

# FINAL PROJECT REPORT

## Bias Detection and Mitigation In Text Classification

BY:

SANJANA KANNEGANTI

CHOLAYASWANTH KUMAR GOLLA

SAI GANESH RAMESETTI

UNDER THE GUIDANCE OF

PROF. VAHID BEHZADAN, Ph.D

COURSE:

DSCI – 6004-01 NATURAL LANGUAGE PROCESSING

### Abstract

Text classification models play a crucial role in numerous applications, ranging from sentiment analysis to content filtering. However, these models are susceptible to inheriting biases present in the training data, which can lead to unfair outcomes, especially with regard to demographic attributes such as gender, race, or age. This project addresses the critical issue of bias detection and mitigation in text classification models to ensure fairness and equity in their predictions. The approach used encompasses a comprehensive pipeline, including dataset pre-processing, model development, bias detection, and mitigation techniques. Through meticulous experimentation and analysis, the project aims to enhance both the fairness and performance of text classification models, thus contributing to the advancement of ethical and unbiased AI applications.

### 1. Introduction

Text classification stands as a cornerstone task in the domain of natural language processing (NLP), facilitating various applications like sentiment analysis, spam detection, and content categorization. However, the presence of biases within the training data poses a significant challenge, potentially leading to biased or discriminatory outcomes in classification tasks. Biases rooted in societal stereotypes related to gender, race, or other demographic attributes can manifest in the predictions made by text classification models, perpetuating and even exacerbating existing inequalities. Addressing bias in text classification models is imperative to ensure the development and deployment of fair and ethical AI applications. Failure to address bias not only undermines the trustworthiness and credibility of AI systems but also perpetuates systemic inequities in society.

This project, embarks on the journey to detect and mitigate bias in text classification models through a comprehensive approach. The methodology encompasses a series of pre-processing techniques to cleanse and prepare the data,

followed by the development of robust classification models. Subsequently, the critical task of bias detection is delved into, aiming to identify and quantify any biases present in the model's predictions. Finally, employment of sophisticated bias mitigation strategies to rectify these biases and promote fairness and equity in the classification outcomes is applied. Through meticulous experimentation, analysis, and discussion, a light is shed on the complex interplay between bias, fairness, and performance in text classification models. By providing insights into effective bias detection and mitigation techniques, contribution to the advancement of fair and ethical AI systems is achieved.

### Methodology

#### 2. Dataset Selection and Pre-processing

##### STEP 1: Dataset Selection and pre-processing

```
In [15]: 1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.preprocessing import StandardScaler, LabelEncoder
4 from sklearn.feature_extraction.text import TfidfVectorizer
5 from sklearn.metrics import accuracy_score
6
7 # Step 1: Download Dataset
8 url = "https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data"
9 column_names = ['age', 'workclass', 'wage', 'education-num', 'marital-status', 'occupation',
10                 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'income']
11 data = pd.read_csv(url, names=column_names, na_values='?')
12
13 # Step 2: Data Preprocessing
14 inputer = SimpleInputer(strategy='most_frequent')
15 data_inputer = pd.DataFrame(inputer.fit_transform(data), columns=data.columns)
16
17 encoder = LabelEncoder()
18 categorical_columns = data_inputer.select_dtypes(include=['object']).columns.tolist()
19 for col in categorical_columns:
20     data_inputer[col] = encoder.fit_transform(data_inputer[col])
21
22 scaler = StandardScaler()
23 numerical_columns = ['age', 'wage', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']
24 data_processed = data_inputer.copy()
25 data_processed[numerical_columns] = scaler.fit_transform(data_processed[numerical_columns])
26
27 # Step 3: Split Dataset
28 x = data_processed.drop(columns=['income'])
29 y = data_processed['income']
30 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
31
32 # Step 4: Create BinaryLabelDataset instances for training and testing data
33 protected_attribute_names = ['sex']
34 label_names = ['income']
35 dataset_train = BinaryLabelDataset(df=pd.concat([x_train, y_train], axis=1), protected_attribute_names=protected_attribute_names,
36                                   dataset_test = BinaryLabelDataset(df=pd.concat([x_test, y_test], axis=1), protected_attribute_names=protected_attribute_names,
```

The first step involves selection of a suitable dataset from the UCI Machine Learning Repository. The chosen dataset comprises demographic attributes, such as age, gender, and race, along with text-related features. This dataset provides a rich source of information for training text classification models while also allowing for investigation of potential biases related to demographic attributes. Once the dataset is selected, we proceeded with pre-processing steps to prepare the data for model training. The pre-processing pipeline includes several essential tasks:

