# Week 9 Deliverables

Name: Riwaj Neupane
Email: neupaneriwaj64@gmail.com
Country: Nepal
College/Company: NA
Specialization: Data Science

## Problem Description:

ABC Bank wants to sell its term deposit product to customers and before launching the product they want to develop a model which helps them in understanding whether a particular customer will buy their product or not (based on customer's past interaction with bank or other Financial Institution).

### Data Description:

The dataset going to be used for the analysis is called "bank-additional-full.csv", which contains 41188 observations and 21 features, encompassing features related to clients' basic information such as age, job, marital status, education, credit in default, housing, and loan; details about contact such as contact communication type, last contact month, last contact day, last contact duration, number of contacts, etc., and information about marketing campaigns like outcome, employment variation rate, consumer price index, consumer confidence index, euribor 3 month rate, and number of employees. We also have the target variable y, which is the answer for the yes-no question "has the client subscribed a term deposit?", and it will be used in future prediction.

| Feature Name | Type | Data Type | Number of unknowns | Number of Outliers | Comments |
|---|---|---|---|---|---|
| age | Numerical | int | 0 | 381 | Replace with upper bound defined as Q3+IQR |
| job | Categorical | str | 330 | 0 | Replace with mode |
| martial | Categorical | str | 80 | 0 | Replace with mode |
| education | Categorical | str | 1731 | 0 | |
| default | Categorical | str | 8597 | 0 | Leave unknown as its own type |

| | | | | | |
|---|---|---|---|---|---|
| housing | Categorical | str | 990 | 0 | Replace with mode |
| loan | Categorical | str | 990 | 0 | Replace with mode |
| contact | Categorical | str | 0 | 0 | |
| month | Categorical | str | 0 | 0 | |
| day_of_week | Categorical | str | 0 | 0 | |
| duration | Numerical | int | 0 | 861 | Replace with upper bound defined as Q3+IQR |
| campaign | Numerical | int | 0 | 0 | |
| previous | Numerical | int | 0 | 0 | |
| poutcome | Categorical | str | 0 | 0 | |
| emp.var.rate | Numerical | float64 | 0 | 0 | |
| cons.price.idx | Numerical | float64 | 0 | 0 | |
| cons.conf.idx | Numerical | float64 | 0 | 0 | |
| euribor3m | Numerical | float64 | 0 | 0 | |
| nr.employed | Numerical | float64 | 0 | 0 | |
| y | Categorical | str | 0 | 0 | |

**Problems in the Data (number of NA values, outliers , skewed etc):**

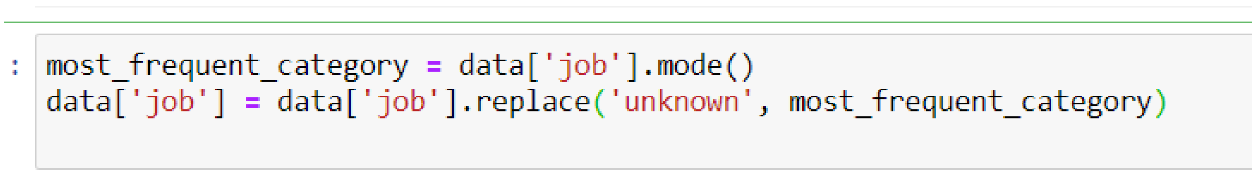
There are 6 categorical features with missing data (job, education, marital, default, housing, & loan). There is one numerical feature (“duration”) that contains outlier data. Specifically, we have the mean for “duration” is around 258, but the maximum value is 4918, which indicates the existence of outliers. And in general, the dataset is imbalanced, as the target variable for the predictive classification model skews highly to the “N” case.

