Bank Deposit Subscription Prediction Report

Team Smart Banker (Data Science Specialization)

April 26, 2023

Agenda

Meet the Team

Group Name: Team Smart Banker

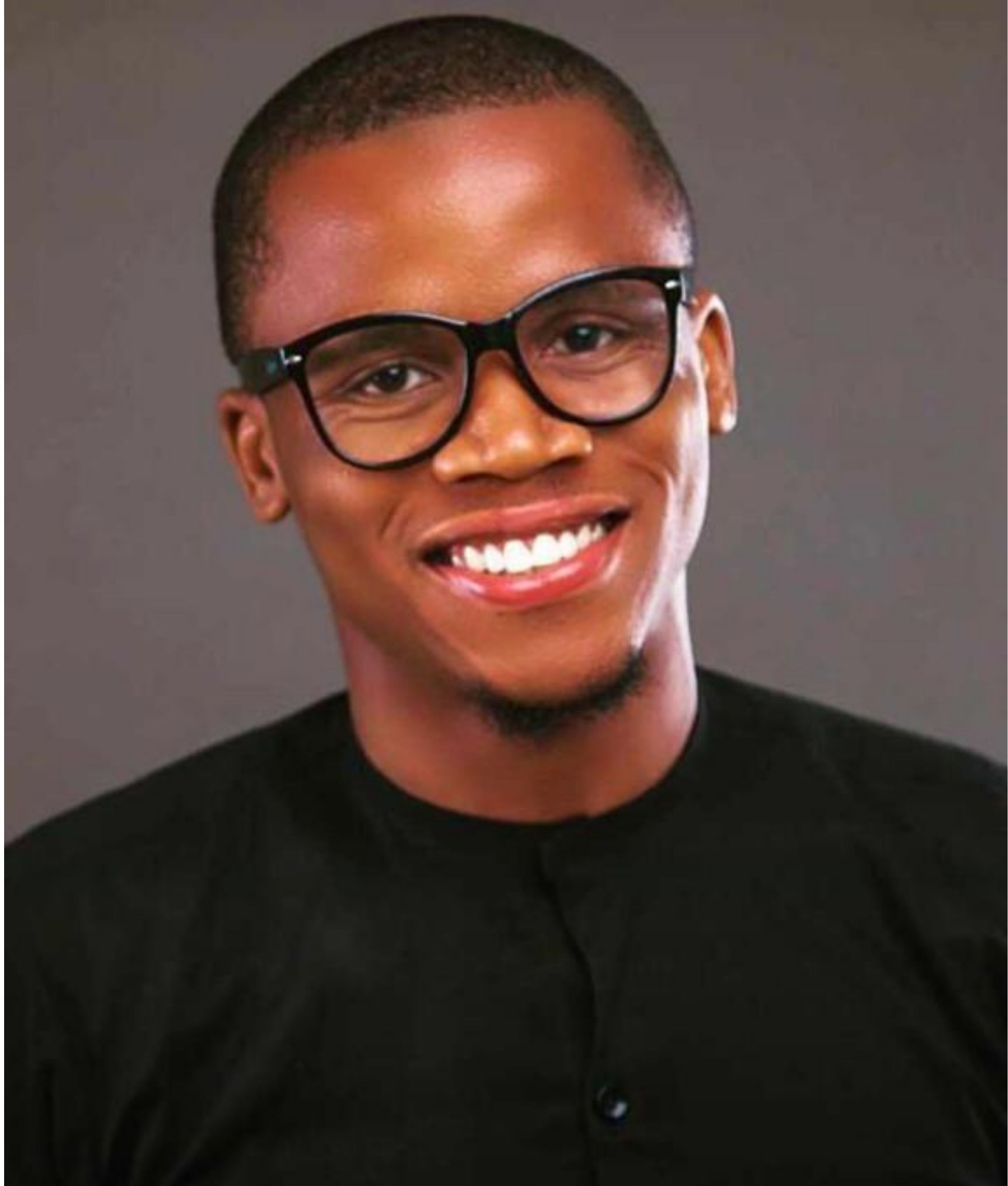
Member Name: Chukwujekwu Joseph Ezema

Email: chukwujekwujeks@gmail.com

School: Teesside University Middlesbrough

Country: United Kingdom

Specialization: Data Science



Bank Marketing Campaign

A Data Science Project

Problem Description

Objective: Predict if a client will subscribe to a term deposit offered by a bank.

Dataset: Information related to direct marketing campaigns conducted through phone calls by a Portuguese banking institution.

Goal: Build a classification model that predicts whether the client will subscribe to the term deposit or not, based on various input variables such as age, job, marital status, education, etc

Business Understanding

Importance: Success of direct marketing campaigns depends on accuracy of targeting potential customers who are more likely to purchase the offered product..

Interest: This is to help a bank identify customers who are more likely to subscribe to a term deposit to optimize their marketing strategy and increase their chances of success while minimizing their costs..

Data Types and Problems in the Data

Data Type: The data was majorly dominated by Categorical Data types – which led to most of the them being converted to viable numeric types for ease of modelling for the Classification problem..

Problem:

1. White spaces in the columns were replaced by “;” thereby making the original data to contain one column instead of 21 columns
2. Some columns were not in their appropriate data types e.g. duration, campaign etc
3. Data contained 0 missing values but 12 duplicated records
4. There were outliers across the numerical columns
5. There is imbalance in the target class, y

2]:

```python
""" Step 1: Get dataset """
# 1.2 Load dataset

# Reading the Bank Additional File - dataset for the features
bank_df = pd.read_csv('bank-additional-full.csv')

bank_df_copy = bank_df.copy() #making a copy
bank_df_copy
```

2]:

	age;"job";"marital";"education";"default";"housing";"loan";"contact";"month";"day_of_week";"duration";"campaign";"pdays";"previous";"poutcome";"emp
0	
1	
2	
3	
4	
...	
41183	
41184	
41185	
41186	
41187	

