Name: Diyora Radhika

Email: radhikadiyora2023@gmail.com

Country: Germany

College: IU Internationale hochschule

📌 PROBLEM DESCRIPTION

The objective of this project is to develop a machine learning model that predicts whether a bank customer will subscribe to a term deposit product. The prediction is based on customer demographic details, previous marketing interactions, and socio-economic indicators.

The bank wants to identify customers with a higher probability of subscription so that marketing efforts such as tele-marketing, SMS, and email campaigns can be targeted efficiently. This helps reduce marketing costs and improves campaign success rates.

📌 DATA UNDERSTANDING

The dataset used in this project is obtained from the **UCI Machine Learning Repository – Bank Marketing Dataset**. The data represents direct marketing campaigns conducted by a Portuguese banking institution, primarily through phone calls.

Each record corresponds to a customer interaction, and the target variable indicates whether the customer subscribed to a term deposit (yes or no).

📌 TYPE OF DATA USED FOR ANALYSIS

The dataset contains **both categorical and numerical data**:

🔹 **Numerical Variables**

- Age
- Duration of last contact
- Campaign (number of contacts in current campaign)
- Pdays (days since last contact)
- Previous (number of previous contacts)
- Economic indicators (employment rate, euribor rate, consumer price index, etc.)

🔹 **Categorical Variables**

- Job
- Marital status
- Education

- Default, housing, loan status
- Contact type
- Month and day of the week
- Outcome of previous campaign

🔹 **Target Variable**

- y – whether the client subscribed to a term deposit (yes / no)

📌 **DATA PROBLEMS IDENTIFIED**

1️⃣ **Missing / Unknown Values**

- Some categorical variables such as **job, education, default, housing, and loan** contain values labeled as "unknown".
- These represent missing or unavailable customer information.

2️⃣ **Imbalanced Dataset**

- The target variable is highly imbalanced.
- Most customers **did not subscribe** to the term deposit, while only a small percentage subscribed.

3️⃣ **Outliers**

- Numerical variables like **duration** and **campaign** contain extreme values.
- Long call durations heavily influence the target variable.

4️⃣ **Skewed Features**

- Features such as **pdays** and **duration** show right-skewed distributions.
- A large number of records have pdays = 999, indicating customers not previously contacted.

5️⃣ **Business Constraint on Duration**

- The duration feature is not available before a call is made.
- Including it creates data leakage and makes the model unrealistic for real-world campaigns.

📌 **APPROACHES USED TO HANDLE DATA ISSUES (AND WHY)**

🔹 **Handling Missing Values**

- "unknown" values were retained as a separate category instead of removal.
- This avoids loss of data and preserves customer records.

🔹 **Handling Imbalanced Data**

- Techniques such as **class weighting and resampling** were explored.
- Performance was evaluated using metrics like **Precision, Recall, and ROC-AUC** instead of accuracy alone.

🔹 **Handling Outliers**

- Extreme values were analyzed using boxplots and distribution plots.
- No aggressive removal was done to avoid losing important customer behavior signals.

