Linear Classification and Logistic Regression

```
df = pd.read_csv('https://query.data.world/s/wh6j7rxy2hvrn4ml75ci62apk5hgae')
#check distribution of target variable
#prints
3A    51481
2A    10576
2B    10096
1B       16
1A       16
Name: QScore, dtype: int64

df['QScore'].value_counts()
df.isna().sum()
#for simplicity, we will drop the rows with missing values.
df = df.dropna()
df.isna().sum()
#An obvious change in our target variable after removing the missing values is that there
are only three classes left #and from the distribution of the 3 classes, we can see that
there is an obvious imbalance between the classes. #There are methods that can be applied to
handle this imbalance such as oversampling and undersampling.
#Oversampling involves increasing the number of instances in the class with fewer instances
while undersampling #involves reducing the data points in the class with more instances.
#For now, we will convert this to a binary classification problem by combining class '2A'
and '1A'.
df['QScore'] = df['QScore'].replace(['1A'], '2A')
df.QScore.value_counts()
#prints
3A    51473
2A      240
Name: QScore, dtype: int64

df_2A = df[df.QScore=='2A']
df_3A = df[df.QScore=='3A'].sample(350)
data_df = df_2A.append(df_3A)

import sklearn.utils
data_df = sklearn.utils.shuffle(data_df)
data_df = data_df.reset_index(drop=True)
data_df.shape
data_df.QScore.value_counts()
#prints
3A    350
2A    240
Name: QScore, dtype: int64

#more preprocessing
data_df = data_df.drop(columns=['country_code', 'country', 'year'])
```

```
X = data_df.drop(columns='QScore')
y = data_df['QScore']

#split the data into training and testing sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
y_train.value_counts()
#prints
3A    242
2A    171
Name: QScore, dtype: int64
#There is still an imbalance in the class distribution. For this, we use SMOTE only on the
training data to handle this.

#encode categorical variable
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
x_train.record = encoder.fit_transform(x_train.record)
x_test.record = encoder.transform(x_test.record)

import imblearn
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=1)
x_train_balanced, y_balanced = smote.fit_sample(x_train, y_train)

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
normalised_train_df = scaler.fit_transform(x_train_balanced.drop(columns=['record']))
normalised_train_df = pd.DataFrame(normalised_train_df,
columns=x_train_balanced.drop(columns=['record']).columns)
normalised_train_df['record'] = x_train_balanced['record']

x_test = x_test.reset_index(drop=True)
normalised_test_df = scaler.transform(x_test.drop(columns=['record']))
normalised_test_df = pd.DataFrame(normalised_test_df,
columns=x_test.drop(columns=['record']).columns)
normalised_test_df['record'] = x_test['record']



#Logistic Regression
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(normalised_train_df, y_balanced)
#returns
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```