Predicting Conflict Escalation

STA130 Course Project

Group: Yiwei Jang, Saeed Al Shrouf, David Kayadarma
Professor: Scott Schwartz

Project Overview

Within this project, we aim to complete a thorough analysis of the transformer, xgboost, and ffnn models responsible for the prediction of conflict escalation in countries. We will be using data provided by UNICEF as well as the Fund for Peace's Fragile States Index. By taking different subsets of this data depending on different demographics and characteristics of countries, we aim to train our own regression model which would allow us to conclude how these demographics affect the performance of the three models identified above.

Statistical Terminology

- Bootstrapped Confidence Interval
    - This is a statistical method in which you resample the observed data with replacement to create multiple simulated datasets. By repeatedly drawing samples and estimating a statistic, you can deduct a certain range of values, quantifying the uncertainty around the true parameter, given a certain level of 'confidence' we have about the prediction.
- Hypothesis Testing
    - Within this statistical method, we make a prediction about a parameter of the population, after which we conduct testing to check whether or not this is an accurate inference and to what degree. To do this we use something called a P-Value, though beyond the fact that we calculate it and judge the accuracy of the prediction based on this table, there is no need to go into further depth on it.

Values of p	Inference
$p > 0.10$	No evidence against the null hypothesis.
$0.05 < p < 0.10$	Weak evidence against the null hypothesis
$0.01 < p < 0.05$	Moderate evidence against the null hypothesis
$0.05 < p < 0.001$	Good evidence against null hypothesis.
$0.001 < p < 0.01$	Strong evidence against the null hypothesis
$p < 0.001$	Very strong evidence against the null hypothesis

Library Imports and Data Loading

Python Library Imports

In [1]:
```
import pandas as pd
import numpy as np
```

Confusion Matrix Display

In [2]:
```
from sklearn.metrics import ConfusionMatrixDisplay
```

Machine Learning Models and Metrics

In [3]:
```
from sklearn import tree, model_selection
from sklearn import metrics
```

Data Visualization

In [4]:
```
import seaborn as sns
import plotly.graph_objects as go
import plotly.io as pio
import plotly.express as px
```

Statistical Modeling

In [5]:
```
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

Pandas Configuration

In [6]:
```
pd.options.mode.chained_assignment = None
```

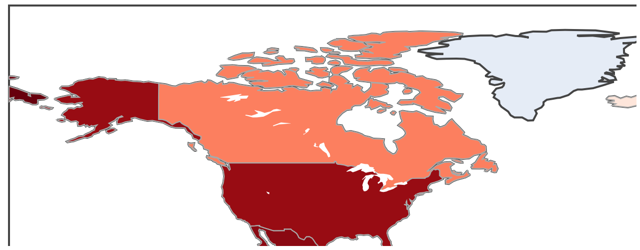
Data Importation

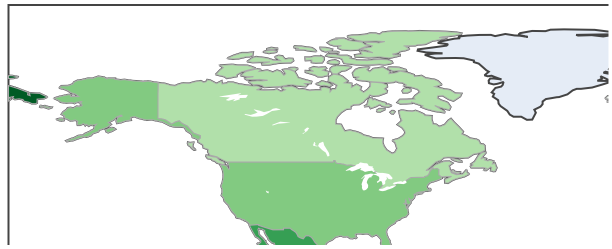
In [7]:
```
cid = pd.read_csv('country_indicators.csv')
tp = pd.read_csv('test_predictions.csv')
```

Geographical Chloropleth Visualisation of Conflict Escalation Probability Across Models

In [8]:
```
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
fig = []
for model,colors in zip(['y_pred_proba_transformer', 'y_pred_proba_ffnn', 'y_pred_p
                        ['Reds', 'Blues', 'Greens']):
    fig += [go.Figure(data = go.Choropleth(
                locations=df['iso3'], text=df['iso3'], z=df[model],
                colorscale = colors, autocolorscale=False, reversescale=False, marke
                marker_line_width=0.5, colorbar_tickprefix='', colorbar_title=model)
fig[0].show()
```

```
fig[1].show()
fig[2].show()
```





Create the Prediciton Probability "Error" results for xgboost

In [9]:
```
tp['xgboost_probability_prediction_error'] = np.abs(tp['y_true_xgboost'].astype(flo
tp[['y_true_xgboost','y_pred_proba_xgboost','xgboost_probability_prediction_error']
```

Out[9]:

	y_true_xgboost	y_pred_proba_xgboost	xgboost_probability_prediction_error
0	False	0.066500	0.066500
1	False	0.099643	0.099643
2	True	0.704086	0.295914
3	True	0.638444	0.361556
4	False	0.608380	0.608380
...	...	...	...
359	False	0.079453	0.079453
360	False	0.060189	0.060189
361	True	0.697625	0.302375
362	False	0.729246	0.729246
363	False	0.591722	0.591722

364 rows × 3 columns

Create the Prediciton Probability "Error" results for ffnn

In [10]:
```
tp['ffnn_probability_prediction_error'] = np.abs(tp['y_true_ffnn'].astype(float) -
tp[['y_true_ffnn','y_pred_proba_ffnn','ffnn_probability_prediction_error']]
```

Out[10]:

	y_true_ffnn	y_pred_proba_ffnn	ffnn_probability_prediction_error
0	False	0.409958	0.409958
1	False	0.406696	0.406696
2	False	0.545236	0.545236
3	False	0.534560	0.534560
4	True	0.538583	0.461417
...	...	...	...
359	False	0.291874	0.291874
360	False	0.300321	0.300321
361	False	0.335496	0.335496
362	False	0.324000	0.324000
363	True	0.332455	0.667545

364 rows × 3 columns

Create the Prediciton Probability "Error" results for transformer

In [11]:
```
tp['transformer_probability_prediction_error'] = np.abs(tp['y_true_transformer'].as
tp[['y_true_transformer','y_pred_proba_transformer','transformer_probability_predic
```

Out[11]:

	y_true_transformer	y_pred_proba_transformer	transformer_probability_prediction_error
0	False	0.183897	0.183897
1	False	0.267831	0.267831
2	False	0.482585	0.482585
3	False	0.187792	0.187792
4	True	0.539319	0.460681
...	...	...	...
359	False	0.182196	0.182196
360	False	0.203236	0.203236
361	False	0.527107	0.527107
362	False	0.555677	0.555677
363	True	0.565700	0.434300

364 rows × 3 columns

Prediction Classification "Correctness" Results

- Binary Classification Predictions
  - Possible scope of results includes False Positive, True Positive, False Negative, False Positive. In this case True and False represent whether the prediction was correct or not, with Positive and Negative representing whether the country in question experienced escalation - Positive meaning it did.
  - Models Inspected:
    - transformer
    - xgboost
    - ffnn

Axis Representations:

- The Y-Axis represents whether or not conflict has in fact occurred.
- The X-Axis represents whether or not the model predicted conflict to occur.

A threshold is a value separating the predicted outcomes made by a model into different classes. Due to the existence class imbalances, bias may be introduced. This bias, however, may be mitigated by adjusting the threshold for the model. In this case, our adjustment results in the rate of prediction of escalation to be around 14.8%.

Unfortunately, adjusting thresholds does not come without drawbacks, paticularly the fact that the model may begin to decline in accuracy - meaning that it may make misclassifications - resulting in False Positives/Negatives. Depending on the value of the threshold there is chance that the rate of either of these is increased. In our case, a False Negative is a catastrophic outcome - failing to predict the conflict escalation in a country would have potentially dire consequences. Therefore, the chosen threshold for each model has a lot of significance as we balanced minimizing errors whilst pushing our rate of prediction of escalation to ≈14.8%.

Transformer *(0.63 Threshold)*

In [12]:
```
threshold_transformer = 0.63

tp['transformer_classifcation_performance_outcome'] = None
tp['xgboost_classifcation_performance_outcome'] = None
tp['ffnn_classifcation_performance_outcome'] = None

tmp = tp['transformer_classifcation_performance_outcome'].copy()
TP_pos_pred_correct = tp.y_true_transformer & (tp.y_pred_proba_transformer>threshol
tmp[TP_pos_pred_correct] = "correctly predicted escalation"
TN_neg_pred_correct = (~tp.y_true_transformer) & (tp.y_pred_proba_transformer<=thre
tmp[TN_neg_pred_correct] = "correctly predicted no escalation"
FP_pos_pred_wrong = (~tp.y_true_transformer) & (tp.y_pred_proba_transformer>thresho
tmp[FP_pos_pred_wrong] = "wrongly predicted escalation"
FN_neg_pred_wrong = tp.y_true_transformer & (tp.y_pred_proba_transformer<=threshold
tmp[FN_neg_pred_wrong] = "wrongly predicted no escalation"

tp['transformer_classifcation_performance_outcome'] = tmp
tp[['y_true_transformer','y_pred_transformer','transformer_classifcation_performanc
tp['transformer_correctness']=((tp.y_true_transformer & (tp.y_pred_proba_transforme
    ~tp.y_true_transformer) & (tp.y_pred_proba_transformer<=threshold_transformer))
((tp['transformer_classifcation_performance_outcome']=='correctly predicted escalat
    tp['transformer_classifcation_performance_outcome']=='wrongly predicted escalat
```

Out[12]: 0.15384615384615385

xgboost *(0.71 Threshold)*

In [13]:
```
threshold_xgboost=0.71
tmp = tp['xgboost_classifcation_performance_outcome'].copy()
TP_pos_pred_correct = tp.y_true_xgboost & (tp.y_pred_proba_xgboost>threshold_xgboos
tmp[TP_pos_pred_correct] = "correctly predicted escalation"
TN_neg_pred_correct = (~tp.y_true_xgboost) & (tp.y_pred_proba_xgboost<=threshold_xg
tmp[TN_neg_pred_correct] = "correctly predicted no escalation"
FP_pos_pred_wrong = (~tp.y_true_xgboost) & (tp.y_pred_proba_xgboost>threshold_xgboo
tmp[FP_pos_pred_wrong] = "wrongly predicted escalation"
```

