Fairness in Recruitment Algorithms

```
#from google.colab import drive
#drive.mount('/content/drive')
```

Task 1 - Preparation

1.1 Import utils

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.utils.class_weight import compute_sample_weight
import matplotlib.pyplot as plt
from xgboost import XGBClassifier
from lime import lime_tabular
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
import shap
from aif360.algorithms.inprocessing import AdversarialDebiasing
from aif360.datasets import BinaryLabelDataset
import tensorflow.compat.v1 as tf
import os

import warnings
warnings.filterwarnings("ignore", message="X does not have valid feature names")

from scipy.stats import chi2_contingency
from sklearn.metrics import confusion_matrix

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import metrics
from sklearn.metrics import f1_score
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import BinaryCrossentropy
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder

from tensorflow.compat.v1.keras.backend import get_session
```

```
from tensorflow.keras.layers import Dense, BatchNormalization
#tf.compat.v1.disable_v2_behavior()



random_seed = 15
```

```
IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.
readthedocs.io/en/stable/user_install.html
WARNING:root:No module named 'fairlearn': ExponentiatedGradientReduction will be u
navailable. To install, run:
pip install 'aif360[Reductions]'
WARNING:root:No module named 'fairlearn': GridSearchReduction will be unavailable.
To install, run:
pip install 'aif360[Reductions]'
WARNING:root:No module named 'inFairness': SenSeI and SenSR will be unavailable. T
o install, run:
pip install 'aif360[inFairness]'
WARNING:root:No module named 'fairlearn': GridSearchReduction will be unavailable.
To install, run:
pip install 'aif360[Reductions]'
```

1.2 Encode the dataset

```
#dataset_path = '/content/drive/MyDrive/Dataset_1.0_Akkodis.xlsx'
dataset_path = 'Dataset_1.0_Akkodis.xlsx'
df = pd.read_excel(dataset_path)
df.head()
```

	Unnamed: 0	ID	Candidate State	Age Range	Citizenship	Residence	Sex	Protected category	
0	0	71470	Hired	31 - 35 years	Pakistani	TURIN » Turin ~ Piedmont	Male	NaN	AUTOSAR, C MATLAB/SIM VEC
1	1	71470	Hired	31 - 35 years	Pakistani	TURIN » Turin ~ Piedmont	Male	NaN	AUTOSAR, C MATLAB/SIM VEC
2	2	71470	Hired	31 - 35 years	Pakistani	TURIN » Turin ~ Piedmont	Male	NaN	AUTOSAR, C MATLAB/SIM VEC
3	3	71470	Hired	31 - 35 years	Pakistani	TURIN » Turin ~ Piedmont	Male	NaN	AUTOSAR, C MATLAB/SIM VEC
4	4	71470	Hired	31 - 35 years	Pakistani	TURIN » Turin ~ Piedmont	Male	NaN	AUTOSAR, C MATLAB/SIM VEC

5 rows × 41 columns

```
# Visualize the shape
print(f"There are {df.shape[0]} rows and {df.shape[1]} columns")
```

```
There are 21377 rows and 41 columns
```

```
# Get some statistics
def get_data(dataframe):
    print("\nBASIC INFORMATION\n")
    print(dataframe.info())
    print("=" * 100)
    print("NULL Values Check")
    print(dataframe.isnull().sum())
    print("=" * 100)
```

```
# Drop non-useful columns
unuseful_columns = ['ID', ' TAG', ' Year of insertion', ' Year of Recruitment', ' R
                    ' Assumption Headquarters', ' event_type__val', ' linked_searc
                      ' Candidate Profile', ' Akkodis headquarters', 'Standing/Pos
df = df.drop(columns=unuseful_columns)
```

```
get_data(df)
```

```
BASIC INFORMATION

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21377 entries, 0 to 21376
Data columns (total 24 columns):
```

```
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0    Candidate State     21377 non-null  object
 1    Age Range           21377 non-null  object
 2    Citizenship         21361 non-null  object
 3    Sex                 21377 non-null  object
 4    Protected category  85 non-null     object
 5    Study area          21332 non-null  object
 6    Study Title         21377 non-null  object
 7    Years Experience    21377 non-null  object
 8    Sector              12214 non-null  object
 9    Job Family Hiring   2382 non-null   object
 10   Job Title Hiring    2382 non-null   object
 11   event_feedback      5846 non-null   object
 12   Overall             5984 non-null   object
 13   Minimum Ral         1169 non-null   object
 14   Ral Maximum         1528 non-null   object
 15   Study Level         2120 non-null   object
 16  Current Ral          4156 non-null   object
 17  Expected Ral         4119 non-null   object
 18  Technical Skills     5955 non-null   float64
 19  Comunication         5968 non-null   float64
 20  Maturity             5964 non-null   float64
 21  Dynamism             5965 non-null   float64
 22  Mobility             5974 non-null   float64
 23  English              5944 non-null   float64
dtypes: float64(6), object(18)
memory usage: 3.9+ MB
None
================================================================================
==================
NULL Values Check
 Candidate State          0
 Age Range                0
 Citizenship             16
 Sex                      0
 Protected category   21292
 Study area              45
 Study Title              0
 Years Experience         0
 Sector                9163
 Job Family Hiring    18995
 Job Title Hiring     18995
 event_feedback       15531
 Overall              15393
 Minimum Ral          20208
 Ral Maximum          19849
 Study Level          19257
Current Ral           17221
Expected Ral          17258
Technical Skills      15422
Comunication          15409
Maturity              15413
Dynamism              15412
Mobility              15403
English               15433
dtype: int64
```

```
==========================================================================================
==================
```

```python
# Visualize the new shape
print(f"There are {df.shape[0]} rows and {df.shape[1]} columns")
```

```
There are 21377 rows and 24 columns
```

```python
# Visualize the possible values
for feature in df.columns:
    print(f'Feature: {feature} -- {list(df[feature].unique())}')
```

```
Feature:  Candidate State -- ['Hired', 'Vivier', 'QM', 'In selection', 'First cont
act', 'Economic proposal', 'Imported']
Feature:  Age Range -- ['31 - 35 years', '40 - 45 years', '36 - 40 years', '> 45 y
ears', '26 - 30 years', '< 20 years', '20 - 25 years']
Feature:  Citizenship -- ['Pakistani', 'Italian', nan, 'Moroccan', 'Iranian', 'Alb
anian', 'Indiana', 'Colombian', 'Ethiopian', 'Romanian', 'Vltava', 'Lebanese', 'Sp
anish', 'Egyptian', 'Russian', 'Tunisian', 'Turkish', 'Chinese', 'Uzbek', 'Brazili
an', 'Cameroonian', 'Sudanese', 'Algerian', 'Croatian', 'Polish', 'Indonesian', 'S
an Marino', 'Argentina', 'Azerbaijan', 'Portuguese', 'Serbian', 'French', 'Swiss',
'German', 'Peruvian', 'British', 'Venezuelan', 'Rwandan', 'Costa Rican', 'South Ko
rean', 'Ukraine', 'Macedonian', 'Nigerian', 'American', 'Kenyan', 'Emirati', 'Ecua
dorian', 'Ivorian', 'Mexican', 'Chilean', 'Japanese', 'Syrian', 'Bangladeshis', 'G
reek', 'Israeli', 'Omani', 'South African', 'Bolivian', 'Filipina', 'Sinhalese', '
Palestinian (Palestinian Territories)', 'Afghan', 'Jordan', 'Cuban', 'Vietnamese',
'Latvian', 'Libyan', 'Bulgarian', 'Togolese', 'Kazakh', 'Austrian', 'Belarusian',
'Saudi', 'Bosnian', 'Kyrgyz', 'Tajik', 'Dutch', 'Qatari', 'Georgian', 'Canadian',
'Australian', 'Salvadoran', 'Congolese', 'Guatemalan', 'Hungarian', 'Tanzanian', '
Gabonese', 'Angolan', 'Maltese']
Feature:  Sex -- ['Male', 'Female']
Feature:  Protected category -- [nan, 'Article 1', 'Article 18']
Feature:  Study area -- ['Automation/Mechatronics Engineering', 'computer engineer
ing', 'chemical engineering', 'Legal', 'Mechanical engineering', 'Telecommunicatio
ns Engineering', 'Economic - Statistics', 'Psychology', 'Materials Science and Eng
ineering', 'Other scientific subjects', 'Biomedical Engineering', 'electronic Engi
neering', 'Information Engineering', 'Aeronautical/Aerospace/Astronautics Engineer
ing', 'Energy and Nuclear Engineering', 'Informatics', 'Management Engineering', '
Automotive Engineering', 'industrial engineering', 'Other', 'Surveyor', 'Civil/Civ
il and Environmental Engineering', 'Electrical Engineering', 'Scientific maturit
y', 'Chemist - Pharmaceutical', 'Political-Social', 'Other humanities subjects', '
Geo-Biological', 'Linguistics', 'Agriculture and veterinary', 'Literary', 'Humanis
tic high school diploma', 'Accounting', 'Communication Sciences', 'Safety Engineer
ing', 'Architecture', 'Mathematics', 'construction Engineering', 'Petroleum Engine
ering', 'Naval Engineering', 'Artistic', nan, 'Mathematical-physical modeling for
engineering', 'Engineering for the environment and the territory', 'Medical', 'Def
ense and Security', 'Physical education', 'Statistics', 'Educational/training scie
nces']
Feature:  Study Title -- ['Five-year degree', 'Doctorate', 'High school graduatio
n', 'Three-year degree', "master's degree", 'Professional qualification', 'Middle
school diploma']
Feature:  Years Experience -- ['[1-3]', '[7-10]', '[3-5]', '[5-7]', '[0]', '[+1
0]', '[0-1]']
Feature:  Sector -- ['Automotive', 'Aeronautics', 'Consulting', 'Telecom', 'Other
s', 'Space', 'Life sciences', nan, 'Railway', 'Defence', 'Naval', 'Services and In
formation Systems', 'Energy', 'Machining - Heavy Industry', 'Oil and Gas']
Feature:  Job Family Hiring -- ['Engineering', nan, 'Support', 'Tech Consulting &
Solutions', 'Business & Sales', 'Enabling Function', 'Business Management & Sale
```

```
s', 'Talent Acquisition']
Feature:  Job Title Hiring -- ['Consultant', nan, 'Support Advanced', 'Support', '
Qualified Consultant', 'Team Leader', 'Advanced Consultant', 'Business Manager', '
Junior Consultant', '???', 'Junior Business Manager', 'Senior Consultant', 'Busine
ss Unit Director', 'Pharmaceutical Consultant', 'Technician', 'Senior Sales Manage
r', 'Advanced Business Manager', 'Talent Acquisition Specialist', 'Specialist', 'S
ourcing Specialist']
Feature:  event_feedback -- [nan, 'OK', 'KO (technical skills)', 'OK (live)', 'OK
(waiting for departure)', 'KO (opportunity closed)', 'OK (other candidate)', 'KO (
proposed renunciation)', 'KO (manager)', 'KO (retired)', 'KO (mobility)', 'OK (hir
ed)', 'KO (seniority)', 'KO (ral)', 'KO (lost availability)', 'KO (language skill
s)']
Feature:  Overall -- [nan, '~ 2 - Medium', '~ 3 - High', '2 - Medium', '~ 4 - To
p', '~ 1 - Low', '3 - High', '1 - Low', '4 - Top']
Feature:  Minimum Ral -- ['26-28K', nan, '22-24K', '28-30K', 'Not Avail.', '38-40
K', '24-26K', '20-22K', '30-32K', '32-34K', '36-38K', '+50K', '- 20K', '40-42K', '
34-36K', '20K']
Feature:  Ral Maximum -- ['30-32K', nan, '- 20K', '28-30K', '32-34K', '24-26K', '3
8-40K', '26-28K', 'Not Avail.', '+50K', '36-38K', '34-36K', '48-50K', '42-44K', '4
0-42K', '22-24K', '20-22K', '44-46K', '20K']
Feature:  Study Level -- ['Five-year degree', nan, "master's degree", 'Three-year
degree', 'High school graduation', 'Professional qualification', 'Middle school di
ploma', 'Doctorate']
Feature: Current Ral -- ['22-24 K', '24-26 K', nan, '20-22 K', '28-30 K', '- 20
K', '36-38 K', '40-42 K', '34-36 K', '30-32 K', '46-48 K', '32-34 K', 'Not availab
le', '+ 50 K', '26-28 K', '38-40 K', '44-46 K', '48-50 K', '42-44 K']
Feature: Expected Ral -- ['24-26 K', '26-28 K', nan, '20-22 K', '28-30 K', '22-24
K', '- 20 K', '30-32 K', '44-46 K', '40-42 K', '32-34 K', '34-36 K', 'Not availabl
e', '+ 50 K', '36-38 K', '48-50 K', '38-40 K', '46-48 K', '42-44 K']
Feature: Technical Skills -- [nan, 2.0, 3.0, 1.0, 4.0]
Feature: Comunication -- [nan, 1.0, 2.0, 3.0, 4.0]
Feature: Maturity -- [nan, 2.0, 3.0, 1.0, 4.0]
Feature: Dynamism -- [nan, 2.0, 3.0, 1.0, 4.0]
Feature: Mobility -- [nan, 3.0, 2.0, 1.0, 4.0]
Feature: English -- [nan, 3.0, 4.0, 2.0, 1.0]
```

1.3 Handle the NANs

```
columns_with_nan = df.columns[df.isnull().any()].tolist()
print(columns_with_nan)
```

```
[' Citizenship', ' Protected category', ' Study area', ' Sector', ' Job Family Hir
ing', ' Job Title Hiring', ' event_feedback', ' Overall', ' Minimum Ral', ' Ral Ma
ximum', ' Study Level', 'Current Ral', 'Expected Ral', 'Technical Skills', 'Comuni
cation', 'Maturity', 'Dynamism', 'Mobility', 'English']
```

```
# Handle NaNs
df[' Citizenship'] = df[' Citizenship'].fillna('Not Specified')

df[' Protected category'] = df[' Protected category'].fillna('Not a protected categ
df[' Protected category'] = df[' Protected category'].replace('Article 18', 'Articl

df[' Study area'] = df[' Study area'].fillna('Not Specified')
df[' Sector'] = df[' Sector'].fillna('Unemployed')
df[' Job Family Hiring'] = df[' Job Family Hiring'].fillna('Not Specified')
df[' Job Title Hiring'] = df[' Job Title Hiring'].fillna('Not Specified')
df[' event_feedback'] = df[' event_feedback'].fillna('Not Specified')
```

```
df[' Overall'] = df[' Overall'].fillna('Not Specified')
df[' Minimum Ral'] = df[' Minimum Ral'].fillna('Not Specified')
df[' Ral Maximum'] = df[' Ral Maximum'].fillna('Not Specified')
df[' Study Level'] = df[' Study Level'].fillna('Not Specified')
df['Current Ral'] = df['Current Ral'].fillna('Not Specified')
df['Expected Ral'] = df['Expected Ral'].fillna('Not Specified')
df['Technical Skills'] = df['Technical Skills'].fillna(df['Technical Skills'].mean(
df['Comunication'] = df['Comunication'].fillna(df['Comunication'].mean())
df['Maturity'] = df['Maturity'].fillna(df['Maturity'].mean())
df['Dynamism'] = df['Dynamism'].fillna(df['Dynamism'].mean())
df['Mobility'] = df['Mobility'].fillna(df['Mobility'].mean())
df['English'] = df['English'].fillna(df['English'].mean())
```

```
# check
print(f'There are {df.isnull().sum().sum()} NANs')
```

```
There are 0 NANs
```

```
df.head()
```

	Candidate State	Age Range	Citizenship	Sex	Protected category	Study area	Study Title	Year Experienc
0	Hired	31 - 35 years	Pakistani	Male	Not a protected category	Automation/Mechatronics Engineering	Five-year degree	[1-3
1	Hired	31 - 35 years	Pakistani	Male	Not a protected category	Automation/Mechatronics Engineering	Five-year degree	[1-3
2	Hired	31 - 35 years	Pakistani	Male	Not a protected category	Automation/Mechatronics Engineering	Five-year degree	[1-3
3	Hired	31 - 35 years	Pakistani	Male	Not a protected category	Automation/Mechatronics Engineering	Five-year degree	[1-3
4	Hired	31 - 35 years	Pakistani	Male	Not a protected category	Automation/Mechatronics Engineering	Five-year degree	[1-3

5 rows × 24 columns

1.4 Work on the features

```
for feature in df.columns:
    print(f'Feature: {feature} -- {list(df[feature].unique())}')
```

```
Feature:  Candidate State -- ['Hired', 'Vivier', 'QM', 'In selection', 'First cont
act', 'Economic proposal', 'Imported']
Feature:  Age Range -- ['31 - 35 years', '40 - 45 years', '36 - 40 years', '> 45 y
ears', '26 - 30 years', '< 20 years', '20 - 25 years']
Feature:  Citizenship -- ['Pakistani', 'Italian', 'Not Specified', 'Moroccan', 'Ir
anian', 'Albanian', 'Indiana', 'Colombian', 'Ethiopian', 'Romanian', 'Vltava', 'Le
banese', 'Spanish', 'Egyptian', 'Russian', 'Tunisian', 'Turkish', 'Chinese', 'Uzbe
```

```
k', 'Brazilian', 'Cameroonian', 'Sudanese', 'Algerian', 'Croatian', 'Polish', 'Ind
onesian', 'San Marino', 'Argentina', 'Azerbaijan', 'Portuguese', 'Serbian', 'Frenc
h', 'Swiss', 'German', 'Peruvian', 'British', 'Venezuelan', 'Rwandan', 'Costa Rica
n', 'South Korean', 'Ukraine', 'Macedonian', 'Nigerian', 'American', 'Kenyan', 'Em
irati', 'Ecuadorian', 'Ivorian', 'Mexican', 'Chilean', 'Japanese', 'Syrian', 'Bang
ladeshis', 'Greek', 'Israeli', 'Omani', 'South African', 'Bolivian', 'Filipina', '
Sinhalese', 'Palestinian (Palestinian Territories)', 'Afghan', 'Jordan', 'Cuban',
'Vietnamese', 'Latvian', 'Libyan', 'Bulgarian', 'Togolese', 'Kazakh', 'Austrian',
'Belarusian', 'Saudi', 'Bosnian', 'Kyrgyz', 'Tajik', 'Dutch', 'Qatari', 'Georgia
n', 'Canadian', 'Australian', 'Salvadoran', 'Congolese', 'Guatemalan', 'Hungaria
n', 'Tanzanian', 'Gabonese', 'Angolan', 'Maltese']
Feature:  Sex -- ['Male', 'Female']
Feature:  Protected category -- ['Not a protected category', 'Article 1']
Feature:  Study area -- ['Automation/Mechatronics Engineering', 'computer engineer
ing', 'chemical engineering', 'Legal', 'Mechanical engineering', 'Telecommunicatio
ns Engineering', 'Economic - Statistics', 'Psychology', 'Materials Science and Eng
ineering', 'Other scientific subjects', 'Biomedical Engineering', 'electronic Engi
neering', 'Information Engineering', 'Aeronautical/Aerospace/Astronautics Engineer
ing', 'Energy and Nuclear Engineering', 'Informatics', 'Management Engineering', '
Automotive Engineering', 'industrial engineering', 'Other', 'Surveyor', 'Civil/Civ
il and Environmental Engineering', 'Electrical Engineering', 'Scientific maturit
y', 'Chemist - Pharmaceutical', 'Political-Social', 'Other humanities subjects', '
Geo-Biological', 'Linguistics', 'Agriculture and veterinary', 'Literary', 'Humanis
tic high school diploma', 'Accounting', 'Communication Sciences', 'Safety Engineer
ing', 'Architecture', 'Mathematics', 'construction Engineering', 'Petroleum Engine
ering', 'Naval Engineering', 'Artistic', 'Not Specified', 'Mathematical-physical m
odeling for engineering', 'Engineering for the environment and the territory', 'Me
dical', 'Defense and Security', 'Physical education', 'Statistics', 'Educational/t
raining sciences']
Feature:  Study Title -- ['Five-year degree', 'Doctorate', 'High school graduatio
n', 'Three-year degree', "master's degree", 'Professional qualification', 'Middle
school diploma']
Feature:  Years Experience -- ['[1-3]', '[7-10]', '[3-5]', '[5-7]', '[0]', '[+1
0]', '[0-1]']
Feature:  Sector -- ['Automotive', 'Aeronautics', 'Consulting', 'Telecom', 'Other
s', 'Space', 'Life sciences', 'Unemployed', 'Railway', 'Defence', 'Naval', 'Servic
es and Information Systems', 'Energy', 'Machining - Heavy Industry', 'Oil and Ga
s']
Feature:  Job Family Hiring -- ['Engineering', 'Not Specified', 'Support', 'Tech C
onsulting & Solutions', 'Business & Sales', 'Enabling Function', 'Business Managem
ent & Sales', 'Talent Acquisition']
Feature:  Job Title Hiring -- ['Consultant', 'Not Specified', 'Support Advanced',
'Support', 'Qualified Consultant', 'Team Leader', 'Advanced Consultant', 'Business
Manager', 'Junior Consultant', '???', 'Junior Business Manager', 'Senior Consultan
t', 'Business Unit Director', 'Pharmaceutical Consultant', 'Technician', 'Senior S
ales Manager', 'Advanced Business Manager', 'Talent Acquisition Specialist', 'Spec
ialist', 'Sourcing Specialist']
Feature:  event_feedback -- ['Not Specified', 'OK', 'KO (technical skills)', 'OK (
live)', 'OK (waiting for departure)', 'KO (opportunity closed)', 'OK (other candid
ate)', 'KO (proposed renunciation)', 'KO (manager)', 'KO (retired)', 'KO (mobilit
y)', 'OK (hired)', 'KO (seniority)', 'KO (ral)', 'KO (lost availability)', 'KO (la
nguage skills)']
Feature:  Overall -- ['Not Specified', '~ 2 - Medium', '~ 3 - High', '2 - Medium',
'~ 4 - Top', '~ 1 - Low', '3 - High', '1 - Low', '4 - Top']
Feature:  Minimum Ral -- ['26-28K', 'Not Specified', '22-24K', '28-30K', 'Not Avai
l.', '38-40K', '24-26K', '20-22K', '30-32K', '32-34K', '36-38K', '+50K', '- 20K',
'40-42K', '34-36K', '20K']
```

