Notes:

Definitions:

Historical dataset: complete dataset collected by the police department and provided

Training set: Subset of the historical dataset containing observations for which feature *VehicleSearchedIndicator* equals ‘True’.

Test set: Subset of the training set that was used to assess model performance.

Production set: complete dataset resulting collected by the web app during the week the deployed model was in production.

Production test set:

# 

# Business Conclusions

## Summary

# Results Analysis

## Model Performance

Confusion matrix for both methods?

In this section, we compare the model performance on the test set with its actual performance on production (naturally, only for those observations for which we know the true class - we have named this subset “production test set”). The results are presented in the table below and officers’ performance was also included in the last column as a baseline (i.e. what would have been the performance in case the model had not been deployed and all searches were performed). The recall is the most significant metric to measure the model performance, as defined in the third requirement of the briefing. The precision score is a byproduct of setting the classification threshold at 50%, according to our interpretation of the first requirement. We also present the percentage of searches performed, as a measure of the cost-effectiveness of the model, and accuracy, since it is a standard metric of common usage.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Model performance | | Officers’ performance |
|  | Test set (40% of the training set) | Production test set | Production test set |
| Recall | 0.5144 | 0.4942 | 1.0 |
| Precision | 0.5998 | 0.6005 | 0.359 |
| Searches [% of total] | 29% | 30% | 100% |
| Accuracy | 0.7231 | 0.7004 | 0.359 |

As we have anticipated in report 1, performance in production remained very close to its figures on the test set. Two main reasons led us to believe the model performance wouldn’t significantly change in production:

1. Since the search selection process was going to be maintained: a first selection of subjects by the police officers, and only then a search decision provided by the model; we didn’t expect the population characteristics to change much.
2. The test set on which the model performance was assessed before deployment had a considerable size (40% of the training set, amounting to 30,697 observations), which gave us confidence in the accuracy of the performance values on this set.

As we also argued in report 1, because the first requirement of the briefing led us to define a specific classification threshold, that the model dynamic assessment becomes less relevant in this case. Even so, on the figures below we compare the dynamic behavior of our model (ROC curve and True/False positives curves) on both the test set and production test set.

|  |  |
| --- | --- |
|  |  |

## Fairness

|  |  |  |
| --- | --- | --- |
|  | Compare with the same metrics on the test set. |  |

We start by noticing that there are 46 observations with class ‘Asian/Pacific’ and 10 with class ‘Indian American’ for which we know the true class. Among these, 14 ‘Asian/Pacific’ and 4 ‘Indian American’ individuals have received a positive prediction by the model. This means that it is prudent to disregard these classes when assessing the fairness requirements, since we could have an extreme value for the precision among these classes just by chance.

‘Asian/Pacific’: 14 observations predicted positive (46 in total); ‘Indian American’: 4 observations predicted positive (10 in total).

df\_obs\_classes[df\_obs\_classes.y\_pred==1].groupby('SubjectRaceCode').SubjectRaceCode.count()

df\_obs\_classes.groupby('SubjectRaceCode').SubjectRaceCode.count()

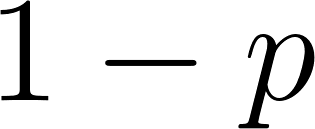
Comparison of the discrimination level between our model and the current process for the feature *SubjectRaceCode* on training and production sets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision score in the training set for feature *SubjectRaceCode* | | Precision score in the production set for feature *SubjectRaceCode* | |
|  | Current method | Our model | Current method | Our model |
| ‘White’ [%] | 34.68 | 61.99 | 36.75 | 60.48 |
| ‘Black’ [%] | 29.97 | 54.82 | 34.51 | 59.29 |
| ‘Asian/Pacific’ [%] | 29.11 | 54.76 | 26.09 | 57.14 (n.a.) |
| ‘Indian American’ [%] | 29.54 | 65.71 | 20.00 | 50.00 (n.a.) |
| Max. difference [p.p.] | 5.58 | 10.95 | 16.75 | 10.48 |
| Max. difference [%] | 16.05 | 16.67 | 45.58 | 17.32 |
| Std. Deviation [p.p.] | 2.59 | 5.45 | 7.73 | 4.69 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision score in the training set for feature *SubjectEthnicityCode* | | Precision score in the production set for feature *SubjectEthnicityCode* | |
|  | Current method | Our model | Current method | Our model |
| ‘Not applicable’ [%] | 34.99 | 60.64 | 37.98 | 61.38 |
| ‘Hispanic’ [%] | 27.71 | 57.64 | 29.84 | 55.75 |
| ‘Middle Eastern’ [%] | 27.97 | 50.00 | n.a. | n.a. |
| Max. difference [p.p.] | 7.28 | 10.64 | 8.14 | 5.63 |
| Max. difference [%] | 20.82 | 17.55 | 21.43 | 9.18 |
| Std. Deviation [p.p.] | 4.13 | 5.48 | 5.76 | 3.98 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision score in the training set for feature *SubjectSexCode* | | Precision score in the production set for feature *SubjectSexCode* | |
|  | Current method | Our model | Current method | Our model |
| ‘Male’ [%] | 33.54 | 59.73 | 36.97 | 60.55 |
| ‘Female’ [%] | 31.96 | 61.13 | 31.87 | 58.20 |
| Max. difference [p.p.] | 1.57 | 1.40 | 5.10 | 2.35 |
| Max. difference [%] | 4.70 | 2.29 | 13.80 | 3.88 |
| Std. Deviation [p.p.] | 1.11 | 0.99 | 3.61 | 1.66 |