```
FN_neg_pred_wrong = tp.y_true_xgboost & (tp.y_pred_proba_xgboost<=threshold_xgboost
tmp[FN_neg_pred_wrong] = "wrongly predicted no escalation"

tp['xgboost_classifcation_performance_outcome'] = tmp
tp[['y_true_xgboost','y_pred_xgboost','xgboost_classifcation_performance_outcome']]
tp['xgboost_correctness']=((tp.y_true_xgboost & (tp.y_pred_proba_xgboost>threshold_
    ~tp.y_true_xgboost) & (tp.y_pred_proba_xgboost<=threshold_xgboost))
((tp['xgboost_classifcation_performance_outcome']=='correctly predicted escalation'
    tp['xgboost_classifcation_performance_outcome']=='wrongly predicted escalation'
```

Out[13]: 0.14560439560439561

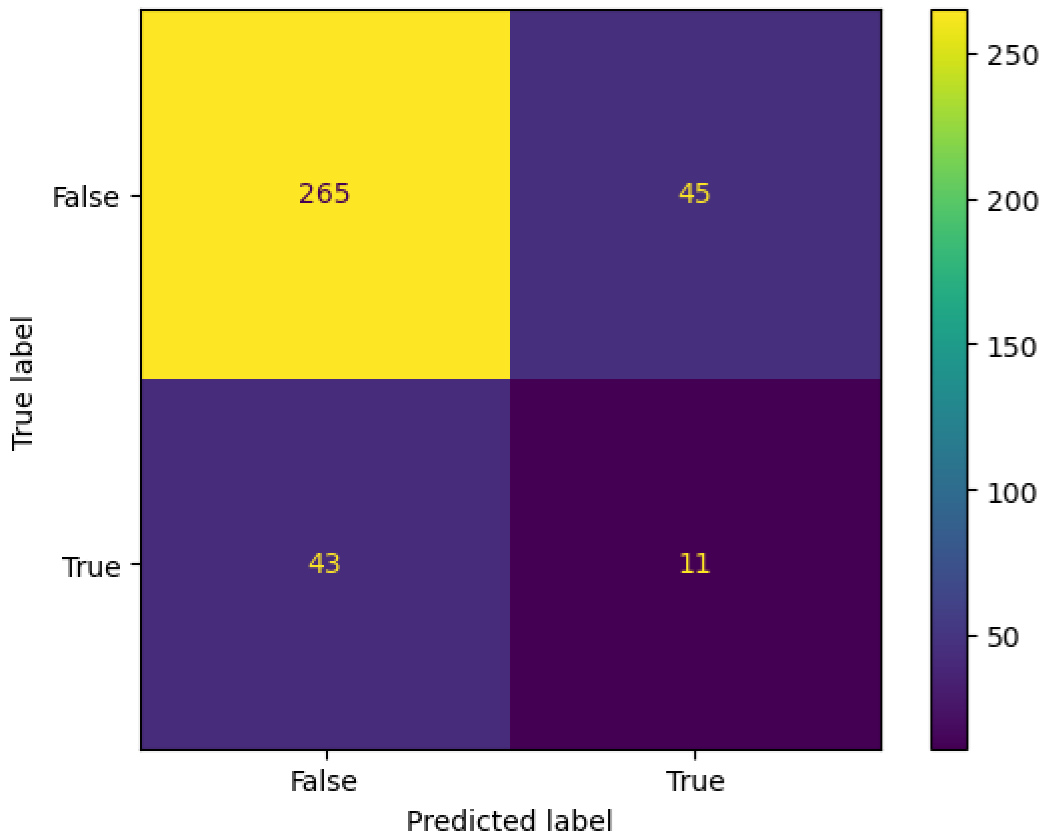
ffnn *(0.54 Threshold)*

In [14]:
```
threshold_ffnn = 0.54
tmp = tp['ffnn_classifcation_performance_outcome'].copy()
TP_pos_pred_correct = tp.y_true_ffnn & (tp.y_pred_proba_ffnn>threshold_ffnn)
tmp[TP_pos_pred_correct] = "correctly predicted escalation"
TN_neg_pred_correct = (~tp.y_true_ffnn) & (tp.y_pred_proba_ffnn<=threshold_ffnn)
tmp[TN_neg_pred_correct] = "correctly predicted no escalation"
FP_pos_pred_wrong = (~tp.y_true_ffnn) & (tp.y_pred_proba_ffnn>threshold_ffnn)
tmp[FP_pos_pred_wrong] = "wrongly predicted escalation"
FN_neg_pred_wrong = tp.y_true_ffnn & (tp.y_pred_proba_ffnn<=threshold_ffnn)
tmp[FN_neg_pred_wrong] = "wrongly predicted no escalation"