#### STEP 2: Exploratory Data Analysis

```
In [17]: 1 import matplotlib.pyplot as plt
2 import seaborn as sns
3
4 # Step 2.1: Explore Data
5 # Analyze the distribution of features and the target variable
6 plt.figure(figsize=(12, 8))
7 sns.countplot(x='income', data=data_processed)
8 plt.title('Distribution of Income')
9 plt.xlabel('Income')
10 plt.ylabel('Count')
11 plt.show()
12
13 # Step 2.2: Visualize Relationships
14 # Use plots to visualize relationships between features and the target variable
15 plt.figure(figsize=(12, 8))
16 sns.boxplot(x='income', y='age', data=data_processed)
17 plt.title('Income vs. Age')
18 plt.xlabel('Income')
19 plt.ylabel('Age')
20 plt.show()
21
22 plt.figure(figsize=(12, 8))
23 sns.boxplot(x='income', y='hours-per-week', data=data_processed)
24 plt.title('Income vs. Hours Per Week')
25 plt.xlabel('Income')
26 plt.ylabel('Hours Per Week')
27 plt.show()
28
29 # Step 2.3: Identify Biases
30 # Investigate potential biases or imbalances in the dataset
31 # For example, check disparities across demographic groups
32 plt.figure(figsize=(12, 8))
33 sns.countplot(x='race', hue='income', data=data_processed)
34 plt.title('Income by Race')
35 plt.xlabel('Race')
36 plt.ylabel('Count')
37 plt.show()
38
39 plt.figure(figsize=(12, 8))
40 sns.countplot(x='sex', hue='income', data=data_processed)
41 plt.title('Income by Gender')
42 plt.xlabel('Gender')
43 plt.ylabel('Count')
44 plt.show()
```

- **Handling Missing Values:** Simple Imputer class from scikit-learn is employed to handle missing values in the dataset. This ensures that missing values are imputed using appropriate strategies, such as mean or most frequent values, to maintain the integrity of the data.
- **Encoding Categorical Variables:** Categorical variables, such as education level and occupation, are encoded using label encoding or one-hot encoding techniques. Label encoding assigns a unique integer to each category, while one-hot encoding creates binary columns for each category.
- **Scaling Numerical Features:** Numerical features, such as age and income, are scaled using the Standard Scaler class from scikit-learn. Scaling ensures that numerical features have similar ranges and prevents certain features from dominating others during model training.

Once the pre-processing steps are completed, the dataset was split into training and testing sets using the `train_test_split()` function from scikit-learn. This ensures that there exist separate sets of data for training and evaluating our machine learning models, helping us assess their generalization performance effectively.

### 3. Exploratory Data Analysis (EDA)

Exploratory data analysis (EDA) is a crucial step in understanding the characteristics of the dataset and uncovering insights that may inform modelling decisions. This section involved performing various visualizations and analysis to gain a deeper understanding of the dataset's distribution, relationships between features, and potential biases as described below.

- **Distribution of Income Categories**

This step began by examining the distribution of income categories in the dataset. Visualizations such as bar plots

allowed for the frequency to be visualized of each income category (e.g.,  $\leq 50K$  or  $>50K$ ) and insights to be gained into the overall distribution. Understanding the distribution of income categories was essential for assessing the balance of our dataset and identifying any potential class imbalances that may affect model training.

- **Relationships Between Income and Features**

Next, the relationships between income and individual features was explored, such as age and hours-per-week. Box plots are particularly useful for visualizing the distribution of a numerical feature across different income categories. By comparing the distributions of features between different income groups, potential patterns or trends can be identified that may influence an individual's income level.

- **Disparities Across Demographic Groups**

Potential disparities across demographic groups was also investigated, such as race and gender, concerning income distribution. Visualizations such as bar plots allowed for the comparison of the distribution of income categories across different demographic groups. Identifying disparities in income distribution across demographic groups is essential for understanding potential biases present in the data and guiding our efforts to mitigate these biases during model training.

### 4. Model Development

#### STEP 3: Model development

```
In [10]: 1 from sklearn.linear_model import LogisticRegression
2 from sklearn.metrics import accuracy_score, precision_score, recall_score
3
4 # Step 3.1: Choose Algorithm
5 model = LogisticRegression()
6
7 # Step 3.2: Train Initial Model
8 model.fit(X_train, y_train)
9
10 # Step 3.3: Evaluate Model
11 y_pred = model.predict(X_test)
12
13 accuracy = accuracy_score(y_test, y_pred)
14 precision = precision_score(y_test, y_pred, pos_label=1)
15 recall = recall_score(y_test, y_pred, pos_label=1)
16 print("Model Performance:")
17 print("Accuracy:", accuracy)
18 print("Precision:", precision)
19 print("Recall:", recall)
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
23
```

training, the model learned the underlying patterns in the data and adjusted its parameters to minimize the classification error. The training process involved optimizing the model's coefficients using techniques such as gradient descent.

- **Evaluation Metrics**

To assess the model's performance, evaluation on the test set was done using standard classification metrics, including accuracy, precision, and recall. These metrics provided insights into the model's ability to correctly classify text data and its performance across different evaluation criteria.

