Business Problem

This is a dataset from one bank in the United States.

Besides usual services, this bank also provides car insurance services.

The bank organizes regular campaigns to attract new clients.

The bank has potential customers' data, and bank's employees call them for advertising available car insurance options.

We are provided with general information about clients (age, job, etc.) as well as more specific information about the current insurance sell campaign (communication, last contact day) and previous campaigns (attributes like previous attempts, outcome).

You have data about 4000 customers who were contacted during the last campaign and for whom the results of campaign (did the customer buy insurance or not) are known.

Task and Approach:

The task is to predict for 1000 customers who were contacted during the current campaign, whether they will buy car insurance or not.

We will be using **Boosting technique (XG BOOST & GBM)** to predict it

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report, accuracy_score

# Load the datasets
train_file = 'tWjuCygrsACDcRTe.csv'
test_file = 'fuuFZK0sm1iSGYEn.csv'

train_data = pd.read_csv(train_file)
test_data = pd.read_csv(test_file)

# Explore the data
print("Training Data Overview:")
print(train_data.head())
print(train_data.info())
print("\nTest Data Overview:")
print(test_data.head())
print(test_data.info())

# Check for missing values
print("\nMissing Values in Training Data:")
print(train_data.isnull().sum())
print("\nMissing Values in Test Data:")
print(test_data.isnull().sum())

# Preprocessing
# Encode categorical variables using Label Encoding
le = LabelEncoder()
for column in train_data.select_dtypes(include=['object']).columns:
    train_data[column] = le.fit_transform(train_data[column].astype(str))

for column in test_data.select_dtypes(include=['object']).columns:
    test_data[column] = le.fit_transform(test_data[column].astype(str))

# Separate features and target variable
X = train_data.drop(columns=['CarInsurance'])  # 'CarInsurance' is the target column
y = train_data['CarInsurance']

X_test = test_data.drop(columns=['CarInsurance'], errors='ignore')  # Ensure target column is excluded

# Split the training data for validation
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# XGBoost Model
xgb_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
xgb_model.fit(X_train, y_train)

# Gradient Boosting Model
gbm_model = GradientBoostingClassifier(random_state=42)
gbm_model.fit(X_train, y_train)

# Evaluate models
xgb_preds = xgb_model.predict(X_val)
gbm_preds = gbm_model.predict(X_val)

print("XGBoost Classification Report:")
print(classification_report(y_val, xgb_preds))
print("Accuracy:", accuracy_score(y_val, xgb_preds))

print("\nGradient Boosting Classification Report:")
print(classification_report(y_val, gbm_preds))
print("Accuracy:", accuracy_score(y_val, gbm_preds))

# Make predictions on the test set
xgb_test_preds = xgb_model.predict(X_test)
gbm_test_preds = gbm_model.predict(X_test)

# Save predictions
pd.DataFrame({'XGBoost_Predictions': xgb_test_preds, 'GBM_Predictions': gbm_test_preds}).to_csv('predictions.csv', index=F

print("Predictions saved to 'predictions.csv'")
```

```
print("Predictions saved to 'predictions.csv'")
```

```
Training Data Overview:
   Id  Age          Job  Marital Education  Default  Balance  HHInsurance  \
0   1   32   management   single  tertiary        0     1218            1
1   2   32  blue-collar  married   primary        0     1156            1
2   3   29   management   single  tertiary        0      637            1
3   4   25      student   single   primary        0      373            1
4   5   30   management  married  tertiary        0     2694            0

   CarLoan Communication  LastContactDay LastContactMonth  NoOfContacts  \
0        0     telephone              28              jan             2
1        0           NaN              26              may             5
2        0     cellular               3              jun             1
3        0     cellular              11              may             2
4        0     cellular               3              jun             1

   DaysPassed  PrevAttempts  Outcome CallStart   CallEnd  CarInsurance
0          -1             0      NaN  13:45:20  13:46:30             0
1          -1             0      NaN  14:49:03  14:52:08             0
2         119             1  failure  16:30:24  16:36:04             1
3          -1             0      NaN  12:06:43  12:20:22             1
4          -1             0      NaN  14:35:44  14:38:56             0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 19 columns):
```

