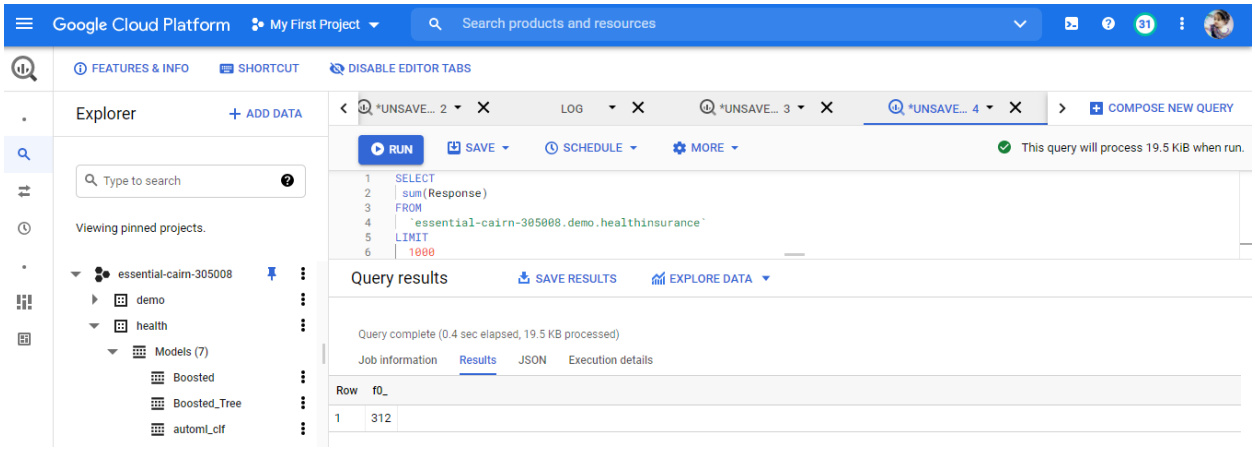
1. Implementation

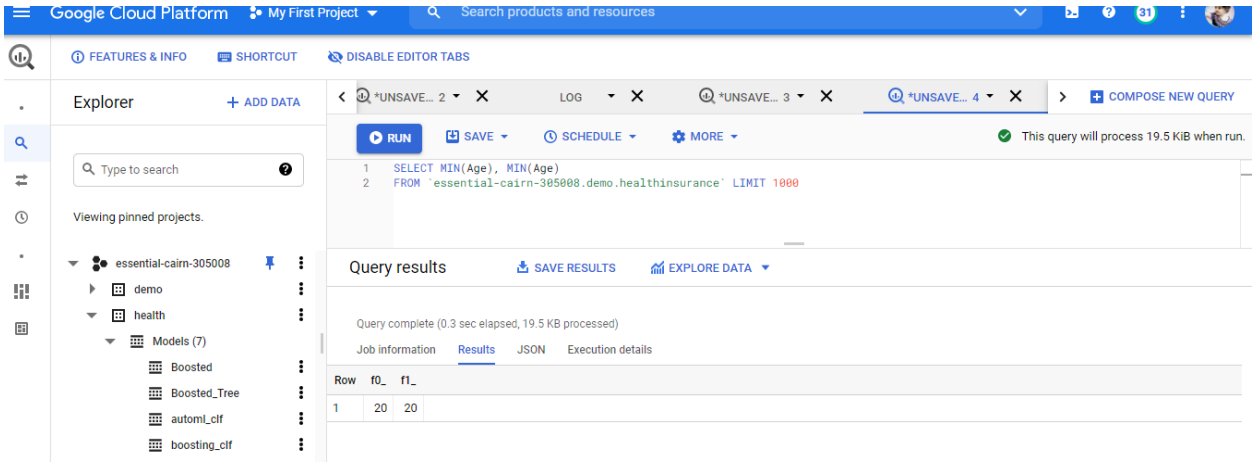
BigQuery is an enterprise data warehouse that solves this problem by enabling super-fast SQL queries using the processing power of Google's infrastructure. Simply move your data into BigQuery and let us handle the hard work. You can control access to both the project and your data based on your business needs, such as giving others the ability to view or query your data.