```
Feature:  Ral Maximum -- ['30-32K', 'Not Specified', '- 20K', '28-30K', '32-34K',
'24-26K', '38-40K', '26-28K', 'Not Avail.', '+50K', '36-38K', '34-36K', '48-50K',
'42-44K', '40-42K', '22-24K', '20-22K', '44-46K', '20K']
Feature:  Study Level -- ['Five-year degree', 'Not Specified', "master's degree",
'Three-year degree', 'High school graduation', 'Professional qualification', 'Midd
le school diploma', 'Doctorate']
Feature: Current Ral -- ['22-24 K', '24-26 K', 'Not Specified', '20-22 K', '28-30
K', '- 20 K', '36-38 K', '40-42 K', '34-36 K', '30-32 K', '46-48 K', '32-34 K', 'N
ot available', '+ 50 K', '26-28 K', '38-40 K', '44-46 K', '48-50 K', '42-44 K']
Feature: Expected Ral -- ['24-26 K', '26-28 K', 'Not Specified', '20-22 K', '28-30
K', '22-24 K', '- 20 K', '30-32 K', '44-46 K', '40-42 K', '32-34 K', '34-36 K', 'N
ot available', '+ 50 K', '36-38 K', '48-50 K', '38-40 K', '46-48 K', '42-44 K']
Feature: Technical Skills -- [2.1776658270361042, 2.0, 3.0, 1.0, 4.0]
Feature: Comunication -- [2.3200402144772116, 1.0, 2.0, 3.0, 4.0]
Feature: Maturity -- [2.2857142857142856, 2.0, 3.0, 1.0, 4.0]
Feature: Dynamism -- [2.2960603520536464, 2.0, 3.0, 1.0, 4.0]
Feature: Mobility -- [2.225979243388015, 3.0, 2.0, 1.0, 4.0]
Feature: English -- [2.7649730820995964, 3.0, 4.0, 2.0, 1.0]
```

```
citizenship_mapping = {
    'Pakistani' : 'Non-European',
    'Italian' : 'European',
    'Not Specified' : 'Non-European',
    'Moroccan' : 'Non-European',
    'Iranian' : 'Non-European',
    'Albanian' : 'European',
    'Indiana' : 'Non-European',
    'Colombian' : 'Non-European',
    'Ethiopian' : 'Non-European',
    'Romanian' : 'European',
    'Vltava' : 'European',
    'Lebanese' : 'Non-European',
    'Spanish' : 'European',
    'Egyptian' : 'Non-European',
    'Russian' : 'European',
    'Tunisian' : 'Non-European',
    'Turkish' : 'European',
    'Chinese' : 'Non-European',
    'Uzbek' : 'Non-European',
    'Brazilian' : 'Non-European',
    'Cameroonian' : 'Non-European',
    'Sudanese' : 'Non-European',
    'Algerian' : 'Non-European',
    'Croatian' : 'European',
    'Polish' : 'European',
    'Indonesian' : 'Non-European',
    'San Marino' : 'European',
    'Argentina' : 'Non-European',
    'Azerbaijan' : 'Non-European',
    'Portuguese' : 'European',
    'Serbian' : 'European',
    'French' : 'European',
    'Swiss' : 'European',
    'German' : 'European',
    'Peruvian' : 'Non-European',
    'British' : 'European',
    'Venezuelan' : 'Non-European',
```

```
    'Rwandan' : 'Non-European',
    'Costa Rican' : 'Non-European',
    'South Korean' : 'Non-European',
    'Ukraine' : 'European',
    'Macedonian' : 'European',
    'Nigerian' : 'Non-European',
    'American' : 'Non-European',
    'Kenyan' : 'Non-European',
    'Emirati' : 'Non-European',
    'Ecuadorian' : 'Non-European',
    'Ivorian' : 'Non-European',
    'Mexican' : 'Non-European',
    'Chilean' : 'Non-European',
    'Japanese' : 'Non-European',
    'Syrian' : 'Non-European',
    'Bangladeshis' : 'Non-European',
    'Greek' : 'European',
    'Israeli' : 'Non-European',
    'Omani' : 'Non-European',
    'South African' : 'Non-European',
    'Bolivian' : 'Non-European',
    'Filipina' : 'Non-European',
    'Sinhalese' : 'Non-European',
    'Palestinian (Palestinian Territories)' : 'Non-European',
    'Afghan' : 'Non-European',
    'Jordan' : 'Non-European',
    'Cuban' : 'Non-European',
    'Vietnamese' : 'Non-European',
    'Latvian' : 'European',
    'Libyan' : 'Non-European',
    'Bulgarian' : 'European',
    'Togolese' : 'Non-European',
    'Kazakh' : 'Non-European',
    'Austrian' : 'European',
    'Belarusian' : 'European',
    'Saudi' : 'Non-European',
    'Bosnian' : 'European',
    'Kyrgyz' : 'Non-European',
    'Tajik' : 'Non-European',
    'Dutch' : 'European',
    'Qatari' : 'Non-European',
    'Georgian' : 'European',
    'Canadian' : 'Non-European',
    'Australian' : 'Non-European',
    'Salvadoran' : 'Non-European',
    'Congolese' : 'Non-European',
    'Guatemalan' : 'Non-European',
    'Hungarian' : 'European',
    'Tanzanian' : 'Non-European',
    'Gabonese' : 'Non-European',
    'Angolan' : 'Non-European',
    'Maltese' : 'European'
}


study_area_mapping = {
    'Automation/Mechatronics Engineering' : 'Engineering',
```

```python
    'computer engineering' : 'Engineering',
    'chemical engineering' : 'Engineering',
    'Legal' : 'Law',
    'Mechanical engineering' : 'Engineering',
    'Telecommunications Engineering' : 'Engineering',
    'Economic - Statistics' : 'Economic',
    'Psychology' : 'Scientific Field',
    'Materials Science and Engineering' : 'Engineering',
    'Other scientific subjects' : 'Scientific Field',
    'Biomedical Engineering' : 'Engineering',
    'electronic Engineering' : 'Engineering',
    'Information Engineering' : 'Engineering',
    'Aeronautical/Aerospace/Astronautics Engineering' : 'Engineering',
    'Energy and Nuclear Engineering' : 'Engineering',
    'Informatics' : 'Informatics',
    'Management Engineering' : 'Engineering',
    'Automotive Engineering' : 'Engineering',
    'industrial engineering' : 'Engineering',
    'Other' : 'Other',
    'Surveyor' : 'NO COLLEGE',
    'Civil/Civil and Environmental Engineering' : 'Engineering',
    'Electrical Engineering' : 'Engineering',
    'Scientific maturity' : 'NO COLLEGE',
    'Chemist - Pharmaceutical' : 'Medical Field',
    'Political-Social' : 'Other Humanities Subjects',
    'Other humanities subjects' : 'Other Humanities Subjects',
    'Geo-Biological' : 'Scientific Field',
    'Linguistics' : 'Linguistics',
    'Agriculture and veterinary' : 'Scientific Field',
    'Literary' : 'Other Humanities Subjects',
    'Humanistic high school diploma' : 'NO COLLEGE',
    'Accounting' : 'NO COLLEGE',
    'Communication Sciences' : 'Other Humanities Subjects',
    'Safety Engineering' : 'Engineering',
    'Architecture' : 'Scientific Field',
    'Mathematics' : 'Scientific Field',
    'construction Engineering' : 'Engineering',
    'Petroleum Engineering' : 'Engineering',
    'Naval Engineering' : 'Engineering',
    'Artistic' : 'NO COLLEGE',
    'Not Specified' : 'Other',
    'Mathematical-physical modeling for engineering' : 'Engineering',
    'Engineering for the environment and the territory' : 'Engineering',
    'Medical' : 'Medical Field',
    'Defense and Security' : 'Other',
    'Physical education' : 'Other',
    'Statistics' : 'Scientific Field',
    'Educational/training sciences' : 'Other Humanities Subjects'

}

age_mapping = {
    '< 20 years': 'Young',
    '20 - 25 years': 'Young',
    '26 - 30 years': 'Young',
    '31 - 35 years': 'Young',
    '36 - 40 years': 'Senior',
```

```
    '40 - 45 years': 'Senior',
    '> 45 years': 'Senior'
}
```

```
df[' Citizenship'] = df[' Citizenship'].replace(citizenship_mapping)
df[' Age Range'] = df[' Age Range'].replace(age_mapping)
df[' Study area'] = df[' Study area'].replace(study_area_mapping)
```

```
df.head()
```

	Candidate State	Age Range	Citizenship	Sex	Protected category	Study area	Study Title	Years Experience	Secto
0	Hired	Young	Non-European	Male	Not a protected category	Engineering	Five-year degree	[1-3]	Automotiv
1	Hired	Young	Non-European	Male	Not a protected category	Engineering	Five-year degree	[1-3]	Automotiv
2	Hired	Young	Non-European	Male	Not a protected category	Engineering	Five-year degree	[1-3]	Automotiv
3	Hired	Young	Non-European	Male	Not a protected category	Engineering	Five-year degree	[1-3]	Automotiv
4	Hired	Young	Non-European	Male	Not a protected category	Engineering	Five-year degree	[1-3]	Automotiv

5 rows × 24 columns

1.5 Create the target column

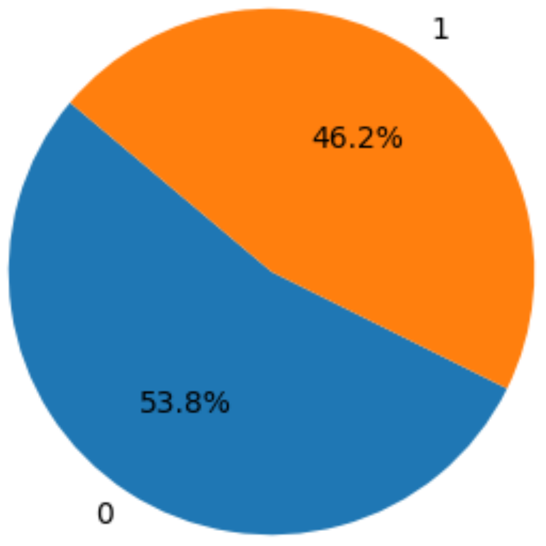
```
statuses_to_remove = ['First contact', 'Imported']
print(df[' Candidate State'].unique())
df = df[~df[' Candidate State'].isin(statuses_to_remove)]
print(df[' Candidate State'].unique())
df.shape
```

```
['Hired' 'Vivier' 'QM' 'In selection' 'First contact' 'Economic proposal'
 'Imported']
['Hired' 'Vivier' 'QM' 'In selection' 'Economic proposal']
```

```
(9857, 24)
```

```
df['STATUS'] = np.where((df[' Candidate State'] == 'Hired') | (df[' Candidate State

distribution = df['STATUS'].value_counts()

plt.figure(figsize=(4, 4))
plt.pie(distribution, labels=distribution.index, autopct='%1.1f%%', startangle=140)
plt.title(f'Distribution of the STATUS column') # 1 means the candidate is consider
plt.show()
```

Distribution of the STATUS column



1.6 Encode the categorical columns

```
categorical_columns = [' Age Range', ' Citizenship', ' Sex',
       ' Protected category', ' Study area', ' Study Title',
       ' Years Experience', ' Sector', ' Job Family Hiring',
       ' Job Title Hiring', ' Overall',
       ' Minimum Ral', ' Ral Maximum', ' Study Level',
       'Current Ral', 'Expected Ral']

encoding_mappings = {}

for column in categorical_columns:
    encoder = LabelEncoder()
    df[f'{column}_encoded'] = encoder.fit_transform(df[column])
    encoding_mappings[column] = dict(zip(encoder.classes_, encoder.transform(encode

df = df.drop(columns=categorical_columns)
df = df.drop(columns=[' Candidate State', ' event_feedback'])
```

```
# Get a look at the new dataset
print(f"The new columns of the dataset are: {df.columns}")
df.head()
```

```
The new columns of the dataset are: Index(['Technical Skills', 'Comunication', 'Ma
turity', 'Dynamism', 'Mobility',
       'English', 'STATUS', ' Age Range_encoded', ' Citizenship_encoded',
       ' Sex_encoded', ' Protected category_encoded', ' Study area_encoded',
       ' Study Title_encoded', ' Years Experience_encoded', ' Sector_encoded',
       ' Job Family Hiring_encoded', ' Job Title Hiring_encoded',
       ' Overall_encoded', ' Minimum Ral_encoded', ' Ral Maximum_encoded',
       ' Study Level_encoded', 'Current Ral_encoded', 'Expected Ral_encoded'],
      dtype='object')
```

	Technical Skills	Comunication	Maturity	Dynamism	Mobility	English	STATUS	A Range_encoc
0	2.177666	2.32004	2.285714	2.29606	2.225979	2.764973	1	
1	2.177666	2.32004	2.285714	2.29606	2.225979	2.764973	1	
2	2.177666	2.32004	2.285714	2.29606	2.225979	2.764973	1	
3	2.000000	1.00000	2.000000	2.00000	3.000000	3.000000	1	
4	2.177666	2.32004	2.285714	2.29606	2.225979	2.764973	1	

5 rows × 23 columns

```
# And at an example of encoding mapping
print(encoding_mappings)
```

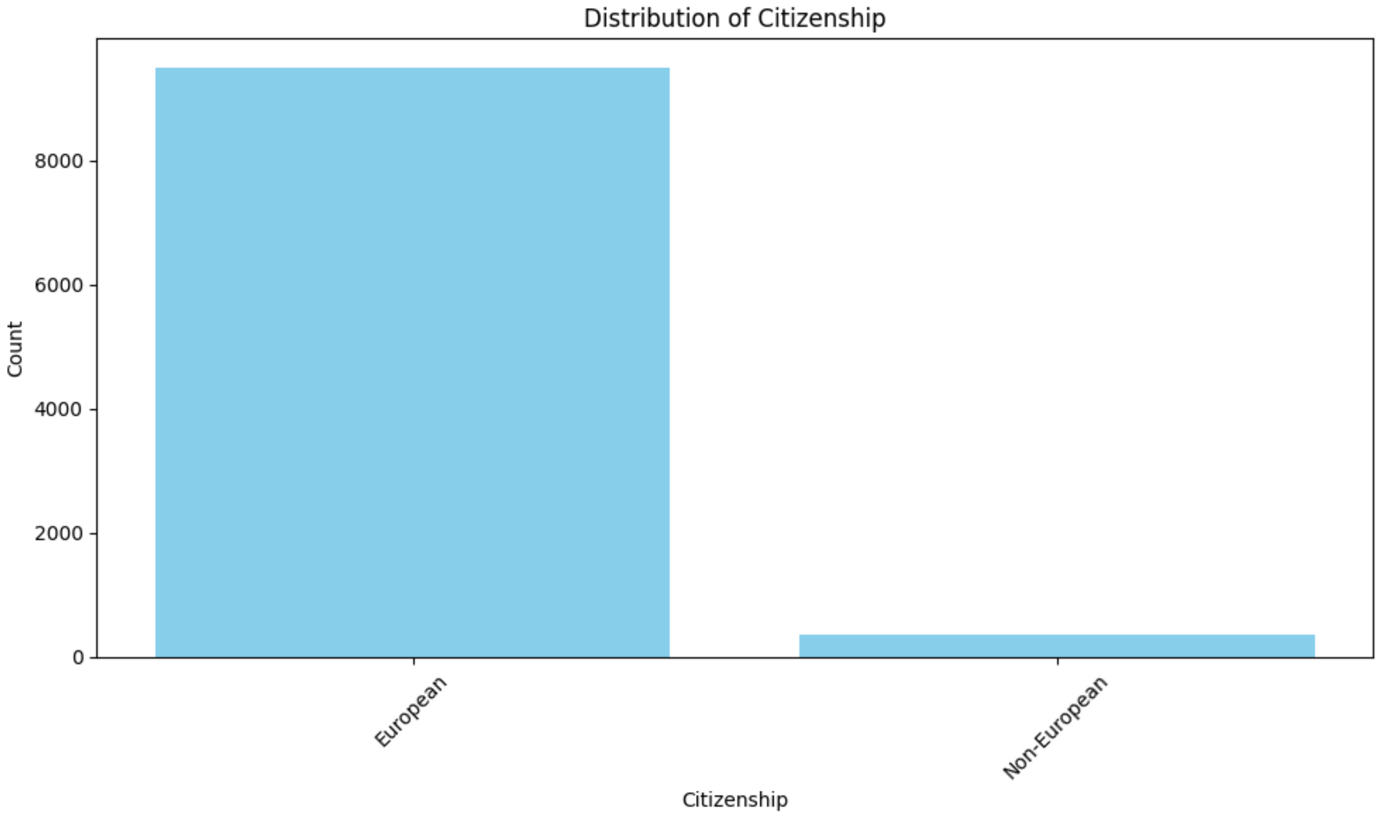
```
{' Age Range': {'Senior': 0, 'Young': 1}, ' Citizenship': {'European': 0, 'Non-Eur
opean': 1}, ' Sex': {'Female': 0, 'Male': 1}, ' Protected category': {'Article 1':
0, 'Not a protected category': 1}, ' Study area': {'Economic': 0, 'Engineering':
1, 'Informatics': 2, 'Law': 3, 'Linguistics': 4, 'Medical Field': 5, 'NO COLLEGE':
6, 'Other': 7, 'Other Humanities Subjects': 8, 'Scientific Field': 9}, ' Study Tit
le': {'Doctorate': 0, 'Five-year degree': 1, 'High school graduation': 2, 'Middle
school diploma': 3, 'Professional qualification': 4, 'Three-year degree': 5, "mast
er's degree": 6}, ' Years Experience': {'[+10]': 0, '[0-1]': 1, '[0]': 2, '[1-3]':
3, '[3-5]': 4, '[5-7]': 5, '[7-10]': 6}, ' Sector': {'Aeronautics': 0, 'Automotiv
e': 1, 'Consulting': 2, 'Defence': 3, 'Energy': 4, 'Life sciences': 5, 'Machining
- Heavy Industry': 6, 'Naval': 7, 'Oil and Gas': 8, 'Others': 9, 'Railway': 10, 'S
ervices and Information Systems': 11, 'Space': 12, 'Telecom': 13, 'Unemployed': 1
4}, ' Job Family Hiring': {'Business & Sales': 0, 'Business Management & Sales':
1, 'Enabling Function': 2, 'Engineering': 3, 'Not Specified': 4, 'Support': 5, 'Ta
lent Acquisition': 6, 'Tech Consulting & Solutions': 7}, ' Job Title Hiring':
{'???': 0, 'Advanced Business Manager': 1, 'Advanced Consultant': 2, 'Business Man
ager': 3, 'Business Unit Director': 4, 'Consultant': 5, 'Junior Business Manager':
6, 'Junior Consultant': 7, 'Not Specified': 8, 'Pharmaceutical Consultant': 9, 'Qu
alified Consultant': 10, 'Senior Consultant': 11, 'Senior Sales Manager': 12, 'Sou
rcing Specialist': 13, 'Specialist': 14, 'Support': 15, 'Support Advanced': 16, 'T
alent Acquisition Specialist': 17, 'Team Leader': 18, 'Technician': 19}, ' Overal
l': {'1 - Low': 0, '2 - Medium': 1, '3 - High': 2, '4 - Top': 3, 'Not Specified':
4, '~ 1 - Low': 5, '~ 2 - Medium': 6, '~ 3 - High': 7, '~ 4 - Top': 8}, ' Minimum
Ral': {'+50K': 0, '- 20K': 1, '20-22K': 2, '20K': 3, '22-24K': 4, '24-26K': 5, '26
-28K': 6, '28-30K': 7, '30-32K': 8, '32-34K': 9, '34-36K': 10, '36-38K': 11, '38-4
0K': 12, '40-42K': 13, 'Not Avail.': 14, 'Not Specified': 15}, ' Ral Maximum': {'+
50K': 0, '- 20K': 1, '20-22K': 2, '20K': 3, '22-24K': 4, '24-26K': 5, '26-28K': 6,
'28-30K': 7, '30-32K': 8, '32-34K': 9, '34-36K': 10, '36-38K': 11, '38-40K': 12, '
40-42K': 13, '42-44K': 14, '44-46K': 15, '48-50K': 16, 'Not Avail.': 17, 'Not Spec
ified': 18}, ' Study Level': {'Doctorate': 0, 'Five-year degree': 1, 'High school
graduation': 2, 'Middle school diploma': 3, 'Not Specified': 4, 'Professional qual
ification': 5, 'Three-year degree': 6, "master's degree": 7}, 'Current Ral': {'+ 5
0 K': 0, '- 20 K': 1, '20-22 K': 2, '22-24 K': 3, '24-26 K': 4, '26-28 K': 5, '28-
30 K': 6, '30-32 K': 7, '32-34 K': 8, '34-36 K': 9, '36-38 K': 10, '38-40 K': 11,
'40-42 K': 12, '42-44 K': 13, '44-46 K': 14, '46-48 K': 15, '48-50 K': 16, 'Not Sp
ecified': 17, 'Not available': 18}, 'Expected Ral': {'+ 50 K': 0, '- 20 K': 1, '20
-22 K': 2, '22-24 K': 3, '24-26 K': 4, '26-28 K': 5, '28-30 K': 6, '30-32 K': 7, '
32-34 K': 8, '34-36 K': 9, '36-38 K': 10, '38-40 K': 11, '40-42 K': 12, '42-44 K':
13, '44-46 K': 14, '46-48 K': 15, '48-50 K': 16, 'Not Specified': 17, 'Not availab
le': 18}}
```


1.7 Visualize data (TBD)

Citizenship

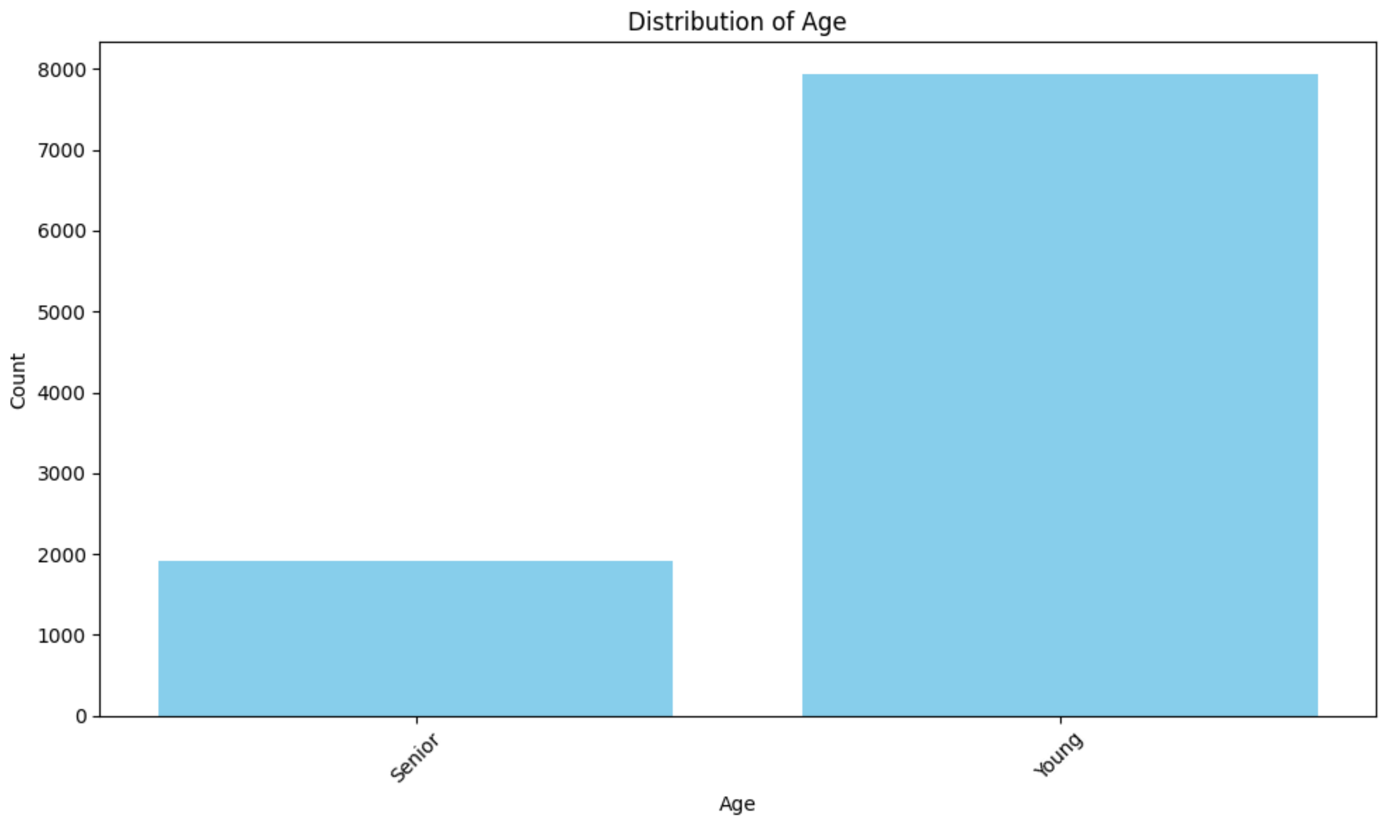
```python
citizenship_mapping = {v: k for k, v in encoding_mappings[' Citizenship'].items()}
distribution = df[' Citizenship_encoded'].value_counts()
plt.figure(figsize=(10, 6))
plt.bar(distribution.index, distribution.values, color='skyblue')