## Population Analysis

In the three plots below we show the population distribution among protected classes comparing the training set (in blue) with the production set (orange). With the naked eye, we can see that the characteristics of the population don’t differ significantly between the two sets when we consider these classes, or at least their presence in the training and production sets is quite similar. From a statistical perspective, however, we can’t accurately say that the population is the same in both sets. To test this hypothesis - the population having the same characteristics in both sets - we may regard an observation’s class, for example being ‘Male’, as a Bernoulli trial, i.e. for a given observation there is a probability [](https://www.codecogs.com/eqnedit.php?latex=p%0) of being ‘Male’ and a probability [](https://www.codecogs.com/eqnedit.php?latex=1-p%0) of not being ‘Male’. In this framework, the number of males in each set forms a Binomial distribution, and we can test if the probability [](https://www.codecogs.com/eqnedit.php?latex=p%0) of being ‘Male’ is the same on both sets. This is done in the [Annexes](#edqwcgnup6bz) for both ‘Male’ and ‘White’ as examples, and the result, with a great level of statistical significance, is that it is not, the probability of ‘Male’ or ‘White’ is different in each of the two sets. The same is true for the remaining classes (to avoid being exhaustive limit the results for these two classes). So, our conclusion is the following: although the populations in the training set and in the production set are very similar (as can be seen just by looking at the plots), from a statistical perspective we cannot say that it is the same population (since the probability for the occurrence of each class is different, with a high level of statistical significance, in the two sets).

|  |  |  |
| --- | --- | --- |
|  |  |  |

In this section, we think of population analysis in an extended sense: not only individual characteristics, which are relevant for fairness assessment, but also the distribution of the features used by the model, which may impact its performance. Thus, we replicate the previous plots for those features that have been included in the model. Below we show the plots for *SearchAuthorizationCode* and *StatuteReason* (similar plots for the remaining variables presented in the [Annexes](#edqwcgnup6bz)).

|  |  |  |
| --- | --- | --- |
|  |  | |

Based on the [report 2 guidelines](https://docs.google.com/spreadsheets/d/1dQlGe1iFCFKrLFtYDcMWdDfm5gDhENhOJz2zuZLX52E/edit#gid=767562645) we believe the previous analysis was required. However, both for fairness and performance requirements, we don’t think this is the most significant analysis to be done. Actually, we don’t see how the population distribution is truly relevant for our purposes, we think that the finding rate among each of the features is the aspect that could have compromised both the performance and fairness requirements of our model, since the metrics that have been chosen - precision and recall - would be affected if the presence of contraband among each feature value had changed (i.e. if the probability of contraband conditional on the feature had changed), but not if the distribution of those feature values happened to be different between the train and production sets (or at least we don’t see how). In other words, even if the occurrence of different classes or values in a given feature changes significantly between the training and the production sets (i.e. even if the orange and blue bars in the plots above were very different), as long as the probability of finding contraband within each feature value doesn’t change much, then we see no reason for or model not to preserve its performance (both from a fairness and a contraband finding perspective). This alternative analysis is presented on the plots below.

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | |

whether or not the probability of

finding contraband changes when a variable changes its value.

The plots below

show us how the presence of contraband changes within the selected features

# Next Steps

In the previous report we have suggested a dynamic threshold instead of the fixed one that we ended up implementing. Actually, there is an intermediate alternative between these two, and it is so obvious that we regard with surprise that we didn’t imagine it before the model deployment time.