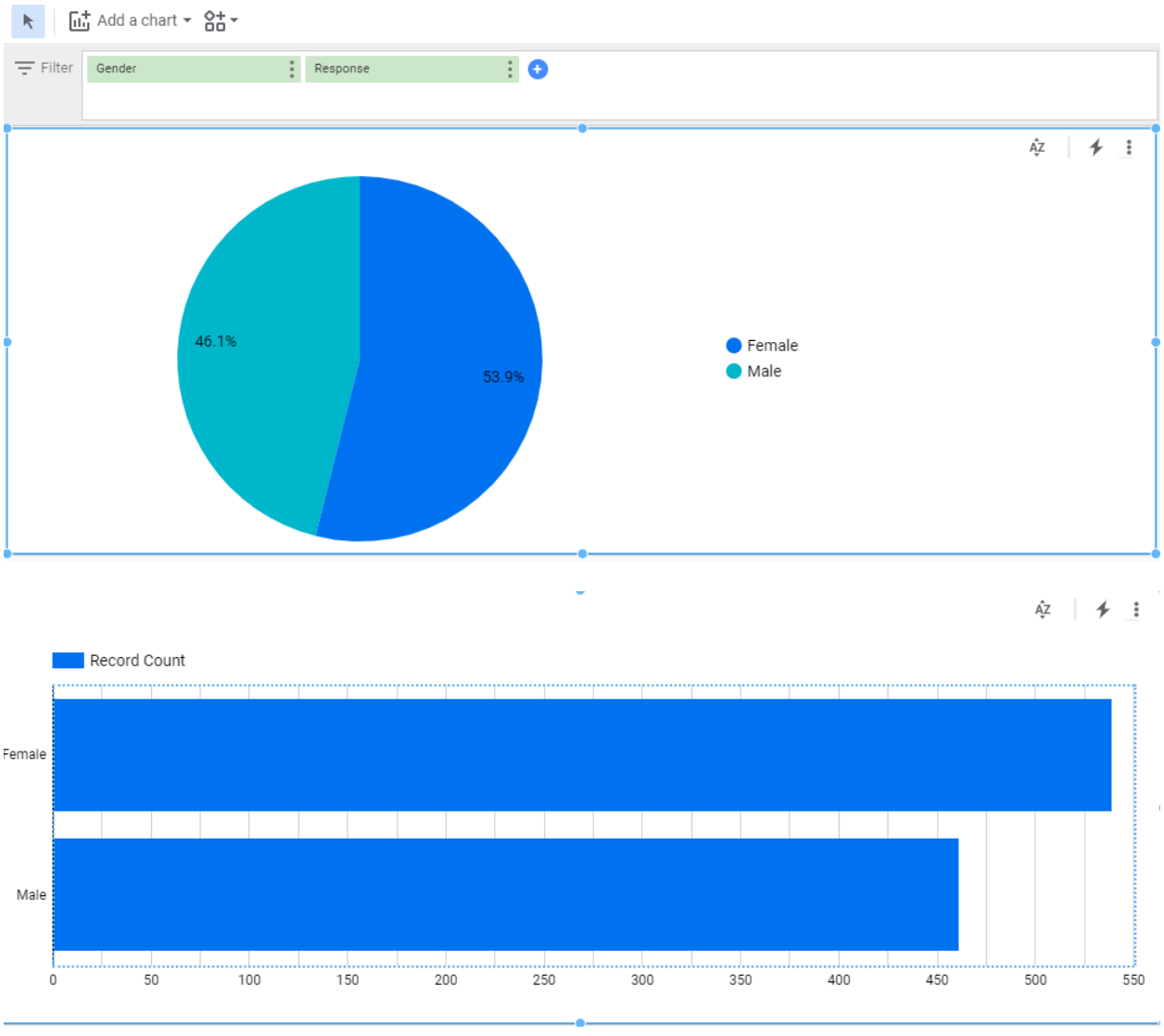



RUN
SAVE
SCHEDULE
MORE
This query will process 219.6 KiB when run.
SELECT * FROM `essential-cairn-305008.demo.healthinsurance` LIMIT 1000
Type to search
Viewing pinned projects.
essential-cairn-305008
demo
health
Models (7)
Boosted
Boosted_Tree
automl_clf
boosting_clf
dnn_clf
Query results
SAVE RESULTS
EXPLORE DATA
Query complete (0.3 sec elapsed, 219.6 KB processed)
Job information
Results
JSON
Execution details
Row id Gender Age Driving_License Region_Code Previously_Insured Vehicle_Age Vehicle_Damage Annual_Premium Policy_Sales_Channel Vintage Res
1 148 Male 20 1 39 1 < 1 Year false 28329 160 78
2 161 Male 20 1 3 1 < 1 Year false 41914 160 206
3 274 Female 20 1 17 0 1-2 Year false 2630 156 181
4 331 Male 20 1 28 1 < 1 Year false 31367 160 255