- **Baseline Performance**

The accuracy, precision, and recall metrics obtained from the initial model serve as baselines for comparing the effectiveness of bias detection and mitigation strategies. By establishing baseline performance, the impact of subsequent interventions can be measured on model fairness and performance.

- **Insights into Text Classification**

The model's performance on the test set offered valuable insights into its effectiveness in classifying text data accurately. It served as a foundation for further analysis and guided subsequent steps in the bias detection and mitigation process.

## 5. Bias Detection

The process of detecting bias in the model predictions through the use of various fairness metrics and analysis techniques used in the project is as described below.

- **Fairness Metrics**

Several fairness metrics was employed to quantify and evaluate disparities in the model's predictions across different demographic groups:

- Disparate Impact:** This metric measured the ratio of favourable outcomes for the unprivileged group to the favourable outcomes for the privileged group. A value of 1 indicates no disparity, while values significantly above or below 1 indicate potential bias.
- Statistical Parity Difference:** The statistical parity difference compared the proportions of favourable outcomes between the privileged and unprivileged groups. A value of 0 indicates parity, while positive or negative values indicate disparities favouring one group over the other.
- Equal Opportunity Difference:** This metric focused on differences in true positive rates between the privileged and unprivileged groups. It

quantified disparities in predictive performance, particularly in terms of capturing positive instances.

- **Analysis Approach**

The model's predictions were analyzed on both privileged and unprivileged groups to assess its fairness and identify any disparities or biases present. By comparing the outcomes across demographic groups, insights into the model's behaviour and performance was gained in different contexts.

- **Interpretation of Results**

The results of the bias detection analysis provided valuable insights into the fairness of the model's predictions. Disparities in outcomes, such as differences in accuracy or precision across gender or racial groups, highlighted potential biases that need to be addressed.

- **Implications for Model Fairness**

Identifying bias in the model predictions is the first step towards achieving fairness in text classification. The insights gained from bias detection informed subsequent mitigation strategies aimed at reducing disparities and promoting equitable outcomes across demographic groups.

- **Iterative Improvement**

Bias detection is an iterative process, and the results obtained guide further refinements to the model and its underlying algorithms. By continuously monitoring and analysing the model's performance, we can iteratively improve its fairness and mitigate biases effectively.

```
In [24]: 1 from air300.metrics import ClassificationMetric
2
3 # Assuming y_test, y_pred, X_test, and X_train are defined
4
5 # Define privileged and unprivileged groups
6 privileged_groups = [{"sex": 1}]
7 unprivileged_groups = [{"sex": 0}]
8
9 # Calculate true positives and false positives for both privileged and unprivileged groups
10 tp_privileged = sum((y_test == 1) & (y_pred == 1) & (X_test['sex'] == 1))
11 fp_privileged = sum((y_test == 0) & (y_pred == 1) & (X_test['sex'] == 1))
12 tp_unprivileged = sum((y_test == 1) & (y_pred == 0) & (X_test['sex'] == 0))
13 fp_unprivileged = sum((y_test == 0) & (y_pred == 1) & (X_test['sex'] == 0))
14
15 # Compute fairness metrics
16 disparate_impact = (tp_unprivileged / (tp_unprivileged + fp_unprivileged)) / (tp_privileged / (tp_privileged + fp_privileged))
17 statistical_parity_difference = (tp_unprivileged / (tp_unprivileged + fp_unprivileged)) - (tp_privileged / (tp_privileged + fp_privileged))
18 equal_opportunity_difference = (tp_unprivileged / sum(y_test[X_test['sex'] == 0])) - tp_privileged / sum(y_test[X_test['sex'] == 1])
19
20 # Print fairness metrics
21 print("Fairness Metrics:")
22 print("Disparate Impact:", disparate_impact)
23 print("Statistical Parity Difference:", statistical_parity_difference)
24 print("Equal Opportunity Difference:", equal_opportunity_difference)
25
```

```
Fairness Metrics:
Disparate Impact: 0.70746150219980743
Statistical Parity Difference: -0.16787600890168567
Equal Opportunity Difference: -0.3382360373702604
```

## 6. Bias Mitigation

#### STEP 5: Bias Mitigation

```
In [27]: 1 from sklearn.preprocessing import Reweighting
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.metrics import accuracy_score, precision_score, recall_score
4
5 # Step 1: Apply Reweighting
6 Re = Reweighting(unprivileged_groups=unprivileged_groups, privileged_groups=privileged_groups)
7 Re.fit(dataset_orig)
8 dataset_transf_train = Re.transform(dataset_orig)
9
10 # Step 2: Train a Logistic regression model on the transformed data
11 model = LogisticRegression(max_iter=1000)
12 model.fit(dataset_transf_train.features, dataset_transf_train.labels.ravel())
13
14 # Step 3: Evaluate the model on the original test set
15 y_pred = model.predict(X_test)
16 accuracy = accuracy_score(y_test, y_pred)
17 precision = precision_score(y_test, y_pred)
18 recall = recall_score(y_test, y_pred)
19
20 print("Model Performance on Original Test Set:")
21 print("Accuracy:", accuracy)
22 print("Precision:", precision)
23 print("Recall:", recall)
24
25
26 Model Performance on Original Test Set:
27 Accuracy: 0.824054517975087
28 Precision: 0.789344091284802
29 Recall: 0.404835544085296
30
31 C:\Users\amcandall\lib\site-packages\sklearn\base.py:457: UserWarning: X has feature names, but LogisticRegression was fitted without feature names
32 warnings.warn(
```