tp['ffnn_classifcation_performance_outcome'] = tmp
tp[['y_true_ffnn','y_pred_ffnn','ffnn_classifcation_performance_outcome']]
tp['ffnn_correctness']=((tp.y_true_ffnn & (tp.y_pred_proba_ffnn>threshold_ffnn))|(
    ~tp.y_true_ffnn) & (tp.y_pred_proba_ffnn<=threshold_ffnn))
((tp['ffnn_classifcation_performance_outcome']=='correctly predicted escalation').s
    tp['ffnn_classifcation_performance_outcome']=='wrongly predicted escalation').s
```

Out[14]: 0.14560439560439561

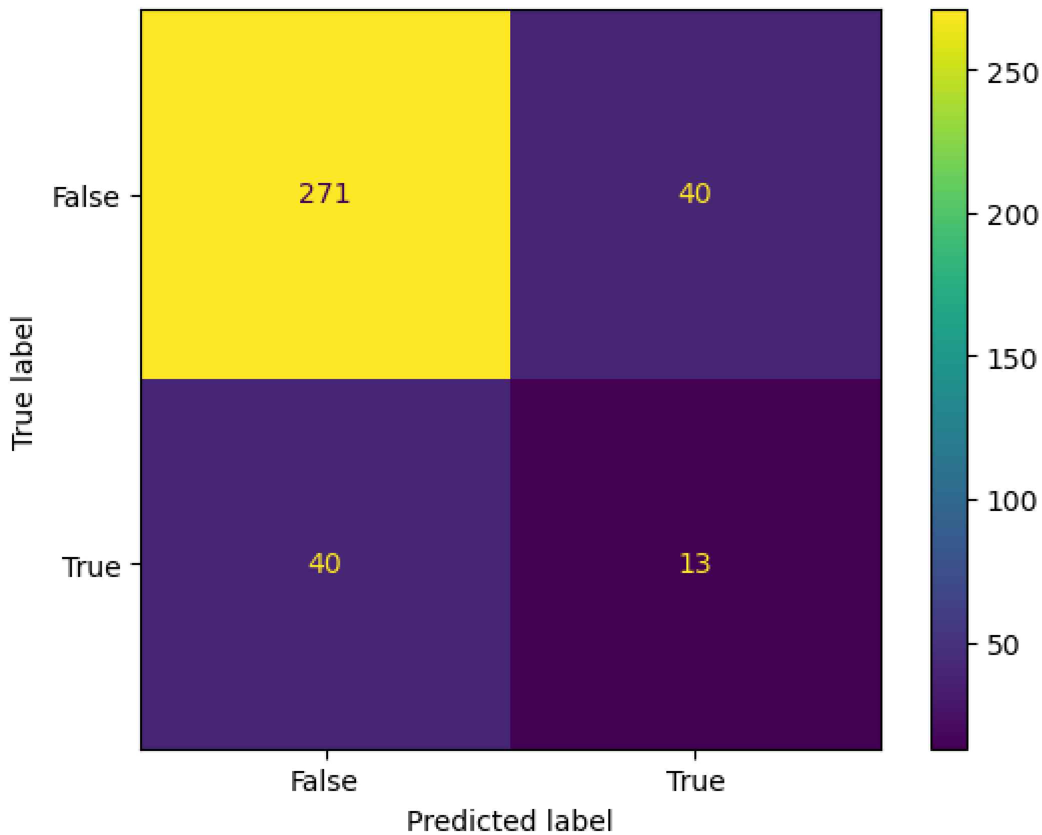
Transformer Threshold Confusion Matrix

In [15]:
```
threshold = threshold_transformer
_ = ConfusionMatrixDisplay.from_predictions(tp.y_true_transformer, tp.y_pred_proba_
```



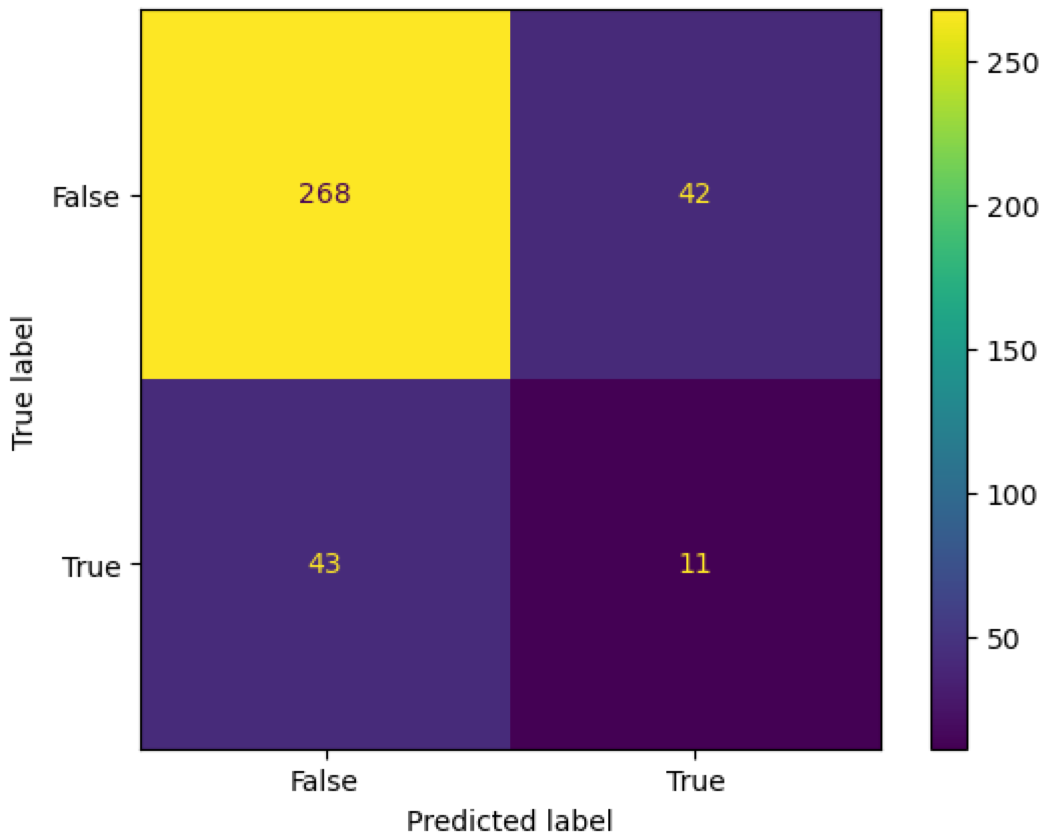
xgboost Threshold Confusion Matrix

In [16]:
```
threshold = threshold_xgboost
_ = ConfusionMatrixDisplay.from_predictions(tp.y_true_xgboost, tp.y_pred_proba_xgbo
```



ffnn Threshold Confusion Matrix

In [17]:
```
threshold = threshold_ffnn
_ = ConfusionMatrixDisplay.from_predictions(tp.y_true_ffnn, tp.y_pred_proba_ffnn>th
```



Data Subset Characteristic: Countries with a GDP Per-Capita Greater than the Median GDP Per-Capita

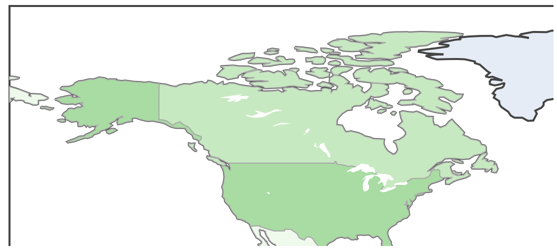
Tuple output: (Num of Subset Countries, Num of Countries outside subset)

In [18]:
```
cid['GDP-per-capita']=cid['sowc_social-protection-and-equity__gdp-per-capita-curren
    'sowc_social-protection-and-equity__gdp-per-capita-current-us-2010-2019-r_botto
cid['GDP-per-capita_values'] = cid['sowc_social-protection-and-equity__gdp-per-capi
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
high_GDP = df[df['GDP-per-capita'] == True]
((df['GDP-per-capita'] == True).sum(),(df['GDP-per-capita'] == False).sum())
```

Out[18]: (165, 199)

Choropleth Data Visualisation Based on Country GDP Per-Capita

In [19]:
```
pio.renderers.default = 'notebook'
progress_indicator = 'sowc_social-protection-and-equity__gdp-per-capita-current-us-
go.Figure(data = go.Choropleth(
          locations=cid['iso3'], text=cid['iso3'], z=cid[progress_indicator],
          colorscale = 'Greens', autocolorscale=False, reversescale=False, marker_l
          marker_line_width=0.5, colorbar_tickprefix='', colorbar_title="GDP per ca
```



Changes in GDP Per-Capita Bootstrap Confidence Interval

Comparing Chosen Data Subset to Remaining Data

Relevant Analysis Objects	Objects Chosen
Model	ffnn
-	-
Data Subset	Countries with GDP Per-Capita > GDP Per-Capita Median
-	-
Method of Analysis	Bootstrapped Confidence Interval

Sub-Set Bootstrapped Confidence Interval (95% Interval)

In [20]:
```
np.random.seed(4)
reps = 1000
df_copy = df[['ffnn_probability_prediction_error','GDP-per-capita']].copy()
bootstrapped_sample_difference = np.zeros(reps)
bootstrapped_value_copy=df.ffnn_probability_prediction_error.values.copy()
```

```
for i in range(reps):

    bootstrapped_value_copy[df['GDP-per-capita'] == True] = df['ffnn_probability_pr
        df['GDP-per-capita'] == True].sample(frac=1, replace=True).values
    bootstrapped_value_copy[df['GDP-per-capita'] == False] = df['ffnn_probability_p
        df['GDP-per-capita'] == False].sample(frac=1, replace=True).values
    df_copy['ffnn_probability_prediction_error']=bootstrapped_value_copy
    bootstrapped_sample_difference[i] = np.diff(df_copy.groupby('GDP-per-capita').f

confidence_interval = np.quantile(bootstrapped_sample_difference, [0.025,0.975])

f'Confidence Interval: Lower Bound ({confidence_interval[0]}) Upper Bound ({confide
```

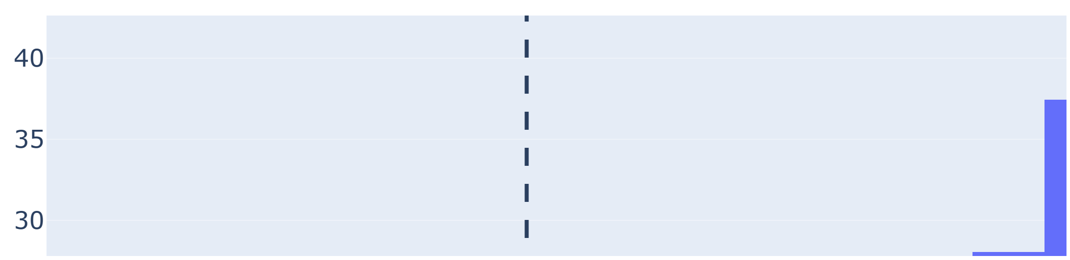
Out[20]:

```
'Confidence Interval: Lower Bound (-0.1345224642740977) Upper Bound (-0.0924704789
9311713)'
```

Visualising Histogram of 95% Confidence Interval

In [21]:

```
fig = go.Figure()
fig.add_trace(go.Histogram(x=bootstrapped_sample_difference,histnorm='probability d
fig.add_vline(x=np.quantile(bootstrapped_sample_difference,0.025),line_dash = 'dash
fig.add_vline(x=np.quantile(bootstrapped_sample_difference,0.975),line_dash = 'dash
```



ffnn Internal Performance Changes:

Comparing Chosen Data Subset to Remaining Data

Relevant Analysis Objects	**Objects Chosen**
Model	ffnn
-	-
Data Subset	Countries with GDP Per-Capita > GDP Per-Capita Median
-	-
H_0	Data Subset == Total Countries - Data Subset
-	-
H_1	H_0 is False

Probability Error Prediction Hypothesis Test

In [22]:
```
df_copy = df[['ffnn_probability_prediction_error','GDP-per-capita']].copy()
np.random.seed(4)
reps = 1000
difference_in_mean = np.zeros(reps)
for i in range(reps):
    shuffled_labels = df['GDP-per-capita'].sample(frac=1)