Original Data Load

41188 rows × 1 columns

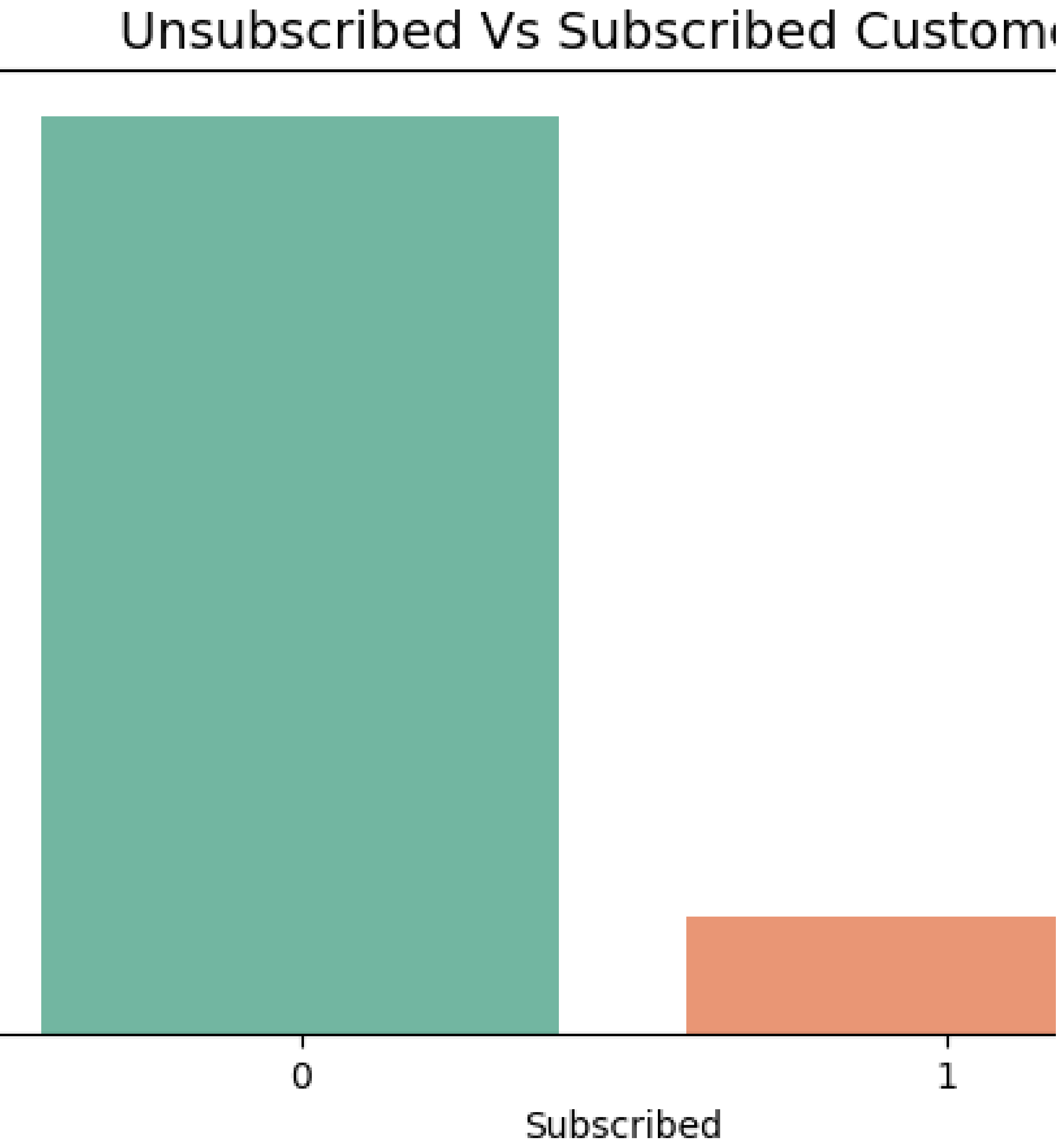
Inappropriate Datatypes

In [4]:
```
# Explore the features of dataset

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             41188 non-null  object
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  object
 11  campaign        41188 non-null  object
 12  pdays           41188 non-null  object
 13  previous        41188 non-null  object
 14  poutcome        41188 non-null  object
 15  emp_var_rate    41188 non-null  object
 16  cons_price_idx  41188 non-null  object
 17  cons_conf_idx   41188 non-null  object
 18  euribor3m       41188 non-null  object
 19  nr_employed     41188 non-null  object
 20  y               41188 non-null  object
dtypes: object(21)
memory usage: 6.6+ MB
```