# Deployment Issues

## Re-deployment

We could have done it to improve the model performance, especially the fairness requirement that we were not able to address. However we don’t think that this would have been an honest approach. We get to the best possible model within the available time and it wouldn’t be honest to continue to change it during the deployment week. We believe this report to be the appropriate moment to suggest changes. Unless we had faced some unexpected behaviour (e.g. web app error) or if the model was performing in a way significantly different from what we were expecting and had presented in the previous report. Besides, once we had a working model running on heroku, any re-deployment would present some degree of risk and of loosing observations.

## Unexpected problems

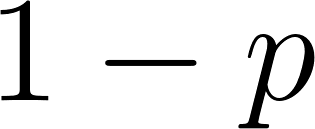
Anaconda environment was not working. We needed to spend 4 to 5 days working on the app before we had a working model.

During the deployment time we didn’t notice any problems or unexpected data.

## What would you do differently next time

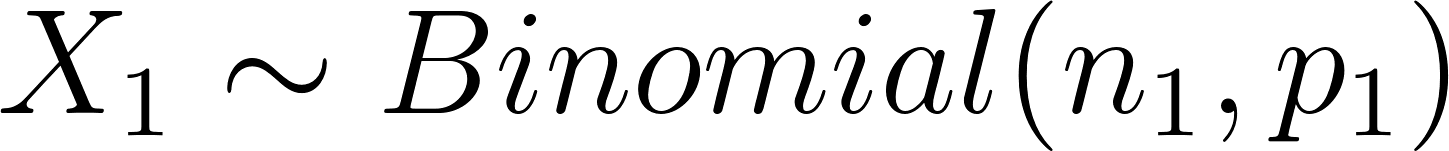
Work on a linux operating system.

# Annexes

Considering a class, e.g. being ‘Male’ or being ‘White’, the presence of this class on a given observation may be interpreted as Bernoulli trial with probability [](https://www.codecogs.com/eqnedit.php?latex=p%0), i.e. the class occurs in the observation with probability [](https://www.codecogs.com/eqnedit.php?latex=p%0) and does not occur with probability [](https://www.codecogs.com/eqnedit.php?latex=1-p%0). Repeating the trial [](https://www.codecogs.com/eqnedit.php?latex=n%0) times, the number of occurrences of this class becomes a Binomial distribution [](https://www.codecogs.com/eqnedit.php?latex=X%0) with parameters [](https://www.codecogs.com/eqnedit.php?latex=n%0) and [](https://www.codecogs.com/eqnedit.php?latex=p%0). That is, the number of occurrences of this class, [](https://www.codecogs.com/eqnedit.php?latex=X%0), is a random variable represented as

[](https://www.codecogs.com/eqnedit.php?latex=X%20%5Csim%20Binomial(n%2C%20p)%0).

If we denote the training set with the subscript 1 and the production set with the subscript 2, then the occurrence of this class in each of the sets is given by

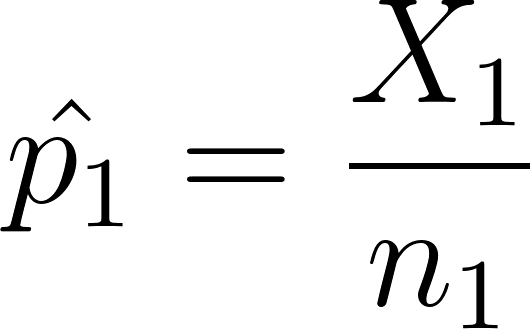
[](https://www.codecogs.com/eqnedit.php?latex=X_1%20%5Csim%20Binomial(n_1%2C%20p_1)%0)

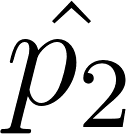
and

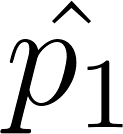
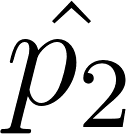
[](https://www.codecogs.com/eqnedit.php?latex=X_2%20%5Csim%20Binomial(n_2%2C%20p_2)%0),

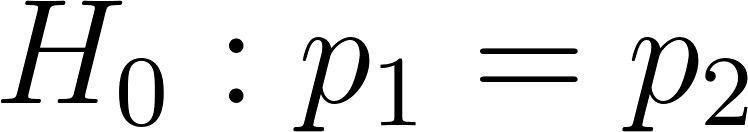
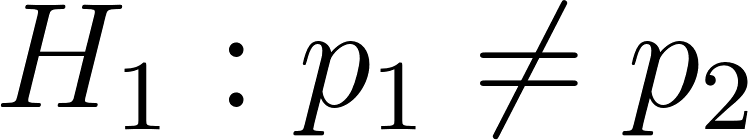
where [](https://www.codecogs.com/eqnedit.php?latex=n_1%0) is the size of the training set and [](https://www.codecogs.com/eqnedit.php?latex=p_1%0) is the probability of class occurrence for each of the observed outcomes of the training set (equivalent for [](https://www.codecogs.com/eqnedit.php?latex=n_2%0) and [](https://www.codecogs.com/eqnedit.php?latex=p_2%0) in the production set).