# Replace the x-tick labels with the mapped values
plt.xticks(distribution.index, distribution.index.map(citizenship_mapping), rotatio

plt.title('Distribution of Citizenship')
plt.xlabel('Citizenship')
plt.ylabel('Count')
plt.tight_layout()
```

```
plt.show()
```



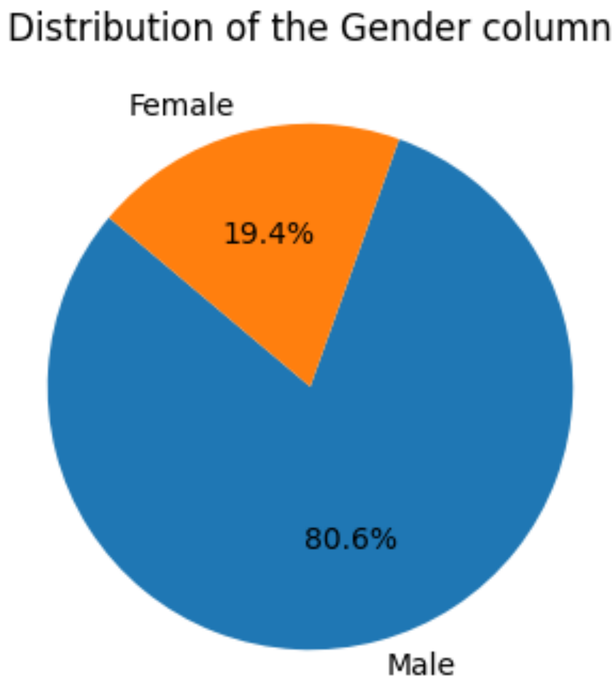
Age

```
age_mapping = {v: k for k, v in encoding_mappings[' Age Range'].items()}
distribution = df[' Age Range_encoded'].value_counts()
plt.figure(figsize=(10, 6))
plt.bar(distribution.index, distribution.values, color='skyblue')

# Replace the x-tick labels with the mapped values
plt.xticks(distribution.index, distribution.index.map(age_mapping), rotation=45)

plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

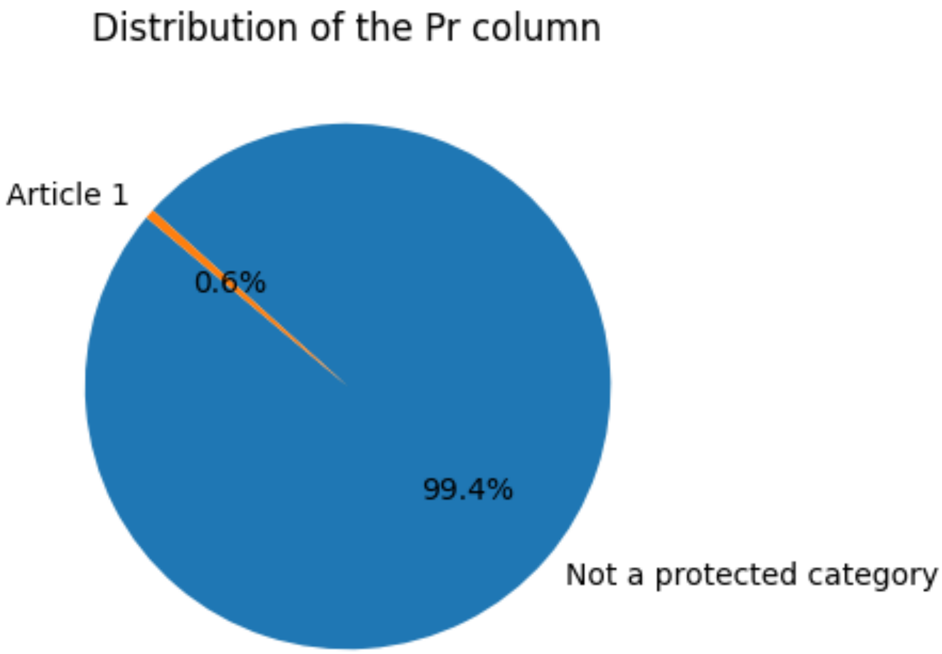


Gender:

```
gender_mapping = {v: k for k, v in encoding_mappings[' Sex'].items()}
distribution = df[' Sex_encoded'].value_counts()
distribution.index = distribution.index.map(gender_mapping)

plt.figure(figsize=(4, 4))
plt.pie(distribution, labels=distribution.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of the Gender column')
plt.show()
```

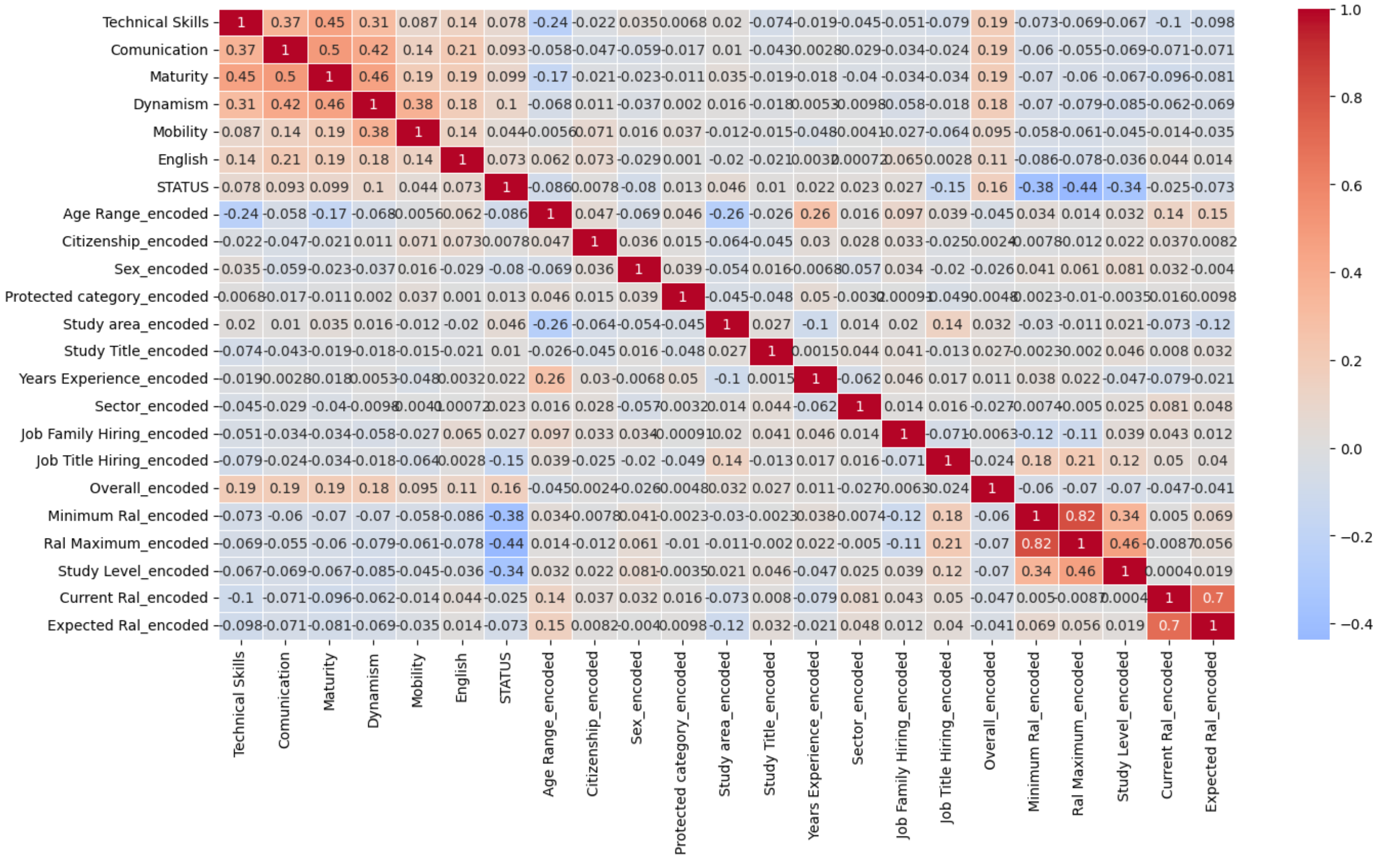


```python
gender_mapping = {v: k for k, v in encoding_mappings[' Protected category'].items()}
distribution = df[' Protected category_encoded'].value_counts()
distribution.index = distribution.index.map(gender_mapping)

plt.figure(figsize=(4, 4))
plt.pie(distribution, labels=distribution.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of the Pr column')
plt.show()
```



Correlation Matrix

```python
# Correlation matrix
corr_matrix = df.corr()
plt.figure(figsize=(16, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0, linewidths=.5)
plt.show()
```



1.8 Visualize percentage of hired inside each class

```
sensitive_features = [' Sex_encoded', ' Age Range_encoded', ' Citizenship_encoded',
for feature in sensitive_features:
    for i in range(len(list(df[feature].unique()))):
        total_elements = len(df[(df[feature] == i) & (df['STATUS'] == 1)])
        total_age = len(df[df[feature] == i])
        percentage = (total_elements / total_age) * 100
        print(f"Percentage of elements where {feature} is {i} and STATUS is HIRED:
```

```
Percentage of elements where  Sex_encoded is 0 and STATUS is HIRED: 54.32%
Percentage of elements where  Sex_encoded is 1 and STATUS is HIRED: 44.26%
Percentage of elements where  Age Range_encoded is 0 and STATUS is HIRED: 54.95%
Percentage of elements where  Age Range_encoded is 1 and STATUS is HIRED: 44.10%
Percentage of elements where  Citizenship_encoded is 0 and STATUS is HIRED: 46.13%
Percentage of elements where  Citizenship_encoded is 1 and STATUS is HIRED: 48.21%
Percentage of elements where  Protected category_encoded is 0 and STATUS is HIRED:
37.93%
Percentage of elements where  Protected category_encoded is 1 and STATUS is HIRED:
46.26%
```

```
encoding_mappings
```

```
{' Age Range': {'Senior': 0, 'Young': 1},
 ' Citizenship': {'European': 0, 'Non-European': 1},
 ' Sex': {'Female': 0, 'Male': 1},
 ' Protected category': {'Article 1': 0, 'Not a protected category': 1},
 ' Study area': {'Economic': 0,
  'Engineering': 1,
  'Informatics': 2,
  'Law': 3,
  'Linguistics': 4,
```

```
 'Medical Field': 5,
 'NO COLLEGE': 6,
 'Other': 7,
 'Other Humanities Subjects': 8,
 'Scientific Field': 9},
' Study Title': {'Doctorate': 0,
 'Five-year degree': 1,
 'High school graduation': 2,
 'Middle school diploma': 3,
 'Professional qualification': 4,
 'Three-year degree': 5,
 "master's degree": 6},
' Years Experience': {'[+10]': 0,
 '[0-1]': 1,
 '[0]': 2,
 '[1-3]': 3,
 '[3-5]': 4,
 '[5-7]': 5,
 '[7-10]': 6},
' Sector': {'Aeronautics': 0,
 'Automotive': 1,
 'Consulting': 2,
 'Defence': 3,
 'Energy': 4,
 'Life sciences': 5,
 'Machining - Heavy Industry': 6,
 'Naval': 7,
 'Oil and Gas': 8,
 'Others': 9,
 'Railway': 10,
 'Services and Information Systems': 11,
 'Space': 12,
 'Telecom': 13,
 'Unemployed': 14},
' Job Family Hiring': {'Business & Sales': 0,
 'Business Management & Sales': 1,
 'Enabling Function': 2,
 'Engineering': 3,
 'Not Specified': 4,
 'Support': 5,
 'Talent Acquisition': 6,
 'Tech Consulting & Solutions': 7},
' Job Title Hiring': {'???': 0,
 'Advanced Business Manager': 1,
 'Advanced Consultant': 2,
 'Business Manager': 3,
 'Business Unit Director': 4,
 'Consultant': 5,
 'Junior Business Manager': 6,
 'Junior Consultant': 7,
 'Not Specified': 8,
 'Pharmaceutical Consultant': 9,
 'Qualified Consultant': 10,
 'Senior Consultant': 11,
 'Senior Sales Manager': 12,
 'Sourcing Specialist': 13,
 'Specialist': 14,
```

```
 'Support': 15,
 'Support Advanced': 16,
 'Talent Acquisition Specialist': 17,
 'Team Leader': 18,
 'Technician': 19},
' Overall': {'1 - Low': 0,
 '2 - Medium': 1,
 '3 - High': 2,
 '4 - Top': 3,
 'Not Specified': 4,
 '~ 1 - Low': 5,
 '~ 2 - Medium': 6,
 '~ 3 - High': 7,
 '~ 4 - Top': 8},
' Minimum Ral': {'+50K': 0,
 '- 20K': 1,
 '20-22K': 2,
 '20K': 3,
 '22-24K': 4,
 '24-26K': 5,
 '26-28K': 6,
 '28-30K': 7,
 '30-32K': 8,
 '32-34K': 9,
 '34-36K': 10,
 '36-38K': 11,
 '38-40K': 12,
 '40-42K': 13,
 'Not Avail.': 14,
 'Not Specified': 15},
' Ral Maximum': {'+50K': 0,
 '- 20K': 1,
 '20-22K': 2,
 '20K': 3,
 '22-24K': 4,
 '24-26K': 5,
 '26-28K': 6,
 '28-30K': 7,
 '30-32K': 8,
 '32-34K': 9,
 '34-36K': 10,
 '36-38K': 11,
 '38-40K': 12,
 '40-42K': 13,
 '42-44K': 14,
 '44-46K': 15,
 '48-50K': 16,
 'Not Avail.': 17,
 'Not Specified': 18},
' Study Level': {'Doctorate': 0,
 'Five-year degree': 1,
 'High school graduation': 2,
 'Middle school diploma': 3,
 'Not Specified': 4,
 'Professional qualification': 5,
 'Three-year degree': 6,
 "master's degree": 7},
```

```
 'Current Ral': {'+ 50 K': 0,
  '- 20 K': 1,
  '20-22 K': 2,
  '22-24 K': 3,
  '24-26 K': 4,
  '26-28 K': 5,
  '28-30 K': 6,
  '30-32 K': 7,
  '32-34 K': 8,
  '34-36 K': 9,
  '36-38 K': 10,
  '38-40 K': 11,
  '40-42 K': 12,
  '42-44 K': 13,
  '44-46 K': 14,
  '46-48 K': 15,
  '48-50 K': 16,
  'Not Specified': 17,
  'Not available': 18},
 'Expected Ral': {'+ 50 K': 0,
  '- 20 K': 1,
  '20-22 K': 2,
  '22-24 K': 3,
  '24-26 K': 4,
  '26-28 K': 5,
  '28-30 K': 6,
  '30-32 K': 7,
  '32-34 K': 8,
  '34-36 K': 9,
  '36-38 K': 10,
  '38-40 K': 11,
  '40-42 K': 12,
  '42-44 K': 13,
  '44-46 K': 14,
  '46-48 K': 15,
  '48-50 K': 16,
  'Not Specified': 17,
  'Not available': 18}}
```

Task 2 - Algorithms

```
# Shuffle the dataset
df = shuffle(df, random_state=random_seed)

# Split in X and y
X = df.drop(columns=['STATUS'])
y = df['STATUS']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

2.1 Machine Learning models

```
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeClassifier(),
```

```
    'Naive Bayes': GaussianNB(),
    'XGBoost': XGBClassifier(),
    'KNN': KNeighborsClassifier(),
}
```

```
metrics = []
predictions = {}

# Fit models and evaluate
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    if name in ['Linear Regression', 'XGBoost']:
        y_pred = (y_pred > 0.5).astype(int)