```
print("Predictions saved to 'predictions.csv'")
```

```
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Id               4000 non-null   int64
 1   Age              4000 non-null   int64
 2   Job              3981 non-null   object
 3   Marital          4000 non-null   object
 4   Education        3831 non-null   object
 5   Default          4000 non-null   int64
 6   Balance          4000 non-null   int64
 7   HHInsurance      4000 non-null   int64
 8   CarLoan          4000 non-null   int64
 9   Communication    3098 non-null   object
 10  LastContactDay   4000 non-null   int64
 11  LastContactMonth 4000 non-null   object
 12  NoOfContacts     4000 non-null   int64
 13  DaysPassed       4000 non-null   int64
 14  PrevAttempts     4000 non-null   int64
 15  Outcome          958 non-null    object
 16  CallStart        4000 non-null   object
 17  CallEnd          4000 non-null   object
 18  CarInsurance     4000 non-null   int64
dtypes: int64(11), object(8)
memory usage: 593.9+ KB
None
```

```
print("Predictions saved to 'predictions.csv'")
```

```
Test Data Overview:
     Id  Age          Job  Marital  Education  Default  Balance  HHInsurance  \
0  4001   25       admin.   single  secondary        0        1            1
1  4002   40   management  married   tertiary        0        0            1
2  4003   44   management   single   tertiary        0    -1313            1
3  4004   27     services   single  secondary        0     6279            1
4  4005   53   technician  married  secondary        0     7984            1

   CarLoan Communication  LastContactDay LastContactMonth  NoOfContacts  \
0        1           NaN              12              may            12
1        1     cellular              24              jul             1
2        1     cellular              15              may            10
3        0     cellular               9              nov             1
4        0     cellular               2              feb             1

   DaysPassed  PrevAttempts Outcome CallStart   CallEnd  CarInsurance
0          -1             0     NaN  17:17:42  17:18:06           NaN
1          -1             0     NaN  09:13:44  09:14:37           NaN
2          -1             0     NaN  15:24:07  15:25:51           NaN
3          -1             0     NaN  09:43:44  09:48:01           NaN
4          -1             0     NaN  16:31:51  16:34:22           NaN
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
```

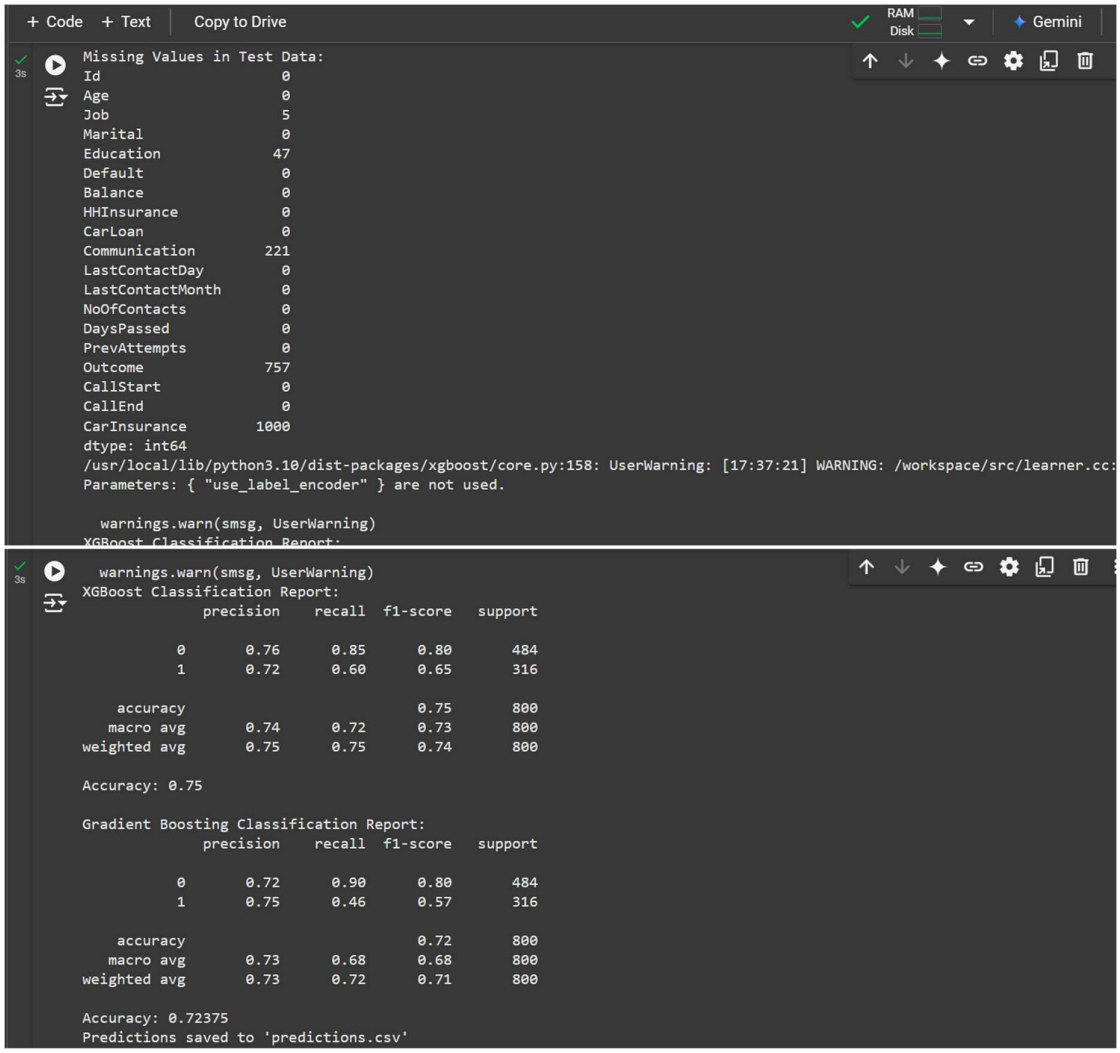
```
print("Predictions saved to 'predictions.csv'")
```