    df_copy['GDP-per-capita'] = shuffled_labels.values
    difference_in_mean[i] = np.diff(df_copy.groupby('GDP-per-capita').mean().ffnn_p
```

Calculating P-Value for Subset Test

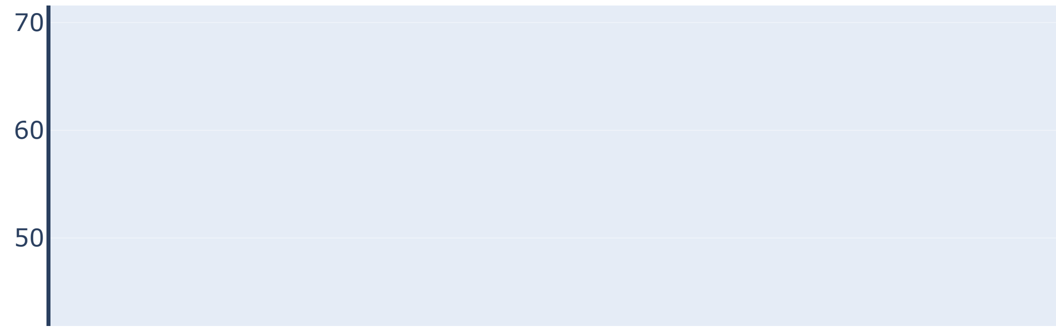
In [23]:
```
(abs(difference_in_mean)-abs(np.diff(df[['ffnn_probability_prediction_error','GDP-p
    'GDP-per-capita').mean().ffnn_probability_prediction_error)[0]) >= 0).sum()/rep
```

Out[23]:
```
0.0
```

Given a p-value this low, from the table shown prior we can conclude to have very strong evidence against the null hypothesis

Graphing Histogram of Subset Test Results

In [24]:
```
fig = px.histogram(pd.DataFrame({'simulated mean difference': difference_in_mean}),
fig.add_vline(x=np.diff(df[['ffnn_probability_prediction_error','GDP-per-capita']].
    'GDP-per-capita').mean().ffnn_probability_prediction_error)[0])
```



xgboost Internal Performance Changes:

Comparing Chosen Data Subset to Remaining Data

Relevant Analysis Objects	Objects Chosen
Model	xgboost
-	-
Data Subset	Countries with GDP Per-Capita > GDP Per-Capita Median
-	-
H_0	Data Subset == Total Countries - Data Subset
-	-
H_1	H_0 is False

Permutation Shuffle Test Using Subset of Data

In [25]:
```
df_copy = df[['xgboost_correctness','GDP-per-capita']].copy()
np.random.seed(4)
reps = 1000
difference_in_mean = np.zeros(reps)
for i in range(reps):
    shuffled_labels = df['GDP-per-capita'].sample(frac=1)

    df_copy['GDP-per-capita'] = shuffled_labels.values
    difference_in_mean[i] = np.diff(df_copy.groupby('GDP-per-capita').mean().xgboos
```

Calculating P-Value for Hypothesis Test

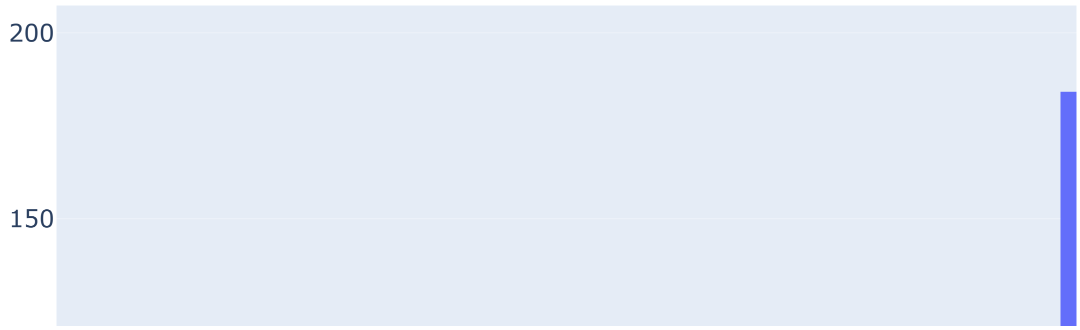
In [26]:
```
(abs(difference_in_mean)-abs(np.diff(df[['xgboost_correctness','GDP-per-capita']].g
    'GDP-per-capita').mean().xgboost_correctness)[0]) >= 0).sum()/reps
```

Out[26]: 0.0

Given a p-value this low, from the table shown prior we can conclude to have very strong evidence against the null hypothesis

Visualising Hypothesis Test Using Histogram

In [27]:
```
fig = px.histogram(pd.DataFrame({'difference_between_high_low': difference_in_mean}
fig.add_vline(x=np.diff(df[['xgboost_correctness','GDP-per-capita']].groupby(
    'GDP-per-capita').mean().xgboost_correctness)[0])
```



Data Subset Characteristic: Countries with a Child Dependency Ratio Greater Than Median Ratio

Tuple output: (Num of Subset Countries, Num of Countries outside subset)

In [28]:
```
cid['child_dependency_ratio']=cid['sowc_demographics__dependency-ratio-2021_child-d
    'sowc_demographics__dependency-ratio-2021_child-dependency-ratio_2021-0'].quant
cid['child_dependency_ratio_values']=cid['sowc_demographics__dependency-ratio-2021_
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
high_dr = df[df['child_dependency_ratio'] == True]
((df['child_dependency_ratio'] == True).sum(),(df['child_dependency_ratio'] == Fals
```

Out[28]: (192, 172)

Choropleth Data Visualisation Based on Country Child Dependency Ratio

In [65]:
```
pio.renderers.default = 'notebook'
progress_indicator = 'sowc_demographics__dependency-ratio-2021_child-dependency-rat
go.Figure(data = go.Choropleth(
          locations=cid['iso3'], text=cid['iso3'], z=cid[progress_indicator],
```

```
        colorscale = 'Blues', autocolorscale=False, reversescale=True, marker_lin
        marker_line_width=0.5, colorbar_tickprefix='', colorbar_title="Child depe
```

ffnn Internal Performance Changes:

Comparing Chosen Data Subset to Remaining Data

Relevant Analysis Objects	Objects Chosen
Model	ffnn
-	-
Data Subset	Countries with Child Dependency Ratio > Child Dependency Ratio Median
-	-
H_0	Data Subset == Total Countries - Data Subset
-	-
H_1	H_0 is False

Permutation Shuffle Test Using Subset of Data

In [30]:
```
df_copy = df[['ffnn_probability_prediction_error','child_dependency_ratio']].copy()
np.random.seed(4)
reps = 1000
difference_in_mean = np.zeros(reps)
for i in range(reps):
    shuffled_labels = df['child_dependency_ratio'].sample(frac=1)

    df_copy['child_dependency_ratio'] = shuffled_labels.values
    difference_in_mean[i] = np.diff(df_copy.groupby('child_dependency_ratio').mean(
```

Calculating Subset P-Value

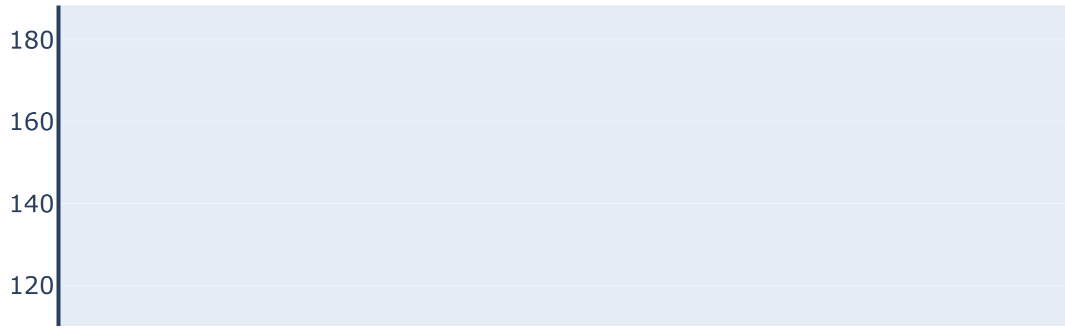
In [31]:
```
(abs(difference_in_mean)-abs(np.diff(df[['ffnn_probability_prediction_error','child
    'child_dependency_ratio').mean().ffnn_probability_prediction_error)[0]) >= 0).s
```

Out[31]:
```
0.0
```

Given a p-value this low, from the table shown prior we can conclude to have very strong evidence against the null hypothesis

Visualising Hypothesis Test Using Histogram

In [32]:
```
fig = px.histogram(pd.DataFrame({'difference_between_high_low': difference_in_mean}
fig.add_vline(x=np.diff(df[['ffnn_correctness','child_dependency_ratio']].groupby(
    'child_dependency_ratio').mean().ffnn_correctness)[0])
```



ffnn Prediction Probability "Error" Result Analysis

Comparing Chosen Data Subset to Remaining Data

Relevant Analysis Objects	Objects Chosen
Model	ffnn
-	-
Data Subset	Countries with Child Dependency Ratio > Child Dependency Ratio Median
-	-
Method of Analysis	Bootstrapped Confidence Interval

Sub-Set Bootstrapped Confidence Interval (95% Interval)

In [33]:
```
np.random.seed(4)
df_copy = df[['ffnn_probability_prediction_error','child_dependency_ratio']].copy()
bootstrapped_sample_difference = np.zeros(reps)
```