    # Store predictions
    predictions[name] = y_pred

    accuracy = round(accuracy_score(y_test, y_pred), 3)
    precision = round(precision_score(y_test, y_pred), 3)
    recall = round(recall_score(y_test, y_pred), 3)
    f1 = round(f1_score(y_test, y_pred), 3)
    roc_auc = round(roc_auc_score(y_test, y_pred), 3)

    # Append metrics to the DataFrame
    metrics.append({
        'Model': name,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-score': f1,
        'ROC AUC': roc_auc
    })

metrics = pd.DataFrame(metrics)
metrics.head()
```

	Model	Accuracy	Precision	Recall	F1-score	ROC AUC
0	Linear Regression	0.727	0.913	0.470	0.620	0.715
1	Decision Tree	0.782	0.781	0.752	0.766	0.781
2	Naive Bayes	0.779	1.000	0.534	0.696	0.767
3	XGBoost	0.816	0.900	0.690	0.781	0.810
4	KNN	0.789	0.820	0.711	0.762	0.785

```
predictions_df = pd.DataFrame({
    'Linear Regression' : predictions['Linear Regression'],
    'Decision Tree' : predictions['Decision Tree'],
    'Naive Bayes' : predictions['Naive Bayes'],
    'XGBoost' : predictions['XGBoost'],
    'kNN' : predictions['KNN']
})
```

2.2 Neural Network

```
def create_model():
    model = Sequential()
    model.add(Dense(128, input_dim=22, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dense(128, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dense(64, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dense(1, activation='sigmoid'))

    # Compile the model
    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

# List to hold the models
neural_models = []

# Create and compile 7 models with different seeds
for seed in range(85,92):
    np.random.seed(seed)
    tf.random.set_seed(seed)
    model = create_model()
    neural_models.append(model)
```

```
# Fit the models
histories = []
for i, model in enumerate(neural_models):
    print(f"Fitting model {i+1}...")
    history = model.fit(X_train, y_train, epochs=15, batch_size=64, validation_spli
    histories.append(history)
    print(f"Model {i+1} fitted.\n")
```

```
Fitting model 1...
Epoch 1/15
99/99 [==============================] - 2s 6ms/step - loss: 0.5270 - accuracy: 0.
7319 - val_loss: 0.7303 - val_accuracy: 0.4838
Epoch 2/15
99/99 [==============================] - 0s 4ms/step - loss: 0.4764 - accuracy: 0.
7600 - val_loss: 0.5367 - val_accuracy: 0.6874
Epoch 3/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4635 - accuracy: 0.
7671 - val_loss: 0.4671 - val_accuracy: 0.7793
Epoch 4/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4491 - accuracy: 0.
7755 - val_loss: 0.4686 - val_accuracy: 0.7622
Epoch 5/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4518 - accuracy: 0.
7712 - val_loss: 0.4457 - val_accuracy: 0.7907
Epoch 6/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4458 - accuracy: 0.
```

```
7765 - val_loss: 0.4792 - val_accuracy: 0.7565
Epoch 7/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4398 - accuracy: 0.
7820 - val_loss: 0.4698 - val_accuracy: 0.7635
Epoch 8/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4333 - accuracy: 0.
7831 - val_loss: 0.4534 - val_accuracy: 0.7838
Epoch 9/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4315 - accuracy: 0.
7866 - val_loss: 0.4573 - val_accuracy: 0.7654
Epoch 10/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4278 - accuracy: 0.
7866 - val_loss: 0.4455 - val_accuracy: 0.7635
Epoch 11/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4233 - accuracy: 0.
7919 - val_loss: 0.4574 - val_accuracy: 0.7964
Epoch 12/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4235 - accuracy: 0.
7904 - val_loss: 0.4712 - val_accuracy: 0.7248
Epoch 13/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4204 - accuracy: 0.
7952 - val_loss: 0.4419 - val_accuracy: 0.7888
Epoch 14/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4195 - accuracy: 0.
7923 - val_loss: 0.4538 - val_accuracy: 0.7736
Epoch 15/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4148 - accuracy: 0.
7945 - val_loss: 0.4575 - val_accuracy: 0.7597
Model 1 fitted.

Fitting model 2...
Epoch 1/15
99/99 [==============================] - 2s 5ms/step - loss: 0.5382 - accuracy: 0.
7170 - val_loss: 0.6687 - val_accuracy: 0.5073
Epoch 2/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4751 - accuracy: 0.
7636 - val_loss: 0.5254 - val_accuracy: 0.7191
Epoch 3/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4659 - accuracy: 0.
7722 - val_loss: 0.4709 - val_accuracy: 0.7781
Epoch 4/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4481 - accuracy: 0.
7820 - val_loss: 0.4703 - val_accuracy: 0.7679
Epoch 5/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4491 - accuracy: 0.
7738 - val_loss: 0.4732 - val_accuracy: 0.7724
Epoch 6/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4498 - accuracy: 0.
7757 - val_loss: 0.4497 - val_accuracy: 0.7844
Epoch 7/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4415 - accuracy: 0.
7869 - val_loss: 0.4570 - val_accuracy: 0.7838
Epoch 8/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4347 - accuracy: 0.
7812 - val_loss: 0.4436 - val_accuracy: 0.7895
Epoch 9/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4289 - accuracy: 0.
```

```
7860 - val_loss: 0.5429 - val_accuracy: 0.6595
Epoch 10/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4242 - accuracy: 0.
7901 - val_loss: 0.4747 - val_accuracy: 0.7812
Epoch 11/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4262 - accuracy: 0.
7877 - val_loss: 0.4553 - val_accuracy: 0.7869
Epoch 12/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4238 - accuracy: 0.
7941 - val_loss: 0.4677 - val_accuracy: 0.7267
Epoch 13/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4199 - accuracy: 0.
7939 - val_loss: 0.4479 - val_accuracy: 0.7660
Epoch 14/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4168 - accuracy: 0.
7949 - val_loss: 0.4665 - val_accuracy: 0.7438
Epoch 15/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4148 - accuracy: 0.
7980 - val_loss: 0.4461 - val_accuracy: 0.7647
Model 2 fitted.

Fitting model 3...
Epoch 1/15
99/99 [==============================] - 2s 5ms/step - loss: 0.5285 - accuracy: 0.
7319 - val_loss: 0.6120 - val_accuracy: 0.6221
Epoch 2/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4782 - accuracy: 0.
7643 - val_loss: 0.5881 - val_accuracy: 0.6392
Epoch 3/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4619 - accuracy: 0.
7724 - val_loss: 0.4790 - val_accuracy: 0.7711
Epoch 4/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4530 - accuracy: 0.
7765 - val_loss: 0.4667 - val_accuracy: 0.7578
Epoch 5/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4493 - accuracy: 0.
7762 - val_loss: 0.4506 - val_accuracy: 0.7749
Epoch 6/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4485 - accuracy: 0.
7803 - val_loss: 0.4645 - val_accuracy: 0.7546
Epoch 7/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4388 - accuracy: 0.
7804 - val_loss: 0.4711 - val_accuracy: 0.7476
Epoch 8/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4316 - accuracy: 0.
7876 - val_loss: 0.4623 - val_accuracy: 0.7654
Epoch 9/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4305 - accuracy: 0.
7899 - val_loss: 0.4555 - val_accuracy: 0.7685
Epoch 10/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4278 - accuracy: 0.
7876 - val_loss: 0.4419 - val_accuracy: 0.7869
Epoch 11/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4230 - accuracy: 0.
7907 - val_loss: 0.4647 - val_accuracy: 0.7831
Epoch 12/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4254 - accuracy: 0.
```

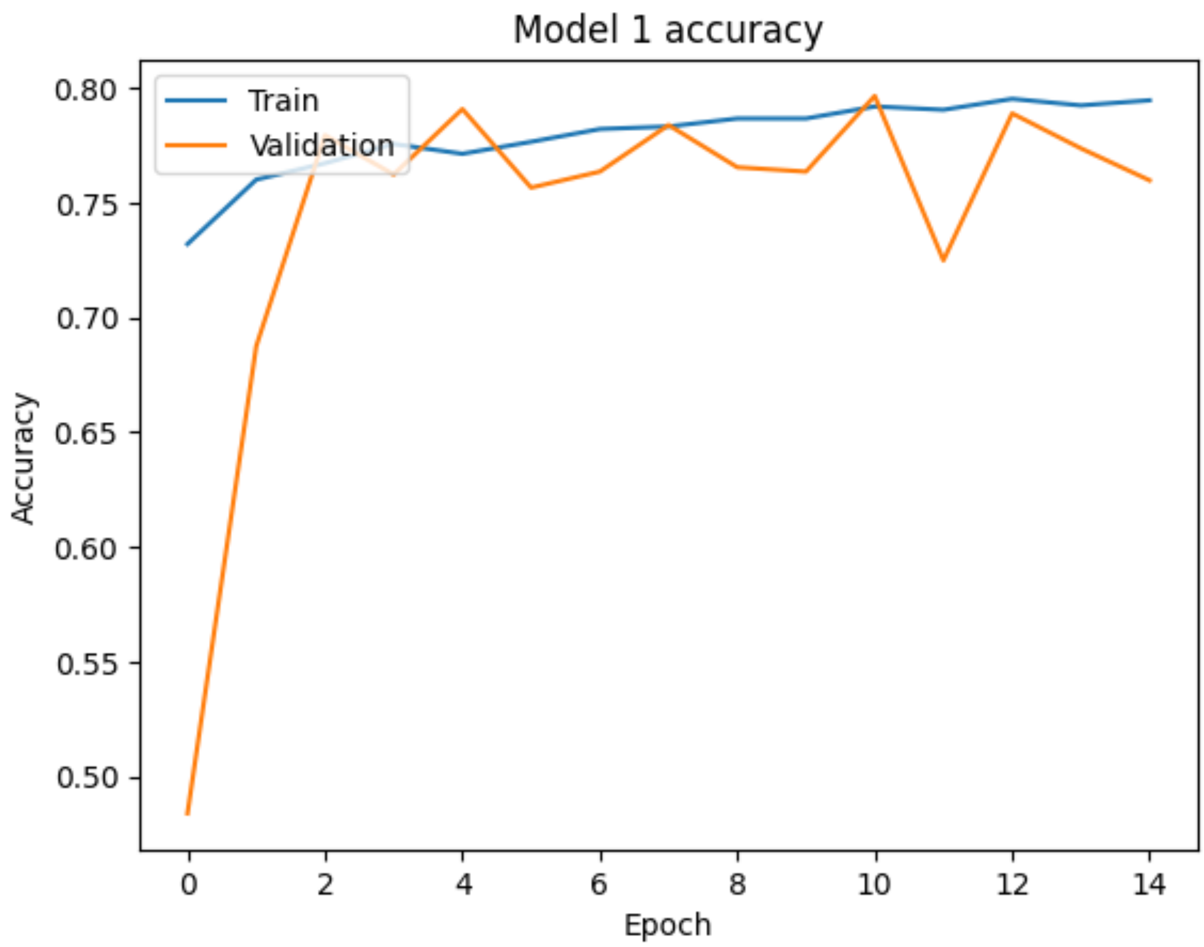
```
7899 - val_loss: 0.4929 - val_accuracy: 0.7292
Epoch 13/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4120 - accuracy: 0.
8050 - val_loss: 0.4542 - val_accuracy: 0.7647
Epoch 14/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4132 - accuracy: 0.
7995 - val_loss: 0.4738 - val_accuracy: 0.7223
Epoch 15/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4117 - accuracy: 0.
7974 - val_loss: 0.4615 - val_accuracy: 0.7483
Model 3 fitted.

Fitting model 4...
Epoch 1/15
99/99 [==============================] - 2s 5ms/step - loss: 0.5335 - accuracy: 0.
7240 - val_loss: 0.6387 - val_accuracy: 0.5688
Epoch 2/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4738 - accuracy: 0.
7649 - val_loss: 0.5473 - val_accuracy: 0.6842
Epoch 3/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4665 - accuracy: 0.
7652 - val_loss: 0.4865 - val_accuracy: 0.7578
Epoch 4/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4479 - accuracy: 0.
7762 - val_loss: 0.4777 - val_accuracy: 0.7476
Epoch 5/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4479 - accuracy: 0.
7808 - val_loss: 0.4552 - val_accuracy: 0.7787
Epoch 6/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4467 - accuracy: 0.
7790 - val_loss: 0.4513 - val_accuracy: 0.7831
Epoch 7/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4364 - accuracy: 0.
7873 - val_loss: 0.4497 - val_accuracy: 0.7895
Epoch 8/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4296 - accuracy: 0.
7890 - val_loss: 0.5045 - val_accuracy: 0.7223
Epoch 9/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4316 - accuracy: 0.
7863 - val_loss: 0.4555 - val_accuracy: 0.7768
Epoch 10/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4246 - accuracy: 0.
7969 - val_loss: 0.4749 - val_accuracy: 0.7762
Epoch 11/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4258 - accuracy: 0.
7912 - val_loss: 0.4685 - val_accuracy: 0.7724
Epoch 12/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4214 - accuracy: 0.
7931 - val_loss: 0.5212 - val_accuracy: 0.6886
Epoch 13/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4152 - accuracy: 0.
7971 - val_loss: 0.4846 - val_accuracy: 0.7413
Epoch 14/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4133 - accuracy: 0.
7976 - val_loss: 0.4878 - val_accuracy: 0.7286
Epoch 15/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4081 - accuracy: 0.
```

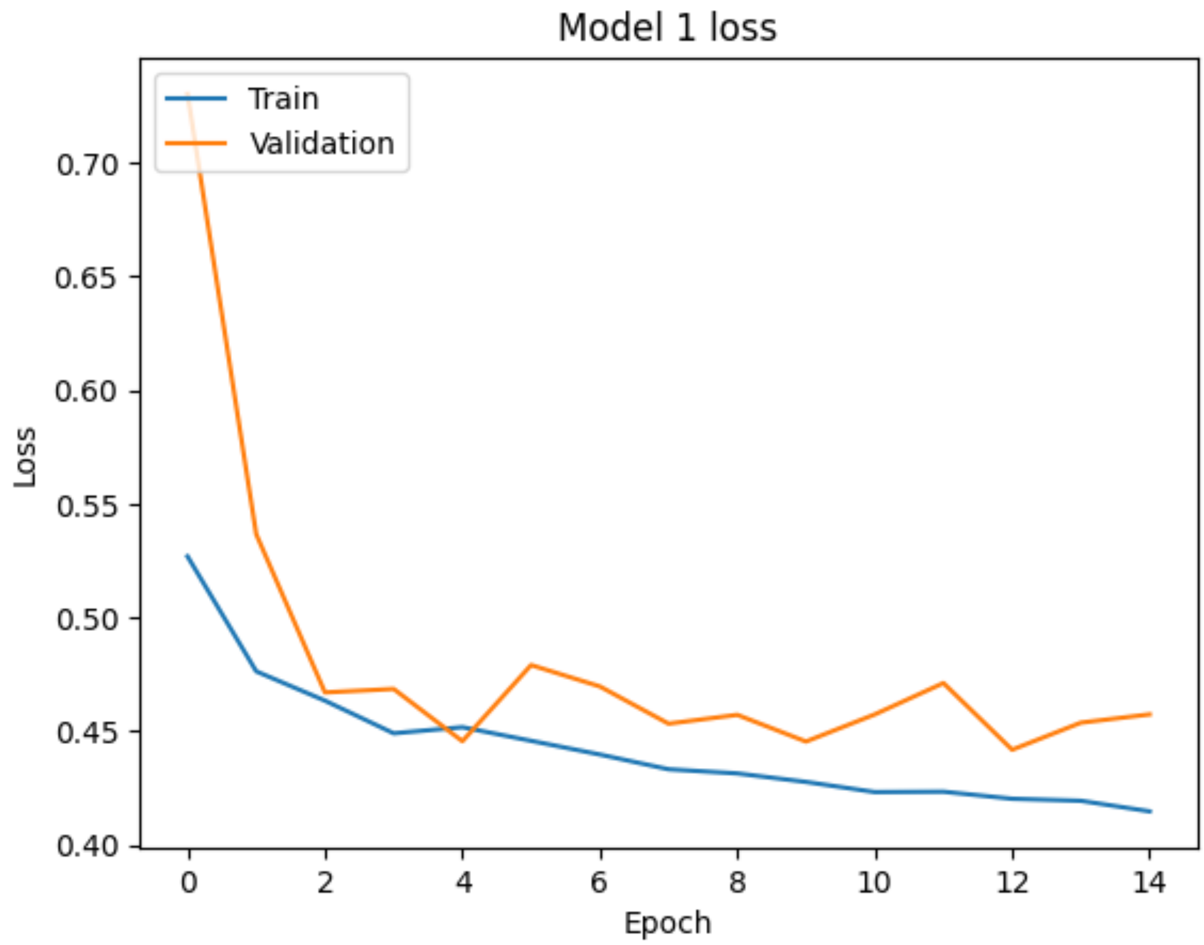
```
7996 - val_loss: 0.4450 - val_accuracy: 0.7768
Model 4 fitted.

Fitting model 5...
Epoch 1/15
99/99 [==============================] - 2s 5ms/step - loss: 0.5322 - accuracy: 0.
7253 - val_loss: 0.7233 - val_accuracy: 0.5136
Epoch 2/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4758 - accuracy: 0.
7647 - val_loss: 0.5111 - val_accuracy: 0.7495
Epoch 3/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4684 - accuracy: 0.
7695 - val_loss: 0.4815 - val_accuracy: 0.7565
Epoch 4/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4528 - accuracy: 0.
7801 - val_loss: 0.4839 - val_accuracy: 0.7470
Epoch 5/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4510 - accuracy: 0.
7792 - val_loss: 0.4752 - val_accuracy: 0.7679
Epoch 6/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4506 - accuracy: 0.
7823 - val_loss: 0.4951 - val_accuracy: 0.7191
Epoch 7/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4389 - accuracy: 0.
7866 - val_loss: 0.4838 - val_accuracy: 0.7432
Epoch 8/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4426 - accuracy: 0.
7795 - val_loss: 0.4587 - val_accuracy: 0.7762
Epoch 9/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4324 - accuracy: 0.
7877 - val_loss: 0.4512 - val_accuracy: 0.7787
Epoch 10/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4247 - accuracy: 0.
7936 - val_loss: 0.4785 - val_accuracy: 0.7261
Epoch 11/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4235 - accuracy: 0.
7963 - val_loss: 0.4598 - val_accuracy: 0.7895
Epoch 12/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4260 - accuracy: 0.
7941 - val_loss: 0.4665 - val_accuracy: 0.7685
Epoch 13/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4184 - accuracy: 0.
7934 - val_loss: 0.4777 - val_accuracy: 0.7223
Epoch 14/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4225 - accuracy: 0.
7899 - val_loss: 0.4643 - val_accuracy: 0.7616
Epoch 15/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4167 - accuracy: 0.
7931 - val_loss: 0.4461 - val_accuracy: 0.7787
Model 5 fitted.

Fitting model 6...
Epoch 1/15
99/99 [==============================] - 2s 5ms/step - loss: 0.5223 - accuracy: 0.
7360 - val_loss: 0.8246 - val_accuracy: 0.4724
Epoch 2/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4720 - accuracy: 0.
```

```
7633 - val_loss: 0.5591 - val_accuracy: 0.6848
Epoch 3/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4623 - accuracy: 0.
7703 - val_loss: 0.4734 - val_accuracy: 0.7685
Epoch 4/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4482 - accuracy: 0.
7757 - val_loss: 0.4598 - val_accuracy: 0.7793
Epoch 5/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4472 - accuracy: 0.
7781 - val_loss: 0.4510 - val_accuracy: 0.7863
Epoch 6/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4422 - accuracy: 0.
7789 - val_loss: 0.4490 - val_accuracy: 0.7857
Epoch 7/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4340 - accuracy: 0.
7838 - val_loss: 0.4723 - val_accuracy: 0.7540
Epoch 8/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4311 - accuracy: 0.
7860 - val_loss: 0.4515 - val_accuracy: 0.7882
Epoch 9/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4288 - accuracy: 0.
7876 - val_loss: 0.4577 - val_accuracy: 0.7685
Epoch 10/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4212 - accuracy: 0.
7923 - val_loss: 0.4546 - val_accuracy: 0.7768
Epoch 11/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4206 - accuracy: 0.
7936 - val_loss: 0.4513 - val_accuracy: 0.7800
Epoch 12/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4241 - accuracy: 0.
7909 - val_loss: 0.4642 - val_accuracy: 0.7647
Epoch 13/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4153 - accuracy: 0.
7952 - val_loss: 0.4620 - val_accuracy: 0.7711
Epoch 14/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4140 - accuracy: 0.
7941 - val_loss: 0.4547 - val_accuracy: 0.7762
Epoch 15/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4116 - accuracy: 0.
7972 - val_loss: 0.4470 - val_accuracy: 0.7800
Model 6 fitted.