```
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Id               1000 non-null   int64
 1   Age              1000 non-null   int64
 2   Job              995 non-null    object
 3   Marital          1000 non-null   object
 4   Education        953 non-null    object
 5   Default          1000 non-null   int64
 6   Balance          1000 non-null   int64
 7   HHInsurance      1000 non-null   int64
 8   CarLoan          1000 non-null   int64
 9   Communication    779 non-null    object
 10  LastContactDay   1000 non-null   int64
 11  LastContactMonth 1000 non-null   object
 12  NoOfContacts     1000 non-null   int64
 13  DaysPassed       1000 non-null   int64
 14  PrevAttempts     1000 non-null   int64
 15  Outcome          243 non-null    object
 16  CallStart        1000 non-null   object
 17  CallEnd          1000 non-null   object
 18  CarInsurance     0 non-null      float64
dtypes: float64(1), int64(10), object(8)
memory usage: 148.6+ KB
```

```
print("Predictions saved to 'predictions.csv'")
```

```
None

Missing Values in Training Data:
Id                   0
Age                  0
Job                 19
Marital              0
Education          169
Default              0
Balance              0
HHInsurance          0
CarLoan              0
Communication      902
LastContactDay       0
LastContactMonth     0
NoOfContacts         0
DaysPassed           0
PrevAttempts         0
Outcome           3042
CallStart            0
CallEnd              0
CarInsurance         0
dtype: int64
```


+ Code
+ Text
Copy to Drive
RAM
Disk
Gemini
3s
Missing Values in Test Data:
Id 0
Age 0
Job 5
Marital 0
Education 47
Default 0
Balance 0
HHInsurance 0
CarLoan 0
Communication 221
LastContactDay 0
LastContactMonth 0
NoOfContacts 0
DaysPassed 0
PrevAttempts 0
Outcome 757
CallStart 0
CallEnd 0
CarInsurance 1000
dtype: int64
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [17:37:21] WARNING: /workspace/src/learner.cc:
Parameters: { "use_label_encoder" } are not used.
warnings.warn(smsg, UserWarning)
XGBoost Classification Report:

3s
warnings.warn(smsg, UserWarning)
XGBoost Classification Report:
precision recall f1-score support
0 0.76 0.85 0.80 484
1 0.72 0.60 0.65 316
accuracy 0.75 800
macro avg 0.74 0.72 0.73 800
weighted avg 0.75 0.75 0.74 800
Accuracy: 0.75
Gradient Boosting Classification Report:
precision recall f1-score support
0 0.72 0.90 0.80 484
1 0.75 0.46 0.57 316
accuracy 0.72 800
macro avg 0.73 0.68 0.68 800
weighted avg 0.73 0.72 0.71 800
Accuracy: 0.72375
Predictions saved to 'predictions.csv'