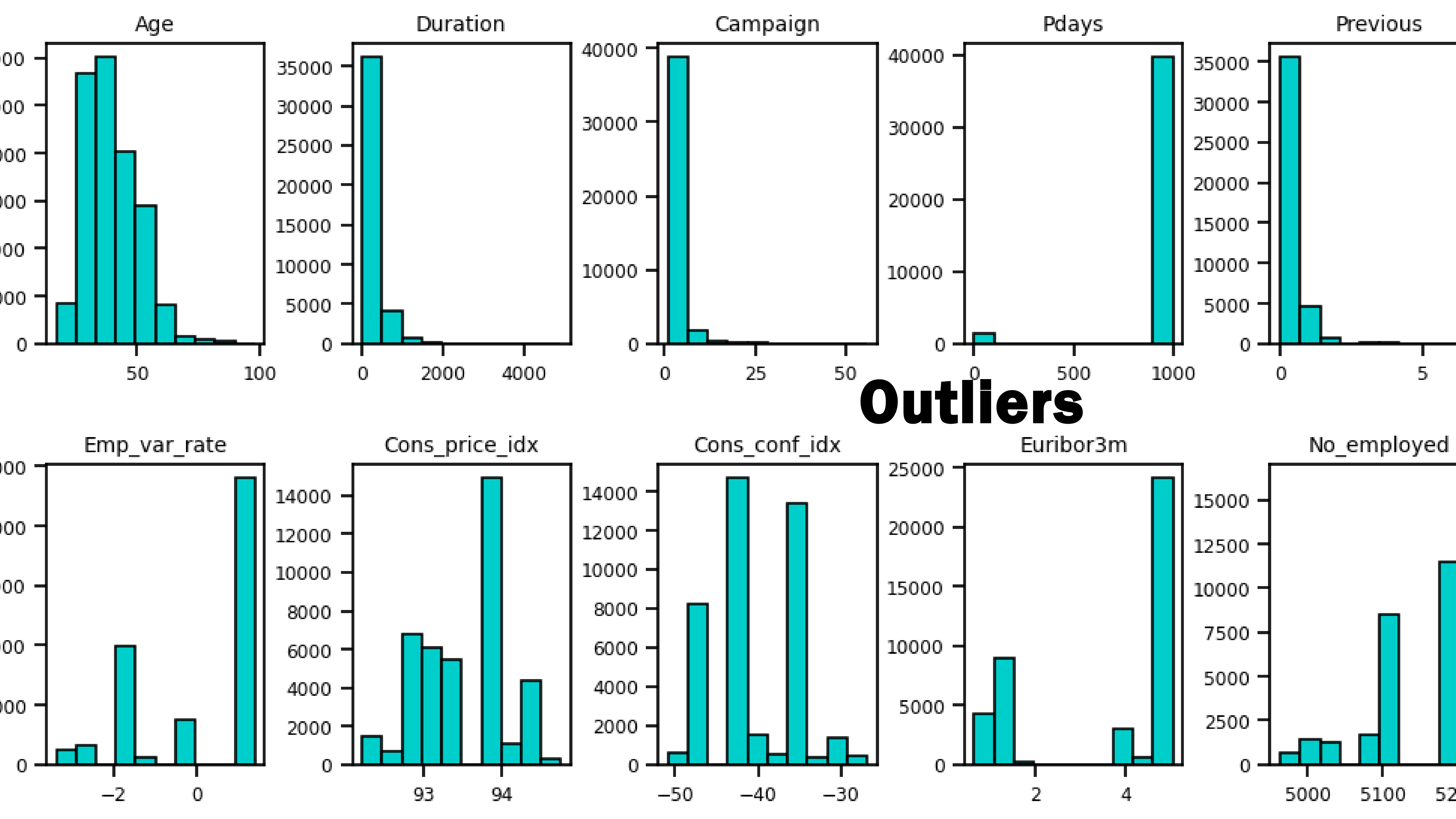
Missing & Duplicated Records

```
In [5]: # Check features with missing value

        data.isnull().sum()

Out[5]: age               0
        job               0
        marital           0
        education         0
        default           0
        housing           0
        loan              0
        contact           0
        month             0
        day_of_week       0
        duration          0
        campaign          0
        pdays             0
        previous          0
        poutcome          0
        emp_var_rate      0
        cons_price_idx    0
        cons_conf_idx     0
        euribor3m         0
        nr_employed       0
        y                 0
        dtype: int64

In [6]: # Check duplicated records:

        print("Number of duplicated records before dropping:

        Number of duplicated records before dropping: 12
```



Imbalanced Target Class

Outliers


Age
50
100
Duration
35000
30000
25000
20000
15000
10000
5000
0
2000
4000
Campaign
40000
30000
20000
10000
0
25
50
Pdays
40000
30000
20000
10000
0
500
1000
Previous
35000
30000
25000
20000
15000
10000
5000
0
5
Emp_var_rate
−2
0
Cons_price_idx
14000
12000
10000
8000
6000
4000
2000
0
93
94
Cons_conf_idx
14000
12000
10000
8000
6000
4000
2000
0
−50
−40
−30
Euribor3m
25000
20000
15000
10000
5000
0
2
4
No_employed
15000
12500
10000
7500
5000
2500
0
5000
5100

Approach to Eliminate Problems in the Data

1. Split the columns using the delimiter, semicolon (;) and removed the double the quotes (“”)

2. Use methods from Pandas library to convert to appropriate datatypes e.g pd.to_numeric()

3. Dropped duplicated rows

4. Use StandardScalar() from sklearn package to limit the effects of the outliers

5. Use smote() oversampling technique from imblearn package to handle the imbalance so as not to lose possible data for the training and testing

Cleaned Dataset

In [3]:

```
""" Step 2: Clean dataset """
# Split the columns into separate columns
data = bank_df.iloc[:,0].str.split(";", expand=True)

# Remove the double quotes from the values
data = data.applymap(lambda x: x.strip('"'))

# Rename the columns with more descriptive names
new_cols = ["age", "job", "marital", "education", "default", "housing", "loan", "contact", "month", "day_of_week", "duration",
data = data.rename(columns=dict(zip(data.columns, new_cols)))
data
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexisten
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	nonexisten
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	nonexisten
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexisten
4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1	999	0	nonexisten
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	334	1	999	0	nonexisten
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	383	1	999	0	nonexisten
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	189	2	999	0	nonexisten
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	442	1	999	0	nonexisten
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	239	3	999	1	failur

41188 rows × 21 columns

```python
# Confirming non duplicated
print("Number of duplicated records after dropping: {}".format(df1.duplicated().sum()))
```

```
Number of duplicated records after dropping: 0
```

Dropped Duplicates and converted to numeric

In [8]:

```python
""" Step 3: Transform data """
# Convert categorical data to numeric values
def to_integers(self):

    # Converting to binary values for best results
    self.replace({'y' : {'yes' : 1,  'no' : 0}}, inplace=True)

    # Converting to numeric values
    self['age'] = pd.to_numeric(self['age'], errors='coerce')
    self['duration'] = pd.to_numeric(self['duration'], errors='coerce')
    self['campaign'] = pd.to_numeric(self['campaign'], errors='coerce')
    self['pdays'] = pd.to_numeric(self['pdays'], errors='coerce')
    self['previous'] = pd.to_numeric(self['previous'], errors='coerce')
    self['emp_var_rate'] = pd.to_numeric(self['emp_var_rate'], errors='coerce')
    self['cons_price_idx'] = pd.to_numeric(self['cons_price_idx'], errors='coerce')
    self['cons_conf_idx'] = pd.to_numeric(self['cons_conf_idx'], errors='coerce')
    self['euribor3m'] = pd.to_numeric(self['euribor3m'], errors='coerce')
    self['nr_employed'] = pd.to_numeric(self['nr_employed'], errors='coerce')