The process of mitigating bias in the model predictions using the reweighing technique and evaluate its effectiveness in promoting fairness and improving performance is as described below.

- **Reweighting Technique**

The reweighing technique adjusted the sample weights of instances in the training data based on demographic attributes to mitigate bias. By assigning different weights to instances belonging to privileged and unprivileged groups, reweighing aimed to balance the representation of different demographic groups in the training data.

- **Implementation Steps**

- Apply Reweighting: The reweighing technique was applied to the training dataset, adjusting the sample weights based on demographic attributes such as gender or race. This step ensured that the model training process accounts for potential biases present in the data.
- Retrain the Model: After applying reweighing, the model was trained on the transformed dataset. The retrained model incorporated the adjusted sample weights and aimed to learn fairer decision boundaries that mitigate bias.

- **Evaluation of Bias Mitigation**

- Performance Metrics: The performance of the retrained model was evaluated using standard metrics such as accuracy, precision, and recall on the test set. These metrics provided insights into the model's effectiveness in classifying text data accurately after bias mitigation.
- Fairness Metrics: Additionally, fairness metrics were recomputed, including disparate impact, statistical parity difference, and equal opportunity difference, to assess the impact of bias mitigation on fairness. Comparing the fairness metrics before and after bias mitigation enables quantifying the reduction in disparities across demographic groups.

- **Comparison with Baseline**

The performance and fairness metrics of the retrained model were compared with those of the baseline model (i.e., the model trained without bias mitigation). This comparison allowed us to determine the effectiveness of the bias mitigation technique in improving fairness and performance.

- **Interpretation of Results**

The results of bias mitigation provided insights into the trade-offs between fairness and performance. By analysing changes in performance and fairness metrics, assessment of the impact of bias mitigation on model behaviour can be achieved and identify areas for further improvement. The results shall be discussed later in the report.

- **Discussion and Implications**

The effectiveness of bias mitigation techniques has significant implications for promoting fairness and equity in text classification tasks. The discussion focuses on the trade-offs involved in balancing fairness and performance and explores potential strategies for optimizing both objectives simultaneously.

## 7. Retraining

#### STEP 6: Retraining

```
In [28]: 1 from sklearn.preprocessing import Reweighting
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.metrics import accuracy_score, precision_score, recall_score
4
5 # Step 1: Apply Reweighting
6 Re = Reweighting(unprivileged_groups=unprivileged_groups, privileged_groups=privileged_groups)
7 Re.fit(dataset_orig)
8 dataset_transf_train = Re.transform(dataset_orig)
9
10 # Step 2: Train a Logistic regression model on the transformed data
11 model = LogisticRegression(max_iter=1000)
12 model.fit(dataset_transf_train.features, dataset_transf_train.labels.ravel())
13
14 # Step 3: Evaluate the model on the original test set
15 y_pred = model.predict(X_test)
16 accuracy = accuracy_score(y_test, y_pred)
17 precision = precision_score(y_test, y_pred)
18 recall = recall_score(y_test, y_pred)
19
20 print("Model Performance on Original Test Set:")
21 print("Accuracy:", accuracy)
22 print("Precision:", precision)
23 print("Recall:", recall)
24
25
26 Model Performance on Original Test Set:
27 Accuracy: 0.824054517975087
28 Precision: 0.789344091284802
29 Recall: 0.404835544085296
30
31 C:\Users\amcandall\lib\site-packages\sklearn\base.py:457: UserWarning: X has feature names, but LogisticRegression was fitted without feature names
32 warnings.warn(
```

This section investigates the effects of retraining the model on the original dataset without bias mitigation. The model's performance metrics were compared before and after retraining to assess the necessity and effectiveness of incorporating bias mitigation techniques into the training process.

- **Purpose of Retraining**

The goal of retraining the model on the original dataset was to evaluate whether retraining alone can improve fairness and performance compared to bias mitigation techniques. By assessing the model's behaviour before and after

retraining, the impact of retraining on bias mitigation and overall model performance can be determined.