## Approaches to Overcome These Problems:

For categorical data feature with unknown as category for (job, education, marital, default, housing, & loan). Replacing unknown we can use 2 method:

1. Replacing with mode:

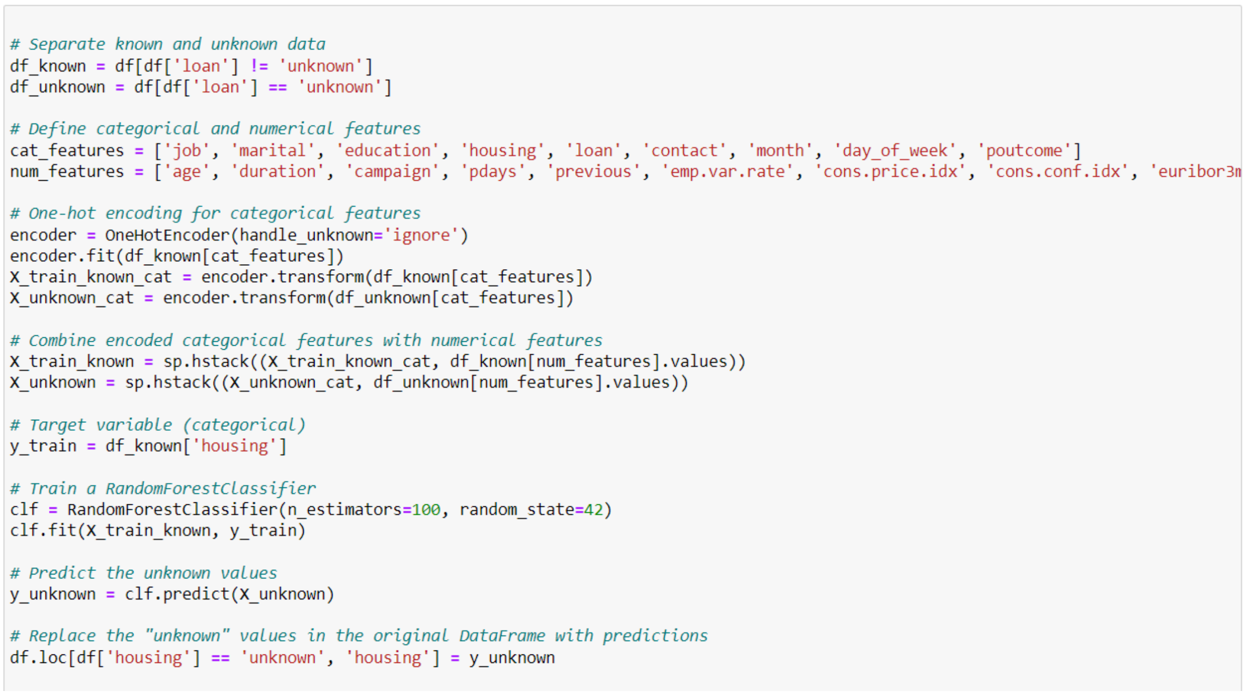
   Example:



```
:  most_frequent_category = data['job'].mode()
   data['job'] = data['job'].replace('unknown', most_frequent_category)
```

2. Replacing using RandomForestClassifier:

   Example:

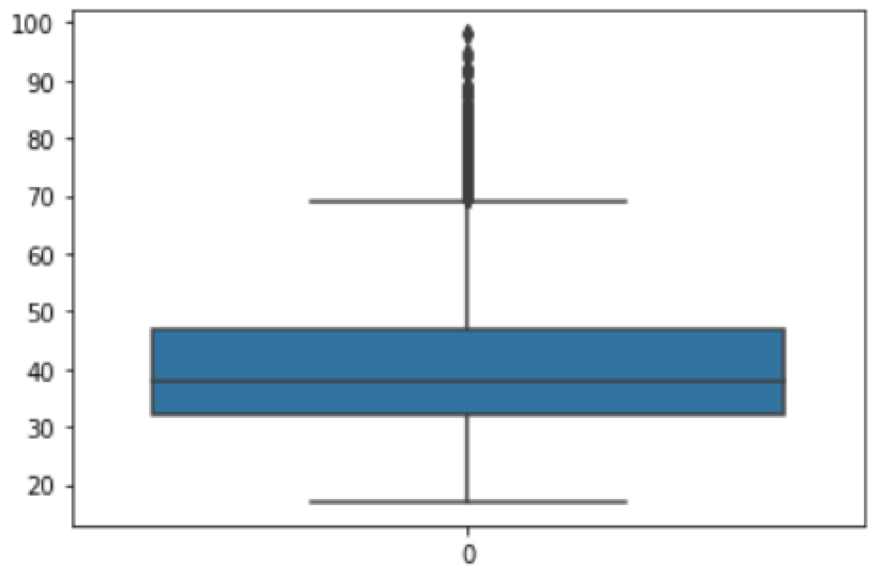


```
# Separate known and unknown data
df_known = df[df['loan'] != 'unknown']
df_unknown = df[df['loan'] == 'unknown']

# Define categorical and numerical features
cat_features = ['job', 'marital', 'education', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'poutcome']
num_features = ['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m

# One-hot encoding for categorical features
encoder = OneHotEncoder(handle_unknown='ignore')
encoder.fit(df_known[cat_features])
X_train_known_cat = encoder.transform(df_known[cat_features])
X_unknown_cat = encoder.transform(df_unknown[cat_features])

# Combine encoded categorical features with numerical features
X_train_known = sp.hstack((X_train_known_cat, df_known[num_features].values))
X_unknown = sp.hstack((X_unknown_cat, df_unknown[num_features].values))

# Target variable (categorical)
y_train = df_known['housing']

# Train a RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train_known, y_train)

# Predict the unknown values
y_unknown = clf.predict(X_unknown)

# Replace the "unknown" values in the original DataFrame with predictions
df.loc[df['housing'] == 'unknown', 'housing'] = y_unknown
```