    return self

to_integers(df1)
```

Out[8]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexistent
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	nonexistent
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	nonexistent
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexistent

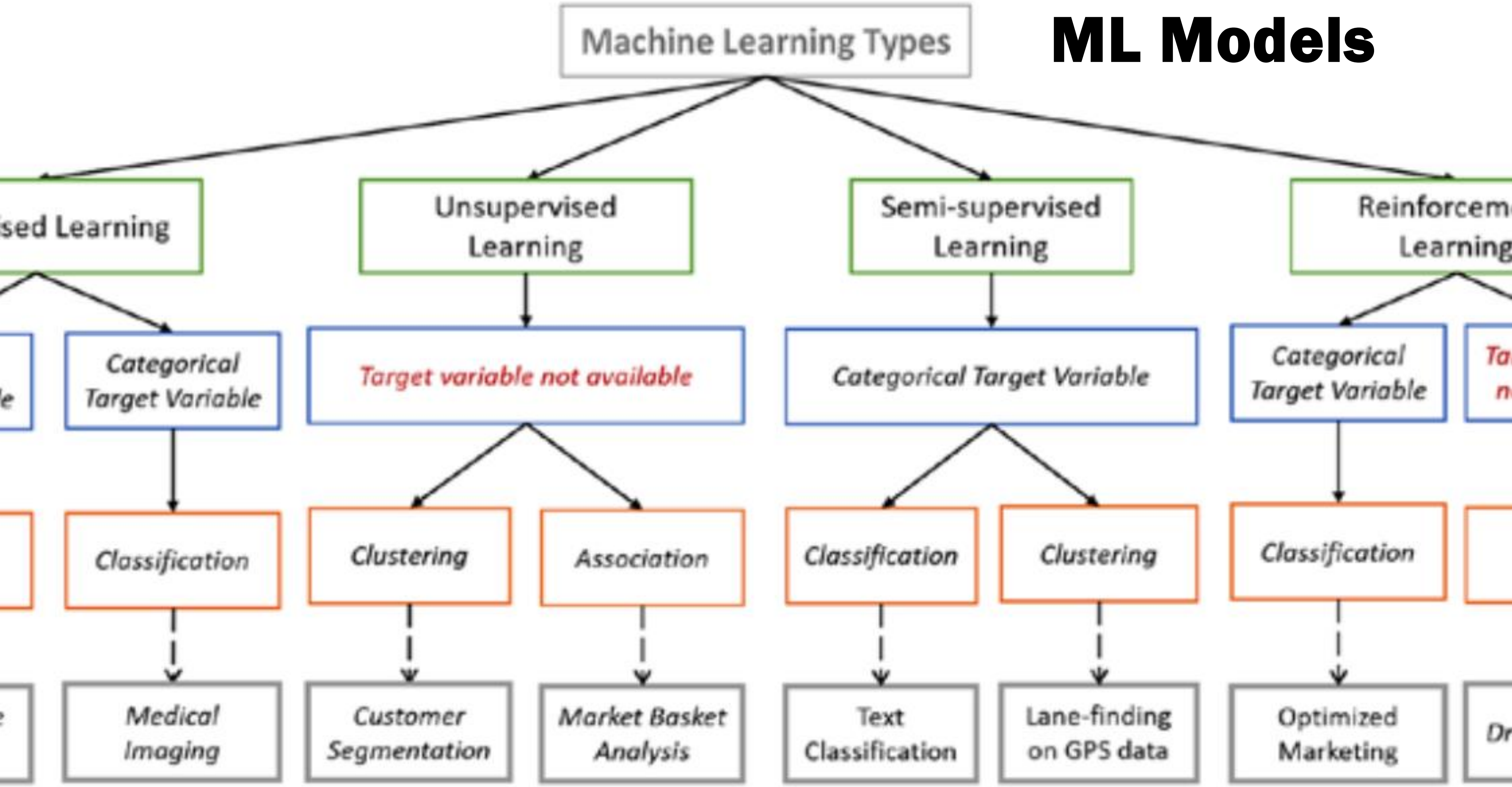
After Transformation

- Cleaned the jam-packed columns into distinct columns
- Removed duplicates
- Converted wrong object datatypes to appropriate numerical types
- Created a function to label encode and further applied one-hot encoding using pd.get_dummies to put the categories into 1’s and 0’s in order to avoid biases
- Note: The increased number of columns will be tackled with dimensionality reduction strategy like PCA

In [11]:
```
# confirm the nature of the data after feature engineering

df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41176 entries, 0 to 41187
Data columns (total 44 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Age                   41176 non-null  int64
 1   Duration              41176 non-null  int64
 2   Campaign              41176 non-null  int64
 3   Pdays                 41176 non-null  int64
 4   Previous              41176 non-null  int64
 5   Emp_var_rate          41176 non-null  float64
 6   Cons_price_idx        41176 non-null  float64
 7   Cons_conf_idx         41176 non-null  float64
 8   Euribor3m             41176 non-null  float64
 9   No_employed           41176 non-null  float64
 10  Subscribed            41176 non-null  int64
 11  Job_Employed          41176 non-null  uint8
 12  Job_Self_employed     41176 non-null  uint8
 13  Job_Unemployed        41176 non-null  uint8
 14  Job_Unknown           41176 non-null  uint8
 15  Marital_Married       41176 non-null  uint8
 16  Marital_Not_married   41176 non-null  uint8
 17  Marital_Unknown       41176 non-null  uint8
 18  Education_Basic       41176 non-null  uint8
 19  Education_Secondary   41176 non-null  uint8
 20  Education_Tertiary    41176 non-null  uint8
 21  Education_Uneducated  41176 non-null  uint8
 22  Education_Unknown     41176 non-null  uint8
 23  Default_no            41176 non-null  uint8
 24  Default_unknown       41176 non-null  uint8
 25  Default_yes           41176 non-null  uint8
 26  Housing_no            41176 non-null  uint8
 27  Housing_unknown       41176 non-null  uint8
 28  Housing_yes           41176 non-null  uint8
 29  Loan_no               41176 non-null  uint8
 30  Loan_unknown          41176 non-null  uint8
```

ML Models


Machine Learning Types
Unsupervised Learning
Semi-supervised Learning
Categorical Target Variable
Target variable not available
Categorical Target Variable
Categorical Target Variable
Classification
Clustering
Association
Classification
Clustering
Classification
Medical Imaging
Customer Segmentation
Market Basket Analysis
Text Classification
Lane-finding on GPS data
Optimized Marketing

ML Model Recommendations

1. Logistics Regression – Linear
2. K-Nearest Neighbour Classifier – Non-parametric
3. Random Forest Classifier - Bagging Ensemble

4. Gradient Boost Classifier - Boosting Ensemble
5. XGBoost Classifier - Boosting Ensemble
6. Neural Networks (Keras) for Deep Learning - Non-linear

Why Choose them?

