Predicting Customer Subscription to Fixed Deposit: A Machine Learning Approach

NAME	**SRIVIDYA IYER**
PROJECT	Predicting Customer Will Subscribe To FD
SUPERVISOR	**Mr.KASHIF**

Business Use Case

There has been a revenue decline for a Portuguese bank and they would like to know what actions to take. After investigation, they found out that the root cause is that their clients are not depositing as frequently as before. Knowing that term deposits allow banks to hold onto a deposit for a specific amount of time, so banks can invest in higher gain financial products to make a profit. In addition, banks also hold better chance to persuade term deposit clients into buying other products such as funds or insurance to further increase their revenues. As a result, the Portuguese bank would like to identify existing clients that have higher chance to subscribe for a term deposit and focus marketing efforts on such clients.

1. Problem Definition

In the modern banking landscape, term deposits stand as a significant revenue source for financial institutions. However, identifying potential customers who are likely to subscribe to term deposits poses a challenge. This project aims to utilize machine learning techniques to predict customer subscription to fixed deposits, aiding the bank in targeted marketing strategies.

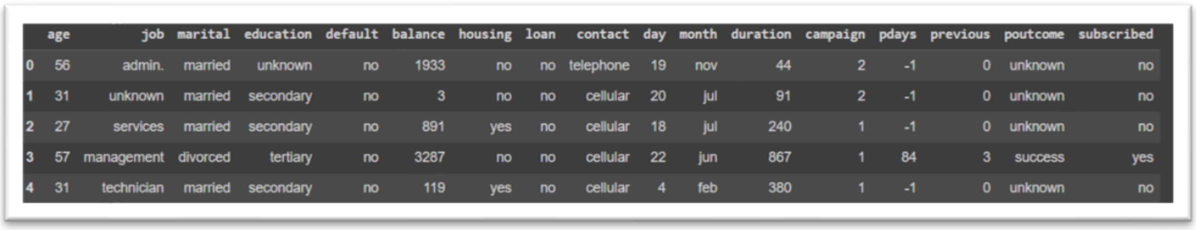
Dataset Attributes

Here is the description of all the variables:

- Variable: Definition
- ID: Unique client ID
- age: Age of the client
- job: Type of job
- marital: Marital status of the client
- education: Education level
- default: Credit in default.
- housing: Housing loan
- loan: Personal loan
- contact: Type of communication
- month: Contact month
- day_of_week: Day of week of contact
- duration: Contact duration
- campaign: number of contacts performed during this campaign to the client
- pdays: number of days that passed by after the client was last contacted
- previous: number of contacts performed before this campaign
- poutcome: outcome of the previous marketing campaign

Output variable (desired target):

- **Subscribed (target):** has the client subscribed a term deposit? (YES/NO)



	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	subscribed
0	56	admin.	married	unknown	no	1933	no	no	telephone	19	nov	44	2	-1	0	unknown	no
1	31	unknown	married	secondary	no	3	no	no	cellular	20	jul	91	2	-1	0	unknown	no
2	27	services	married	secondary	no	891	yes	no	cellular	18	jul	240	1	-1	0	unknown	no
3	57	management	divorced	tertiary	no	3287	no	no	cellular	22	jun	867	1	84	3	success	yes
4	31	technician	married	secondary	no	119	yes	no	cellular	4	feb	380	1	-1	0	unknown	no

2. Data Analysis

2.1 Load the Data

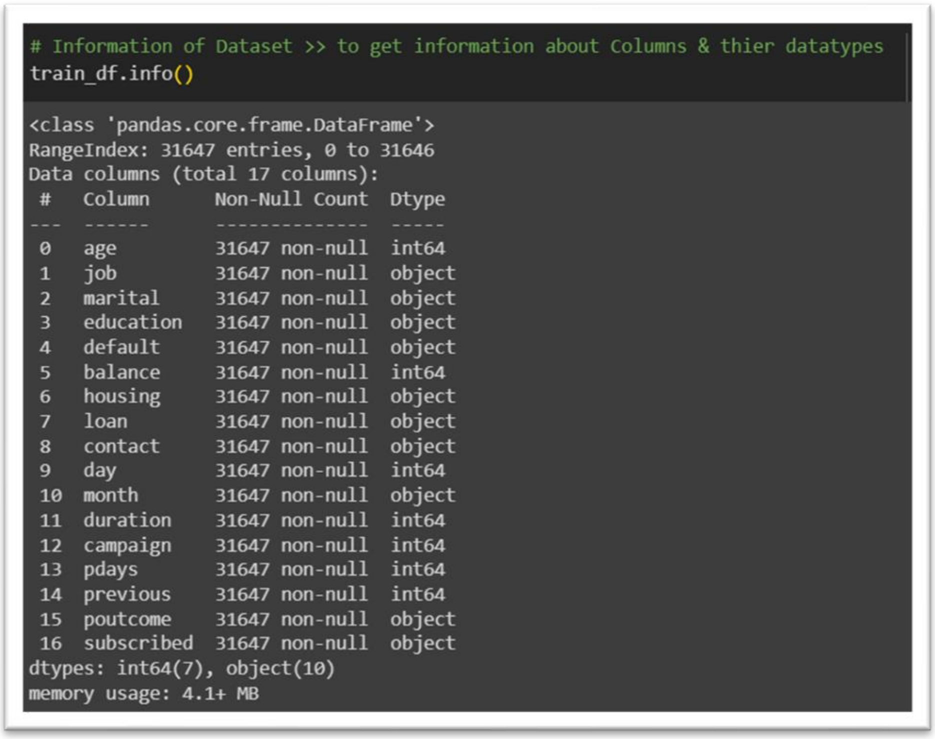
The dataset consists of client information such as age, job type, marital status, along with call details such as call duration and month.

```
# Load the dataset into df variable >>
train_df = pd.read_csv(/termdeposit_train.csv')
test_df = pd.read_csv(/termdeposit_test.csv')
```