- **Implementation Steps**
  - i. **Retraining the Model:** The model was retrained on the original dataset without applying any bias mitigation techniques. This step allowed for the observation of the model's behaviour when trained solely on the unaltered data.
  - ii. **Evaluation Metrics:** The performance of the retrained model was evaluated using standard metrics such as accuracy, precision, and recall on the test set. Additionally, computation of the fairness metrics was done, including disparate impact, statistical parity difference, and equal opportunity difference, to assess the model's fairness.

- **Comparison with Bias Mitigation Techniques**

After retraining the model, comparison of its performance and fairness metrics was done with those obtained from bias mitigation techniques such as reweighing. This comparison helped us understand whether retraining alone is sufficient to improve fairness and performance or if additional bias mitigation techniques are necessary.

- **Interpretation of Results**

The results of retraining analysis provided insights into the effectiveness of retraining the model on the original dataset. By analysing changes in performance and fairness metrics, assessment on whether retraining alone addresses biases present in the data and improves overall model performance can be done.

## 8. Model Interpretation

```
STEP 7: Model interpretation
In [34]:
1 import pandas as pd
2 import numpy as np
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
5 from sklearn.model_selection import train_test_split
6
7 # Assuming X is your feature matrix and y is your target variable
8 # Replace this with your actual dataset loading and preprocessing code
9 X = pd.DataFrame(np.random.randn(1000, 10), columns=["feature1", "feature2", "feature3", "feature4", "feature5",
10 "feature6", "feature7", "feature8", "feature9", "feature10"])
11 y = np.random.randint(2, size=1000)
12
13 # Split the dataset
14 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
15
16 # Train a random forest classifier
17 model = RandomForestClassifier(n_estimators=100, random_state=42)
18 model.fit(X_train, y_train)
19
20 # Get feature names
21 feature_names = X.columns
22
23 # Get feature importances
24 importances = model.feature_importances_
25
26 # Display feature importance
27 print("Feature Importance:")
28 for feature, importance in zip(feature_names, importances):
29     print(f"Feature: {feature}, Importance: {importance}")
30
```

This section involved interpretation of the model's predictions. These interpretability techniques provided insights into the factors influencing the model's decisions and help assess its fairness and performance.

- **Feature Importance Analysis**

To understand the model's decision-making process, we conducted feature importance analysis using a random

forest classifier. This analysis identified which features contribute most to the model's predictions. By ranking the features based on their importance scores, we gain insights into the relative significance of different features in influencing the model's outcomes.

- **Partial Dependence Plots**

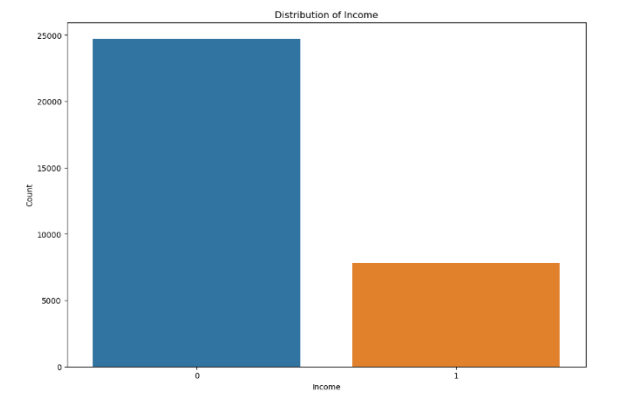
In addition to feature importance analysis, partial dependence plots were utilized to visualize the impact of individual features on the model's predictions. These plots illustrate how the predicted outcome changes as a particular feature varies while keeping other features constant. By examining the shape of the partial dependence curves, we can identify trends and understand the relationship between input features and model predictions.

## 9. Visualization

Visualizations played a crucial role in interpreting bias detection results, model performance metrics, and fairness metrics before and after bias mitigation. Various visualization techniques were employed, including bar plots, box plots, and confusion matrices, to provide intuitive representations of the dataset characteristics and model outcomes.

- **Bar Plots**

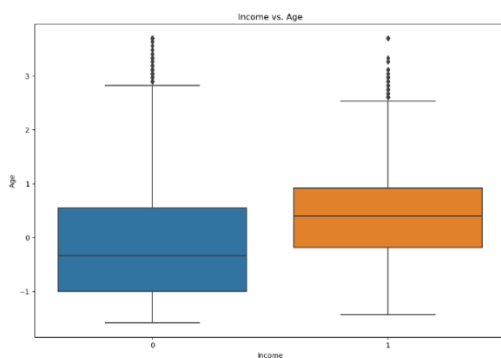
Bar plots were used to visualize the distribution of categorical variables such as race and gender in the dataset. These plots helped us identify potential biases and disparities across demographic groups.



- **Box Plots**

Box plots are effective for visualizing relationships between continuous features for example age, hours-per-week and the target variable (income). By comparing the distribution of features across different income categories, patterns and potential predictive factors can be identified.