```python
""" Step 6: Model the Algorithms for Prediction"""

# Design ML model - Set up

# Classifiers
classifiers = {
            "LogisiticRegression": LogisticRegression(random_state=random_state),
            "KNearest": KNeighborsClassifier(),
            "GradientBoost": GradientBoostingClassifier(random_state=random_state),
            "Random Forest Classifier": RandomForestClassifier(random_state=random_state),
            "XGBClassifier": XGBClassifier(random_state=random_state)
}
```

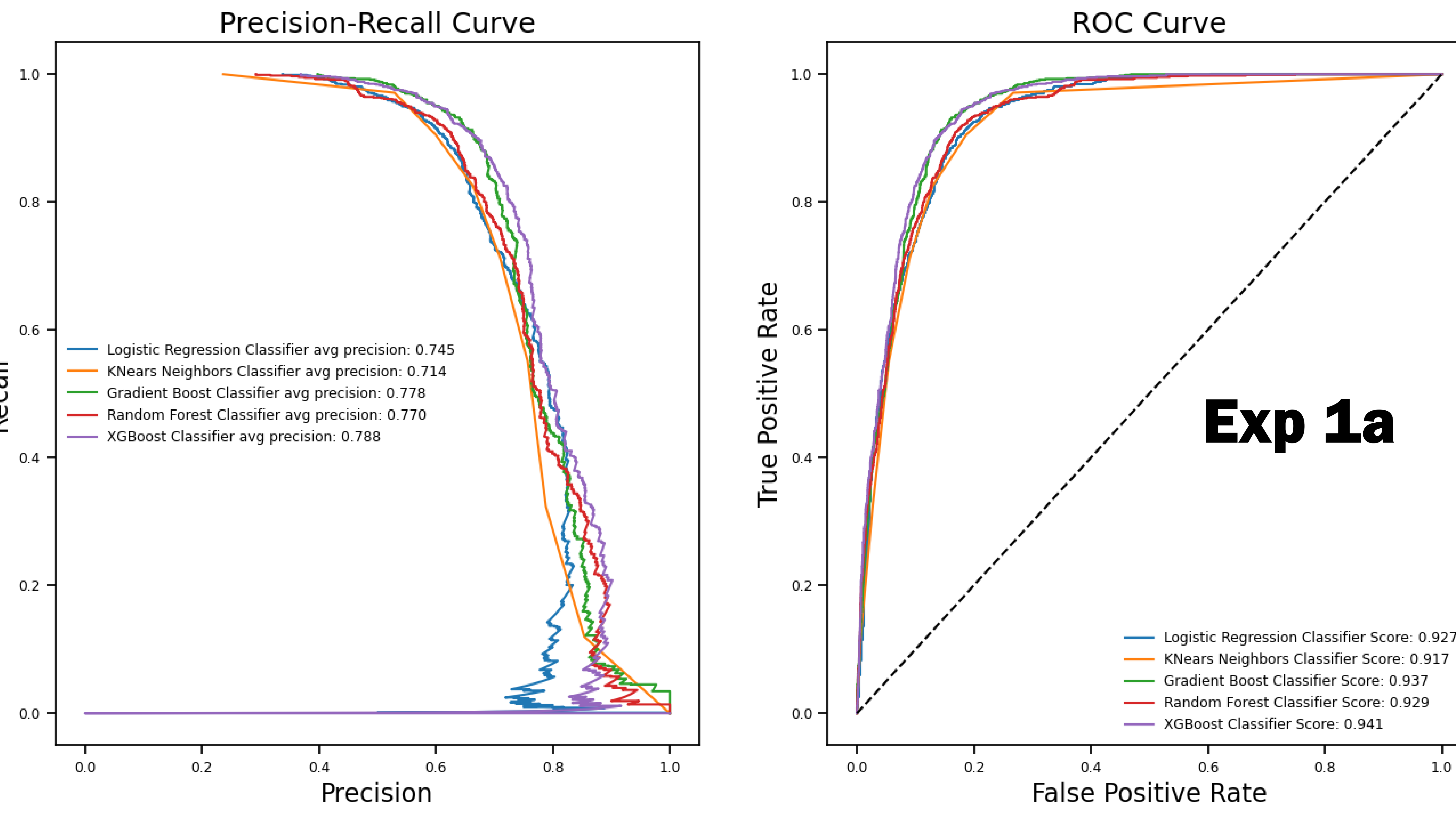
- Logistics Regression: Linear model. Logistic regression is commonly used in classification problems because it provides a probabilistic interpretation of the outputs, allowing for easy thresholding for binary classification tasks. It can also handle imbalanced classes by adjusting the threshold for classification.
- Gradient Boost Classifier: Ensemble model. Gradient boosting is a technique that combines multiple weak learners to create a strong learner. It can handle imbalanced data because it can assign more weight to the minority class during the boosting process
- Neural Networks (Keras) for Deep Learning: Non-linear model. Neural networks are powerful models that can learn complex patterns in data. They can handle imbalanced data through techniques such as class weighting, oversampling of the minority class, and the use of specialized loss functions

```python
# Setup for Deep Learning Model
def ANN_model(X_train, y_train, X_test, y_test, epochs=5):

    #design model
    model = keras.Sequential([
        keras.layers.Dense(18, input_shape=(15,), activation='relu'),
        keras.layers.Dense(1, activation='sigmoid')
    ])

    #compile model
    model.compile(optimizer = 'adam',
                  loss = 'binary_crossentropy',
                  metrics =['accuracy'])

    #fit model
    model.fit(X_train, y_train, epochs=epochs)

    # evaluate model
    print(f'model evaluation :', model.evaluate(X_test, y_test))

    # build prediction series
    yp = model.predict(X_test)
    y_pred =[]
    for element in yp:
        if element > 0.5:
            y_pred.append(1)
        else:
            y_pred.append(0)

    mse = np.mean(np.power(X_test - yp, 2), axis=1)
    error = pd.DataFrame({'reconstruction_error': mse,
                          'true_class': y_test})

    print(error.describe())