2.2 Explore the Data

2.2.1 Overview

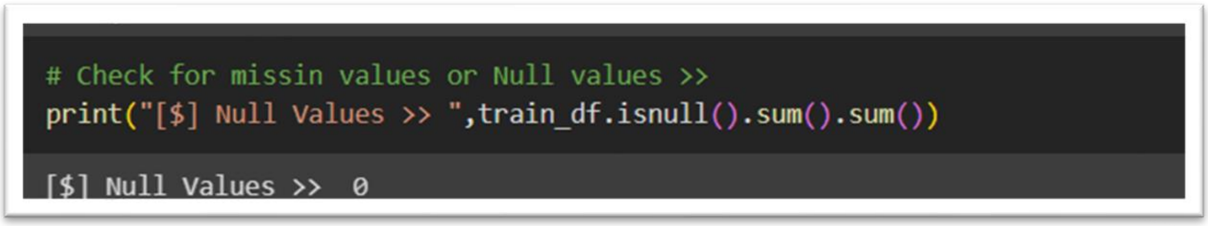
- The dataset contains X features and Y observations.



```
# Information of Dataset >> to get information about Columns & thier datatypes
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31647 entries, 0 to 31646
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         31647 non-null  int64
 1   job         31647 non-null  object
 2   marital     31647 non-null  object
 3   education   31647 non-null  object
 4   default     31647 non-null  object
 5   balance     31647 non-null  int64
 6   housing     31647 non-null  object
 7   loan        31647 non-null  object
 8   contact     31647 non-null  object
 9   day         31647 non-null  int64
 10  month       31647 non-null  object
 11  duration    31647 non-null  int64
 12  campaign    31647 non-null  int64
 13  pdays       31647 non-null  int64
 14  previous    31647 non-null  int64
 15  poutcome    31647 non-null  object
 16  subscribed  31647 non-null  object
dtypes: int64(7), object(10)
memory usage: 4.1+ MB
```

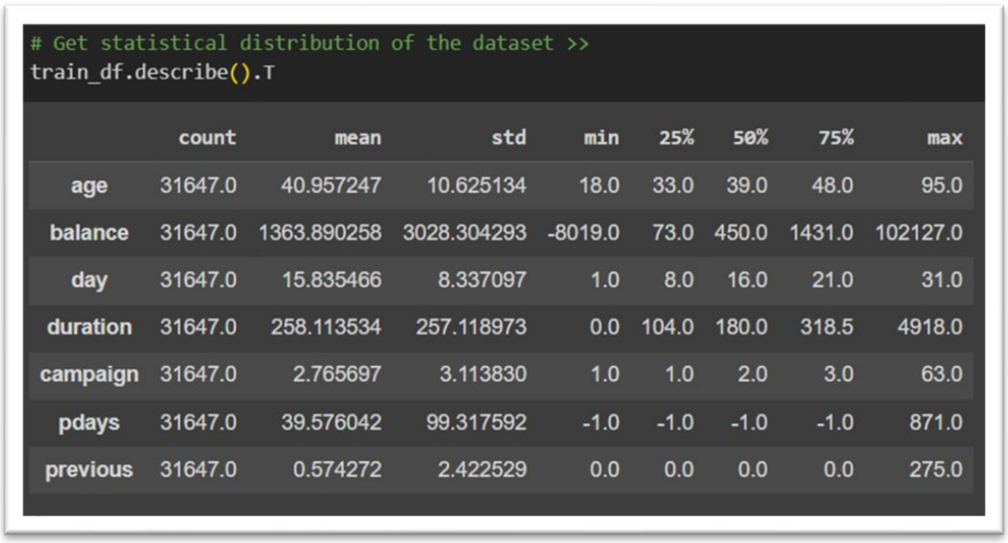
- No missing values were found in the dataset.



```
# Check for missin values or Null values >>
print("[$] Null Values >> ",train_df.isnull().sum().sum())
```

```
[$] Null Values >>  0
```