```
bootstrapped_value_copy=df.ffnn_probability_prediction_error.values.copy()
for i in range(reps):

    bootstrapped_value_copy[df['child_dependency_ratio'] == True] = df['ffnn_probab
        df['child_dependency_ratio'] == True].sample(frac=1, replace=True).values
    bootstrapped_value_copy[df['child_dependency_ratio'] == False] = df['ffnn_proba
        df['child_dependency_ratio'] == False].sample(frac=1, replace=True).values
    df_copy['ffnn_probability_prediction_error']=bootstrapped_value_copy
    bootstrapped_sample_difference[i] = np.diff(df_copy.groupby('child_dependency_r
confidence_interval = np.quantile(bootstrapped_sample_difference,[0.025,0.975])
f'Confidence Interval: Lower Bound ({confidence_interval[0]}) Upper Bound ({confide
```

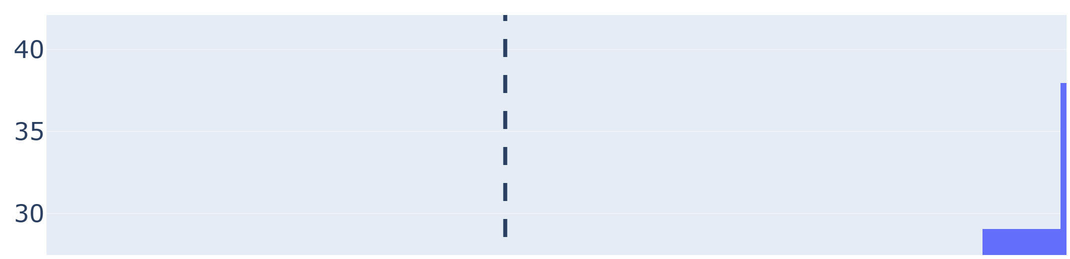
Out[33]:

```
'Confidence Interval: Lower Bound (0.12377195375753997) Upper Bound (0.16247826404
972146)'
```

Visualising Histogram of 95% Confidence Interval

In [34]:

```
fig = go.Figure()
fig.add_trace(go.Histogram(x=bootstrapped_sample_difference,histnorm='probability d
fig.add_vline(x=np.quantile(bootstrapped_sample_difference,0.025),line_dash = 'dash
fig.add_vline(x=np.quantile(bootstrapped_sample_difference,0.975),line_dash = 'dash
```



xgboost Internal Performance Changes:

Comparing Chosen Data Subset to Remaining Data

Relevant Analysis Objects	Objects Chosen
Model	xgboost
-	-
Data Subset	Countries with Child Dependency Ratio > Child Dependency Ratio Median
-	-
H_0	Data Subset == Total Countries - Data Subset
-	-
H_1	H_0 is False

Permutation Shuffle Test Using Subset of Data

In [35]:
```
df_copy = df[['xgboost_correctness','child_dependency_ratio']].copy()
np.random.seed(4)
reps = 1000
difference_in_mean = np.zeros(reps)
for i in range(reps):
    shuffled_labels = df['child_dependency_ratio'].sample(frac=1)

    df_copy['child_dependency_ratio'] = shuffled_labels.values
    difference_in_mean[i] = np.diff(df_copy.groupby('child_dependency_ratio').mean(
```

Calculating P-Value for Hypothesis Test

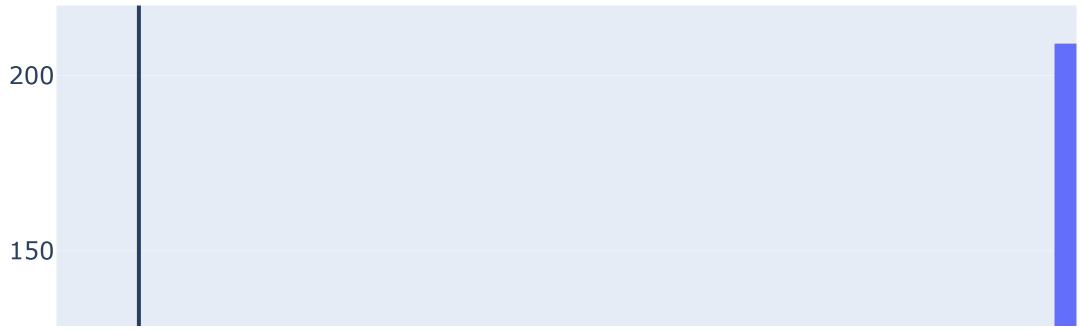
In [36]:
```
(abs(difference_in_mean)-abs(np.diff(df[['xgboost_correctness','child_dependency_ra
    'child_dependency_ratio').mean().xgboost_correctness)[0]) >= 0).sum()/reps
```

Out[36]:
```
0.003
```

This p-value of 0.003 indicates strong evidence against the null hypothesis - as mentioned in the prior table.

Visualising Hypothesis Test Using Histogram

In [37]:
```
fig = px.histogram(pd.DataFrame({'difference_between_high_low': difference_in_mean}
fig.add_vline(x=np.diff(df[['xgboost_correctness','child_dependency_ratio']].groupb
    'child_dependency_ratio').mean().xgboost_correctness)[0])
```



Designing Our Regression Model

Discovering More Relevant Features

Thus far, we have concluded that both a country's GDP Per-Capita and Child Dependency Ratio are both relevant in determining whether or not it is likely to experience escalation. It is now time to find more features which will be useful to use as our predictor variables.

features are the independent variables or predictors that are initially included in the model.

Data Subset Characteristic: Countries with Household Basic Drinking Water Services Greater Than the Median

Tuple output: (Num of Subset Countries, Num of Countries outside subset)

In [38]:
```
cid['water']=cid['sowc_wash__households-2020_at-least-basic-drinking-water-services
    'sowc_wash__households-2020_at-least-basic-drinking-water-services_total'].quan
cid['water_values']=cid['sowc_wash__households-2020_at-least-basic-drinking-water-s
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
high_dr = df[df['water'] == True]
((df['water'] == True).sum(),(df['water'] == False).sum())
```

Out[38]: (161, 203)

Data Subset Characteristic: Countries an Under-18 Population (Thousands) Greater Than the Median

Tuple output: (Num of Subset Countries, Num of Countries outside subset)

In [39]:
```
cid['pop18']=cid['sowc_demographics__population-thousands-2021_under-18'] > cid[
    'sowc_demographics__population-thousands-2021_under-18'].quantile(0.5)
cid['pop18_values']=cid['sowc_demographics__population-thousands-2021_under-18']
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
high_18 = df[df['pop18'] == True]
((df['pop18'] == True).sum(),(df['pop18'] == False).sum())
```

Out[39]: (238, 126)

Data Subset Characteristic: Countries with Annual Rate of Reduction in Under-Five Mortality Rate Greater Than the Median

Tuple output: (Number of Subset Countries, Number of Countries outside Subset)

In [40]:
```
cid['reduction_mortality5']=cid['sowc_child-mortality__annual-rate-of-reduction-in-
    'sowc_child-mortality__annual-rate-of-reduction-in-under-five-mortality-rate_20
cid['reduction_mortality5_values']=cid['sowc_child-mortality__annual-rate-of-reduct
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
high_reduction_mortality5 = df[df['reduction_mortality5'] == True]
((df['reduction_mortality5'] == True).sum(),(df['reduction_mortality5'] == False).s
```

Out[40]: (210, 154)

Data Subset Characteristic: Countries with Immunization Coverage for Vaccine-Preventable Diseases Greater Than the Median

Tuple output: (Number of Subset Countries, Number of Countries outside Subset)

In [41]:
```
cid['coverage_immunization']=cid['sowc_child-health__intervention-coverage_immuniza
    'sowc_child-health__intervention-coverage_immunization-for-vaccine-preventable-
cid['coverage_immunization_values']=cid['sowc_child-mortality__annual-rate-of-reduc
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
high_reduction_mortality5 = df[df['coverage_immunization'] == True]
((df['coverage_immunization'] == True).sum(),(df['coverage_immunization'] == False)
```

Out[41]: (155, 209)

Data Subset Characteristic: Countries with Malnutrition Among School-Aged Children (5-19) Greater Than the Median

Tuple output: (Number of Subset Countries, Number of Countries outside Subset)

In [42]:
```
cid['thin']=cid['sowc_nutrition-newborns-preschool-school-age-children-women-and-ho
    'sowc_nutrition-newborns-preschool-school-age-children-women-and-households__ma
cid['thin_values']=cid['sowc_nutrition-newborns-preschool-school-age-children-women
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
```