    #result
    print(classification_report(y_test, y_pred, labels=[1,0]))
```

Predictions, Results and Evaluation

Exp 1a: Classifiers on Best Features of Normal Sample Dataset (without oversampling)

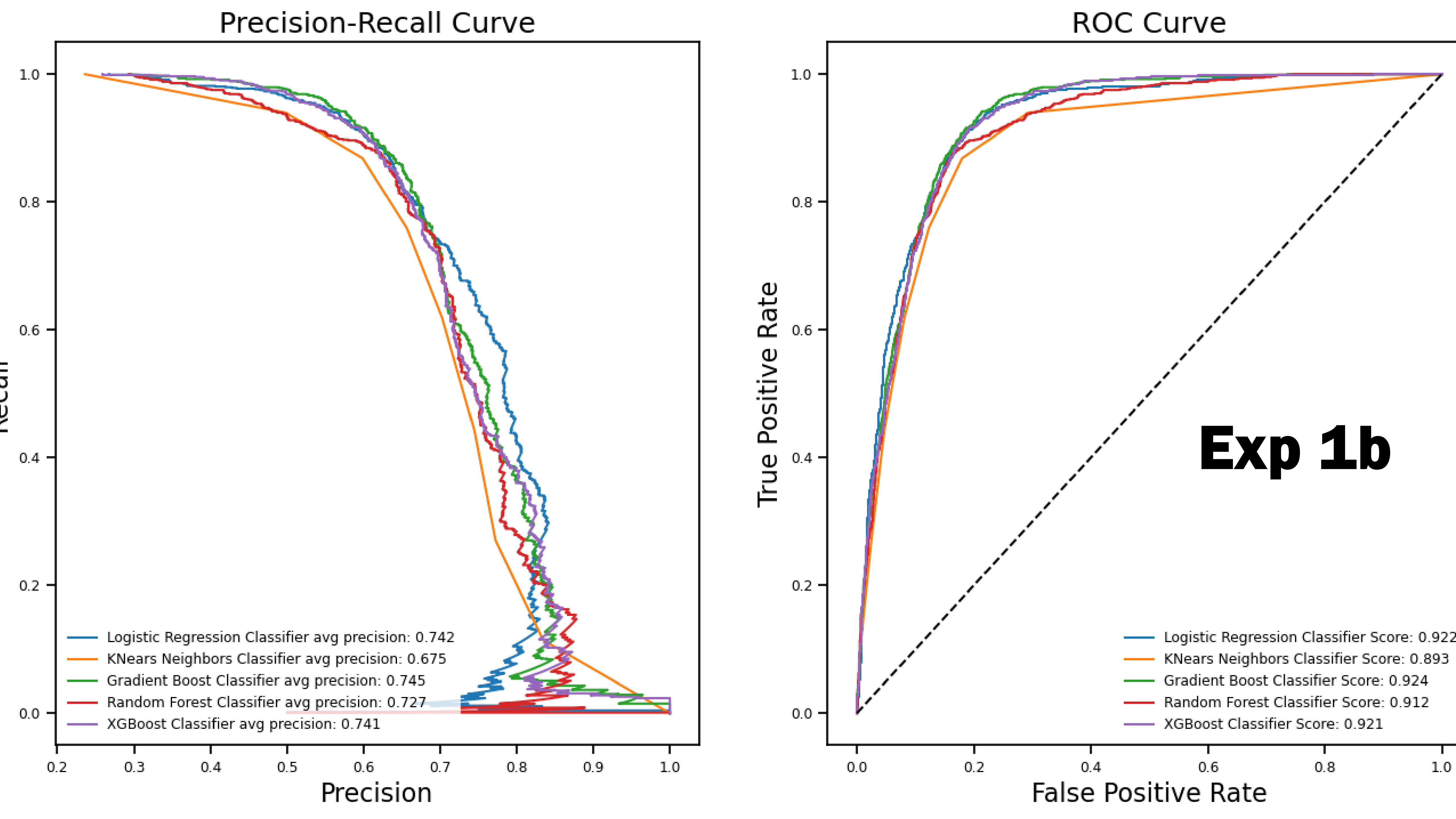
Classifier	Accur. Score	Macro F1-Score	Weighted F1-Score	P-R Curve	AUC-ROC
Log. Regression	0.87	0.80	0.86	0.75	0.93
KNN	0.87	0.82	0.87	0.71	0.92
Gradient Boost	0.89	0.85	0.89	0.78	0.94
Random Forest	0.86	0.79	0.85	0.77	0.93
XG Boost	0.89	0.85	0.89	0.79	0.94


Precision-Recall Curve
Recall
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Logistic Regression Classifier avg precision: 0.745
KNears Neighbors Classifier avg precision: 0.714
Gradient Boost Classifier avg precision: 0.778
Random Forest Classifier avg precision: 0.770
XGBoost Classifier avg precision: 0.788
ROC Curve
True Positive Rate
False Positive Rate
Logistic Regression Classifier Score: 0.927
KNears Neighbors Classifier Score: 0.917
Gradient Boost Classifier Score: 0.937
Random Forest Classifier Score: 0.929
XGBoost Classifier Score: 0.941
Exp 1a

Predictions, Results and Evaluation

Exp 1b: PCA applied on Normal Sample Dataset (without oversampling)

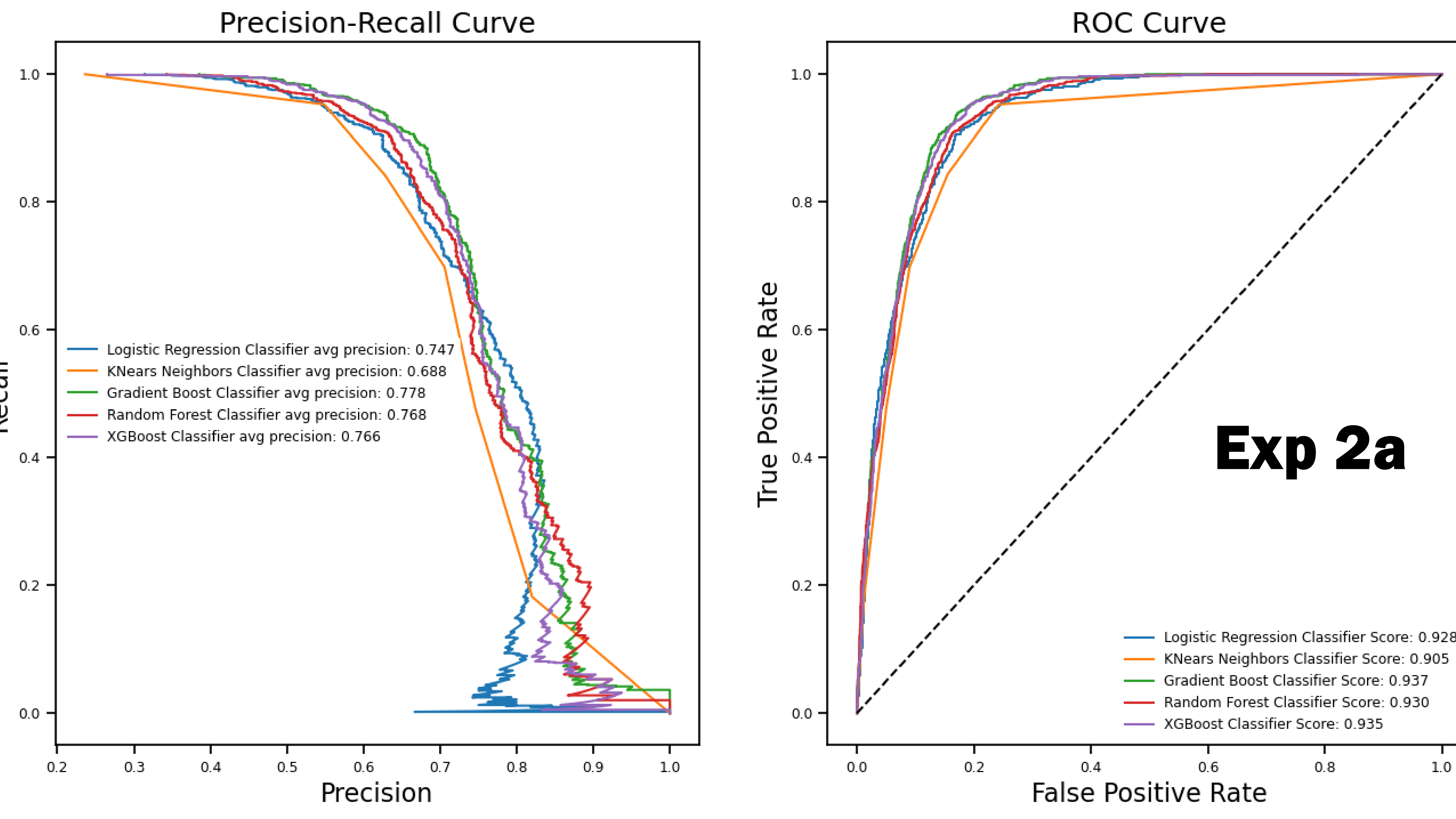
Classifier	Accur. Score	Macro F1-Score	Weighted F1-Score	P-R Curve	AUC-ROC
Log. Regression	0.87	0.81	0.86	0.74	0.92
KNN	0.86	0.80	0.85	0.68	0.89
Gradient Boost	0.87	0.82	0.87	0.75	0.92
Random Forest	0.85	0.76	0.84	0.73	0.91
XG Boost	0.88	0.83	0.88	0.74	0.92


Precision-Recall Curve
Precision
Logistic Regression Classifier avg precision: 0.742
KNears Neighbors Classifier avg precision: 0.675
Gradient Boost Classifier avg precision: 0.745
Random Forest Classifier avg precision: 0.727
XGBoost Classifier avg precision: 0.741
ROC Curve
True Positive Rate
False Positive Rate
Exp 1b
Logistic Regression Classifier Score: 0.922
KNears Neighbors Classifier Score: 0.893
Gradient Boost Classifier Score: 0.924
Random Forest Classifier Score: 0.912
XGBoost Classifier Score: 0.921

Predictions, Results and Evaluation

Exp 2a: Smote Technique on Best Features - Oversampling

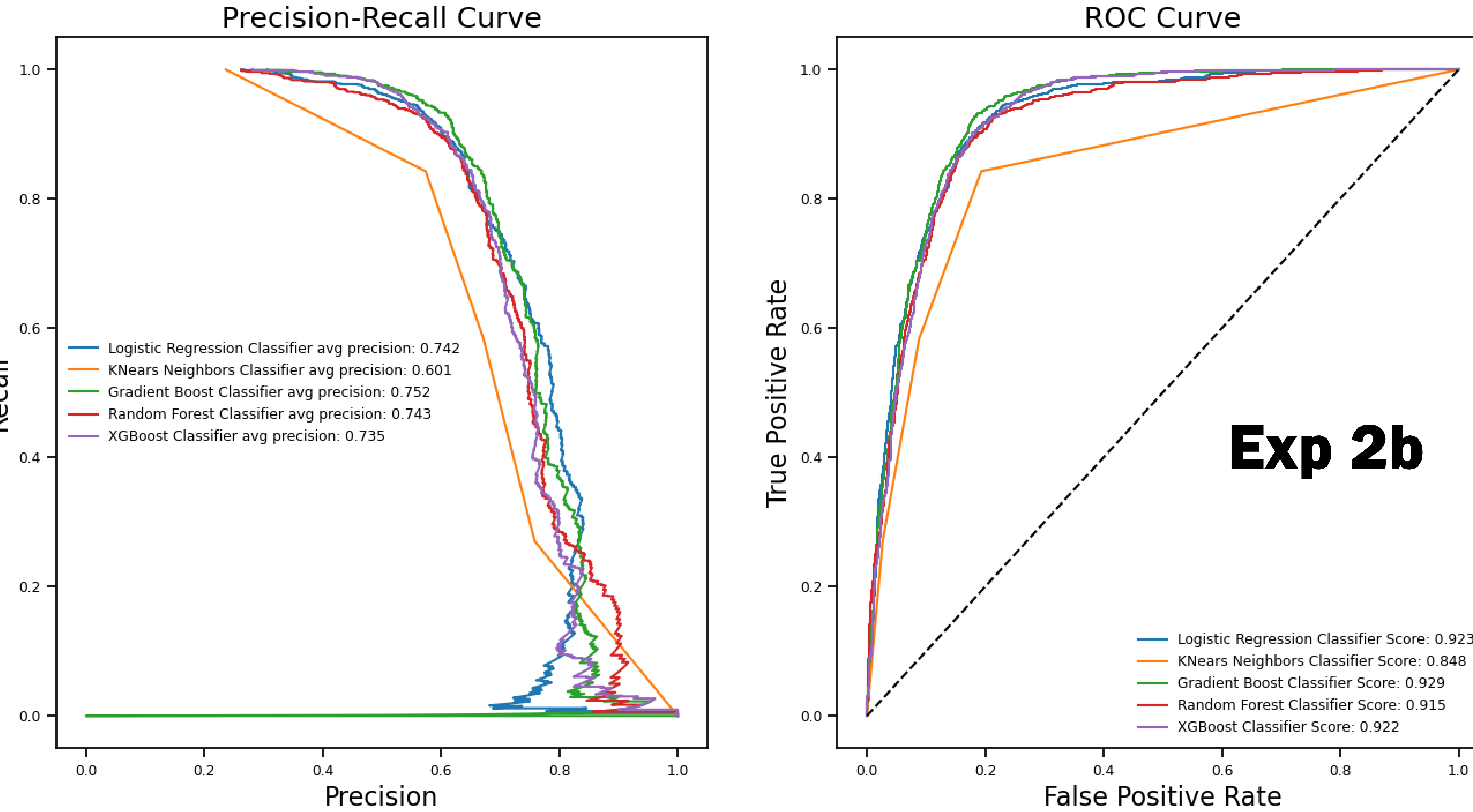
Classifier	Accur. Score	Macro F1-Score	Weighted F1-Score	P-R Curve	AUC-ROC