FEATURES & INFO
SHORTCUT
DISABLE EDITOR TABS
Explorer
ADD DATA
Type to search
Viewing pinned projects.
essential-cairn-305008
demo
health
Models (7)
Boosted
Boosted_Tree
automl_clf
boosting_clf
dnn_clf
log
*UNSAVE... 2
LOG
*UNSAVE... 3
*UNSAVE... 4
COMPOSE NEW QUERY
RUN
SAVE
SCHEDULE
MORE
This query will process 36.5 KiB when run.
SELECT
Gender, Response
FROM
`essential-cairn-305008.demo.healthinsurance`
LIMIT
1000
Query results
SAVE RESULTS
EXPLORE DATA
Query complete (0.3 sec elapsed, 36.5 KB processed)
Job information
Results
JSON
Execution details
Row Gender Response
1 Male 0
2 Male 0
3 Female 0
4 Male 0


Google Cloud Platform
My First Project
Search products and resources
FEATURES & INFO
SHORTCUT
DISABLE EDITOR TABS
Explorer
ADD DATA
Type to search
Viewing pinned projects.
essential-cairn-305008
demo
health
Models (7)
Boosted
Boosted_Tree
automl_clf
*UNSAVE... 2
LOG
*UNSAVE... 3
*UNSAVE... 4
COMPOSE NEW QUERY
RUN
SAVE
SCHEDULE
MORE
This query will process 19.5 KiB when run.
SELECT
sum(Response)
FROM
`essential-cairn-305008.demo.healthinsurance`
LIMIT
1000
Query results
SAVE RESULTS
EXPLORE DATA
Query complete (0.4 sec elapsed, 19.5 KB processed)
Job information
Results
JSON
Execution details
Row f0_
1 312


Google Cloud Platform
My First Project
Search products and resources
FEATURES & INFO
SHORTCUT
DISABLE EDITOR TABS
Explorer
ADD DATA
Type to search
Viewing pinned projects.
essential-cairn-305008
demo
health
Models (7)
Boosted
Boosted_Tree
automl_clf
boosting_clf
*UNSAVE... 2
LOG
*UNSAVE... 3
*UNSAVE... 4
COMPOSE NEW QUERY
RUN
SAVE
SCHEDULE
MORE
This query will process 19.5 KiB when run.
SELECT MIN(Age), MIN(Age)
FROM `essential-cairn-305008.demo.healthinsurance` LIMIT 1000
Query results
SAVE RESULTS
EXPLORE DATA
Query complete (0.3 sec elapsed, 19.5 KB processed)
Job information
Results
JSON
Execution details
Row
f0_
f1_
1
20
20


Add a chart
Filter
Gender
Response
46.1%
53.9%
Female
Male
Record Count
Female
Male
0
50
100
150
200
250
300
350
400
450
500
550

2.Results

Machine learning Algorithms

1. Logistic Regression

```
CREATE OR REPLACE MODEL
  `essential-cairn-305008.health.log` OPTIONS (model_type='logistic_reg',
    input_label_cols=['Response']) AS

SELECT
Age,
Driving_License,
Region_Code,
Previously_Insured,
Annual_Premium,
Policy_Sales_Channel,
Vintage,
Response
FROM `essential-cairn-305008.health.healthinsu`
LIMIT 1000
```

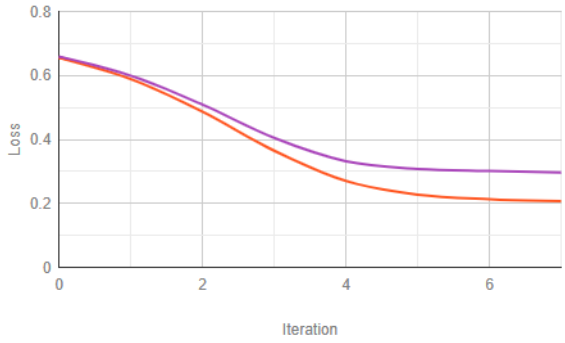
Model type	Data location
LOGISTIC_REGRESSION	US

Training Options

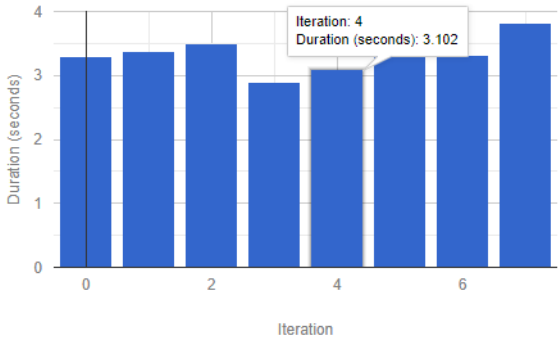
Training options are the optional parameters that were added in the script to create this model.