```
high_reduction_mortality5 = df[df['thin'] == True]
((df['thin'] == True).sum(),(df['thin'] == False).sum())
```

Out[42]: (181, 183)

Data Subset Characteristic: Countries with Female Educational Attainment (Upper Secondary or Higher) Greater Than the Median

Tuple output: (Number of Subset Countries, Number of Countries outside Subset)

In [43]:
```
cid['female_education']=cid['sowc_women-s-economic-empowerment__educational-attainm
    'sowc_women-s-economic-empowerment__educational-attainment-2008-2021-r_upper-se
cid['female_education_values']=cid['sowc_women-s-economic-empowerment__educational-
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
high_reduction_mortality5 = df[df['female_education'] == True]
((df['female_education'] == True).sum(),(df['female_education'] == False).sum())
```

Out[43]: (166, 198)

Data Subset Characteristic: Countries with New Internal Displacements of Individuals Under-18 (Rural Areas) Greater Than the Median

Tuple output: (Number of Subset Countries, Number of Countries outside Subset)

In [44]:
```
cid['displace18']=cid['sowc_migration__new-internal-displacements-2021_under-18-ru'
    'sowc_migration__new-internal-displacements-2021_under-18-ru'].quantile(0.5)
cid['displace18_values']=cid['sowc_migration__new-internal-displacements-2021_under
df = tp.merge(cid,left_on='iso3', right_on='iso3', how='inner')
high_reduction_mortality5 = df[df['displace18'] == True]
((df['displace18'] == True).sum(),(df['displace18'] == False).sum())
```

Out[44]: (201, 163)

Building the Regression Model

We chose to create a backwards stepwise regression model. This approach essentially means that we begin by first using all of the predictor variables we identified above, and iteratively remove the variable with the least-significant contribution towards variation in the dependent variable - Predicting Conflict Escalation.

Dataframe Preparation

In [45]:
```
df_cols = pd.DataFrame(df.dtypes, columns=('coldtype',)).reset_index().rename(colum
df_cols['coldtype'] = df_cols['coldtype'].astype('string')
df['fsi_rank'] = df['fsi_rank'].astype('string').str.replace(r'\D', '', regex=True)
num_vars = df_cols.query("coldtype=='float64'")['colname'].values
df['fsi_rank']=df['fsi_rank'].astype(int)
```

In [46]:
```
df['fsi_rank_cat'] = np.select([df['fsi_rank'] <= df['fsi_rank'].median(), df['fsi_
```

One-hot Encoding Dataframe

This is a technique used to convert categorical variables to binary vectors - allowing the model to interpret the data from a numerical perspective.

In [47]:
```
import itertools
def one_hot(df, cols):
    """ One-hot encode given `cols` and add as new columns
        to `df`

        Returns tuple of `df` with new columns and list of
        new column names.
    """
    new_cols = list()
    new_col_names = list()
    for each in cols:
        dummies = pd.get_dummies(df[each], prefix=each)
        new_cols.append(dummies)
        new_col_names.append(dummies.columns.values)

    df = pd.concat([df]+new_cols, axis=1)
    new_col_names = list(itertools.chain.from_iterable(new_col_names))
    return df, new_col_names
cat_vars = ['fsi_category', 'hdr_hdicode', 'hdr_region',
            'wbi_income_group', 'wbi_lending_category','wbi_other_(emu_or_hipc)','f
con_vars = ['GDP-per-capita_values','child_dependency_ratio_values','water_values',
            'reduction_mortality5_values','coverage_immunization_values','thin_valu
df_oh, oh_cols = one_hot(df, cat_vars)
df_oh = df_oh.drop(columns=cat_vars)
df_co = df[con_vars]
df_oh[['transformer_probability_prediction_error', 'ffnn_probability_prediction_err
```

Out[47]:

	transformer_probability_prediction_error	ffnn_probability_prediction_error	xgboost_prol
0	0.183897	0.409958	
1	0.267831	0.406696	
2	0.482585	0.545236	
3	0.187792	0.534560	
4	0.460681	0.461417	
...	...	...	
359	0.182196	0.291874	
360	0.203236	0.300321	
361	0.527107	0.335496	
362	0.555677	0.324000	
363	0.434300	0.667545	

364 rows × 28 columns

Hypothesis Test for Difference in Prediction Probability "Error" Result Analysis of ffnn vs fsi_rank_cat:

Relevant Analysis Objects	Objects Chosen
Model	ffnn
-	-
Data Set	fsi_rank_cat
-	-
H_0	ffnn Prediction Probability "Error" == fsi_rank_cat Prediction Probability "Error"
-	-
H_1	H_0 is False

Permutation Shuffle Test Using Subset of Data

In [48]:
```
df_copy = df[['ffnn_probability_prediction_error','fsi_rank_cat']].copy()
np.random.seed(4)
reps = 1000
difference_in_mean = np.zeros(reps)
for i in range(reps):
    shuffled_labels = df['fsi_rank_cat'].sample(frac=1)
```

```
    df_copy['fsi_rank_cat'] = shuffled_labels.values
    difference_in_mean[i] = np.diff(df_copy.groupby('fsi_rank_cat').mean().ffnn_pro
```

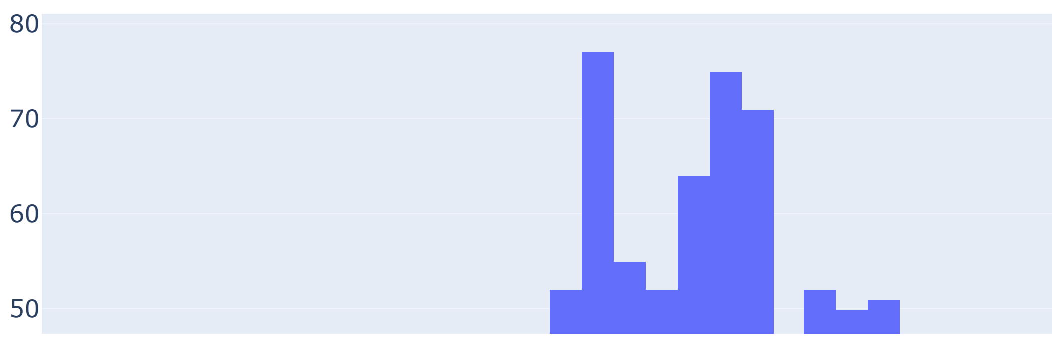
Calculating P-Value for Hypothesis Test

In [49]:
```
(abs(difference_in_mean)-abs(np.diff(df[['ffnn_probability_prediction_error','fsi_r
    'fsi_rank_cat').mean().ffnn_probability_prediction_error)[0]) >= 0).sum()/reps
```

Out[49]: 0.0

Visualising Hypothesis Test Using Histogram

In [50]:
```
fig = px.histogram(pd.DataFrame({'simulated mean difference': difference_in_mean}),
fig.add_vline(x=np.diff(df[['ffnn_probability_prediction_error','fsi_rank_cat']].gr
    'fsi_rank_cat').mean().ffnn_probability_prediction_error)[0])
```



Rocky Ethical Terrain of fsi_rank in International Relations

Through the hypothesis test which we have conducted, it is clear that there is a substantial difference in the high and low fsi_rank subsets. However, we are concerned with the implementation of this feature into considerations. Firstly, there is a possibility of fsi_rank introducing further stereotyping of countries, which could lead to negative perception of them being reinforced.

Due to it being comprised of sensitive components - Human Rights and Rule of Law, Group Grievance, etc. - if it is associated with conflict escalation, international relations may be impacted negatively. In example, Group Grievance focuses on separations between different groups in society, particularly when grounded by social and political characteristics - it is not a stretch to predict that this type of data may be misinterpreted. Moreover, we are concerned by the risk of a Self-Fulfilling Prophecy. People's reactions to predictions influenced by fsi_rank could trigger actions or policy changes, ultimately bringing about the predicted conflict escalation. In conclusion, the delicate nature of the elements within fsi_rank run the risk of misinterpretations and unintended consequences in international relations.

Source for Ethical Concern

The indicators within the Fragile States Index are available to be publically accessed at: https://fragilestatesindex.org/indicators/

Constructing a Comprehensive Design Matrix: Integrating Model Predictions and Log-Transformed Variables

In [51]:
```
df_ohco = df_oh.copy()
df_ohco[con_vars]=df_co[con_vars].copy()
df_ohco['model_transformer'] = df_oh['transformer_probability_prediction_error'].as
df_ohco['model_ffnn'] = df_oh['ffnn_probability_prediction_error'].astype(str)*0+"f
df_ohco['model_xgboost'] = df_oh['xgboost_probability_prediction_error'].astype(str
df_ohco['prediction_transformer'] = df.y_true_transformer.astype(int)
df_ohco['prediction_ffnn'] = df.y_pred_ffnn.astype(int)
df_ohco['prediction_xgboost'] = df.y_true_xgboost.astype(int)
design_matrix = \
pd.concat([df_ohco[['transformer_probability_prediction_error', 'model_transformer'
           df_ohco[['ffnn_probability_prediction_error', 'model_ffnn', 'prediction_
           df_ohco[['xgboost_probability_prediction_error', 'model_xgboost', 'predi
          ignore_index=True)
design_matrix['transformer']=(design_matrix['model']=='transformer').astype(int)
design_matrix['ffnn']=(design_matrix['model']=='ffnn').astype(int)
design_matrix['xgboost']=(design_matrix['model']=='xgboost').astype(int)
design_matrix['log_GDP-per-capita']=np.log(design_matrix['GDP-per-capita_values'])
design_matrix['log_displace18']=np.log(design_matrix['displace18_values']+1)
design_matrix['log_pop18']=np.log(design_matrix['pop18_values'])
```

Cleaning the Design Matrix

In [52]:
```
design_matrix=design_matrix.dropna()
design_matrix=design_matrix.drop(columns = ['ffnn','xgboost','transformer'])
```

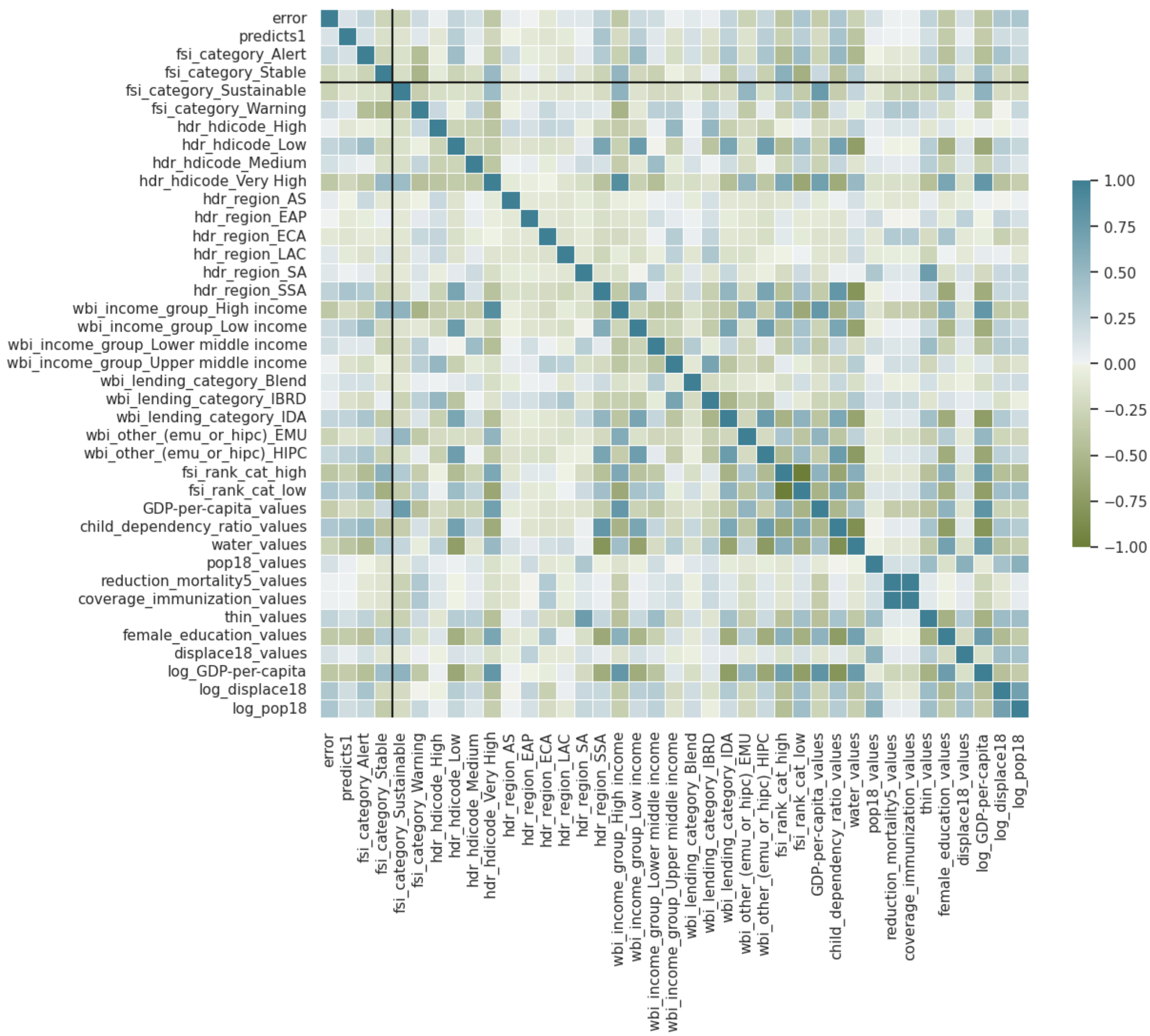
Correlation Heatmap: Visualizing Relationships in the Design Matrix

Each cell in the heatmap represents the correlation coefficient between two variables, with color intensity indicating the strength and direction of the correlation.

In [53]:
```
import matplotlib.pyplot as plt