Fitting model 7...
Epoch 1/15
99/99 [==============================] - 2s 5ms/step - loss: 0.5414 - accuracy: 0.
7229 - val_loss: 0.5724 - val_accuracy: 0.7191
Epoch 2/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4748 - accuracy: 0.
7625 - val_loss: 0.5343 - val_accuracy: 0.6944
Epoch 3/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4637 - accuracy: 0.
7673 - val_loss: 0.4786 - val_accuracy: 0.7736
Epoch 4/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4460 - accuracy: 0.
7806 - val_loss: 0.4817 - val_accuracy: 0.7425
Epoch 5/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4454 - accuracy: 0.
```

```
7766 - val_loss: 0.4556 - val_accuracy: 0.7926
Epoch 6/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4429 - accuracy: 0.
7808 - val_loss: 0.4562 - val_accuracy: 0.7838
Epoch 7/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4369 - accuracy: 0.
7839 - val_loss: 0.4676 - val_accuracy: 0.7685
Epoch 8/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4319 - accuracy: 0.
7874 - val_loss: 0.4637 - val_accuracy: 0.7711
Epoch 9/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4248 - accuracy: 0.
7934 - val_loss: 0.4571 - val_accuracy: 0.7793
Epoch 10/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4230 - accuracy: 0.
7966 - val_loss: 0.4809 - val_accuracy: 0.7292
Epoch 11/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4232 - accuracy: 0.
7911 - val_loss: 0.4547 - val_accuracy: 0.7888
Epoch 12/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4211 - accuracy: 0.
7968 - val_loss: 0.5225 - val_accuracy: 0.6728
Epoch 13/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4138 - accuracy: 0.
7974 - val_loss: 0.5032 - val_accuracy: 0.7254
Epoch 14/15
99/99 [==============================] - 0s 3ms/step - loss: 0.4171 - accuracy: 0.
7945 - val_loss: 0.4701 - val_accuracy: 0.7495
Epoch 15/15
99/99 [==============================] - 0s 4ms/step - loss: 0.4146 - accuracy: 0.
7961 - val_loss: 0.4492 - val_accuracy: 0.7850
Model 7 fitted.
```

```python
# Check training procedure
plt.plot(histories[0].history['accuracy'])
plt.plot(histories[0].history['val_accuracy'])
plt.title('Model 1 accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(histories[0].history['loss'])
plt.plot(histories[0].history['val_loss'])
plt.title('Model 1 loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```






```
neural_predictions = []
```

```python
for i, model in enumerate(neural_models):
    print(f"Predicting with model {i+1}...")
    y_pred = (model.predict(X_test) > 0.5).astype("int32")
    neural_predictions.append(y_pred)
    print(f"Predictions from model {i+1} stored.\n")
```

```
Predicting with model 1...
62/62 [==============================] - 0s 1ms/step
Predictions from model 1 stored.

Predicting with model 2...
62/62 [==============================] - 0s 1ms/step
Predictions from model 2 stored.

Predicting with model 3...
62/62 [==============================] - 0s 1ms/step
Predictions from model 3 stored.

Predicting with model 4...
62/62 [==============================] - 0s 1ms/step
Predictions from model 4 stored.

Predicting with model 5...
62/62 [==============================] - 0s 1ms/step
Predictions from model 5 stored.

Predicting with model 6...
62/62 [==============================] - 0s 1ms/step
Predictions from model 6 stored.

Predicting with model 7...
62/62 [==============================] - 0s 1ms/step
Predictions from model 7 stored.
```

```python
nn_metrics = []

for i, y_pred in enumerate(neural_predictions):
    accuracy = round(accuracy_score(y_test, y_pred), 3)
    precision = round(precision_score(y_test, y_pred), 3)
    recall = round(recall_score(y_test, y_pred), 3)
    f1 = round(f1_score(y_test, y_pred), 3)
    roc_auc = round(roc_auc_score(y_test, y_pred), 3)

    nn_metrics.append({
        "Model": f"Neural Network {i+1}",
        "Accuracy": accuracy,
        "Precision": precision,
        "Recall": recall,
        "F1-score": f1,
        "ROC AUC": roc_auc
    })

# Display the 7 models performances
nn_metrics = pd.DataFrame(nn_metrics)
nn_metrics
```

	Model	Accuracy	Precision	Recall	F1-score	ROC AUC
0	Neural Network 1	0.772	0.825	0.659	0.732	0.766
1	Neural Network 2	0.778	0.848	0.649	0.735	0.772
2	Neural Network 3	0.755	0.766	0.694	0.728	0.752
3	Neural Network 4	0.790	0.872	0.651	0.746	0.783
4	Neural Network 5	0.777	0.833	0.662	0.738	0.771
5	Neural Network 6	0.786	0.882	0.633	0.737	0.778
6	Neural Network 7	0.786	0.887	0.629	0.736	0.778

```
combined_metrics = pd.concat([metrics, nn_metrics], ignore_index=True)
combined_metrics
```

	Model	Accuracy	Precision	Recall	F1-score	ROC AUC
0	Linear Regression	0.727	0.913	0.470	0.620	0.715
1	Decision Tree	0.782	0.781	0.752	0.766	0.781
2	Naive Bayes	0.779	1.000	0.534	0.696	0.767
3	XGBoost	0.816	0.900	0.690	0.781	0.810
4	KNN	0.789	0.820	0.711	0.762	0.785
5	Neural Network 1	0.772	0.825	0.659	0.732	0.766
6	Neural Network 2	0.778	0.848	0.649	0.735	0.772
7	Neural Network 3	0.755	0.766	0.694	0.728	0.752
8	Neural Network 4	0.790	0.872	0.651	0.746	0.783
9	Neural Network 5	0.777	0.833	0.662	0.738	0.771
10	Neural Network 6	0.786	0.882	0.633	0.737	0.778
11	Neural Network 7	0.786	0.887	0.629	0.736	0.778

```
# Add the NN to the models and to the predicitons
for i, model in enumerate(neural_models):
    models[f"Neural Network {i+1}"] = model

for i, prediction_list in enumerate(neural_predictions):
    predictions_df[f'Neural Network {i+1}'] = prediction_list.flatten()
    predictions[f'Neural Network {i+1}'] = prediction_list.flatten()
```

Task 3 - Fairness Metrics

3.1 Demographic Parity

```
# Columns groups of interest
sensitive_features = [' Sex_encoded', ' Age Range_encoded', ' Citizenship_encoded',
non_sensitive_features = ['Technical Skills', 'Comunication', 'Maturity', 'Dynamism
       'English', ' Study area_encoded', ' Study Title_encoded', ' Years Experience
       ' Job Title Hiring_encoded', ' Overall_encoded', ' Years Experience.1_encode
       ' Study Level_encoded', 'Current Ral_encoded', 'Expected Ral_encoded']

models_list = [model for model in models]

# Tresholds
tolerance = 0.15
significance_level = 0.1
```

```
def calculate_demographic_parity(predictions, sensitive_attribute, name, significar

    df = pd.DataFrame({
        'predictions': predictions,
        'sensitive_attribute': sensitive_attribute
    })

    # Proportion of positive predictions for each group
    positive_proportions = df.groupby('sensitive_attribute')['predictions'].mean()
    num_class = positive_proportions.shape[0]
    min_proportion = positive_proportions.min()
    max_proportion = positive_proportions.max()
    percentage_difference = (max_proportion - min_proportion)

    # Case for binary sensitive attribute
    if num_class == 2:

        if activate_check == True:
            print("===")
            print(name)
            print(positive_proportions)

        if percentage_difference <= tolerance:
            return 'T'
        else:
            return False

    # Case for multiclass sensitive attribute
    if num_class > 2:
        contingency_table = pd.crosstab(df['predictions'], df['sensitive_attribute'
        chi2, p, dof, expected = chi2_contingency(contingency_table)

        if activate_check == True:
            print("===")
            print(name)
            print(positive_proportions)
            if (expected < 5).any():
                print(f"Sparse contigency for {name}")

        if p > significance_level:
            return 'T'
```

```
        else:
            return False
```

```
# Models behaviours over sensitive features
table = []

for model in models:
     temp = []
     for i in range(len(sensitive_features)):
        Boolean_Output = calculate_demographic_parity(predictions[model], X_test[se
        temp.append(Boolean_Output)
     table.append(temp)

sf_df = pd.DataFrame(table, index = models_list, columns=sensitive_features)
sf_df.head(len(models_list))
```

```
===
 Sex_encoded
sensitive_attribute
0    0.392208
1    0.207940
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.384422
1    0.208386
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.247499
1    0.150685
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.333333
1    0.243505
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.566234
1    0.429742
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.522613
1    0.439644
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.453923
1    0.520548
```

```
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.456444
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.374026
1    0.223693
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.324121
1    0.235070
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.255398
1    0.191781
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.252165
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.485714
1    0.333963
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.482412
1    0.333545
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.361243
1    0.424658
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.555556
1    0.362710
Name: predictions, dtype: float64
===
 Sex_encoded
```

```
sensitive_attribute
0    0.524675
1    0.383743
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.500000
1    0.388818
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.413902
1    0.342466
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.411105
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.480519
1    0.354127
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.479899
1    0.353240
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.379147
1    0.369863
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.555556
1    0.377993
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.475325
1    0.335854
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.547739
1    0.316391
```

```
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.364402
1    0.328767
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.666667
1    0.361691
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.574026
1    0.394455
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.505025
1    0.410419
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.428647
1    0.452055
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.555556
1    0.428935
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.488312
1    0.321361
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.497487
1    0.317662
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.353344
1    0.369863
Name: predictions, dtype: float64
===
 Protected category_encoded
```

```
sensitive_attribute
0    0.555556
1    0.353031
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.498701
1    0.347196
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.474874
1    0.351970
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.377041
1    0.369863
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.376465
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.462338
1    0.310649
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.479899
1    0.304956
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.339126
1    0.369863
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.555556
1    0.339277
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.483117
1    0.300567
```

```
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.464824
1    0.303685
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.335440
1    0.356164
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.555556
1    0.335201
Name: predictions, dtype: float64
```

	Sex_encoded	Age Range_encoded	Citizenship_encoded	Protected category_encoded
Linear Regression	False	False	T	T
Decision Tree	T	T	T	T
Naive Bayes	False	T	T	False
XGBoost	False	T	T	False
KNN	T	T	T	T
Neural Network 1	T	T	T	False
Neural Network 2	T	False	T	False
Neural Network 3	False	T	T	T
Neural Network 4	False	False	T	False
Neural Network 5	False	T	T	T
Neural Network 6	False	False	T	False
Neural Network 7	False	False	T	False

3.2 Equalized odds

```
def calculate_equalized_odds(predictions, true_labels, sensitive_attribute, name, t
    df = pd.DataFrame({
        'predictions': predictions,
```

```
        'true_labels': true_labels,
        'sensitive_attribute': sensitive_attribute
    })

    # Calculate TPR and FPR for each group
    groups = df['sensitive_attribute'].unique()
    metrics = {}
    for group in groups:
        group_df = df[df['sensitive_attribute'] == group]
        cm = confusion_matrix(group_df['true_labels'], group_df['predictions'], lab
        tn, fp, fn, tp = cm.ravel()

        tpr = tp / (tp + fn) if tp + fn != 0 else 0
        fpr = fp / (fp + tn) if fp + tn != 0 else 0
        metrics[group] = {'TPR': tpr, 'FPR': fpr}

    # Check if TPR and FPR are within the tolerance
    tprs = [metrics[group]['TPR'] for group in groups]
    fprs = [metrics[group]['FPR'] for group in groups]

    max_tpr_diff = max(tprs) - min(tprs)
    max_fpr_diff = max(fprs) - min(fprs)

    if activate_check == True:
        print("===")
        print(name)
        print(max_fpr_diff)
        print(max_tpr_diff)

    tpr_within_tolerance = max_tpr_diff <= tolerance*2
    fpr_within_tolerance = max_fpr_diff <= tolerance*2

    if tpr_within_tolerance and fpr_within_tolerance:
        return 'T'
    else:
        return False
```

```
# Equalized odds
table = []

for model in models:
    temp = []
    for i in range(len(sensitive_features)):
        Boolean_Output = calculate_equalized_odds(predictions[model], y_test, X_tes
        temp.append(Boolean_Output)
    table.append(temp)

# DataFrame
equalized_df = pd.DataFrame(table, index = models_list, columns=sensitive_features)
equalized_df.head(len(models_list))
```

	Sex_encoded	Age Range_encoded	Citizenship_encoded	Protected category_encoded
Linear Regression	T	T	T	T
Decision Tree	T	T	T	T
Naive Bayes	T	T	T	T
XGBoost	T	T	T	False
KNN	T	T	T	T
Neural Network 1	T	T	T	False
Neural Network 2	T	T	T	False
Neural Network 3	T	T	T	False
Neural Network 4	T	T	T	False
Neural Network 5	T	T	T	T
Neural Network 6	T	T	T	False
Neural Network 7	T	T	T	False

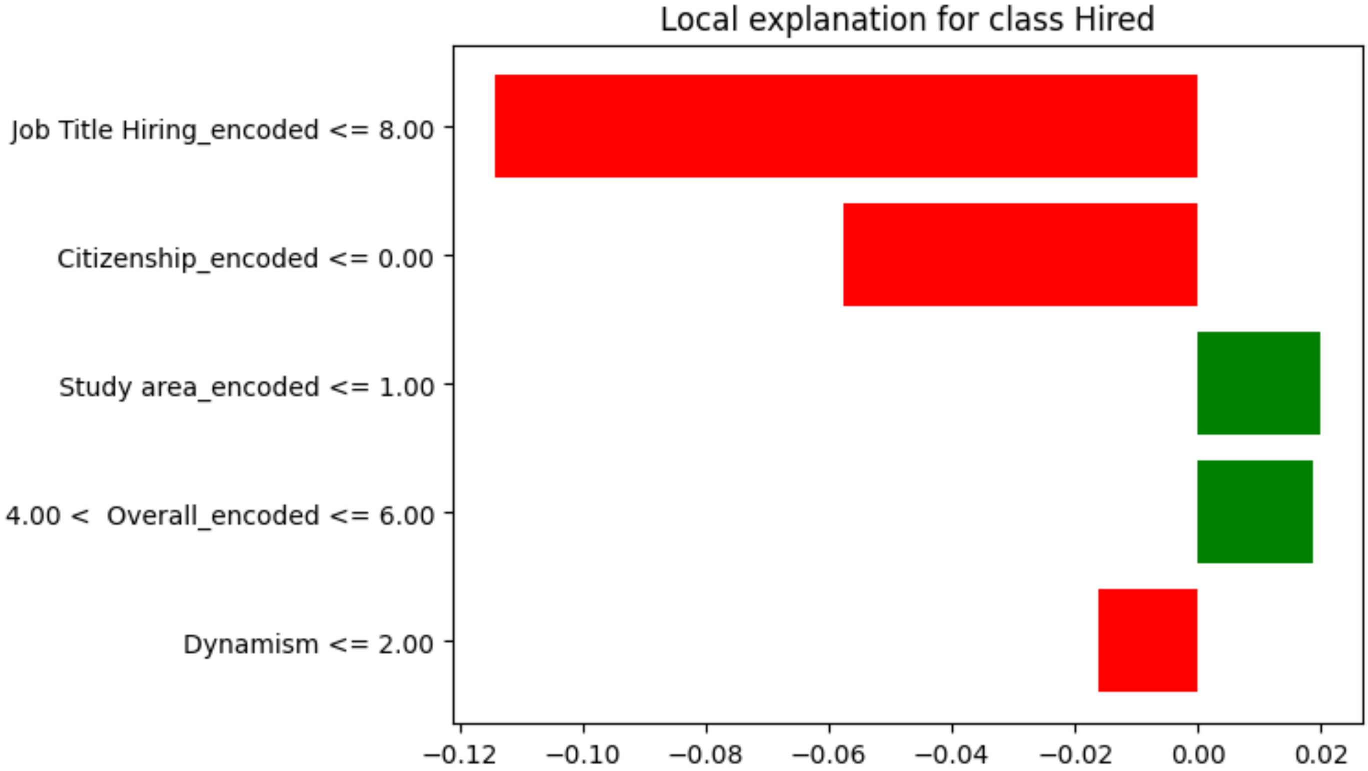
Task 4 - Explainable AI

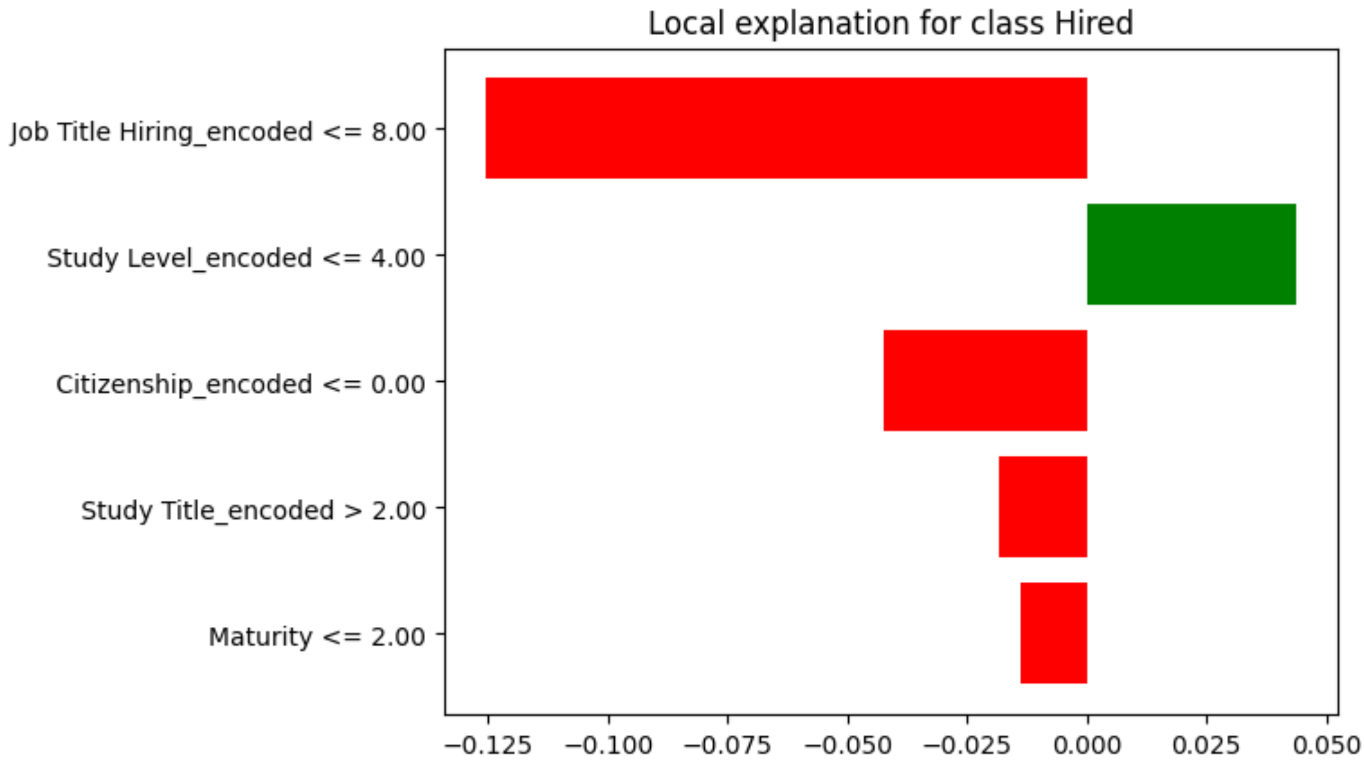
4.1 Lime

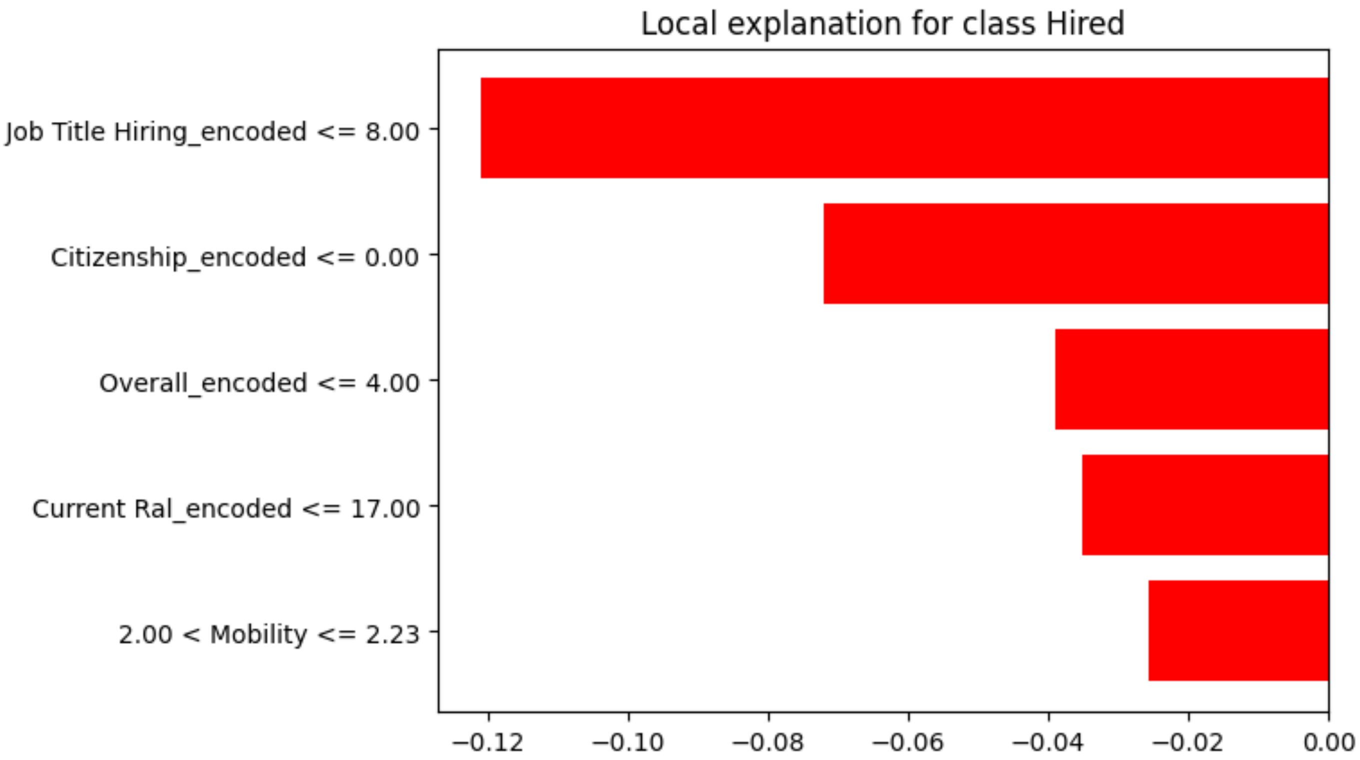
```
# Initialize LimeTabularExplainer using the training data
explainer = lime_tabular.LimeTabularExplainer(X_train.values,
                                               mode="classification",
                                               feature_names=X_train.columns.tolist(),
                                               class_names=['Not Hired', 'Hired'])
```

```
# Initialize LimeTabularExplainer using the training data
explainer = lime_tabular.LimeTabularExplainer(X_train.values,
                                               mode="classification",
                                               feature_names=X_train.columns.tolist(),
                                               class_names=['Not Hired', 'Hired'])
plt.figure(figsize=(5, 3))

print("LIME EXPLANATIONS FOR DECISION TREE")
for i in range(10,13):
    exp_dt = explainer.explain_instance(X_test.values[i], models['Decision Tree'].p
    fig_dt = exp_dt.as_pyplot_figure()
    plt.show()
```

```
LIME EXPLANATIONS FOR DECISION TREE
<Figure size 500x300 with 0 Axes>
```

