Big Data (Classification)

Rofita Siti Musdalifah

05111840000034

1. Business Understanding

- Health Insurance Cross Sell Prediction dataset merupakan contoh dataset problem klasifikasi.
- Diberikan data customer asuransi kesehatan, kita diminta apakah customer tersebut potensial untuk diberi tawaran asuransi

2. Data Understanding

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

2. Data Understanding

	id	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel	Vintage	Response
count	304887.000000	304887	304887.000000	304887.000000	304887.000000	304887.000000	304887	304887	304887.000000	304887.000000	304887.000000	304887.000000
unique	NaN	2	NaN	NaN	NaN	NaN	3	2	NaN	NaN	NaN	NaN
top	NaN	Male	NaN	NaN	NaN	NaN	1-2 Year	Yes	NaN	NaN	NaN	NaN
freq	NaN	164821	NaN	NaN	NaN	NaN	160373	153937	NaN	NaN	NaN	NaN
mean	190594.098072	NaN	38.835582	0.997829	26.413340	0.458373	NaN	NaN	30571.851319	111.966591	154.272609	0.121970
std	110022.144283	NaN	15.509522	0.046547	13.218019	0.498265	NaN	NaN	17254.243887	54.230069	83.642389	0.327251
min	1.000000	NaN	20.000000	0.000000	0.000000	0.000000	NaN	NaN	2630.000000	1.000000	10.000000	0.000000
25%	95342.500000	NaN	25.000000	1.000000	15.000000	0.000000	NaN	NaN	24406.500000	26.000000	82.000000	0.000000
50%	190639.000000	NaN	36.000000	1.000000	28.000000	0.000000	NaN	NaN	31675.000000	131.000000	154.000000	0.000000
75%	285785.500000	NaN	49.000000	1.000000	35.000000	1.000000	NaN	NaN	39414.000000	152.000000	227.000000	0.000000
max	381109.000000	NaN	85.000000	1.000000	52.000000	1.000000	NaN	NaN	540165.000000	163.000000	299.000000	1.000000

3. Data Preparation

- Berdasarkan data yang ada perlu dilakukan normalisasi skala, penanganan missing value (jika ada), penanganan data yang duplikat (jika ada), penanganan imbalanced data
- Kita perlu melakukan **partisi data**:
  - Training: 80%
  - Testing: 20%

  Metode yang digunakan adalah **Random Split** dengan seed 1237

3. Data Preparation

- Dilakukan Load Dataset dan Penghapusan value null

In [2]:
```
# load dataset
data = spark.read.csv('dataset/vehicle_insurance.csv', sep=',', header=True, inferSchema=True, nullValue='NA')
data.show(10)
```

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

```
from pyspark.sql.functions import isnan, when, count, col


print (data.count())
data.dropna().show(truncate=False)
print (data.count())
```

```
381109
```

3. Data Preparation

- Melakukan pengecekan terhadap data duplicate dan menghapusnya

```
# Menghapus data duplicate

print(data_clean.count())
df1=data_clean.groupBy('Age', 'Driving_License', 'Region_Code', 'Previously_Insured', 'Annual_Premium',
                       'Policy_Sales_Channel', 'Vintage', 'Response', 'Gender_Index', 'Vehicle_Age_Index',
                       'Vehicle_Damage_Index'
                            ).count()
df1.show(5)
print ("after delete duplicates:")
print(df1.count())
df2 = df1.drop('count')
df2.show(5)
```

3. Data Preparation

- Karena masih ada beberapa data kategorikal, maka perlu diubah agar dapat di proses

```
# Change kategorikal menjadi numerik
from pyspark.ml.feature import StringIndexer

indexer1 = StringIndexer(inputCol="Gender", outputCol="Gender_Index")
indexer2 = StringIndexer(inputCol="Vehicle_Age", outputCol="Vehicle_Age_Index")
indexer3 = StringIndexer(inputCol="Vehicle_Damage", outputCol="Vehicle_Damage_Index")
indexed1 = indexer1.fit(data_drop_column).transform(data_drop_column)
indexed2 = indexer2.fit(indexed1).transform(indexed1)
indexed3 = indexer3.fit(indexed2).transform(indexed2)
indexed3.show(5)
```

3. Data Preparation

- Agar model baik, maka dilakukan handle imbalance data label

```
data.groupBy('Response').count().show()
```

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

```
# Handle Imbalance data menggunakan Undersampling

def resample(base_features,ratio,class_field,base_class):
    pos = base_features.filter(col(class_field)==base_class)
    neg = base_features.filter(col(class_field)!=base_class)
    total_pos = pos.count()
    total_neg = neg.count()
    fraction=float(total_pos*ratio)/float(total_neg)
    sampled = neg.sample(False,fraction)
    return sampled.union(pos)

df2 = resample(data_clean,1,'Response',1)

df2.groupBy('Response').count().show()
df2.show(5)
```

```
+--------+-----+
|Response|count|
+--------+-----+
|       1|46710|
|       0|46502|
+--------+-----+
```

3. Data Preparation

- Perlu dilakukan normalisasi untuk menstandarkan rentang nilai tiap kolomnya

```
from pyspark.ml.feature import StandardScaler

scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures",
                        withStd=True, withMean=False)

# Compute summary statistics by fitting the StandardScaler
scalerModel = scaler.fit(data_assembled1)

# Normalize each feature to have unit standard deviation.
data_scaled = scalerModel.transform(data_assembled1)
```

```
# membagi data train 80% dan data test 20%
data_train, data_test = data_scaled.randomSplit([0.8, 0.2], seed=1237)
print(data_train.count())
print(data_test.count())
```

```
74483
18729
```