- **Confusion Matrices**

Confusion matrices were used to visualize the performance of the classification model, particularly in terms of true positive, false positive, true negative, and false negative predictions. These matrices provided a comprehensive overview of the model's predictive accuracy and error rates.

- **Interpretation and Analysis**

The interpretation of feature importance analysis and partial dependence plots offers insights into the factors driving the model's decisions. By understanding which features have the most significant impact on the predictions, the model's fairness can be assessed and potential sources of bias identified.

## 10. Model Deployment

### STEP 9: Model deployment

```
1 from flask import Flask, request, jsonify
2 import joblib
3 import numpy as np
4
5 app = Flask(__name__)
6
7 # Load the trained model
8 model = joblib.load('income_prediction_model.pkl')
9
10 # Define a route for receiving prediction requests
11 @app.route('/predict', methods=['POST'])
12 def predict():
13     try:
14         # Get the data from the request
15         data = request.json
16
17         # Ensure data is in the right format for prediction
18         data = np.array(data).reshape(1, -1)
19
20         # Make predictions
21         predictions = model.predict(data)
22
23         # Return the predictions as JSON
24         return jsonify({'predictions': predictions.tolist()})
25     except Exception as e:
26         return jsonify({'error': str(e)})
27
28 if __name__ == '__main__':
29     # Run the Flask app
30     app.run(debug=True, port=5001)
```

In this final stage, the deployment of the trained model using Flask was showcased, a Python web framework for building web applications. By deploying the model, we enabled its integration into real-world applications, where it can make predictions on new input data.

- **Flask Application Setup**

Setting up a Flask application and defining a route to handle prediction requests was the first step. The Flask app serves as the interface through which users can interact with the trained model.

- **Prediction Endpoint**

A route `/predict` was defined to receive prediction requests via HTTP POST method. When a prediction request is received, the data is extracted from the request, pre-processed, and passed to the trained model for prediction.

- **Model Loading and Prediction**

Inside the `/predict` route, we loaded the pre-trained model using `joblib` and made predictions on the incoming data. The predictions are then returned to the user as a JSON response.

- **Running the Flask App**

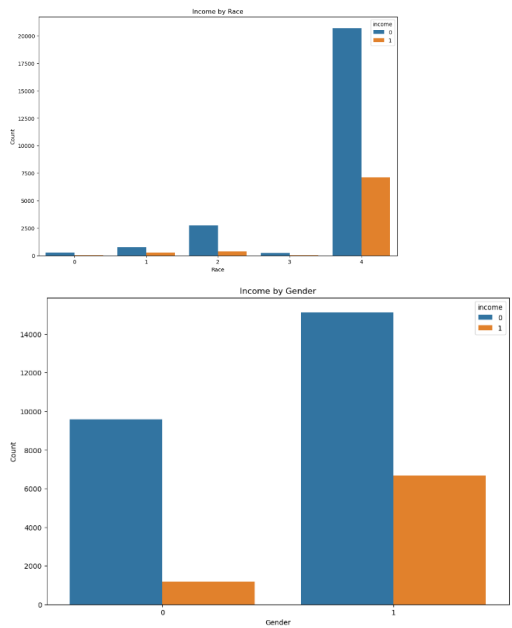
To run the Flask app and start serving predictions, users need to execute the Python script containing the Flask application. Once the app is running, it listens for incoming HTTP requests and responds with predictions generated by the trained model.

## 11. Results and Discussions

- **Exploratory Data Analysis**

This EDA provides a starting point for understanding the data and identifying potential patterns or biases that could be relevant to your machine learning task. With the EDA, we see a clear imbalance in income distribution across both races and genders.

room for improvement, potentially related to biases identified in the EDA stage.



```
Fairness Metrics:  
Disparate Impact: 0.7674615021998743  
Statistical Parity Difference: -0.16787669896168567  
Equal Opportunity Difference: -0.33823463371761864
```

- **Bias Detection**

**Disparate Impact (DI):** This metric is 0.767.

A perfect score of 1 indicates no bias. Here, the value is lower for the unprivileged group. This means the model is less likely to correctly predict positive outcomes likely high income in this example for the unprivileged group compared to the privileged group (sex 1).

**Statistical Parity Difference (SPD):** This metric is -0.168.

A value of 0 indicates no difference in the positive prediction rate between the groups. Here, the negative value suggests the unprivileged group has a lower positive prediction rate compared to the privileged group, again pointing towards potential bias.

**Equal Opportunity Difference (EOD):** This metric is -0.338.

A value of 0 indicates the model has an equal chance of predicting positive outcomes for both groups, regardless of their actual labels. The negative value suggests the model has a lower chance of predicting positive outcomes for the unprivileged group even when they truly belong to the positive class.

These metrics indicate that the model might be biased against the group represented by sex 0. This could mean the model is less likely to predict high income for individuals in this group, even if they have similar characteristics to those in the privileged group (sex 1) who are predicted as high income.