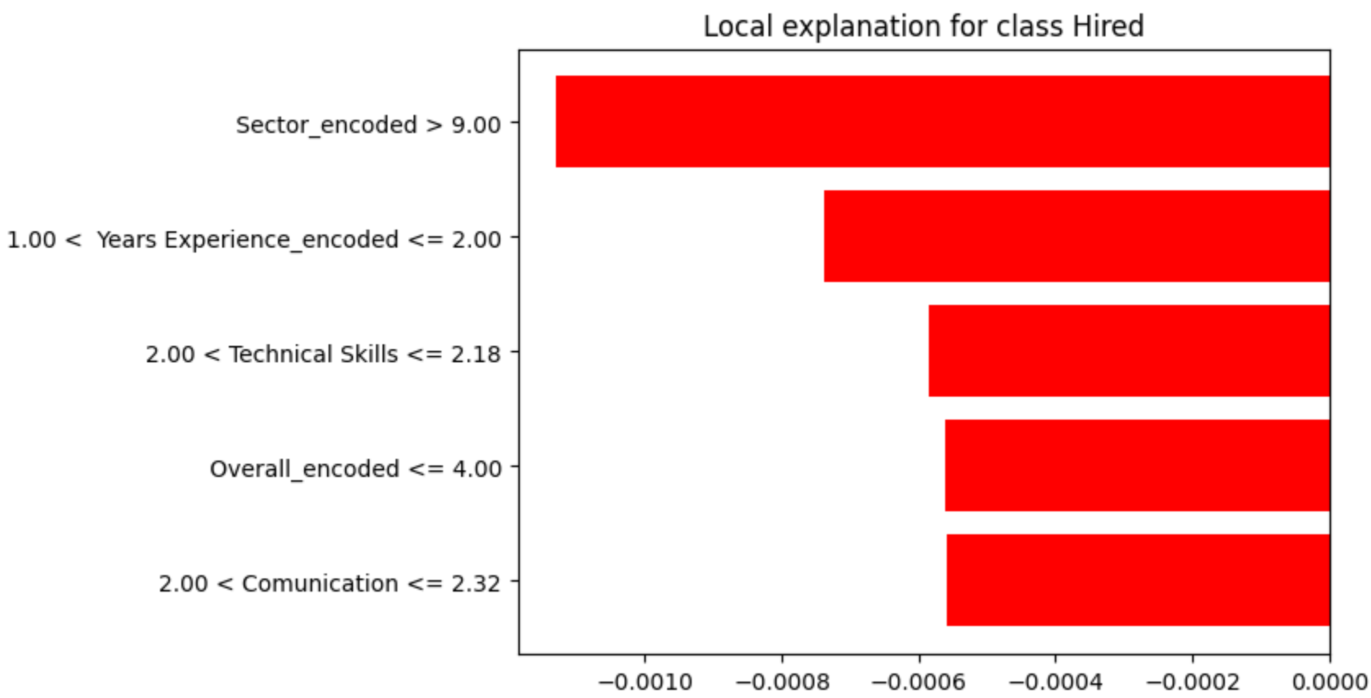


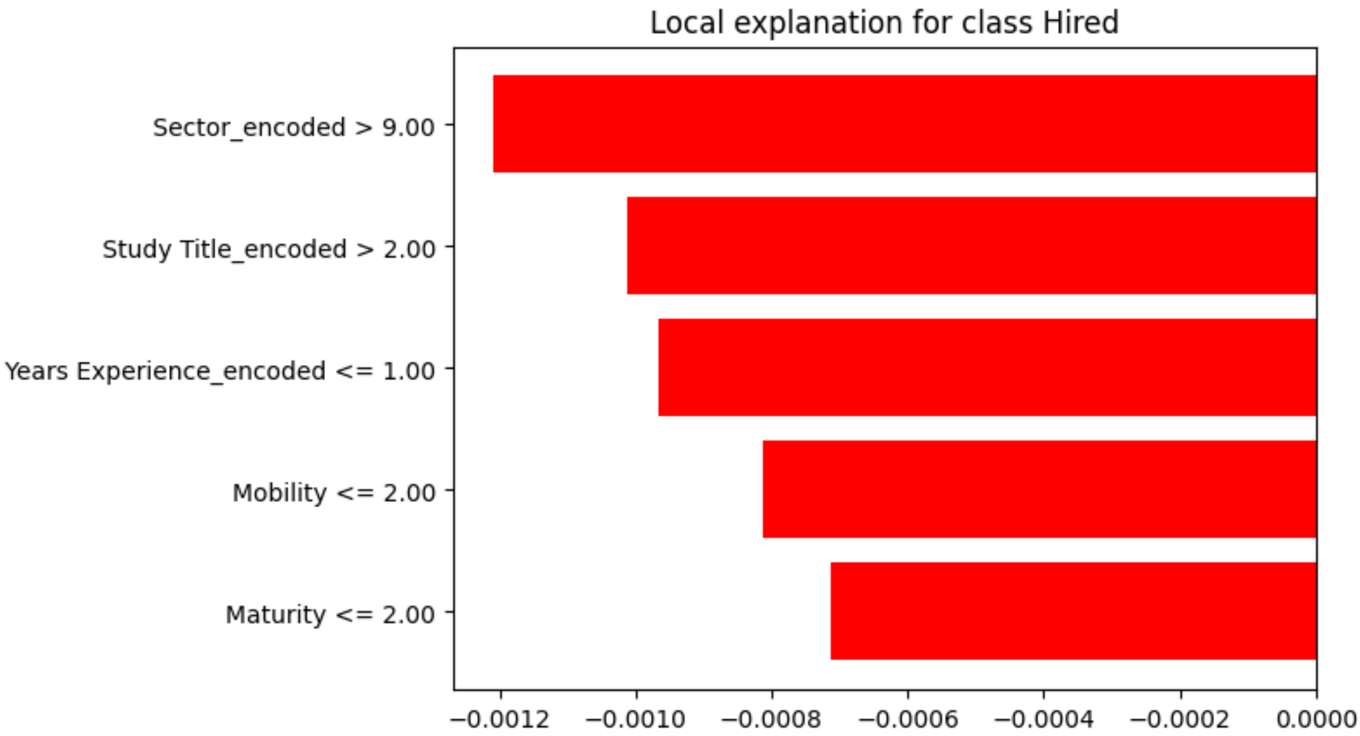


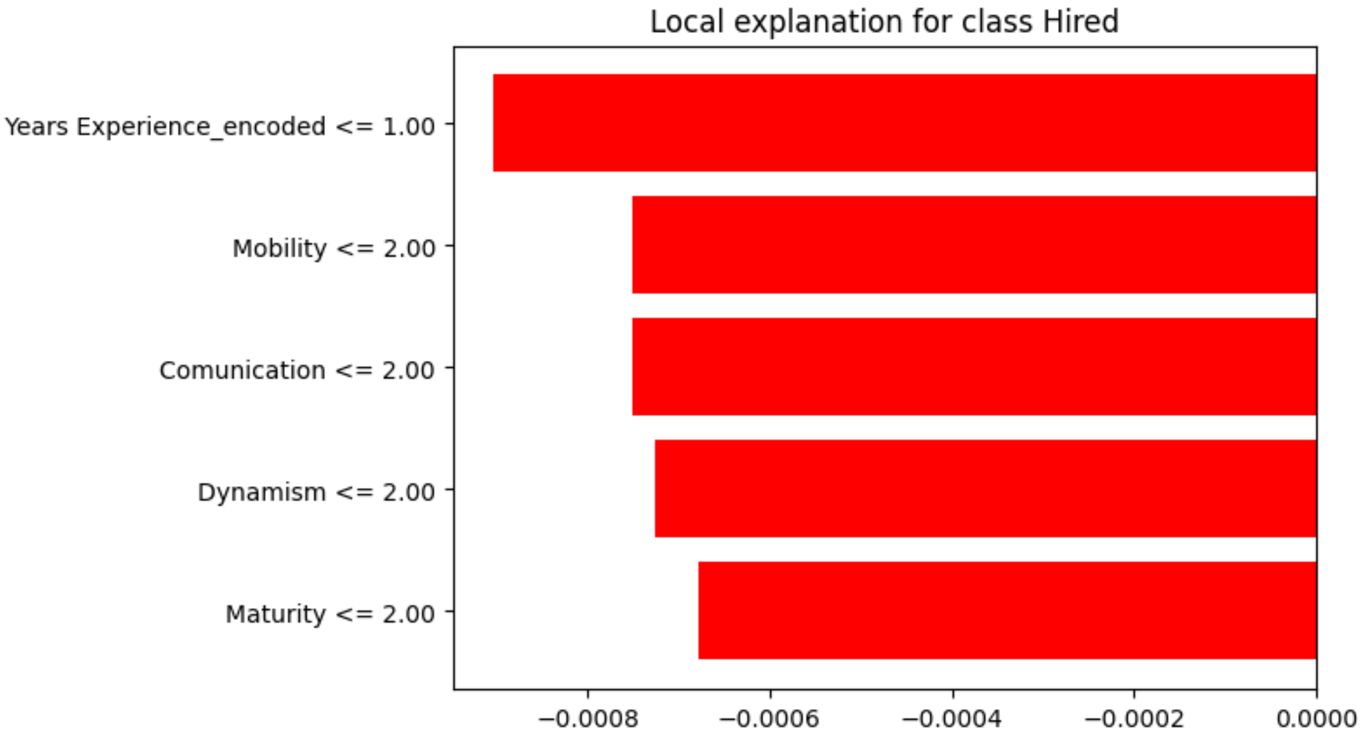



```
print("LIME EXPLANATIONS FOR NAIVE BAYES")
for i in range(21,24):
    exp_dt = explainer.explain_instance(X_test.values[i], models['Naive Bayes'].pre
    fig_dt = exp_dt.as_pyplot_figure()
    plt.show()
```

```
LIME EXPLANATIONS FOR NAIVE BAYES
```

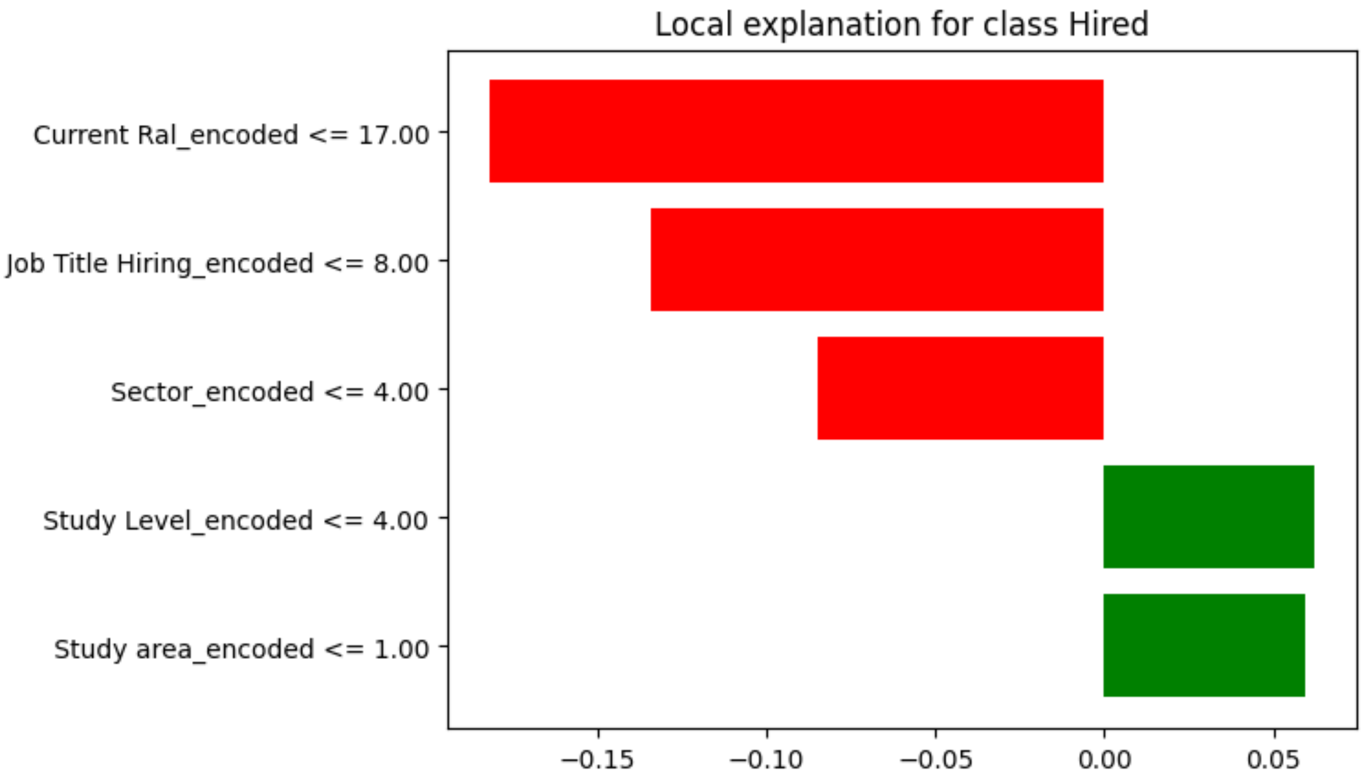


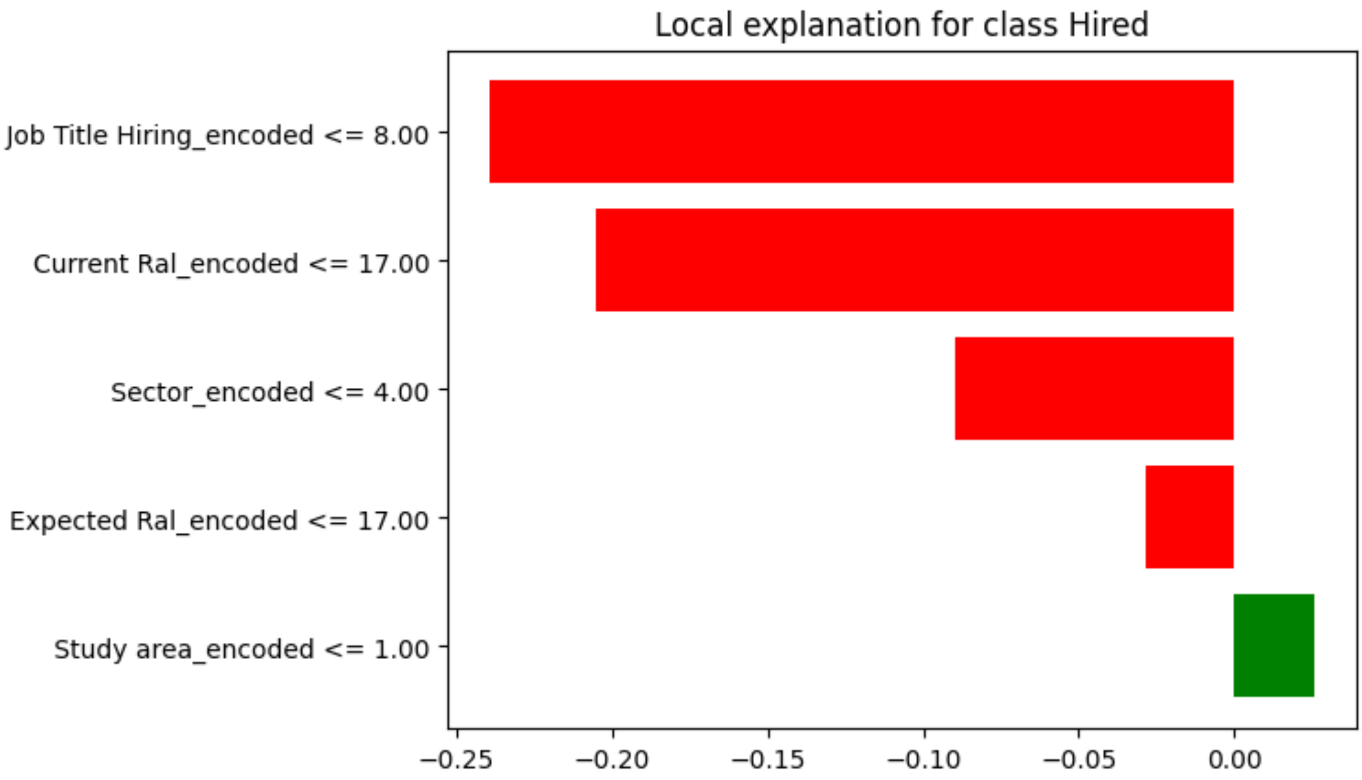


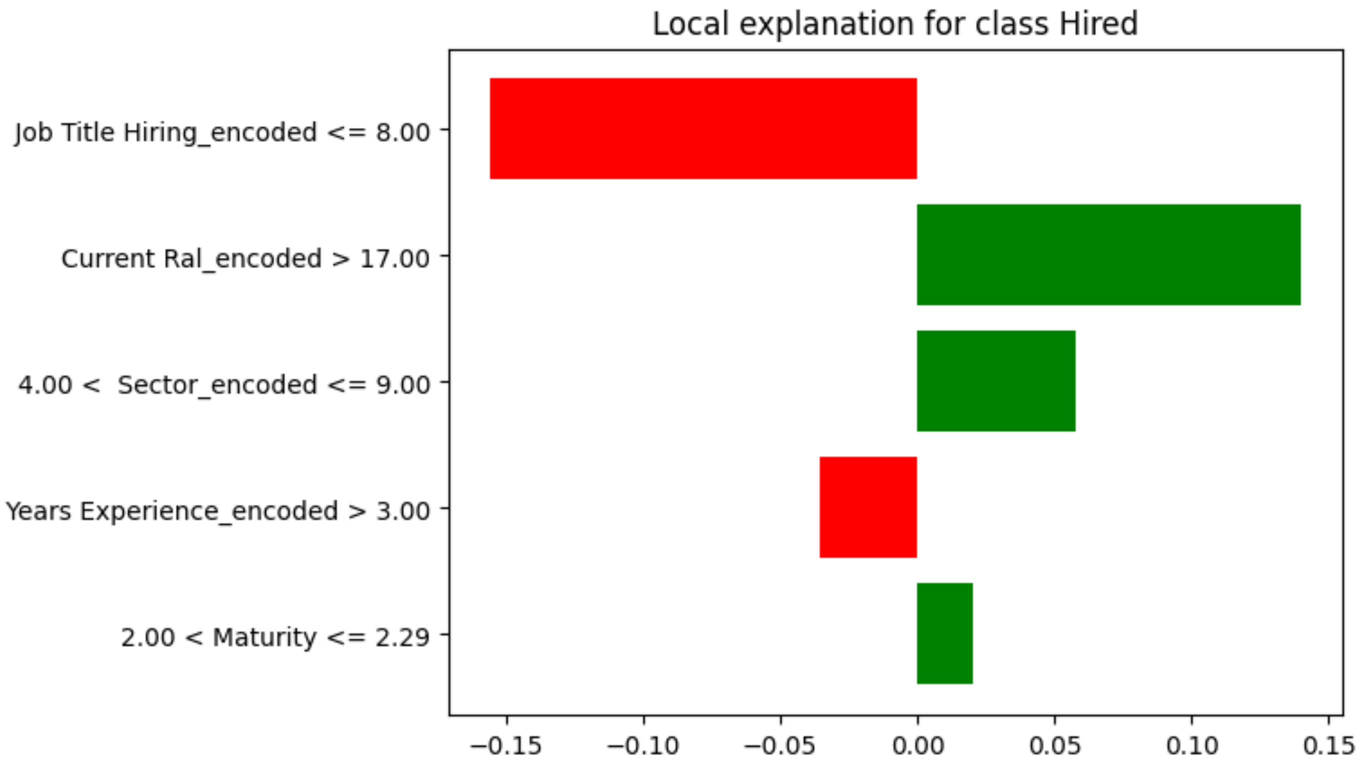



```
print("LIME EXPLANATIONS FOR KNN")
for i in range(16,19):
    exp_dt = explainer.explain_instance(X_test.values[i], models['KNN'].predict_pro
    fig_dt = exp_dt.as_pyplot_figure()
    plt.show()
```

```
LIME EXPLANATIONS FOR KNN
```