Max allowed iterations	20
Actual iterations	8
L1 regularization	0.00
L2 regularization	0.00
Early stop	true
Min relative progress	0.01
Learn rate strategy	Line search
Line search initial learn rate	0.10

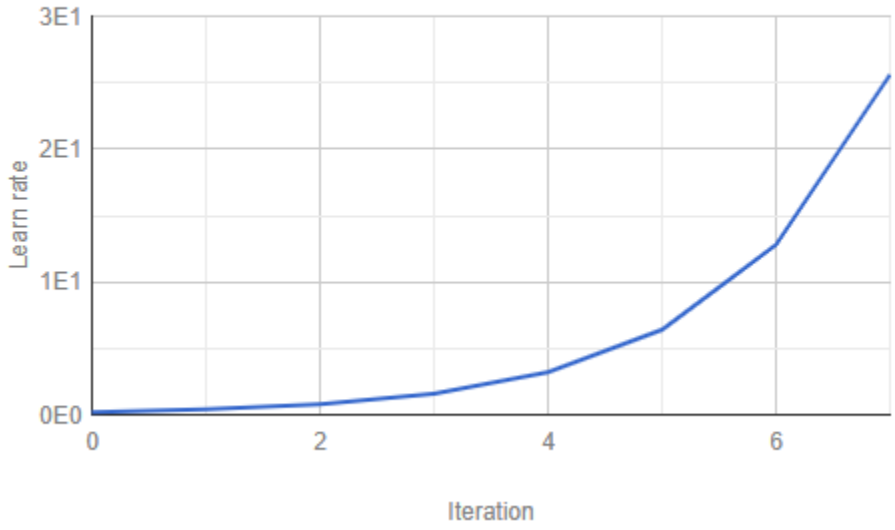
Loss



Duration (seconds)



Learn rate

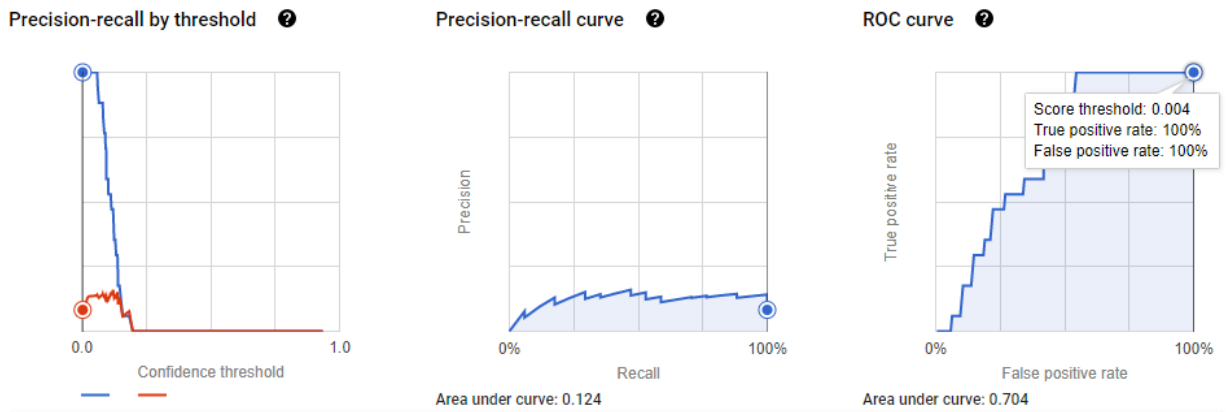


Aggregate Metrics

Log loss	0.2964
ROC AUC	0.7040

Score threshold

Positive class threshold	0.0039
Positive class	1
Negative class	0
Precision	0.0825
Recall	1.0000
Accuracy	0.0825
F1 score	0.1525



Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in gray).