def corr_heatmap(df):
    sns.set(style="white")
    corr = df.corr()
    mask = np.zeros_like(corr, dtype=bool)
    fig, ax = plt.subplots(figsize=(12, 10))
    cmap = sns.diverging_palette(100, 220, as_cmap=True)
    return sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0, squar
                       linewidths=.5, annot=False, cbar_kws={"shrink": .5})

corr_heatmap(design_matrix)
_ = plt.axhline(y=4, c='k'); plt.axvline(x=4, c='k')
```

```
/tmp/ipykernel_209/1508737882.py:5: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future versi
on, it will default to False. Select only valid columns or specify the value of nume
ric_only to silence this warning.
```

Out[53]: <matplotlib.lines.Line2D at 0x7f9ae57adc90>



Creating Training and Test Sets

In [54]:
```
np.random.seed(4)
train, test = model_selection.train_test_split(design_matrix, train_size=0.8)
```

Preparing Training and Test Sets for ffnn, xgboost, and transformer

In [56]:
```
train_ffnn = train[train['model']=='ffnn']
train_xgboost = train[train['model']=='xgboost']
train_transformer = train[train['model']=='transformer']
test_ffnn = test[test['model']=='ffnn']
test_xgboost = test[test['model']=='xgboost']
test_transformer = test[test['model']=='transformer']
```

Variable Selection for ffnn Model: Initial Model Configuration

In [57]:
```
model_0_variables = train_ffnn.iloc[:,2:].columns.tolist()
model_0_variables.remove('fsi_rank_cat_high')
model_0_variables.remove('fsi_rank_cat_low')
```

```
model_0_variables.remove('pop18_values')
model_0_variables.remove('GDP-per-capita_values')
model_0_variables.remove('displace18_values')

model_0_variables.remove('log_pop18')
model_0_variables.remove('hdr_hdicode_Low')
model_0_variables.remove('fsi_category_Sustainable')
model_0_variables.remove('wbi_income_group_High income')
model_0_variables.remove('hdr_hdicode_Medium')
model_0_variables.remove('wbi_lending_category_Blend')
model_0_variables.remove('wbi_other_(emu_or_hipc)_HIPC')
model_0_variables.remove('water_values')
model_0_variables.remove('predicts1')
model_0_variables.remove('wbi_other_(emu_or_hipc)_EMU')
model_0_variables.remove('wbi_lending_category_IBRD')
model_0_variables.remove('reduction_mortality5_values')
model_0_variables.remove('coverage_immunization_values')
model_0_variables.remove('log_GDP-per-capita')
model_0_variables.remove('hdr_hdicode_High')
model_0_variables.remove('fsi_category_Stable')
model_0_variables.remove('hdr_region_ECA')
model_0_variables.remove('wbi_lending_category_IDA')
model_0_variables.remove('hdr_hdicode_Very High')
model_0_variables.remove('hdr_region_AS')
model_0_variables.remove('thin_values')
model_0_variables.remove('hdr_region_EAP')
model_0_variables.remove('hdr_region_SA')
model_0_variables.remove('wbi_income_group_Lower middle income')
model_0_variables.remove('wbi_income_group_Upper middle income')
model_0_variables.remove('fsi_category_Warning')
model_0_variables.remove('fsi_category_Alert')
model_0_variables.remove('wbi_income_group_Low income')
model_0_variables.remove('hdr_region_SSA')
model_0_variables.remove('hdr_region_LAC')
```

Statistical Summary

In [58]:

```
model_0 = sm.OLS(train_ffnn.error, sm.add_constant(train_ffnn[model_0_variables]))
model_0.fit().summary()
```

Out[58]:

OLS Regression Results

Dep. Variable:	error	**R-squared:**	0.475
Model:	OLS	**Adj. R-squared:**	0.469
Method:	Least Squares	**F-statistic:**	78.64
Date:	Wed, 06 Dec 2023	**Prob (F-statistic):**	2.88e-36
Time:	15:28:21	**Log-Likelihood:**	281.29
No. Observations:	265	**AIC:**	-554.6
Df Residuals:	261	**BIC:**	-540.3
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.3136	0.030	10.605	0.000	0.255	0.372
child_dependency_ratio_values	0.0024	0.000	6.345	0.000	0.002	0.003
female_education_values	-0.0008	0.000	-3.218	0.001	-0.001	-0.000
log_displace18	0.0042	0.002	2.531	0.012	0.001	0.007

Omnibus:	161.661	**Durbin-Watson:**	2.090
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	924.909
Skew:	2.566	**Prob(JB):**	1.44e-201
Kurtosis:	10.578	**Cond. No.**	373.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Variable Selection for xgboost Model: Initial Model Configuration

In [59]:
```
model_1_variables = train_xgboost.iloc[:,2:].columns.tolist()
model_1_variables.remove('fsi_rank_cat_high')
model_1_variables.remove('fsi_rank_cat_low')



model_1_variables.remove('pop18_values')
model_1_variables.remove('GDP-per-capita_values')
model_1_variables.remove('displace18_values')