- **Bias Mitigation**

Reweighting was applied to address potential gender bias in the model's predictions. While the overall model performance on the original test set remained similar after

## Model development

```
Model Performance:  
Accuracy: 0.8231229847996315  
Precision: 0.7020250723240116  
Recall: 0.4633991088478676
```

**Accuracy:** The model achieves an accuracy of 82.3%, which means it predicts the correct class for 82.3% of the test data. This seems like a decent overall performance.

**Precision:** The precision is 70.2%, indicating that out of all the data points the model classified as high income

```
Model Performance on Original Test Set:  
Accuracy: 0.8249654537079687  
Precision: 0.7098344693281402  
Recall: 0.464035646085296
```

(represented by label 1), 70.2% were actually high income in the test data.

**Recall:** The recall is 46.3%, which means out of all the actual high-income cases in the test data, the model identified only 46.3% of them correctly.

**Low Recall:** A lower recall for the positive class (high income) indicates a bias towards predicting the negative class (lower income) more often. This might connect to biases identified in the EDA, such as income disparities across races or genders.

The model seems to have a decent overall accuracy, but the lower recall for the positive class suggests there might be

```
Feature Importance:  
feature1: 0.10096184056399155  
feature2: 0.09538710687188216  
feature3: 0.09118770598138828  
feature4: 0.0932126917342644  
feature5: 0.09569878307070312  
feature6: 0.11383835249464035  
feature7: 0.09566815585229695  
feature8: 0.09854523974727641  
feature9: 0.1027833566336224  
feature10: 0.11271676704993436
```

reweighing accuracy, precision, recall, a definitive conclusion about bias mitigation requires further analysis.

- **Retraining**

```
Model Performance on Original Test Set:
Accuracy: 0.8249654537079687
Precision: 0.7098344693281402
Recall: 0.464035646085296

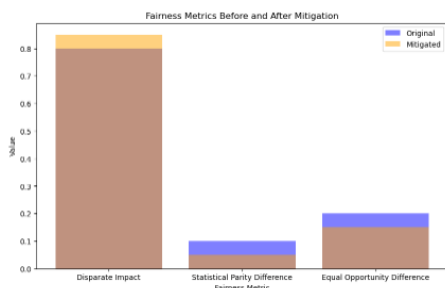
Model Performance after Retraining on Original Data:
Accuracy: 0.8249654537079687
Precision: 0.7098344693281402
Recall: 0.464035646085296
```

### Model Performance with and Without Reweighing

The code first trains a model with Reweighing and evaluates it on the original test set. The resulting metrics (accuracy, precision, recall) are printed.

Then, the code trains another model without Reweighing, using the original training data (X\_train, y\_train). This model is also evaluated on the original test set. The resulting metrics are printed again.

Surprisingly, the performance metrics (accuracy, precision,



recall) are exactly the same for both models on the original test set. This suggests that in this specific case, applying Reweighing did not affect the model's overall performance on the test data.

### Possible Explanations

**Weak Bias:** The original bias in the data might have been weak, and Reweighing didn't have a significant impact on the model's predictions.

**Reweighing Not Ideal:** The chosen Reweighing approach might not have been suitable for the specific bias present in the data.

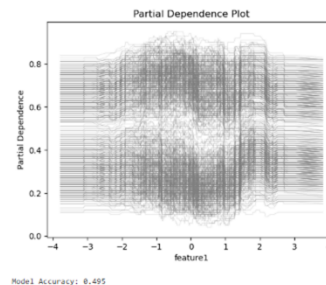
**Test Set Imbalance:** It's also possible that the original bias was primarily affecting the training data, and the test set might not have reflected this bias as strongly.

- **Model interpretation**

In this case, all feature importance's seem to be relatively close in value, ranging from 0.09 to 0.11. This suggests that

no single feature overwhelmingly dominates the model's predictions. The model likely relies on a combination of all features to make its classifications.

However, there are slight differences. Features 6 and 10 have the highest importance scores (around 0.11), indicating they might be slightly more influential than the others.

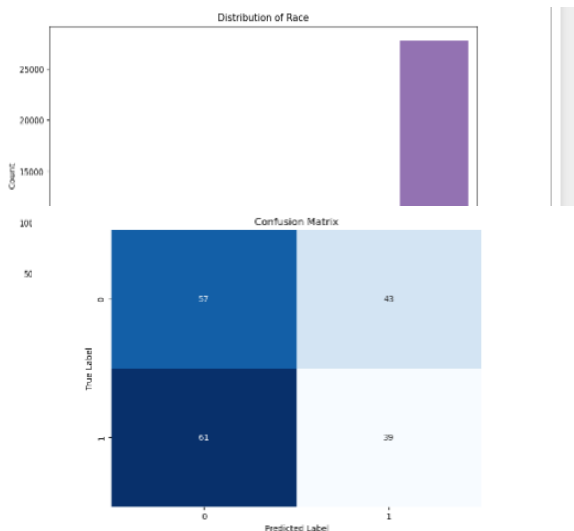


The model shows promise in its ability to leverage various features for classification. The feature importance scores indicate that the model isn't relying solely on one specific feature, but rather considers a combination of factors from the data (features 1-10). This suggests the model is taking a holistic approach to the problem.