Log. Regression	0.86	0.82	0.86	0.75	0.93
KNN	0.85	0.82	0.86	0.69	0.91
Gradient Boost	0.88	0.85	0.89	0.78	0.94
Random Forest	0.84	0.81	0.85	0.77	0.93
XG Boost	0.88	0.85	0.89	0.77	0.94


Precision-Recall Curve
Logistic Regression Classifier avg precision: 0.747
KNears Neighbors Classifier avg precision: 0.688
Gradient Boost Classifier avg precision: 0.778
Random Forest Classifier avg precision: 0.768
XGBoost Classifier avg precision: 0.766
Precision
ROC Curve
True Positive Rate
Exp 2a
Logistic Regression Classifier Score: 0.928
KNears Neighbors Classifier Score: 0.905
Gradient Boost Classifier Score: 0.937
Random Forest Classifier Score: 0.930
XGBoost Classifier Score: 0.935
False Positive Rate

Predictions, Results and Evaluation

Exp 2b: SMOTE Technique on PCA - Oversampling

Classifier	Accur. Score	Macro F1-Score	Weighted F1-Score	P-R Curve	AUC-ROC
Log. Regression	0.86	0.82	0.86	0.74	0.92
KNN	0.85	0.80	0.85	0.60	0.85
Gradient Boost	0.87	0.83	0.87	0.75	0.93
Random Forest	0.83	0.80	0.84	0.74	0.92
XG Boost	0.87	0.83	0.87	0.74	0.92



Precision-Recall Curve
Recall
Precision
Logistic Regression Classifier avg precision: 0.742
KNears Neighbors Classifier avg precision: 0.601
Gradient Boost Classifier avg precision: 0.752
Random Forest Classifier avg precision: 0.743
XGBoost Classifier avg precision: 0.735
ROC Curve
True Positive Rate
False Positive Rate
Logistic Regression Classifier Score: 0.923
KNears Neighbors Classifier Score: 0.848
Gradient Boost Classifier Score: 0.929
Random Forest Classifier Score: 0.915
XGBoost Classifier Score: 0.922
Exp 2b

Predictions, Results and Evaluation

Exp 3a/3b: Artificial Neural Network Classification Model

Sample	Accur. Score	Macro F1-Score	Weighted F1-Score	P-R Curve	AUC-ROC
Best Features	0.87	0.82	0.87	0.63	0.81
SMOTE Sample	0.86	0.83	0.87	0.66	0.85

Reflection and Conclusion

This project was to build a classification model, at least one from a family of algorithm, that predicts whether clients will subscribe to the term deposit or not, based on various input variables such as age, job, marital status, education etc. The dataset was properly cleaned and preprocessed. To reduce training time, a resampling was carried out, this was also to reduce the adverse effect of imbalance in the data. The data were scaled using StandardScaler. The feature selection technique proved to be quite good on the models, with less impact by SMOTE techniques and PCA feature selection.