Local explanation for class Hired
Current Ral_encoded <= 17.00
Job Title Hiring_encoded <= 8.00
Sector_encoded <= 4.00
Study Level_encoded <= 4.00
Study area_encoded <= 1.00
−0.15
−0.10
−0.05
0.00
0.05


Local explanation for class Hired
Job Title Hiring_encoded <= 8.00
Current Ral_encoded <= 17.00
Sector_encoded <= 4.00
Expected Ral_encoded <= 17.00
Study area_encoded <= 1.00
−0.25
−0.20
−0.15
−0.10
−0.05
0.00

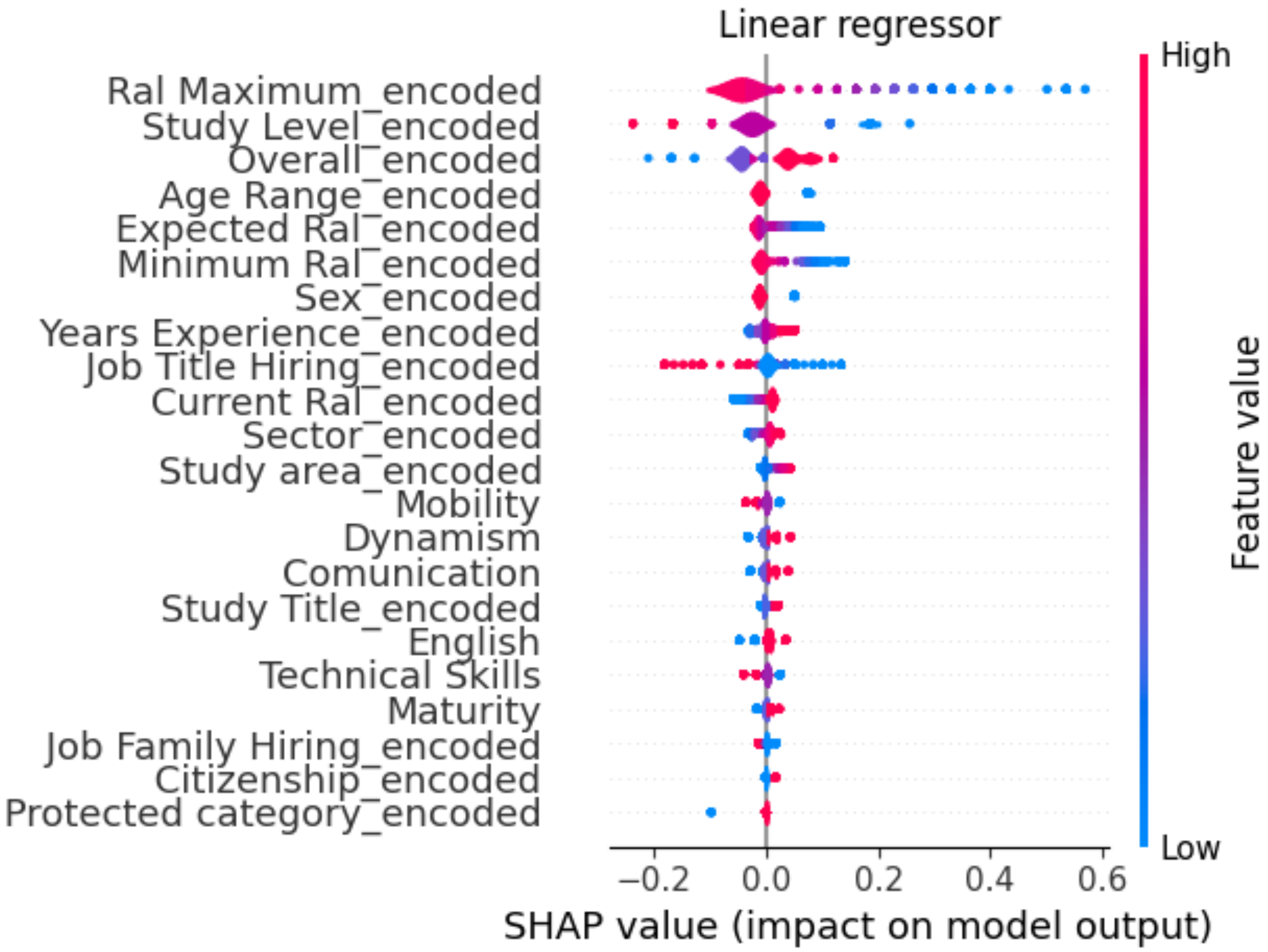


3.4 SHAP

```
def create_explanations(model, X, name):

    if name == 'Neural Network 1':
        # We need a slightly different explainer for NN
        explainer = shap.GradientExplainer (models[models_list[5]], X_test.values)
        #print(explainer(X_test.values))
        shap_values = explainer.shap_values(X_test.values)[:1000][:][0]
        print(shap_values.shape)
        return shap_values, X

    if name == 'Neural Network 2':
        # We need a slightly different explainer for NN
        explainer = shap.GradientExplainer (models[models_list[6]], X_test.values)
        shap_values = explainer.shap_values(X_test.values)[:1000][:][0]
        return shap_values, X

    if name == 'Neural Network 3':
        # We need a slightly different explainer for NN
        explainer = shap.GradientExplainer (models[models_list[7]], X_test.values)
        shap_values = explainer.shap_values(X_test.values)[:1000][:][0]
        return shap_values, X

    if name == 'Neural Network 4':
        # We need a slightly different explainer for NN
        explainer = shap.GradientExplainer (models[models_list[8]], X_test.values)
        shap_values = explainer.shap_values(X_test.values)[:1000][:][0]
        return shap_values, X

    if name == 'Neural Network 5':
        # We need a slightly different explainer for NN
        explainer = shap.GradientExplainer (models[models_list[9]], X_test.values)
        shap_values = explainer.shap_values(X_test.values)[:1000][:][0]
```

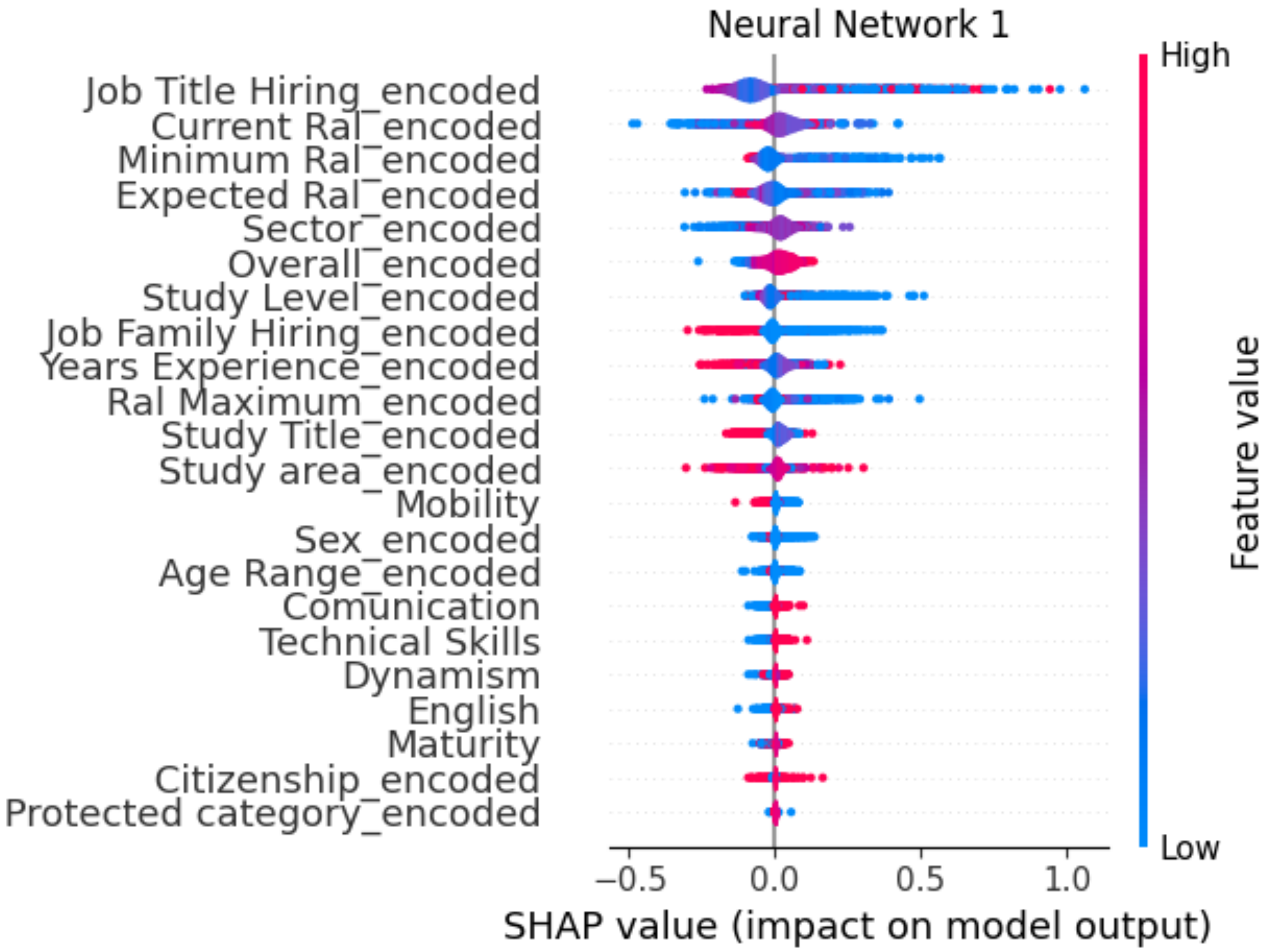
```python
        return shap_values, X

    if name == 'Neural Network 6':
        # We need a slightly different explainer for NN
        explainer = shap.GradientExplainer (models[models_list[10]], X_test.values)
        shap_values = explainer.shap_values(X_test.values)[:1000][:][0]
        return shap_values, X

    if name == 'Neural Network 7':
        # We need a slightly different explainer for NN
        explainer = shap.GradientExplainer (models[models_list[11]], X_test.values)
        shap_values = explainer.shap_values(X_test.values)[:1000][:][0]
        return shap_values, X

    # Add feature names
    explainer = shap.Explainer(model, X)
    explanations = explainer(X)
    return explanations, X

def summaryPlot(model, X, lf, plot_type, plot_name):
    explanations, X = create_explanations(model, X, plot_name)

    # Create plot
    fig, ax = plt.subplots()
    plt.title(f"{plot_name}")
    shap.summary_plot(explanations, X, lf, show=False, plot_size=None, plot_type=pl
    plt.tight_layout()
    plt.show()
    plt.close()
```

```python
tot_columns = list(X_test.columns)
summaryPlot(models[models_list[0]], X_test, tot_columns, plot_type='violin', plot_r
summaryPlot(models[models_list[3]], X_test, tot_columns, plot_type='violin', plot_r
for i in range(5, 12):
    print(i)
    summaryPlot(models[models_list[i]], X_test, tot_columns, plot_type='violin', pl
```

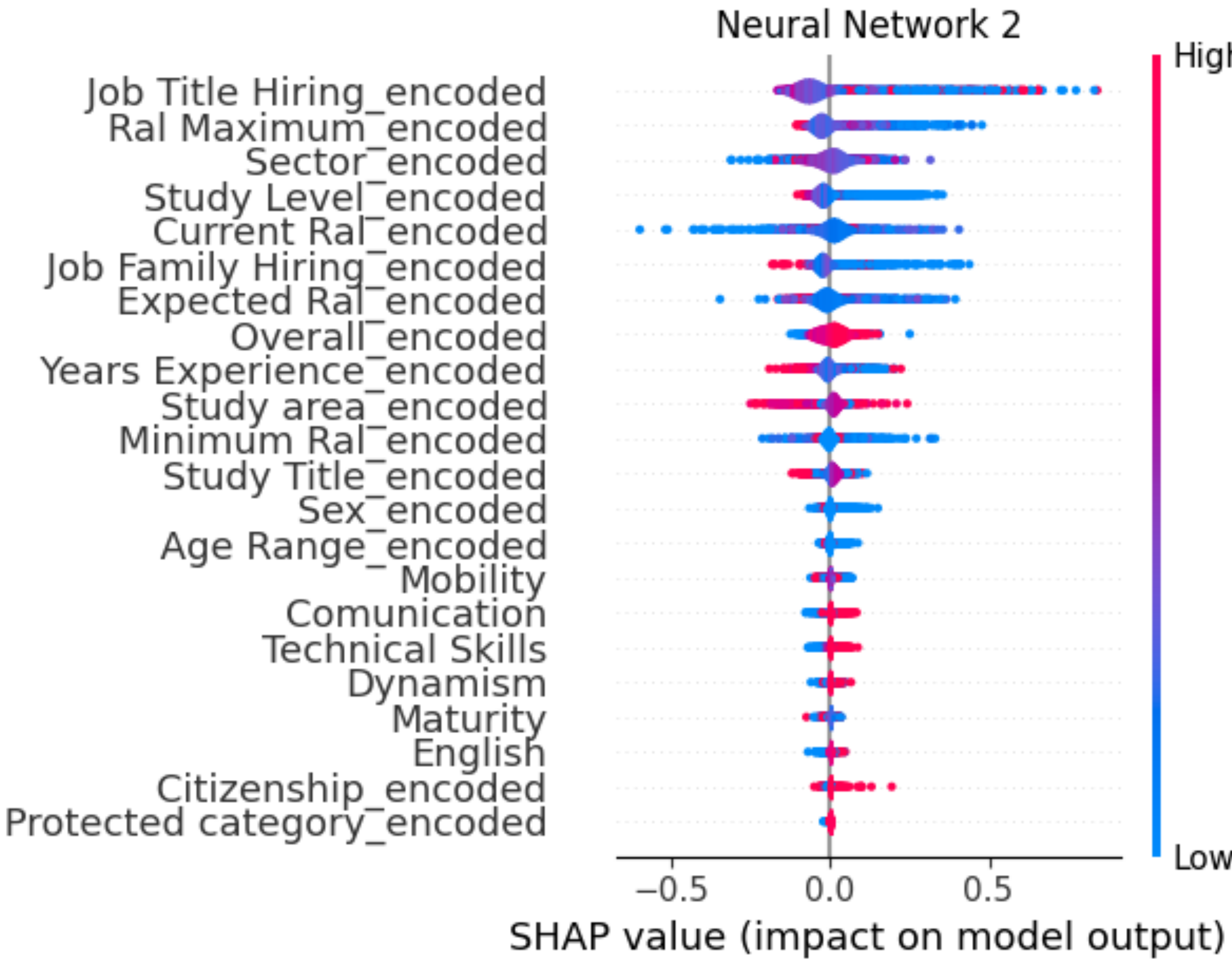

Linear regressor
Ral Maximum_encoded
Study Level_encoded
Overall_encoded
Age Range_encoded
Expected Ral_encoded
Minimum Ral_encoded
Sex_encoded
Years Experience_encoded
Job Title Hiring_encoded
Current Ral_encoded
Sector_encoded
Study area_encoded
Mobility
Dynamism
Comunication
Study Title_encoded
English
Technical Skills
Maturity
Job Family Hiring_encoded
Citizenship_encoded
Protected category_encoded
−0.2
0.0
0.2
0.4
0.6
SHAP value (impact on model output)
High
Low
Feature value


XGBoost
Job Title Hiring_encoded
Sector_encoded
Current Ral_encoded
Job Family Hiring_encoded
Overall_encoded
Years Experience_encoded
Expected Ral_encoded
Study Level_encoded
Comunication
Study area_encoded
Study Title_encoded
Mobility
Age Range_encoded
Maturity
Ral Maximum_encoded
Sex_encoded
Dynamism
Technical Skills
English
Citizenship_encoded
Protected category_encoded
Minimum Ral_encoded
−2.5
0.0
2.5
5.0
SHAP value (impact on model output)
High
Low
Feature value

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
(1972, 22)
```

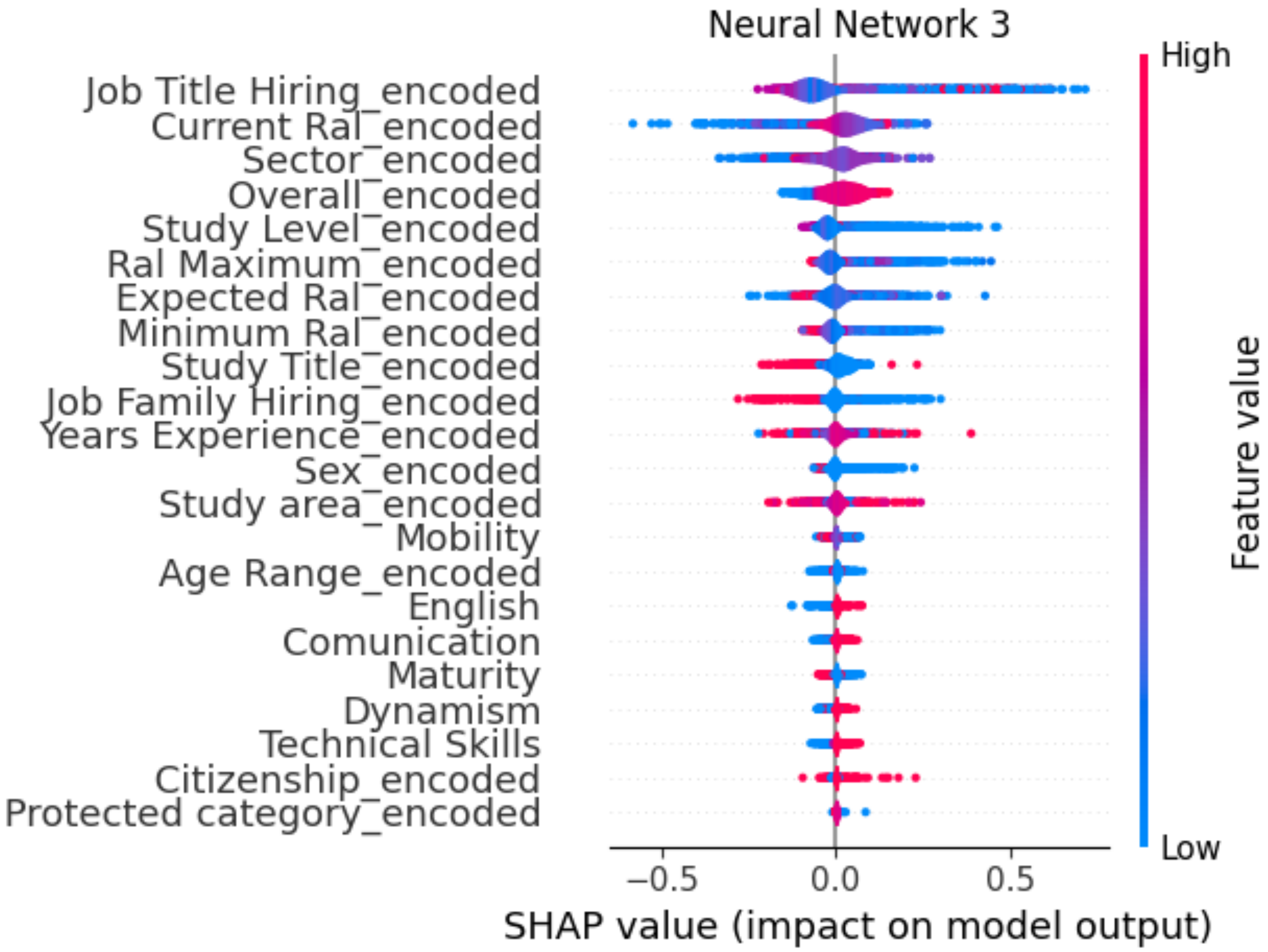