The most natural estimator for [](https://www.codecogs.com/eqnedit.php?latex=p_1%0) (actually, the maximum likelihood estimator) is

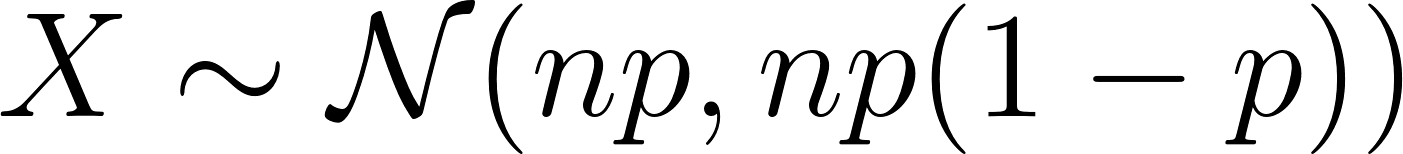
[](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bp_1%7D%20%3D%20%5Cfrac%7BX_1%7D%7Bn_1%7D%0)

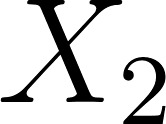
and the same for [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bp_2%7D%0).

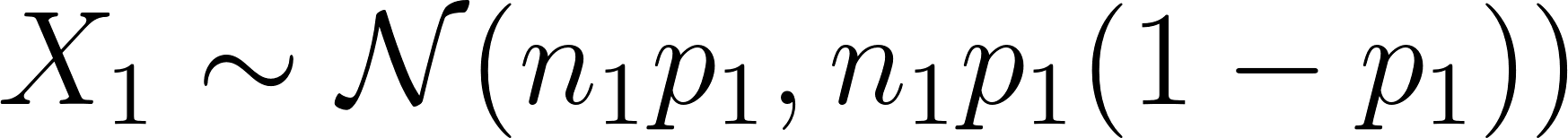
Having computed the estimates [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bp_1%7D%0) and [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7Bp_2%7D%0) we can test the hypothesis that [](https://www.codecogs.com/eqnedit.php?latex=p_1%0) is equal to [](https://www.codecogs.com/eqnedit.php?latex=p_2%0), i.e. that the probability of class occurrence on the training set is the same as in the production set. We will test

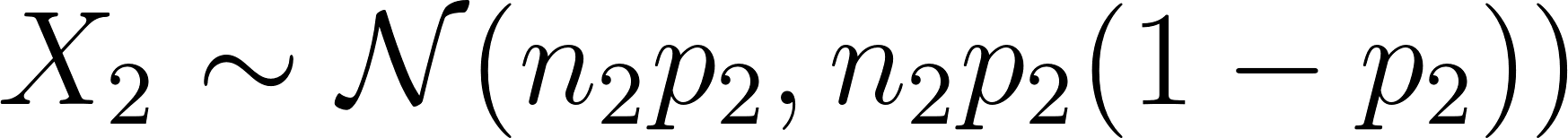
[](https://www.codecogs.com/eqnedit.php?latex=H_0%3A%20p_1%3Dp_2%0) against [](https://www.codecogs.com/eqnedit.php?latex=H_1%3A%20p_1%20%5Cneq%20p_2%0).

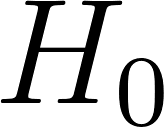
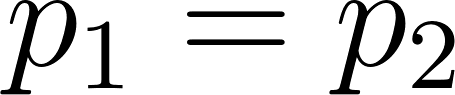
With this purpose in mind, we take advantage of the fact that when the sample size [](https://www.codecogs.com/eqnedit.php?latex=n%0) is large enough, as we consider to be the case, then the Normal distribution can be a reasonable approximation to the Binomial distribution of [](https://www.codecogs.com/eqnedit.php?latex=X%0), i.e.