True label	Predicted label 1	Predicted label 0
1	100%	0%
0	100%	0%

```
SELECT * FROM ML.EVALUATE(MODEL `essential-cairn-305008.health.log`,

(
    SELECT
   Age,
   Driving_License,
   Region_Code,
   Previously_Insured,
   Annual_Premium,
   Policy_Sales_Channel,
   Vintage,
   Response
FROM
  `essential-cairn-305008.health.healthinsu`
)
)
```

Query complete (2.1 sec elapsed, 23.3 MB processed)

Job information | Results | JSON | Execution details

Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.21895848437580098	0.6400984799828731	0.6760428118989581	0.3262996491342948	0.7299451134220115	0.7073856143856144

```
SELECT
predicted_Response,
FROM ML.PREDICT(MODEL `essential-cairn-305008.health.log`,
(
    SELECT
   Age,
   Driving_License,
   Region_Code,
   Previously_Insured,
   Annual_Premium,
   Policy_Sales_Channel,
   Vintage
FROM
  `essential-cairn-305008.health.healthinsu`
)
)
```

Query complete (1.8 sec elapsed, 20.4 MB processed)

Job information | Results | JSON | Execution details

Row	predicted_Response
1	0
2	0
3	0
4	0

2.Boosted Tree Classifier

```
CREATE OR REPLACE MODEL
  `essential-cairn-305008.health.Boosted` OPTIONS (model_type='BOOSTED_TREE_CLASSIFIER',
    input_label_cols=['Response']) AS

SELECT
Age,
Driving_License,
Region_Code,
Previously_Insured,
Annual_Premium,
Policy_Sales_Channel,
Vintage,
Response
FROM `essential-cairn-305008.health.healthinsu`
LIMIT 1000
```

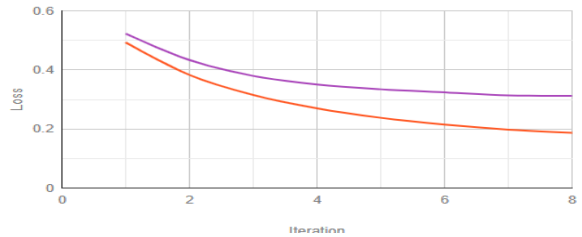
Model type	Data location
BOOSTED_TREE_CLASSIFIER	US

Training Options

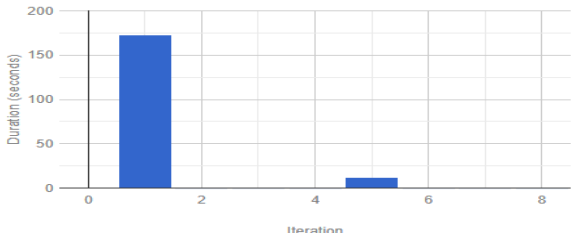
Training options are the optional parameters that were added in the script to create this model.

Max allowed iterations	20
Actual iterations	8
L1 regularization	0.00
L2 regularization	1.00
Learn rate	0.30
Early stop	true
Min relative progress	0.01

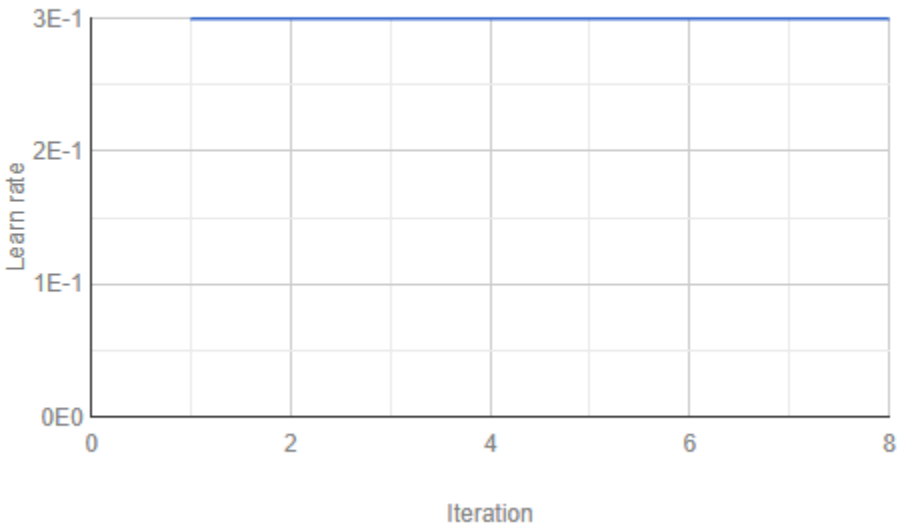
Loss



Duration (seconds)