model_1_variables.remove('female_education_values')
```

```
model_1_variables.remove('log_GDP-per-capita')
model_1_variables.remove('wbi_income_group_Low income')
model_1_variables.remove('log_displace18')
model_1_variables.remove('wbi_lending_category_IBRD')
model_1_variables.remove('fsi_category_Alert')
model_1_variables.remove('hdr_hdicode_Low')
model_1_variables.remove('hdr_hdicode_Medium')
model_1_variables.remove('hdr_hdicode_Very High')
model_1_variables.remove('hdr_hdicode_High')
model_1_variables.remove('wbi_other_(emu_or_hipc)_HIPC')
model_1_variables.remove('thin_values')
model_1_variables.remove('child_dependency_ratio_values')
model_1_variables.remove('water_values')
model_1_variables.remove('wbi_lending_category_Blend')
model_1_variables.remove('fsi_category_Sustainable')
model_1_variables.remove('fsi_category_Warning')
model_1_variables.remove('fsi_category_Stable')
model_1_variables.remove('wbi_other_(emu_or_hipc)_EMU')
model_1_variables.remove('coverage_immunization_values')
model_1_variables.remove('reduction_mortality5_values')
model_1_variables.remove('hdr_region_EAP')
model_1_variables.remove('hdr_region_SA')
```

Statistical Summary

In [60]:
```
model_1 = sm.OLS(train_xgboost.error, sm.add_constant(train_xgboost[model_1_variabl
model_1.fit().summary()
```

Out[60]:

OLS Regression Results

Dep. Variable:	error	**R-squared:**	0.499
Model:	OLS	**Adj. R-squared:**	0.478
Method:	Least Squares	**F-statistic:**	24.38
Date:	Wed, 06 Dec 2023	**Prob (F-statistic):**	1.16e-31
Time:	15:28:25	**Log-Likelihood:**	120.81
No. Observations:	256	**AIC:**	-219.6
Df Residuals:	245	**BIC:**	-180.6
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.3857	0.091	-4.257	0.000	-0.564	-0.207
predicts1	-0.1482	0.027	-5.469	0.000	-0.202	-0.095
hdr_region_AS	0.1413	0.040	3.534	0.000	0.063	0.220
hdr_region_ECA	0.1264	0.041	3.110	0.002	0.046	0.206
hdr_region_LAC	0.1978	0.033	6.063	0.000	0.134	0.262
hdr_region_SSA	0.1911	0.034	5.575	0.000	0.124	0.259
wbi_income_group_High income	0.1418	0.058	2.446	0.015	0.028	0.256
wbi_income_group_Lower middle income	0.1112	0.039	2.842	0.005	0.034	0.188
wbi_income_group_Upper middle income	0.2052	0.055	3.760	0.000	0.098	0.313
wbi_lending_category_IDA	0.1458	0.036	4.018	0.000	0.074	0.217
log_pop18	0.0674	0.007	9.818	0.000	0.054	0.081

Omnibus:	2.311	**Durbin-Watson:**	1.891
Prob(Omnibus):	0.315	**Jarque-Bera (JB):**	2.384
Skew:	0.220	**Prob(JB):**	0.304
Kurtosis:	2.829	**Cond. No.**	109.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Variable Selection for transformer Model: Initial Model Configuration

In [61]:
```
model_2_variables = train_transformer.iloc[:,2:].columns.tolist()

model_2_variables.remove('fsi_rank_cat_high')
model_2_variables.remove('fsi_rank_cat_low')

model_2_variables.remove('pop18_values')
model_2_variables.remove('GDP-per-capita_values')
model_2_variables.remove('displace18_values')

model_2_variables.remove('thin_values')
model_2_variables.remove('fsi_category_Warning')
model_2_variables.remove('fsi_category_Alert')
model_2_variables.remove('hdr_hdicode_Medium')
model_2_variables.remove('hdr_hdicode_Very High')
model_2_variables.remove('water_values')
model_2_variables.remove('wbi_income_group_Low income')
model_2_variables.remove('reduction_mortality5_values')
model_2_variables.remove('coverage_immunization_values')
model_2_variables.remove('hdr_region_EAP')
model_2_variables.remove('wbi_lending_category_IBRD')
model_2_variables.remove('wbi_lending_category_IDA')
model_2_variables.remove('child_dependency_ratio_values')
model_2_variables.remove('wbi_income_group_Lower middle income')
model_2_variables.remove('wbi_income_group_Upper middle income')
model_2_variables.remove('fsi_category_Stable')
model_2_variables.remove('hdr_hdicode_Low')
model_2_variables.remove('wbi_other_(emu_or_hipc)_HIPC')
model_2_variables.remove('hdr_region_ECA')
model_2_variables.remove('predicts1')
model_2_variables.remove('wbi_other_(emu_or_hipc)_EMU')
model_2_variables.remove('wbi_lending_category_Blend')
model_2_variables.remove('hdr_region_AS')
model_2_variables.remove('hdr_region_SSA')
model_2_variables.remove('hdr_region_SA')
model_2_variables.remove('female_education_values')
model_2_variables.remove('log_displace18')
model_2_variables.remove('hdr_hdicode_High')
model_2_variables.remove('hdr_region_LAC')
```

Statistical Summary

In [62]:
```
model_2 = sm.OLS(train_transformer.error, train_transformer[model_2_variables])
model_2.fit().summary()
```

Out[62]:

OLS Regression Results

Dep. Variable:	error	**R-squared (uncentered):**	0.892
Model:	OLS	**Adj. R-squared (uncentered):**	0.891
Method:	Least Squares	**F-statistic:**	528.1
Date:	Wed, 06 Dec 2023	**Prob (F-statistic):**	4.71e-122
Time:	15:28:27	**Log-Likelihood:**	113.28
No. Observations:	259	**AIC:**	-218.6
Df Residuals:	255	**BIC:**	-204.3
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
fsi_category_Sustainable	-0.1327	0.042	-3.134	0.002	-0.216	-0.049
wbi_income_group_High income	-0.0735	0.032	-2.291	0.023	-0.137	-0.010
log_GDP-per-capita	0.0219	0.006	3.752	0.000	0.010	0.033
log_pop18	0.0336	0.005	6.538	0.000	0.023	0.044

Omnibus:	0.511	**Durbin-Watson:**	2.083
Prob(Omnibus):	0.775	**Jarque-Bera (JB):**	0.643
Skew:	-0.073	**Prob(JB):**	0.725
Kurtosis:	2.805	**Cond. No.**	54.9

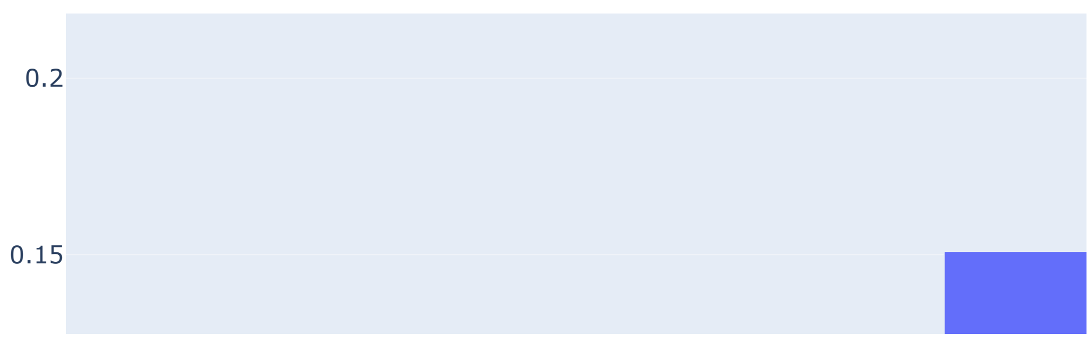
Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Model Performance Evaluation: Root Mean Squared Error Comparison for Models on Training and Testing Sets

In [63]:

```
model_0_train_test_fit = sm.OLS(train_ffnn.error, sm.add_constant(train_ffnn.iloc[:
model_0_train_RMSE = ((train_ffnn.error - model_0_train_test_fit.predict())**2).mea
model_0_test_RMSE = ((test_ffnn.error -
                      model_0_train_test_fit.predict(sm.add_constant(test_ffnn.iloc
                     )**2).mean()**.5
model_1_train_test_fit = sm.OLS(train_xgboost.error, sm.add_constant(train_xgboost.
model_1_train_RMSE = ((train_xgboost.error - model_1_train_test_fit.predict())**2).
model_1_test_RMSE = ((test_xgboost.error -
```

```
                          model_1_train_test_fit.predict(sm.add_constant(test_xgboost.i
                         )**2).mean()**.5
model_2_train_test_fit = sm.OLS(train_transformer.error, train_transformer.iloc[:,2
model_2_train_RMSE = ((train_transformer.error - model_2_train_test_fit.predict())*
model_2_test_RMSE = ((test_transformer.error -
                          model_2_train_test_fit.predict(test_transformer.iloc[:,2:][mo
                         )**2).mean()**.5
px.bar(pd.DataFrame({'RMSE': [model_0_train_RMSE,model_1_train_RMSE,model_2_train_R
                              [model_0_test_RMSE,model_1_test_RMSE,model_2_test_RMSE
                         'Score': ['Training']*3+['Testing']*3,
                         'Model': [0,1,2]+[0,1,2]}),
        y='RMSE', x='Model', color='Score', barmode='group')
```



Analysis Conclusions

From this analysis we can conclude which predictor variables lead to greater error being made by the model in its predictions.

- transformer:
    - A Greater Child Dependency Ratio
    - Lower Female Education

  - Greater Displacement of Individuals Under-18
- xgboost:
  - Greater Total Population Under-18
- transformer:
  - Greater GDP Per-Capita
  - Greater Total Population Under-18