- Statistical distribution of the dataset reveals insights into the numerical variables.



```
# Get statistical distribution of the dataset >>
train_df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	31647.0	40.957247	10.625134	18.0	33.0	39.0	48.0	95.0
balance	31647.0	1363.890258	3028.304293	-8019.0	73.0	450.0	1431.0	102127.0
day	31647.0	15.835466	8.337097	1.0	8.0	16.0	21.0	31.0
duration	31647.0	258.113534	257.118973	0.0	104.0	180.0	318.5	4918.0
campaign	31647.0	2.765697	3.113830	1.0	1.0	2.0	3.0	63.0
pdays	31647.0	39.576042	99.317592	-1.0	-1.0	-1.0	-1.0	871.0
previous	31647.0	0.574272	2.422529	0.0	0.0	0.0	0.0	275.0

Certainly! Here's a simplified, single-line summary for each numerical variable:

1. **Age**:- The average age of customers is approximately 41 years, with a minimum age of 18 and a maximum age of 95.

2. **Balance**: - The average balance in customer accounts is around $1,364, with a wide range from -$8,019 to $102,127.

3. **Day**: - The calls were made mostly evenly throughout the month, with an average call day around the 16th.

4. **Duration**: - The average call duration is approximately 258 seconds, ranging from very short calls to as long as 4918 seconds.

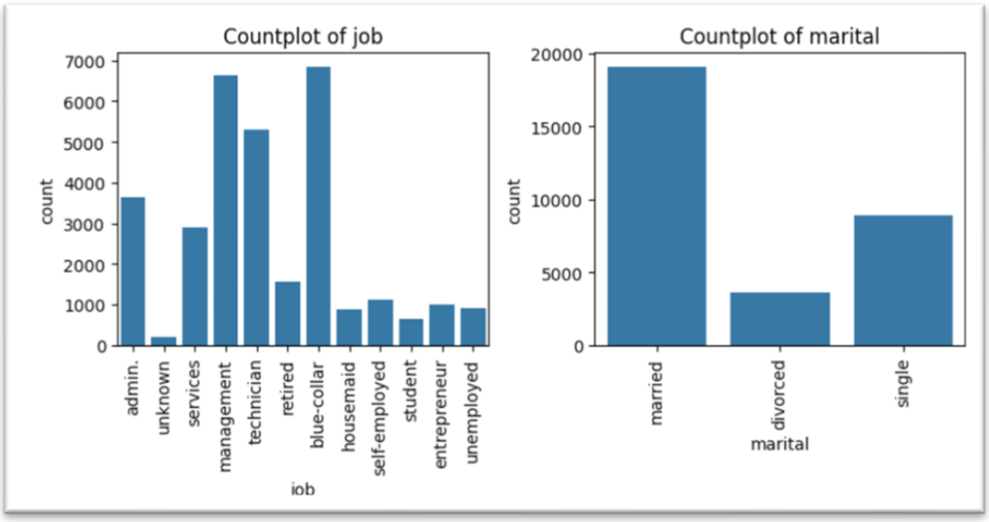
5. **Campaign**: - On average, each customer was contacted about 2-3 times during the campaign, with some customers being contacted up to 63 times.

6. **Pdays**: - The majority of customers were not previously contacted, with an average of around 40 days since their last contact.

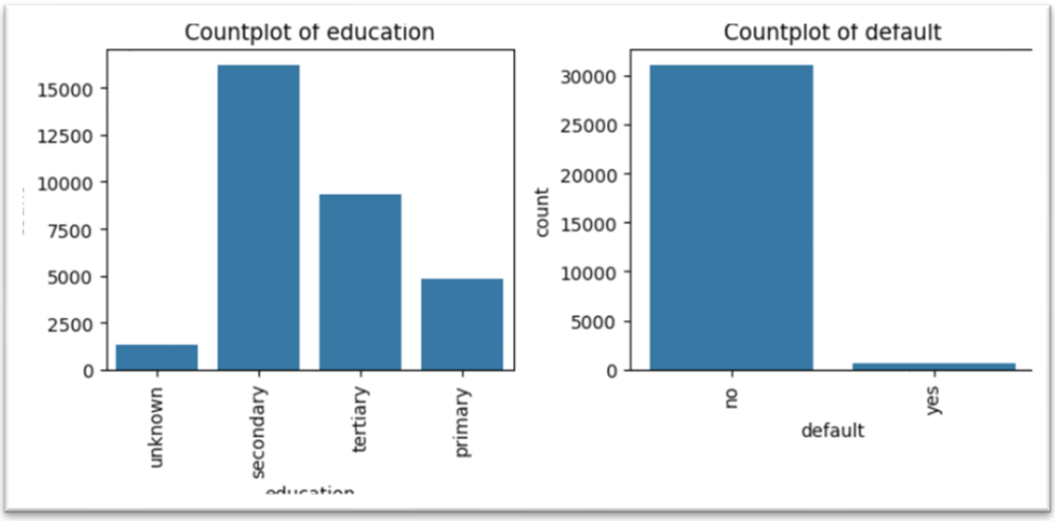
7. **Previous**: - On average, customers had less than one previous contact, with a maximum of 275 previous contacts.

2.2.2 Categorical Variable Analysis

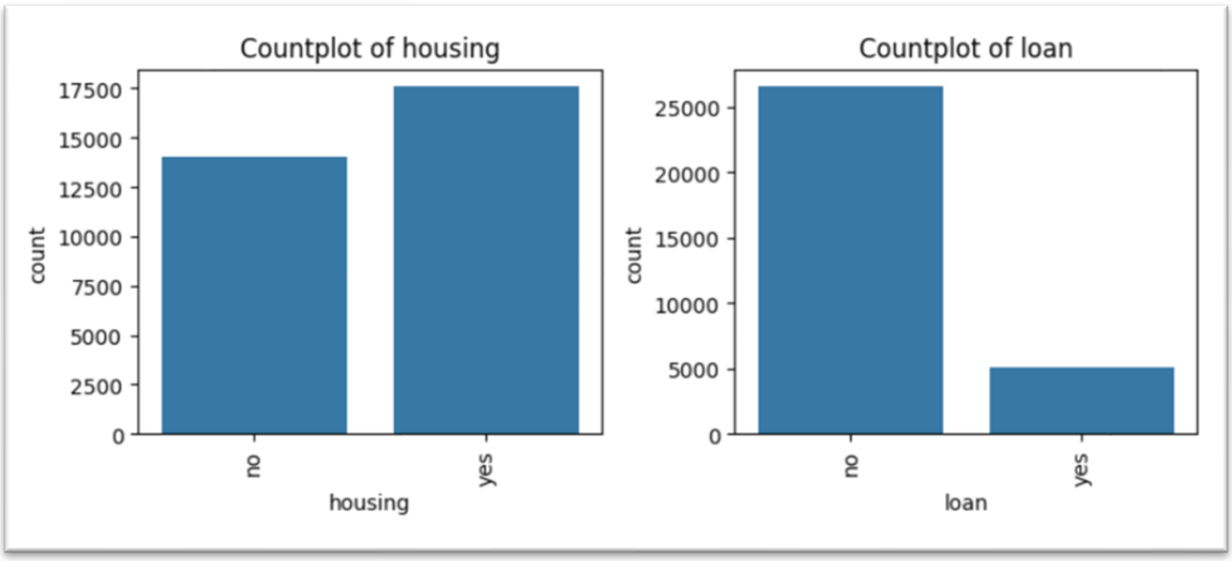
- Exploration of categorical variables provides insights into job type, marital status, education, etc., concerning subscription rates.



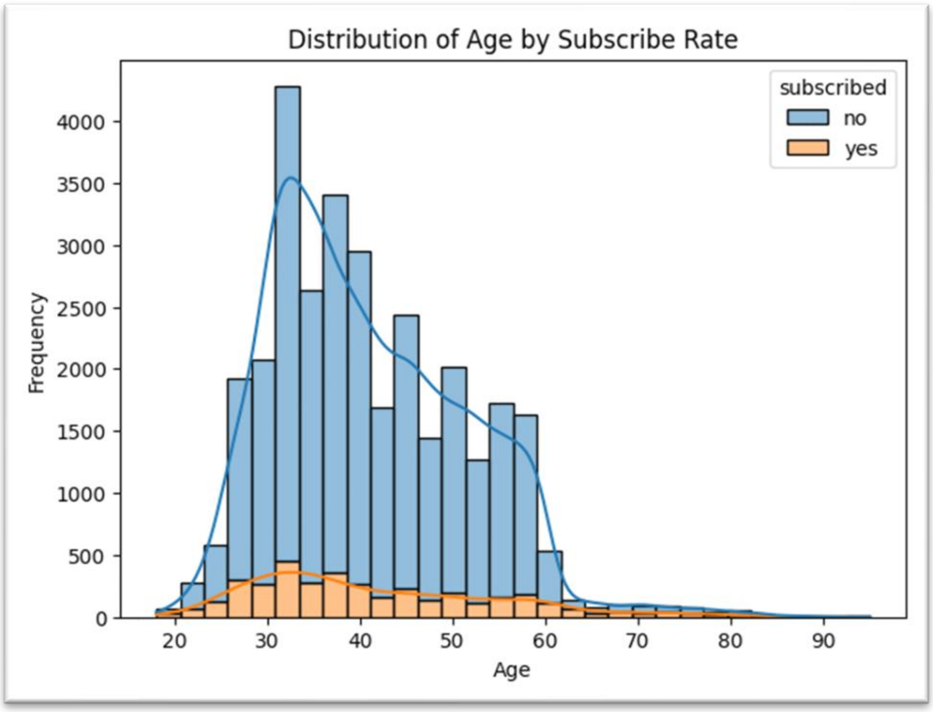
- We can see we have more no of Management & Blue-Collar jobs followed by technician & admin jobs
- Also, If we checked marital status, we have more no of married people counts



