Inferential Statistics

The data is stored in the dataframe df.

```
neg = df[df['airline_sentiment'] == 'negative']
pos = df[df['airline_sentiment'] == 'positive']
pos_len = len(pos)
neg_len = len(neg)
sentisize = len(df['airline_sentiment'])
print("Total number of negative sentiments: " + str(neg_len))
print("Total number of positive sentiments: " + str(pos_len))
print("The total entries in the field 'airline sentiment':" + str(sentisize))
```

```
Total number of negative sentiments: 9178
Total number of positive sentiments: 2363
The total entries in the field 'airline sentiment':14640
```

The central limit theorem states that the sampling distribution of the mean of any independent, random variable will be normal or nearly normal, if the sample size is large enough.

Central Limit Theorem assumes the following conditions:

i) Sample size should be "large enough".
Since the data is large (n > 30), this condition is met.

ii) np>= 10 and nq >= 10. The number of successes and failures of the samples should be greater than or equal to 10.

Success: There are 2363 positive sentiments out of 14640 > 10. Similarly, there are 9178 negative sentiments out of 14640 > 10.

Failure: There are 12277(14640 - 2363) sentiments which are not positive > 10. Similarly, there are 5462(14640 - 9178) which are negative > 10.
Therefore, this condition is met.

iii) Independence. Sample size should be less than or equal to 10% of the population size. We have seen this already in (i), hence this condition is also met.

iv) Randomization. Assuming that the samples are randomly chosen unless and until it is specified.
Hence CLT is applicable in this scenario and it is appropriate to use a significance test.

Null Hypothesis: Proportion of positive sentiments is equal to proportion of negative sentiments.

Alternate Hypothesis: Proportion of positive sentiments is not equal to proportion of negative sentiments.

Let us try Bootstraping two sample test:

```
alpha = 0.05
empirical_diff = np.mean(neg.airline_sentiment_confidence) - np.mean(pos.airline_sentiment_confidence)

def bootstrap_sample(data,func):
    return func(np.random.choice(data,size=len(data)))


def bootstrap_replicate(data,func,size=1):

    bs_replicate = np.empty(size)

    for i in range(size):
        bs_replicate[i] = bootstrap_sample(data,func)

    return bs_replicate


shifted_pos = pos.airline_sentiment_confidence - np.mean(pos.airline_sentiment_confidence) + np.mean(df.airline_sentime
shifted_neg = neg.airline_sentiment_confidence - np.mean(neg.airline_sentiment_confidence) + np.mean(df.airline_sentime


pos_bs_replicate = bootstrap_replicate(shifted_pos,np.mean,10000)
neg_bs_replicate = bootstrap_replicate(shifted_neg,np.mean,10000)

bs_replicate_diff = pos_bs_replicate - neg_bs_replicate

p = np.sum(bs_replicate_diff >= empirical_diff)/len(bs_replicate_diff)
```

```
if p <= alpha:
    print("We can reject the null hypothesis with p-value:" + str(p))
else:
    print("We cannot reject the null hypothesis with p-value: " + str(p))
```

```
We can reject the null hypothesis with p-value:0.0
```

Let us now try the frequentist approach:

```
pos_prop = (pos_len/sentisize)
neg_prop = (neg_len/sentisize)
print("Proportion of positive sentiments (P1): %.2f" % pos_prop)
print("Proportion of negative sentiments (P2): %.2f" % neg_prop)
```

```
Proportion of positive sentiments (P1): 0.16
Proportion of negative sentiments (P2): 0.63
```

Since the inference conditions are met and sample size is large enough (n > 30), two sample Z-test will be performed.

z statistic for two sample is calculated as P1 - P2/sigma1 - sigma2

sigma = sqrt(Pc * (1-Pc) * (1/n1 + 1/n2)) where Pc is combined proportion; n1 and n2 are totals of each proportion.

```
Pc = (pos_len + neg_len)/(sentisize + sentisize)
print("Combined sample proportion is " + str(Pc))
```

```
Combined sample proportion is 0.39415983606557375
```

```
StandardError = np.sqrt(Pc * (1- Pc) * (1/sentisize + 1/sentisize))
print("Standard Error : " + str(StandardError))
```

```
Standard Error : 0.005711624850139768
```

```
from scipy import stats

z_statistic = (neg_prop - pos_prop)/StandardError


p = stats.norm.sf(abs(z_statistic))

print("p-value is : %.5f" % p )

if p <= alpha:
    print("We can reject the null hypothesis.")
else:
    print("We cannot reject the null hypothesis.")
```

```
p-value is : 0.00000
We can reject the null hypothesis.
```

After simulating the conditions for 10,000 trials, we got the same results following both the bootstrap and frequentist approaches, hereby concluding that the proportion of negative sentiments is NOT equal to positive sentiments. Hence the best recommendation that I found to balance this imbalanced data is to downsample the majority class without replacement.