GridSearchCV was used to get the best-fit parameters on each of the models, which was evident in the model evaluation of 70 to 90% at almost all levels. The results from the different experiments and classification models didn't show much differences except the impacts by ensemble boosting models. In conclusion, the prediction was a success as proper machine learning workflow was followed for a better result. The imbalanced dataset was handled by resampling technique – which can heavily affect model evaluation. Thus, these steps are highly recommended for classification problems.

Project Timeline

Wk1. 19 Mar – 26 Mar 2023

Data Collection: Download the dataset from the given source and load it into a Jupyter notebook.

Data Exploration: Perform preliminary data exploration to understand the dataset's structure, quality, and potential challenges.

Wk2. 26 Mar – 2 Apr 2023

Data Preparation: Prepare the dataset for modeling by performing data cleaning, feature selection, feature engineering, and data transformation as necessary.

Modeling: Train different classification models on the prepared dataset and evaluate their performance using appropriate metrics.

Wk3. 2 Apr – 9 Apr 2023

Model Tuning: Tune the hyperparameters of the best-performing model to improve its performance further.

Model Interpretation: Interpret the trained model and identify the most significant factors that contribute to the prediction.

Wk4. 9 Apr – 16 Apr 2023

Finalize the project report: Create a comprehensive report documenting the entire project, including problem definition, methodology, results, and interpretation.

Submission: Submit the final report within 2 weeks by April 28, 2023, as per the given deadline..

Thank you

Thanks for going through this interesting journey with us.

We hope we have added value to this work.

GitHub Link

https://github.com/Jeks042/Data-Science-Projects