Learn rate



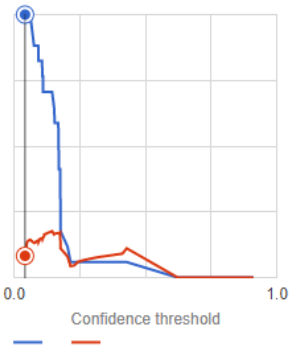
Aggregate Metrics

Log loss	0.3121
ROC AUC	0.7231

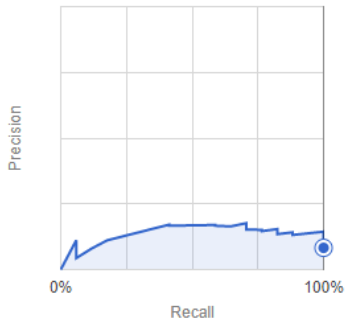
Score threshold

Positive class threshold	0.0420
Positive class	1
Negative class	0
Precision	0.0825
Recall	1.0000
Accuracy	0.0825
F1 score	0.1525

Precision-recall by threshold

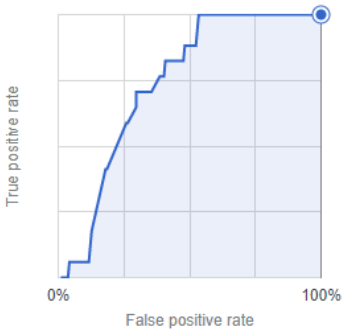


Precision-recall curve



Area under curve: 0.137

ROC curve



Area under curve: 0.723

Confusion matrix

Item

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in gray).

True label	Predicted label: 1	Predicted label: 0
1	100%	0%
0	100%	0%

```
SELECT * FROM ML.EVALUATE(MODEL `essential-cairn-305008.health.boosting_clf`,

(
    SELECT
   Age,
   Driving_License,
   Region_Code,
   Previously_Insured,
   Annual_Premium,
   Policy_Sales_Channel,
   Vintage,
   Response
FROM
  `essential-cairn-305008.health.healthinsu`
)
)
```

Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.24203886980154865	0.40419610361806896	0.7718395524639933	0.30277274403835974	0.5308435842108878	0.6777592407592408

```
SELECT
predicted_Response,
FROM ML.PREDICT(MODEL `essential-cairn-305008.health.boosting_clf`,
(
    SELECT
   Age,
   Driving_License,
   Region_Code,
   Previously_Insured,
   Annual_Premium,
   Policy_Sales_Channel,
   Vintage
FROM
  `essential-cairn-305008.health.healthinsu`
)
)
```

Job information | Results

Row	predicted_Response
1	0
2	0
3	0
4	0
5	0

3.AUTOML CLASSIFIER

```
CREATE OR REPLACE MODEL
  `essential-cairn-305008.health.automl_clf` OPTIONS (model_type='AUTOML_CLASSIFIER',
    input_label_cols=['Response']) AS

SELECT
Age,
Driving_License,
Region_Code,
Previously_Insured,
Annual_Premium,
Policy_Sales_Channel,
Vintage,
Response
FROM `essential-cairn-305008.health.healthinsu`
LIMIT 1000
```

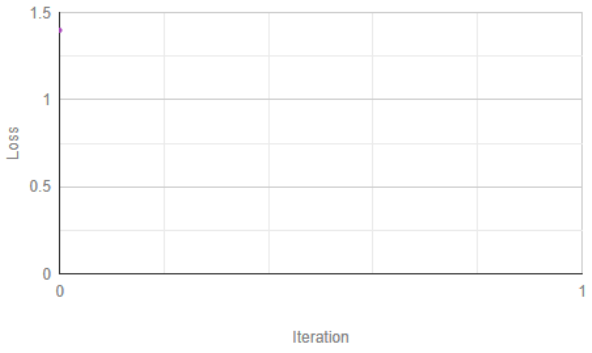
Model type	Data location
AUTOML_CLASSIFIER	US

Training Options

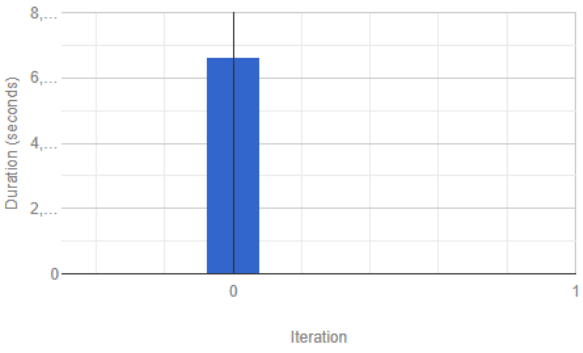
Training options are the optional parameters that were added in the script to create this model.