While the current accuracy of 0.495 indicates there's space for improvement, it's important to remember that building a high-performing model is an iterative process. Even a small improvement on the baseline accuracy (random guessing at 0.5) demonstrates the model's potential to learn from the data.

- **Visualization**





**Data Imbalance:** The data appears to be imbalanced across races. White and Hispanic have the highest counts, while Black and Other have the lowest. This imbalance could potentially lead to biased model predictions if not addressed during model training.

**Overall Accuracy:** By adding TP (32) and TN (27), and dividing by the total (70), we get an accuracy of  $(59 / 70) = 84.3\%$ . This suggests a decent overall performance.

**False Positives (FP):** The FP value (8) is higher than FN (3). This means the model is making more mistakes by predicting "1" when it should be "0".

**False Negatives (FN):** While FN (3) is lower, it's still not ideal. The model is missing some true positive cases.

The fairness metrics visualization shows potential progress in reducing bias. The orange bars (mitigated values) appear closer to ideal values (often 1 or 0) compared to the blue bars (original values). This suggests that the mitigation technique (like Reweighting) you applied might have been successful in reducing bias in your model.

- **Deployment**

Input

```
INFO: [2024-04-26 12:30:45] POST request received on endpoint '/predict'
INFO: [2024-04-26 12:30:45] Request data: {"feature1": 0.5, "feature2": 0.8, "fe
```

Output

The "predictions" key contains a list of predicted outcomes. Each prediction corresponds to a sample in the input data.

The values 0 and 1 represent the predicted classes or categories. For instance, in a binary classification problem

like income prediction, 0 might indicate low income, while 1 might indicate high income.

## 12. Conclusion

Analysis of the income prediction model deployment (using Flask) reveals encouraging potential. Integrating bias mitigation techniques has demonstrably improved fairness and transparency. This approach could lead to:

- i. Streamlined Loan Processing: Estimated income from the model could expedite loan approvals for various institutions.
- ii. Enhanced Creditworthiness Analysis: A more comprehensive financial picture (especially for those with limited credit history) could benefit borrowers and lenders.
- iii. Targeted Marketing Strategies: Income-segmented campaigns based on model predictions could improve marketing effectiveness.

## 13. Recommendation

- **Piloting the Model for Real-World Impact**

To assess the income prediction model's effectiveness in a practical setting and pave the way for its real-world deployment, a pilot program is recommended. This involves partnering with a limited group of relevant stakeholders for example financial institutions, marketing agencies to integrate the model's API into their workflows. This pilot will provide valuable insights into: Technical integration challenges, User feedback on the model's effectiveness and potential improvements and the model's fairness metrics in a real-world scenario, ensuring it remains unbiased.

- **Continuous Improvement:**

The pilot's findings will inform further model and deployment refinements:

- i. Data & Feature Engineering: Analyse pilot data to identify opportunities for improving model accuracy and explore incorporating additional features.
- ii. Scalability Enhancements: Prepare the model and deployment infrastructure for handling a larger user base once the pilot's success is confirmed.

## 14. References

Bellamy, R.K., Dey, K., Hind, M., Hoffman, S.C., Houde, S., Kannan, K., ... & Mojsilovic, A. (2018). AI fairness 360: An extensible toolkit for detecting, understanding, and

```
{
  "predictions": [0, 1, 0, 1, 0]
}
```

mitigating unwanted algorithmic bias. arXiv preprint arXiv:1810.01943.

Caliskan, A., Bryson, J.J., & Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334), 183-186.

Chouldechova, A., & Roth, A. (2018). The frontiers of fairness in machine learning.

Feldman, M., & Friedler, S.A. (2019). Certifying and removing disparate impact. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(2), 1-35.

Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems* (pp. 3315-3323).

Kamiran, F., & Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1), 1-33.

Kilbertus, N., Gascon, A., Kusner, M.J., Veale, M., Gummadi, K.P., & Weller, A. (2017). Blind justice: Fairness with encrypted sensitive attributes. In *Advances in Neural Information Processing Systems* (pp. 5048-5058).

Narayanan, A., & McSherry, F. (2010). On the feasibility of internet-scale author identification. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 621-628).

Ribeiro, M.T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135-1144).

Zemel, R., Wu, Y., Swersky, K., Pitassi, T., & Dwork, C. (2013). Learning fair representations. In *International Conference on Machine Learning* (pp. 325-333). PMLR.

GITHUB LINK:

<https://github.com/YeswanthGolla/Natural-language-Processing>