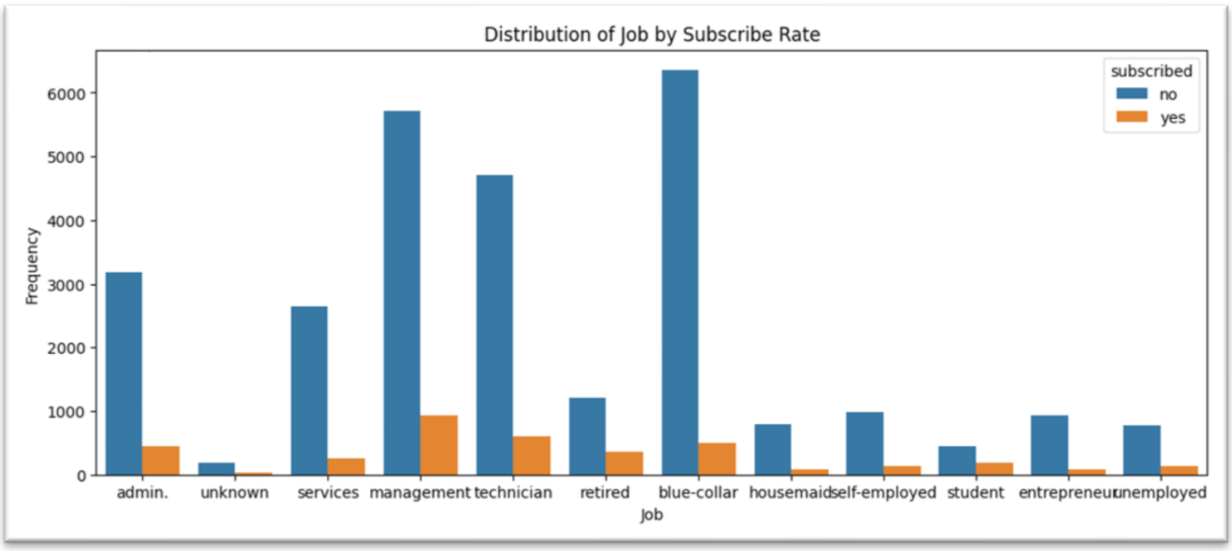
- We have our customer count with more secondary education Background followed by territory & Primary
- We have very less no of defaults count



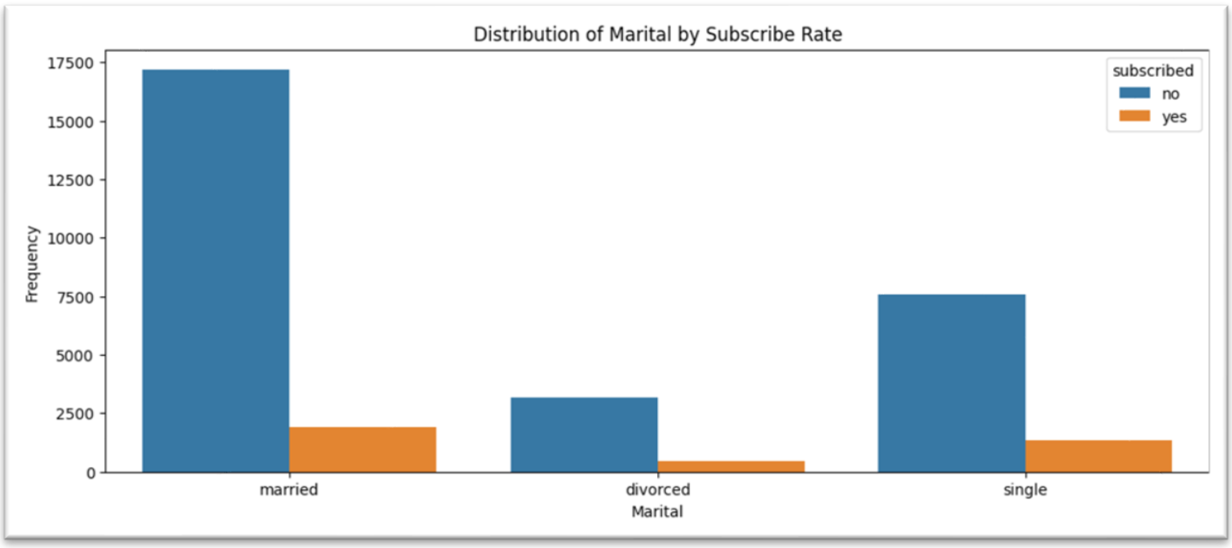
- We have more no of people who has housing
- We have very less people who has loans



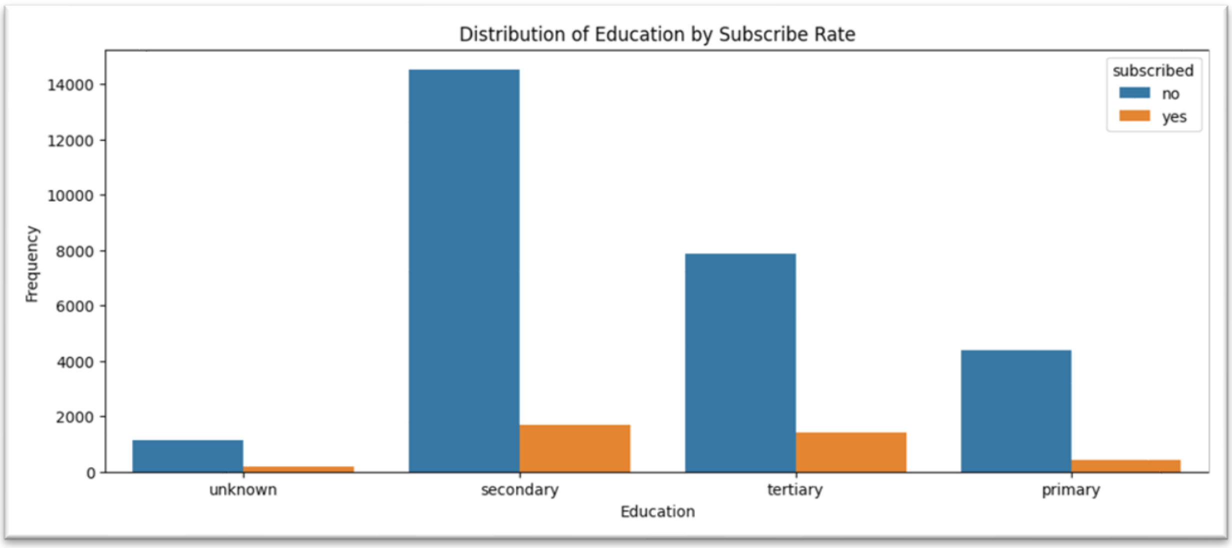
- We can see distribution of age who has subscribed & who not subscribed , Age group of 30 people mostly have max count for subscription as compared to other groups



- We can see We have max count for blue-collars who has not subscribed so we can target them accordingly
- Also, we have more no of management job people who has subscribed



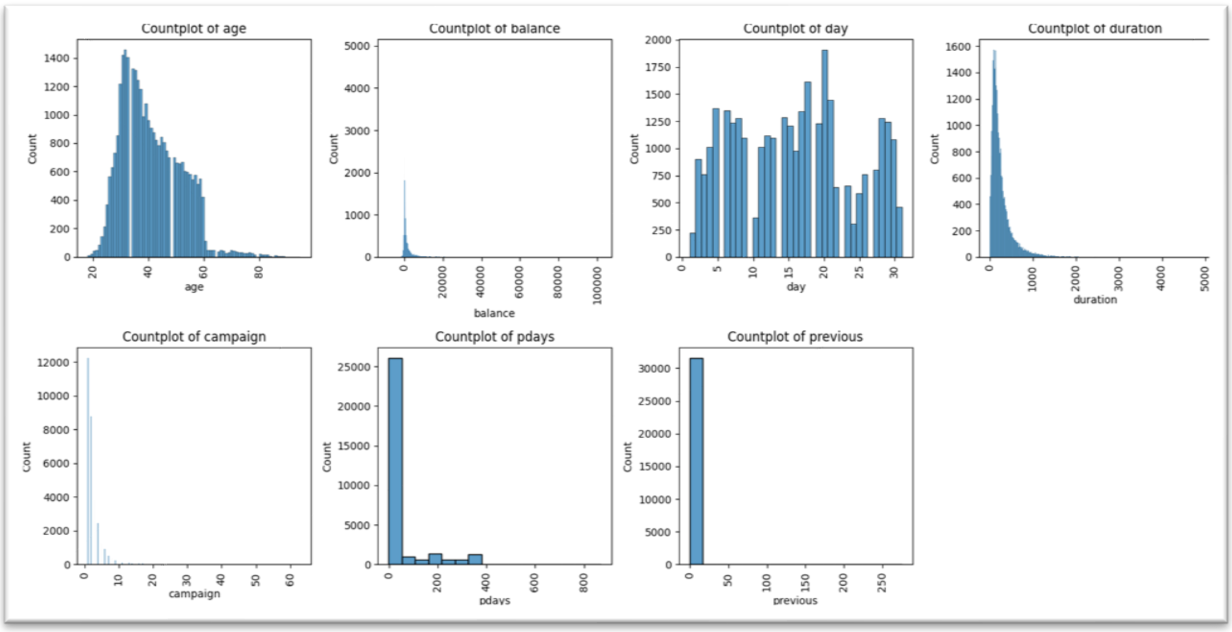
- Married & Single people are mostly interested for subscription as compared to divorced people



- Secondary & Tertiary education background people are most interested to subscribed than others

2.2.3 Numerical Variable Analysis

- Analysis of numerical variables such as age provides insights into their distribution concerning subscription rates.



3. Exploratory Data Analysis (EDA) Concluding Remarks

Exploratory data analysis revealed significant insights into the distribution of variables concerning subscription rates. Categorical variables such as job type and marital status exhibit varying subscription rates, indicating their importance in prediction models.

- We can see distribution of age who has subscribed & who not subscribed , Age group of 30 people mostly have max count for subscription as compared to other groups
- We can see We have max count for blue-collars who has not subscribed so we can target them accordingly
- Also, we have more no of management job people who has subscribed
- Married & Single people are mostly interested for subscription as compared to divorced people
- Secondary & Tertiary education background people are most interested to subscribed than others

4. Pre-processing Pipeline

4.1 Data Preprocessing

- Label encoding categorical variables to prepare them for model training.