Actual iterations	1

Loss



Duration (seconds)



Aggregate Metrics

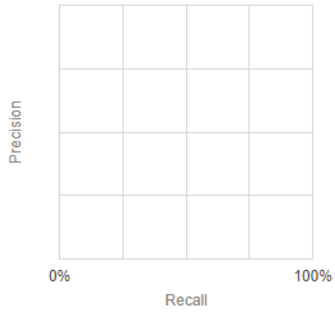
Log loss	1.3984
ROC AUC	0.5000

Score threshold

Positive class threshold	0.0000
Positive class	1
Negative class	0
Precision	
Recall	0.0000
Accuracy	0.9545
F1 score	0.0000

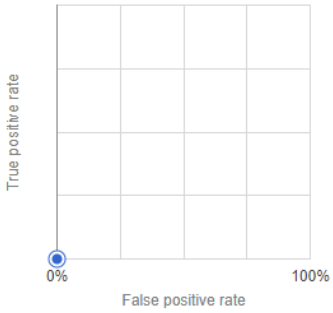
Precision-recall by threshold

Precision-recall curve



Area under curve:

ROC curve



Area under curve: 0.5

Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in gray).

True label	Predicted label: 1	Predicted label: 0
1	0%	100%
0	0%	100%

```sql
SELECT * FROM ML.EVALUATE(MODEL `essential-cairn-305008.health.automl_clf`,

(
    SELECT
   Age,
   Driving_License,
   Region_Code,
   Previously_Insured,
   Annual_Premium,
   Policy_Sales_Channel,
   Vintage,
   Response
FROM
  `essential-cairn-305008.health.healthinsu`
)
)
```

Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.2564102564102564	8.563476771569257E-4	0.877237220847579	0.0017069944095933086	0.383350156086901	0.7335364635364635

```sql
SELECT
predicted_Response,
FROM ML.PREDICT(MODEL `essential-cairn-305008.health.automl_clf`,
(
    SELECT
   Age,
   Driving_License,
   Region_Code,
   Previously_Insured,
   Annual_Premium,
   Policy_Sales_Channel,
   Vintage
FROM
  `essential-cairn-305008.health.healthinsu`
)
)
```


```
[
  {
    "predicted_Response": "0"
  },
  {
    "predicted_Response": "0"
  },
  {
    "predicted_Response": "0"
  },
  {
    "predicted_Response": "0"
  },
  {
```

4.DNN CLASSIFIER

```
CREATE OR REPLACE MODEL
  `essential-cairn-305008.health.dnn_clf` OPTIONS (model_type='DNN_CLASSIFIER',
    input_label_cols=['Response']) AS

SELECT
Age,
Driving_License,
Region_Code,
Previously_Insured,
Annual_Premium,
Policy_Sales_Channel,
Vintage,
Response
FROM `essential-cairn-305008.health.healthinsu`
LIMIT 1000
```

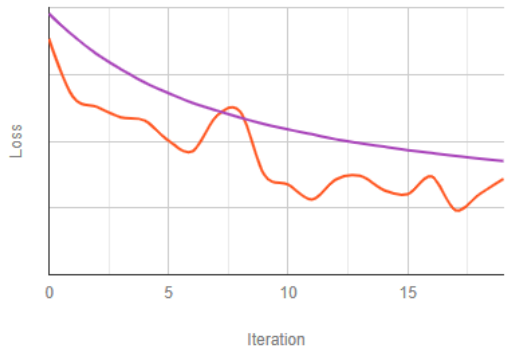
Model type	Data location
DNN_CLASSIFIER	US

Training Options

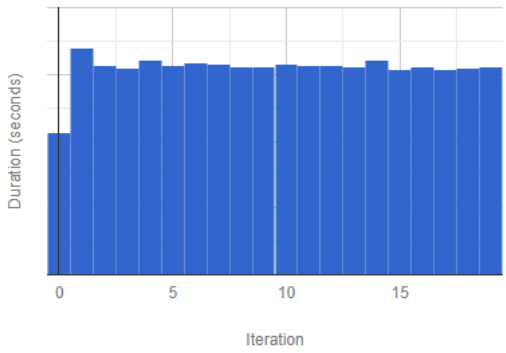
Training options are the optional parameters that were added in the script to create this model.