4. Modeling

- kita akan menggunakan 2 modeling.
- Untuk kasus ini, kita mencoba melakukan klasifikasi menggunakan algoritma berikut:
  - Decision Tree
  - Random Forest

4. Modeling (Decision Tree)

Modeling

In [16]:
```
from pyspark.ml.classification import DecisionTreeClassifier

# buat classifier object
tree = DecisionTreeClassifier(labelCol="Response", featuresCol="scaledFeatures")
```

In [17]:
```
# train model
tree_model = tree.fit(data_train)
```

In [18]:
```
# buat prediksi dari data test
prediction = tree_model.transform(data_test)
prediction.select('prediction', 'Response')
# data_train.filter(col("Response")==1).show()
```

Random Forest Tunning Parameter

Tunning Parameter

In [21]:
```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

for x in range(10, 101,10):
    # membuat classifier object
    rf = RandomForestClassifier(labelCol="Response", featuresCol="scaledFeatures", numTrees=x)
    print("numTrees = %g \t" % x)
    # train model
    rf_model = rf.fit(data_train)
    # memprediksi dari data test
    prediction0 = rf_model.transform(data_test)
    evaluator0 = MulticlassClassificationEvaluator(
        labelCol="Response", predictionCol="prediction", metricName="accuracy")
    accuracy0 = evaluator.evaluate(prediction0)
    print("Test Accuracy = %g \n" % accuracy0)
```

```
numTrees = 10
Test Accuracy = 0.793155

numTrees = 20
Test Accuracy = 0.794917

numTrees = 30
Test Accuracy = 0.793102

numTrees = 40
Test Accuracy = 0.794917

numTrees = 50
Test Accuracy = 0.795024

numTrees = 60
Test Accuracy = 0.795451

numTrees = 70
Test Accuracy = 0.794703

numTrees = 80
Test Accuracy = 0.793956

numTrees = 90
Test Accuracy = 0.793796

numTrees = 100
Test Accuracy = 0.795184
```

4. Modeling (Random Forest numTrees 60)

Modeling 2

In [27]:
```
from pyspark.ml.classification import RandomForestClassifier

# membuat classifier object
rf = RandomForestClassifier(labelCol="Response", featuresCol="scaledFeatures", numTrees=60)
```

In [28]:
```
# train model
rf_model = rf.fit(data_train)
```

In [29]:
```
# memprediksi dari data test
prediction2 = rf_model.transform(data_test)
prediction2.select('prediction', 'Response').show(5)
```

```
+----------+--------+
|prediction|Response|
+----------+--------+
|       0.0|       0|
|       0.0|       0|
|       0.0|       0|
```

5. Evaluation (Decision Tree)

- Karena ini adalah permasalahan klasifikasi, maka kita menggunakan **akurasi**

In [19]:
```
# tampilkan confusion matrix
prediction.groupBy('Response', 'prediction').count().show()
```

```
+--------+----------+-----+
|Response|prediction|count|
+--------+----------+-----+
|       1|       0.0|  219|
|       0|       0.0| 5635|
|       1|       1.0| 9177|
|       0|       1.0| 3698|
+--------+----------+-----+

0.7908590955203161
```

In [20]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Hitung test error dari label dan prediction
evaluator = MulticlassClassificationEvaluator(
 labelCol="Response", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(prediction)
print("Test Accuracy = %g " % accuracy)
print("Test Error = %g " % (1.0 - accuracy))
```

```
Test Accuracy = 0.790859
Test Error = 0.209141
```

5. Evaluation (Random Forest numTrees 60)

Evaluation & Deployment 2

In [30]:
```
# menampilkan confusion matrix
prediction2.groupBy('Response', 'prediction').count().show()
# Hitung accuracy
TN = prediction2.filter('prediction = 0 AND Response = prediction').count()
TP = prediction2.filter('prediction = 1 AND Response = prediction').count()
FN = prediction2.filter('prediction = 0 AND Response = 1').count()
FP = prediction2.filter('prediction = 1 AND Response = 0').count()
accuracy2 = (TN + TP) / (TN + TP + FN + FP)
print(accuracy2)
```

```
+--------+----------+-----+
|Response|prediction|count|
+--------+----------+-----+
|       1|       0.0|  581|
|       0|       0.0| 6083|
|       1|       1.0| 8815|
|       0|       1.0| 3250|
+--------+----------+-----+

0.7954509050136153
```

In [31]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# menghitung test error dari label dan prediction
evaluator2 = MulticlassClassificationEvaluator(
 labelCol="Response", predictionCol="prediction", metricName="accuracy")
accuracy2 = evaluator.evaluate(prediction2)
print("Test Accuracy = %g " % accuracy2)
print("Test Error = %g " % (1.0 - accuracy2))
```

```
Test Accuracy = 0.795451
Test Error = 0.204549
```

Kesimpulan

- Didapat bahwa akurasi terbaik untuk dataset ini adalah menggunakan metode klasifikasi random forest dibandingkan dengan descission tree.
- Dan menggunakan tunning parameter 10-100 diperoleh jumlah tree pada random forest untuk mendapat akurasi terbaik pada data ini yaitu 60