```
from sklearn.preprocessing import LabelEncoder

# Selecting categorical columns >>
categorical_columns = ['job', 'marital', 'education', 'default', 'housing',
'loan', 'contact', 'month', 'poutcome']

# Applying LabelEncoder to each categorical column >>
label_encoders = {}
for column in categorical_columns:
    # Applying labelencoder >>
    label_encoders[column] = LabelEncoder()
    train_df[column] = label_encoders[column].fit_transform(train_df[column])
    test_df[column] = label_encoders[column].transform(test_df[column])
```

Here's a detailed explanation of each step:

- For each categorical column, a new LabelEncoder instance is created and stored in the label_encoders dictionary.

- The fit_transform() method is called on the training dataset column to both fit the encoder to the unique categories in that column and transform the categories into numerical labels simultaneously. This ensures that the labels are consistent across the training data.

- The transform() method is used on the test dataset column to apply the same transformation learned from the training data. This ensures that the same encoding scheme is used consistently across both datasets.

This process effectively converts categorical variables into numerical labels, making them suitable for consumption by machine learning algorithms.

5. Building Machine Learning Models

5.1 Data Splitting

- Splitting the dataset into training and validation sets for model training and evaluation.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report

# Split the training data into features and target variable
X = train_df.drop(columns=['subscribed'])
y = train_df['subscribed']

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

- First, we have divided our data into Features & Target variable

- Here we are going to use train_test_split function so we can divide our data into train & test data so we can use train data for model building & test data for model evaluation

5.2 Model Building

5.2.1 Model Selection

- Utilizing Logistic Regression, Random Forest, and Gradient Boosting algorithms for prediction.

```python
# Initialize models

models = {
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(),
    'Gradient Boosting': GradientBoostingClassifier()
}
```

```python
# Train and evaluate models
for name, model in models.items():
    print("-"*70)
    print(f"Training and evaluating {name}...")
    # Train the model
    model.fit(X_train, y_train)

    # Predict on the validation data
    y_pred = model.predict(X_val)

    # Evaluate the model
    accuracy = accuracy_score(y_val, y_pred)
    report = classification_report(y_val, y_pred)

    print(f"Accuracy of {name}: {accuracy:.2f}")
    print(f"Classification Report of {name}:\n{report}\n")
print("-"*70)
```

In this phase, we initialized three distinct models: Logistic Regression, Random Forest, and Gradient Boosting. Each of these models offers unique characteristics and is commonly employed in predictive modeling tasks.

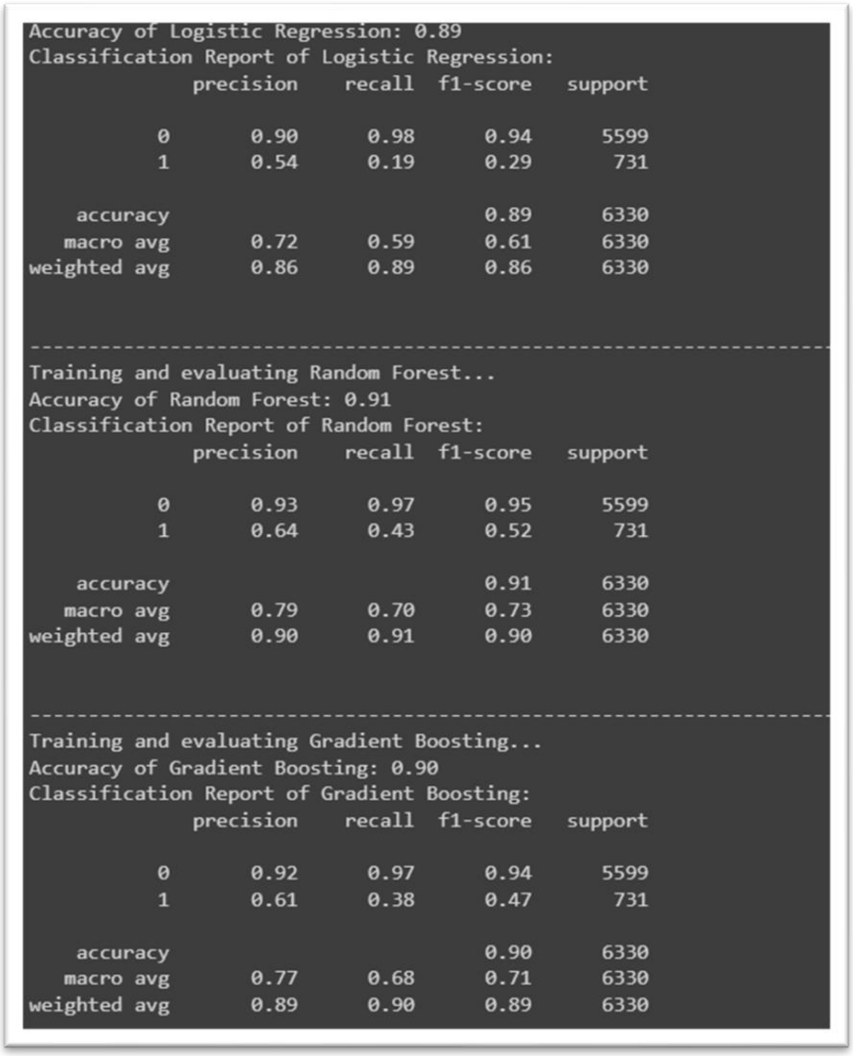
Subsequently, we iterated over each model and conducted the training and evaluation process. For each model:

- Training Phase: We trained the model using the training data, which consists of features (X_train) and their corresponding target labels (y_train).

- Evaluation Phase: After training, we assessed the model's performance using the validation dataset (X_val) by predicting the target labels and comparing them against the actual labels (y_val). The evaluation metrics used include accuracy and a detailed classification report.

5.2.2 Model Evaluation

- Evaluation metrics include accuracy, precision, recall, and F1-score for each model.