Max allowed iterations	20
Actual iterations	20
L1 regularization	0.00
L2 regularization	0.00
Learn rate	0.00
Early stop	true
Min relative progress	0.01

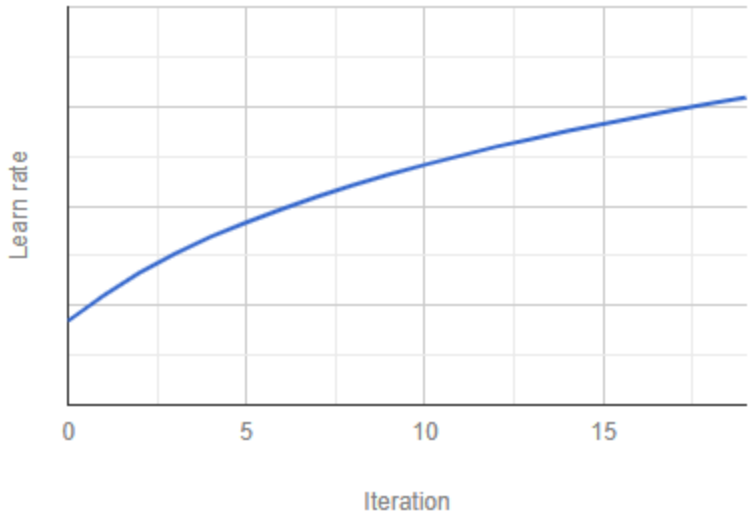
Loss



Duration (seconds)



Learn rate



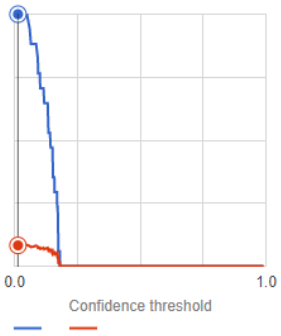
Aggregate Metrics

Log loss	0.4265
ROC AUC	0.3283

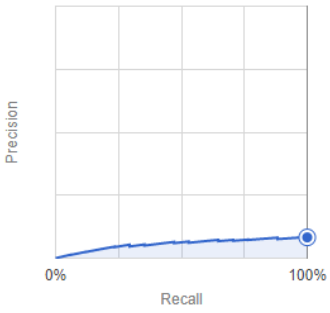
Score threshold

Positive class threshold	0.0145
Positive class	1
Negative class	0
Precision	0.0825
Recall	1.0000
Accuracy	0.0825
F1 score	0.1525

Precision-recall by threshold

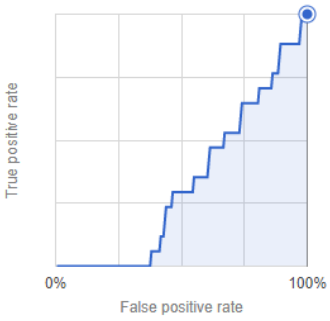


Precision-recall curve



Area under curve: 0.057

ROC curve



Area under curve: 0.328

Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in gr

True label	Predicted label 1	0
1	100%	0%
0	100%	0%

```
SELECT * FROM ML.EVALUATE(MODEL `essential-cairn-305008.health.dnn_clf`,

(
    SELECT
   Age,
   Driving_License,
   Region_Code,
   Previously_Insured,
   Annual_Premium,
   Policy_Sales_Channel,
   Vintage,
   Response
FROM
  `essential-cairn-305008.health.healthinsu`
)
)
```

Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.12277471650633459	0.9964675658317277	0.12694530960958675	0.2186140120567659	436.2974612061208	0.005987012987012987

```
SELECT
predicted_Response,
FROM ML.PREDICT(MODEL `essential-cairn-305008.health.dnn_clf`,
(
    SELECT
   Age,
   Driving_License,
   Region_Code,
   Previously_Insured,
   Annual_Premium,
   Policy_Sales_Channel,
   Vintage
FROM
  `essential-cairn-305008.health.healthinsu`
)
)
```

Job information | Results

Row	predicted_Response
1	0
2	0
3	0
4	0

6. INFERNCE

1. In GCP BIGQUERY Machine Learning Algorithms are trained & Evaluated, Accuracies are:-

   Logistic Regression - 82%
   Boosted Tree Classifier - 82%
   Automl Classifier - 95%
   DNN Classifier - 82%

Therefore, The above 4 models gives good accuracy. But, The best model Is AutoML Classifier got 95% Accuracy.

2. Data Insights and Big Query Pipeline was done on Data Fusion.
3. Machine Learning Training & Testing was done on AutoML Tables In GCP VertexAI.

   Training Accuracy is 87.8%
   Test on online Prediction Deployment is:- Baseline Prediction value :- 0.767
   Response 0 - Confidence score is 0.976
   Response 1 - Confidence Score is 0.024.
4. Pyspark Implementation of machine learning algorithms Got Good Accuracy 85%.so, All models(Logistic, Random Forest Classifier, Gradient Boosted Classifier, LVSM, FM Classifier) are fit for this dataset.

7.REFERENCE

- https://gutentagworld.wordpress.com/2020/12/13/health-insurance-cross-sell-prediction/
- https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction?select=train.csv