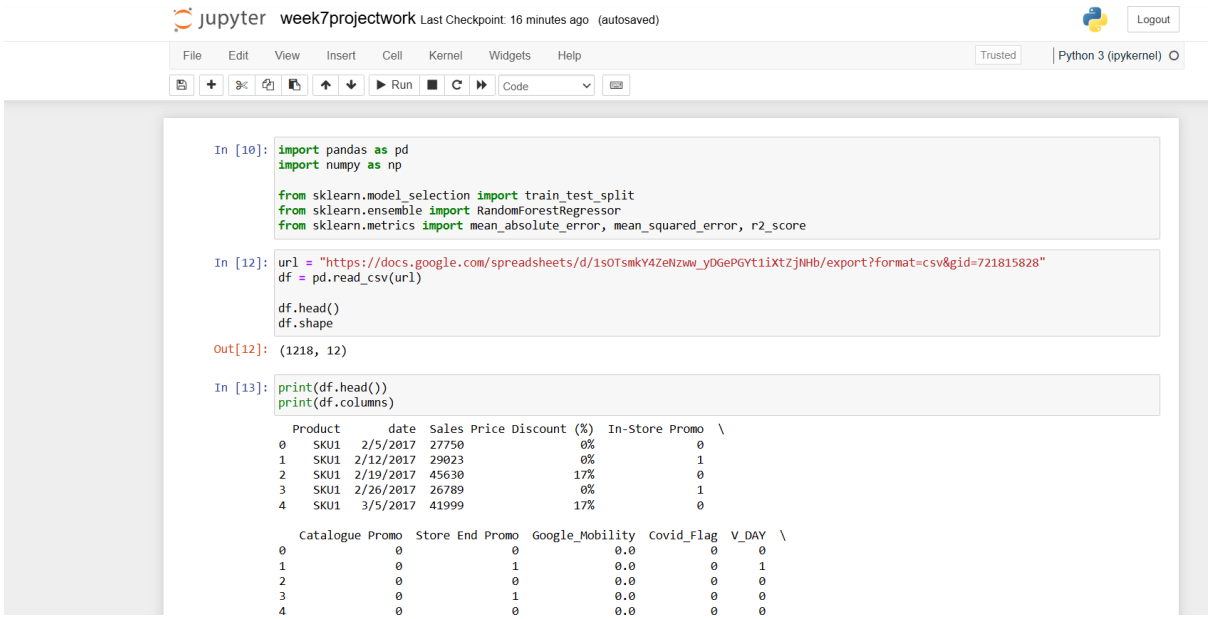
🔹 **Handling Skewness**

- Distribution analysis was performed to understand skewed features.
- Tree-based models were preferred as they are less sensitive to skewness.

🔹 **Duration Feature Strategy**

- Two models were developed:
  - **Model with duration** (benchmark model)
  - **Model without duration** (business-realistic model)
- This allows comparison while maintaining explainability and practical usability.

Code:



File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [13]:
```
print(df.head())
print(df.columns)
```

```
  Product       date  Sales Price Discount (%)  In-Store Promo  \
0    SKU1   2/5/2017  27750                 0%               0
1    SKU1  2/12/2017  29023                 0%               1
2    SKU1  2/19/2017  45630                17%               0
3    SKU1  2/26/2017  26789                 0%               1
4    SKU1   3/5/2017  41999                17%               0

   Catalogue Promo  Store End Promo  Google_Mobility  Covid_Flag  V_DAY  \
0                0                0              0.0           0      0
1                0                1              0.0           0      1
2                0                0              0.0           0      0
3                0                1              0.0           0      0
4                0                0              0.0           0      0

   EASTER  CHRISTMAS
0       0          0
1       0          0
2       0          0
3       0          0
4       0          0
Index(['Product', 'date', 'Sales', 'Price Discount (%)', 'In-Store Promo',
       'Catalogue Promo', 'Store End Promo', 'Google_Mobility', 'Covid_Flag',
       'V_DAY', 'EASTER', 'CHRISTMAS'],
      dtype='object')
```

In [14]:
```
df['Price Discount (%)'] = df['Price Discount (%)'].astype(str)
df['Price Discount (%)'] = df['Price Discount (%)'].str.replace('%', '', regex=False)
df['Price Discount (%)'] = pd.to_numeric(df['Price Discount (%)'])
```

In [15]:
```
df['date'] = pd.to_datetime(df['date'])
```

jupyter week7projectwork Last Checkpoint: 16 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [14]:
```
df['Price Discount (%)'] = df['Price Discount (%)'].astype(str)
df['Price Discount (%)'] = df['Price Discount (%)'].str.replace('%', '', regex=False)
df['Price Discount (%)'] = pd.to_numeric(df['Price Discount (%)'])
```

In [15]:
```
df['date'] = pd.to_datetime(df['date'])
```

In [16]:
```
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
df['week'] = df['date'].dt.isocalendar().week.astype(int)

df.drop(columns=['date'], inplace=True)
```

In [17]:
```
df = pd.get_dummies(df, columns=['Product'], drop_first=True)
```

In [18]:
```
X = df.drop(columns=['Sales'])
y = df['Sales']
```

In [19]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

In [20]:
```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(
    n_estimators=200,
    random_state=42
)

model.fit(X_train, y_train)
```

Out[20]:
```
RandomForestRegressor(n_estimators=200, random_state=42)
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [18]:
```
X = df.drop(columns=['Sales'])
y = df['Sales']
```

In [19]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

In [20]:
```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(
    n_estimators=200,
    random_state=42
)

model.fit(X_train, y_train)
```

Out[20]:
```
RandomForestRegressor(n_estimators=200, random_state=42)
```

In [22]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

y_pred = model.predict(X_test)

print("MAE:", mean_absolute_error(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R2 Score:", r2_score(y_test, y_pred))
```

```
MAE: 5766.219651639343
RMSE: 11688.289222644062
R2 Score: 0.8334071057220762
```