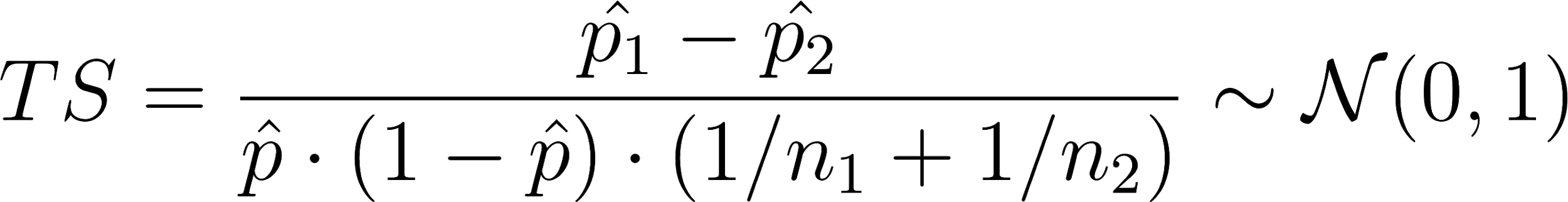
[](https://www.codecogs.com/eqnedit.php?latex=X%20%5Csim%20%5Cmathcal%7BN%7D(np%2C%20np(1-p))%0).

Using this approximation, the distributions for [](https://www.codecogs.com/eqnedit.php?latex=X_1%0) and [](https://www.codecogs.com/eqnedit.php?latex=X_2%0) become

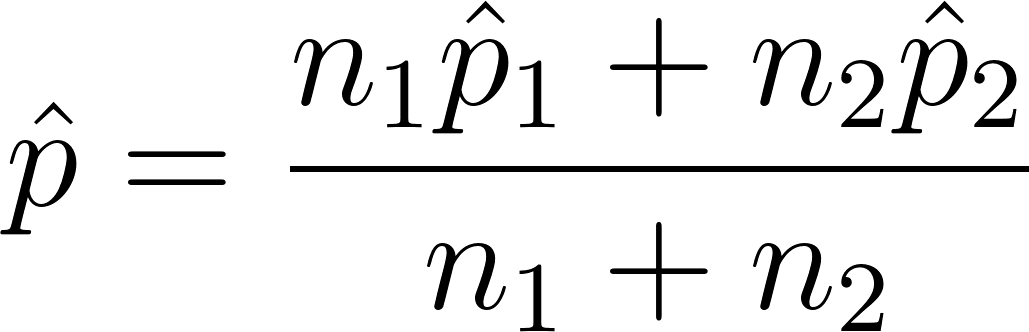
[](https://www.codecogs.com/eqnedit.php?latex=X_1%20%5Csim%20%5Cmathcal%7BN%7D(n_1%20p_1%2C%20n_1%20p_1(1-p_1))%0),

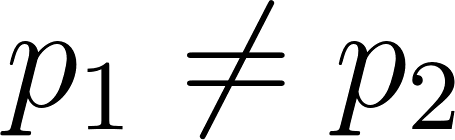
[](https://www.codecogs.com/eqnedit.php?latex=X_2%20%5Csim%20%5Cmathcal%7BN%7D(n_2%20p_2%2C%20n_2%20p_2%20(1-p_2))%0)

and, under the null hypothesis [](https://www.codecogs.com/eqnedit.php?latex=H_0%0) that [](https://www.codecogs.com/eqnedit.php?latex=p_1%20%3D%20p_2%0), we can build the test statistic [](https://www.codecogs.com/eqnedit.php?latex=TS%0) which follows a standard normal distribution

[](https://www.codecogs.com/eqnedit.php?latex=TS%20%3D%20%5Cfrac%7B%5Chat%7Bp_1%7D-%5Chat%7Bp_2%7D%7D%7B%5Chat%7Bp%7D%20%5Ccdot%20(1-%5Chat%7Bp%7D)%20%5Ccdot%20(1%2Fn_1%20%2B%201%2Fn_2)%7D%20%5Csim%20%5Cmathcal%7BN%7D%20(0%2C1)%0)

where

[](https://www.codecogs.com/eqnedit.php?latex=%5Chat%20p%3D%5Cfrac%7Bn_1%5Chat%20p_1%2Bn_2%5Chat%20p_2%7D%7Bn_1%2Bn_2%7D%0).

Now, for each class, we just have to compute the value of [](https://www.codecogs.com/eqnedit.php?latex=TS%0) (based on the occurrence of this class in the training and test sets) and see how likely this value is to happen under the null hypothesis (which led us to accept the standard normal distribution of [](https://www.codecogs.com/eqnedit.php?latex=TS%0)). If it is very unlikely, then we reject the null hypothesis and accept that [](https://www.codecogs.com/eqnedit.php?latex=p_1%20%5Cneq%20p_2%0).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class |  |  |  |  | Test result |
| ‘Male’ | 0.8168 | 0.7986 | 4.4135 | 0.000 | Reject |
| ‘White’ | 0.6974 | 0.6534 | 8.9766 | 0.000 | Reject |
| ‘Black’ | 0.2932 | 0.3362 | -8.8417 | 0.000 | Reject |
|  |  |  |  |  |  |