3. Detecting and Removing Outliers:

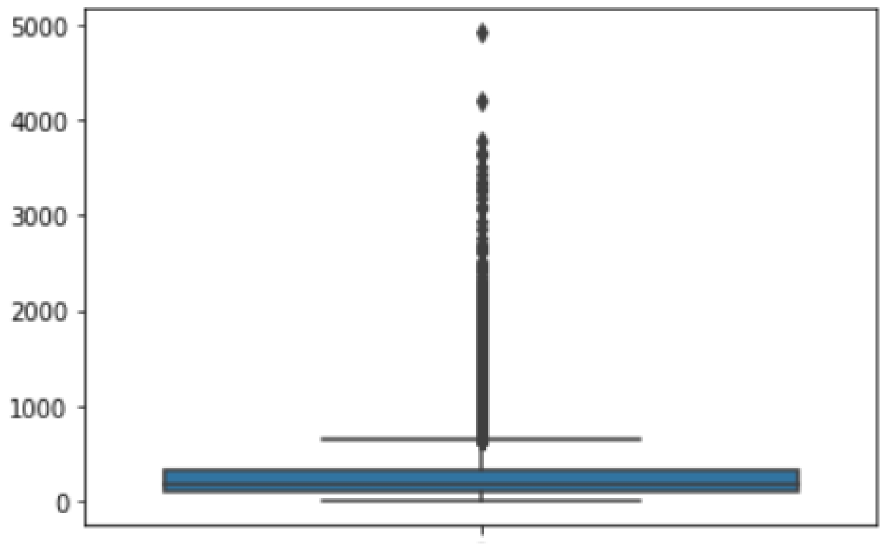
```
sns.boxplot(data.age)
```

```
<Axes: >
```



```
sns.boxplot(data.duration)
```

```
<Axes: >
```



```python
def remove_outliers_iqr(df):
    outliers = {}
    for col in df.columns:
        if np.issubdtype(df[col].dtype, np.number):  # Check if the column contains numerical data
            # Calculate the IQR (Interquartile Range) for the column
            Q1 = df[col].quantile(0.25)
            Q3 = df[col].quantile(0.75)
            IQR = Q3 - Q1

            # Define lower and upper bounds for outliers
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR

            # Find the indices of outliers
            outlier_indices = (df[col] < lower_bound) | (df[col] > upper_bound)

            # Create a DataFrame containing the outliers
            col_outliers = df[outlier_indices]

            # Add the outliers to the dictionary
            outliers[col] = col_outliers

    # Remove the rows containing outliers from the original DataFrame
    for col, col_outliers in outliers.items():
        df = df[~df.index.isin(col_outliers.index)]

    # Print information about removed outliers and the new shape
    if outliers:
        print('These outliers have been removed from your dataset:')
        for col, col_outliers in outliers.items():
            print(f'\nOutliers in column "{col}":')
            print(col_outliers)
    else:
        print('No outliers were found in the dataset.')

    print('\nNew shape is:', df.shape)
    return df
```