```
Accuracy of Logistic Regression: 0.89
Classification Report of Logistic Regression:
              precision    recall  f1-score   support

           0       0.90      0.98      0.94      5599
           1       0.54      0.19      0.29       731

    accuracy                           0.89      6330
   macro avg       0.72      0.59      0.61      6330
weighted avg       0.86      0.89      0.86      6330


--------------------------------------------------------------------
Training and evaluating Random Forest...
Accuracy of Random Forest: 0.91
Classification Report of Random Forest:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      5599
           1       0.64      0.43      0.52       731

    accuracy                           0.91      6330
   macro avg       0.79      0.70      0.73      6330
weighted avg       0.90      0.91      0.90      6330


--------------------------------------------------------------------
Training and evaluating Gradient Boosting...
Accuracy of Gradient Boosting: 0.90
Classification Report of Gradient Boosting:
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      5599
           1       0.61      0.38      0.47       731

    accuracy                           0.90      6330
   macro avg       0.77      0.68      0.71      6330
weighted avg       0.89      0.90      0.89      6330
```

The classification report provides insights into the model's performance, including precision, recall, and F1-score for each class (subscribed or not subscribed), along with their weighted average.

The training and evaluation process enables us to gauge the effectiveness of each model in predicting customer subscription to fixed deposits. By comparing their performance metrics, we can determine which model exhibits the most promising results for deployment in real-world scenarios.

This iterative approach to model evaluation ensures that we select the most suitable algorithm for our predictive task, ultimately enhancing the effectiveness of our solution in identifying potential fixed deposit subscribers.

5.3 Model Comparison and Selection

- Comparative analysis of model performance to identify the most suitable algorithm for customer subscription prediction.

1. Logistic Regression:
   - Accuracy: 0.89
   - Precision:
     - Class 0: 0.90
     - Class 1: 0.54
   - Recall:
     - Class 0: 0.98
     - Class 1: 0.19
   - F1-score:
     - Class 0: 0.94
     - Class 1: 0.29
   - Evaluation: Logistic Regression achieved an accuracy of 0.89. While it demonstrated high precision and recall for Class 0, it struggled to correctly classify instances of Class 1, resulting in a lower recall and F1-score for that class.

2. Random Forest:

- Accuracy: 0.91
- Precision:
  - Class 0: 0.93
  - Class 1: 0.64
- Recall:
  - Class 0: 0.97
  - Class 1: 0.43
- F1-score:
  - Class 0: 0.95
  - Class 1: 0.52
- Evaluation: Random Forest achieved an accuracy of 0.91. It demonstrated higher precision and recall for both classes compared to Logistic Regression, resulting in better overall performance and F1-scores for both classes.

3. Gradient Boosting:

- Accuracy: 0.90
- Precision:
  - Class 0: 0.92
  - Class 1: 0.61
- Recall:
  - Class 0: 0.97
  - Class 1: 0.38
- F1-score:
  - Class 0: 0.94
  - Class 1: 0.47
- Evaluation: Gradient Boosting achieved an accuracy of 0.90. It showed similar precision for Class 0 as Random Forest but lower precision for Class 1. Additionally, it demonstrated lower recall and F1-score for Class 1 compared to Random Forest.

6. Concluding Remarks

- Among the three models evaluated, Random Forest exhibited the highest accuracy, precision, recall, and F1-scores for both classes.

- Logistic Regression struggled to correctly classify instances of Class 1, resulting in lower recall and F1-score for that class.

- Gradient Boosting showed competitive performance but lagged slightly behind Random Forest in terms of precision, recall, and F1-score for both classes.

Given these results, Random Forest emerges as the preferred choice due to its overall better performance across all evaluation metrics.