```
6
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



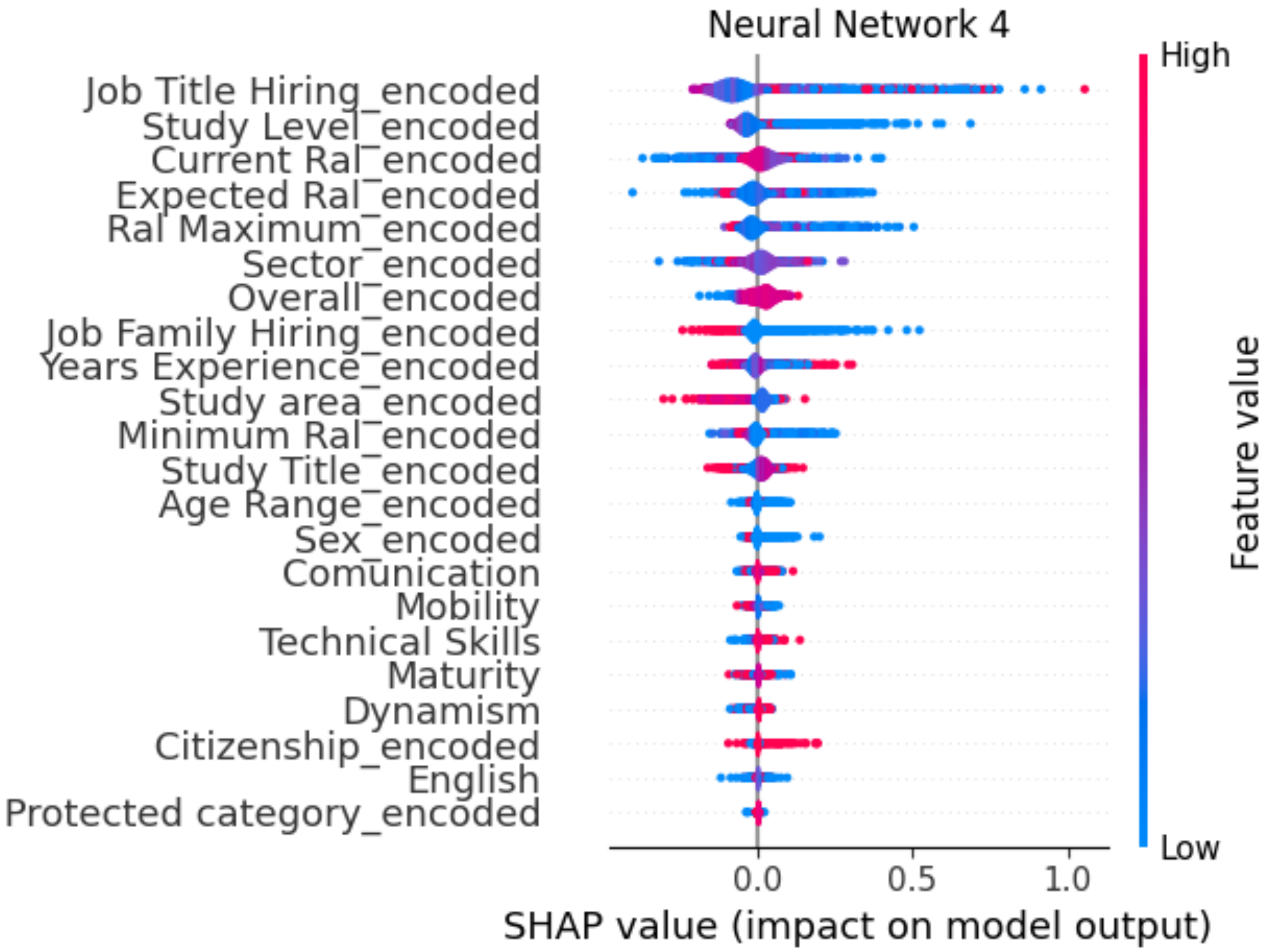
7

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



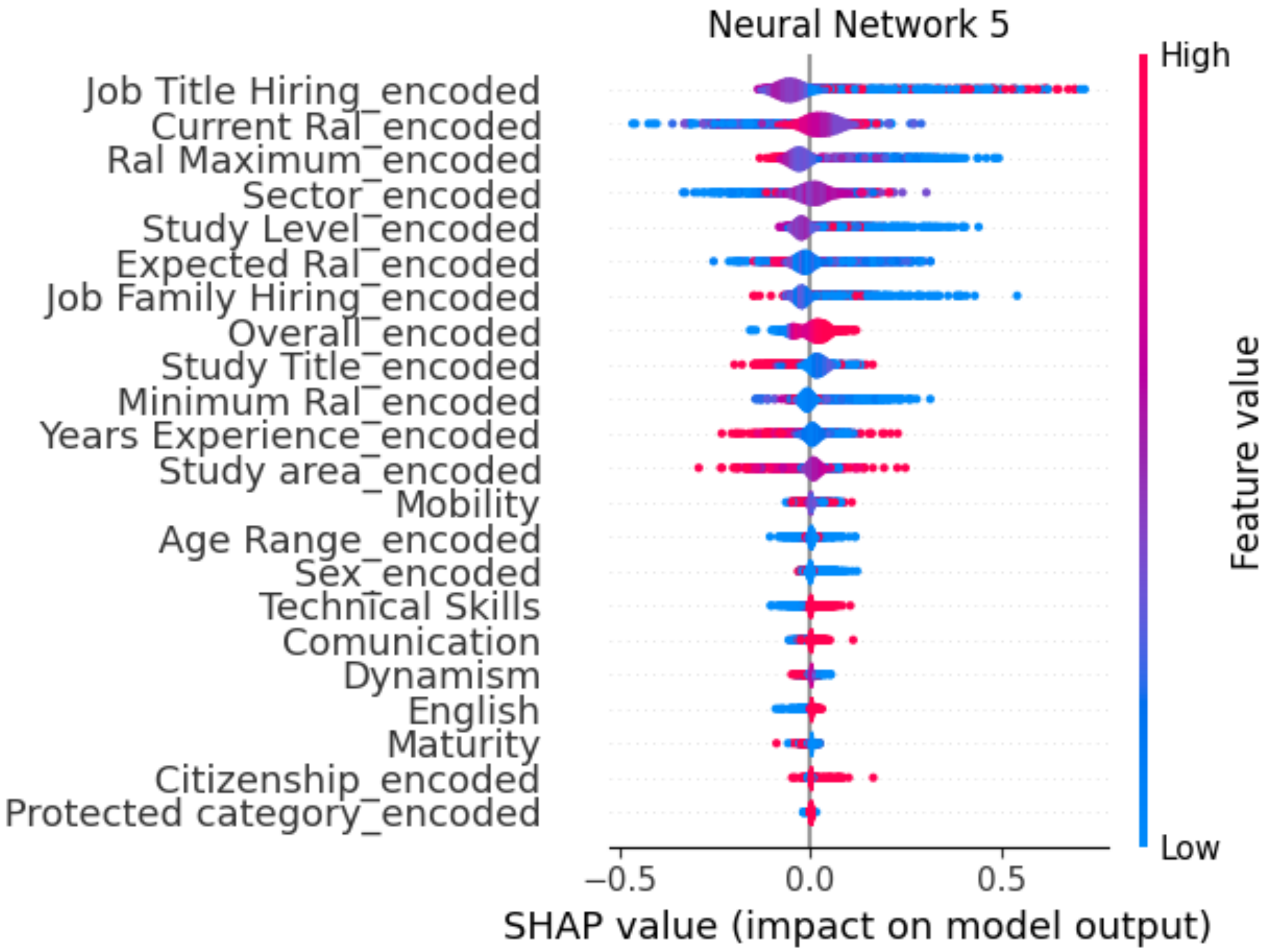
8

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



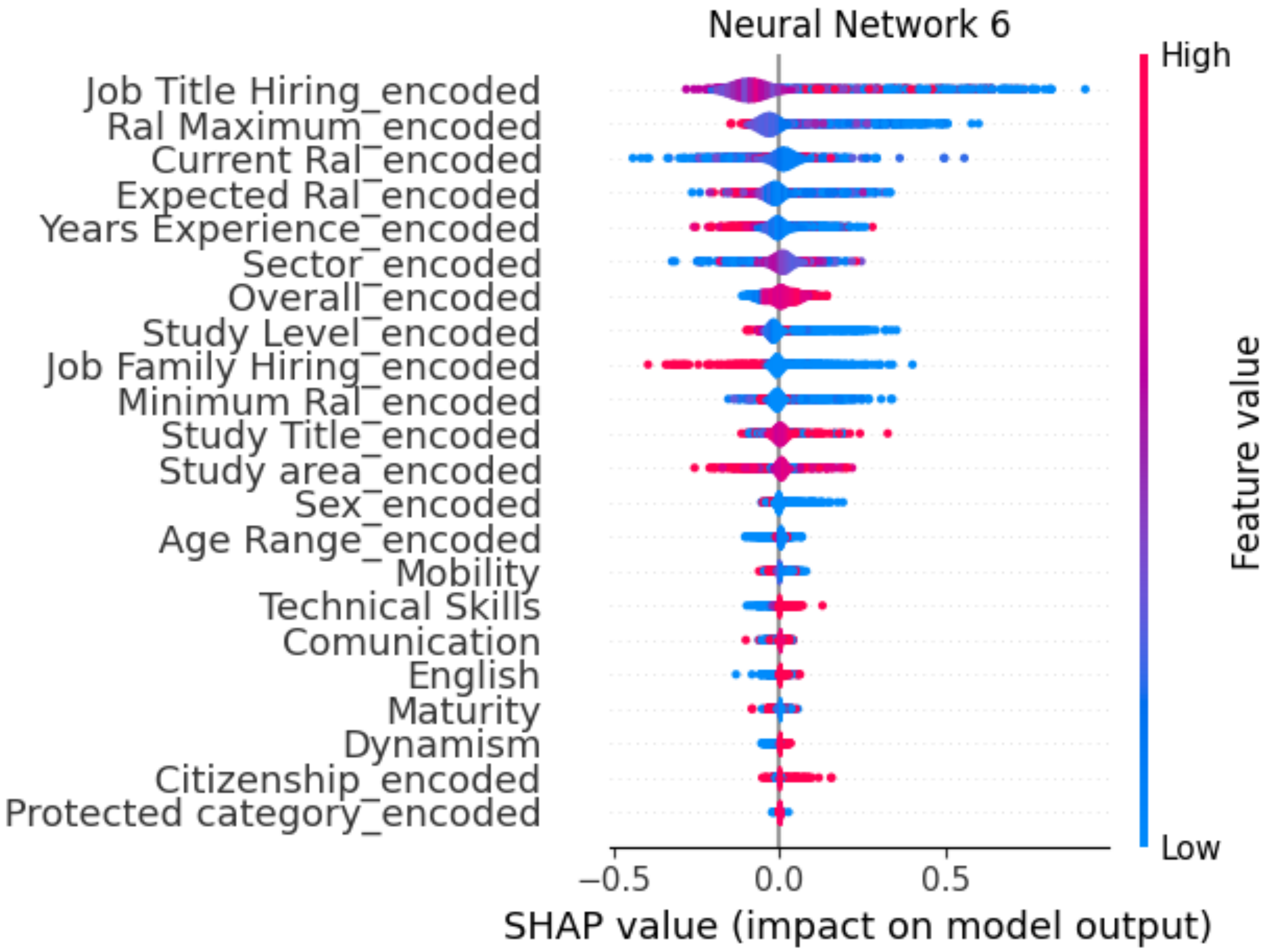
9

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



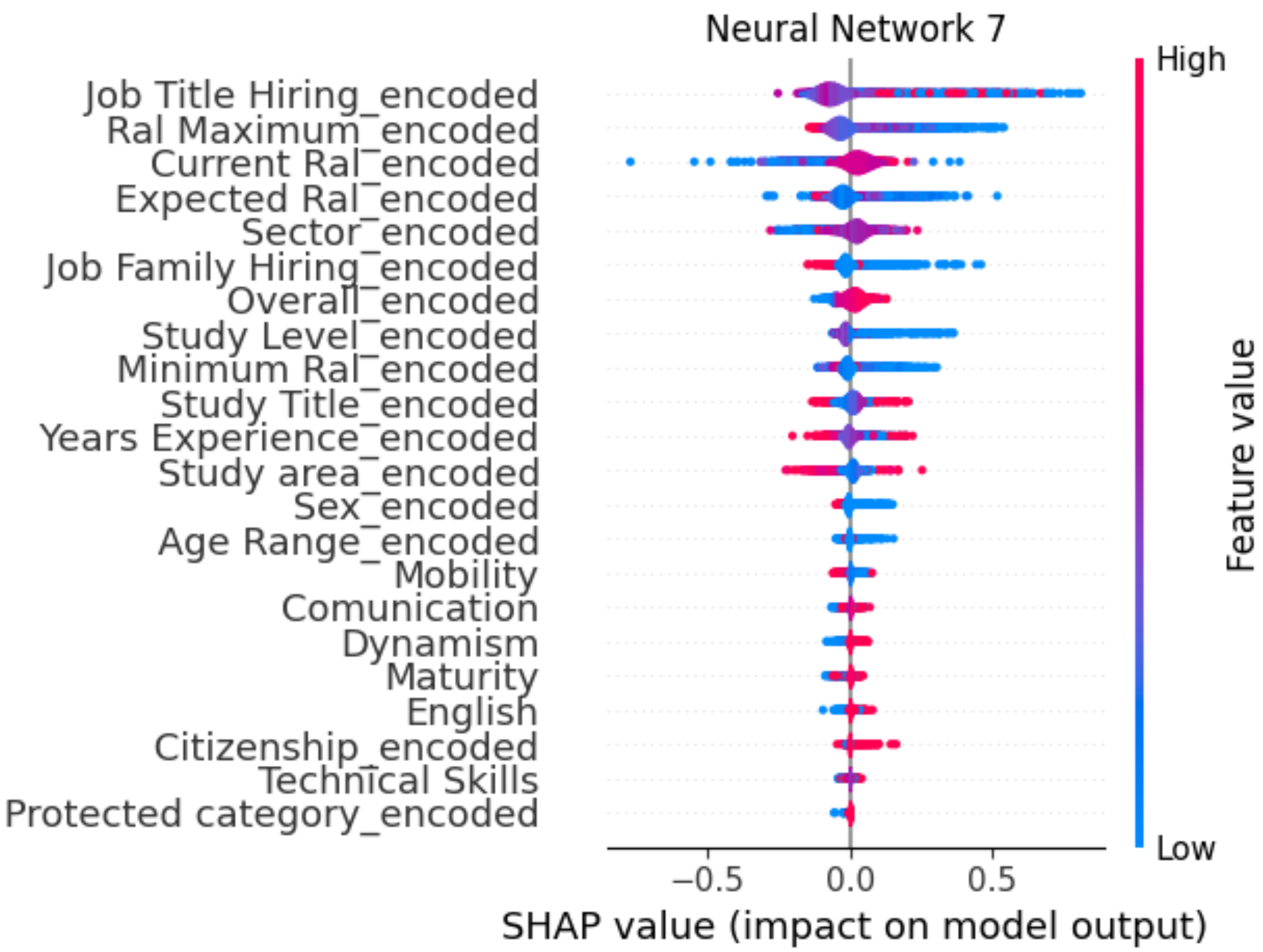

```
10
```

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



11

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



Task 5 - Fairness improvement techniques

5.1 Reweighing

5.1.1 Build the rewrighted dataset

```
def reweight_dataset(df, sensitive_features, target_col='STATUS', random_seed=None)
    if random_seed is not None:
        np.random.seed(random_seed)

    X = df.drop(columns=[target_col])
    y = df[target_col]

    # Calculate weights based on sensitive feature distribution
    group_counts = df.groupby(sensitive_features).size()
    group_weights = 1 / group_counts
    group_weights /= group_weights.sum()

    # Map weights to each sample based on its group membership
    sample_weights = df[sensitive_features].apply(tuple, axis=1).map(group_weights)

    # Normalize sample weights
    sample_weights /= sample_weights.sum()
```

```
    # Resample the dataset based on the calculated sample weights
    reweighted_indices = np.random.choice(df.index, size=len(df), replace=True, p=s
    df_reweighted = df.loc[reweighted_indices]

    X_train_reweighted = df_reweighted.drop(columns=[target_col])
    y_train_reweighted = df_reweighted[target_col]
    #print(X_train_reweighted.shape)

    return X_train_reweighted, y_train_reweighted
```

```
X_train_reweighted, y_train_reweighted = reweight_dataset(df, sensitive_features, '
```

```
print("Distribution before reweighting:")
print(df[sensitive_features].value_counts())

print("\n\nDistribution after reweighting:")
print(X_train_reweighted[sensitive_features].value_counts())
```

```
Distribution before reweighting:
 Sex_encoded   Age Range_encoded   Citizenship_encoded   Protected category_encode
d
1              1                   0                     1
5980
               0                   0                     1
1612
0              1                   0                     1
1597
1              1                   1                     1
292
0              0                   0                     1
247
               1                   1                     1
35
1              0                   1                     1
27
               1                   0                     0
19
               0                   0                     0
16
0              1                   0                     0
14
               0                   0                     0
9
                                   1                     1
9
Name: count, dtype: int64


Distribution after reweighting:
 Sex_encoded   Age Range_encoded   Citizenship_encoded   Protected category_encode
d
0              1                   0                     0
844
1              0                   0                     1
844
0              0                   0                     0
```

```
840
                                                             1

833
              1                   1                          1
833
1             1                   1                          1
823
0             0                   1                          1
821
              1                   0                          1
820
1             0                   1                          1
819
                                  0                          0

806
              1                   0                          0
787
                                                             1

787
Name: count, dtype: int64
```

5.1.2 Re-fit and evaluate the machine learning models

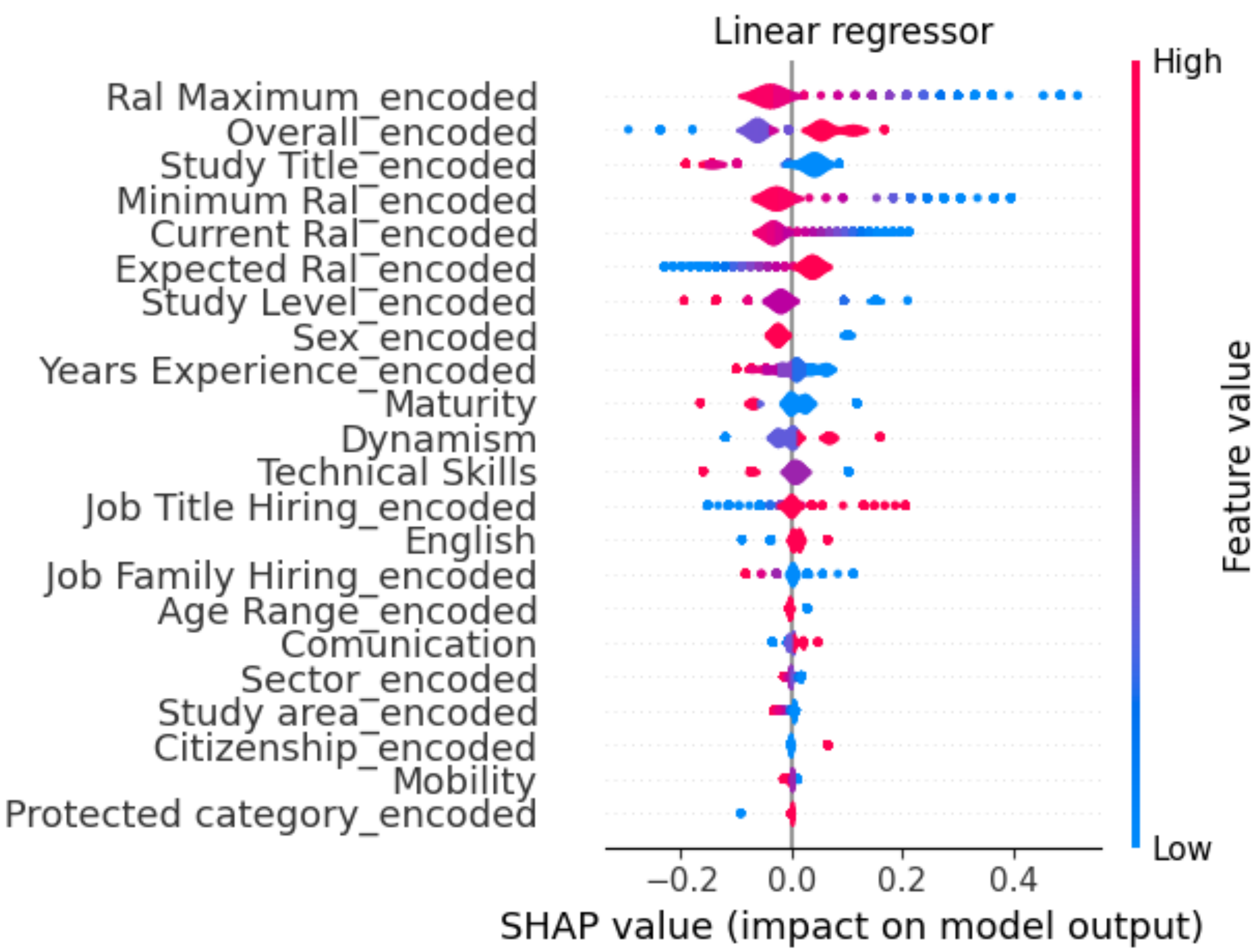
```
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Naive Bayes': GaussianNB(),
    'XGBoost': XGBClassifier(),
    'KNN': KNeighborsClassifier(),
}
```

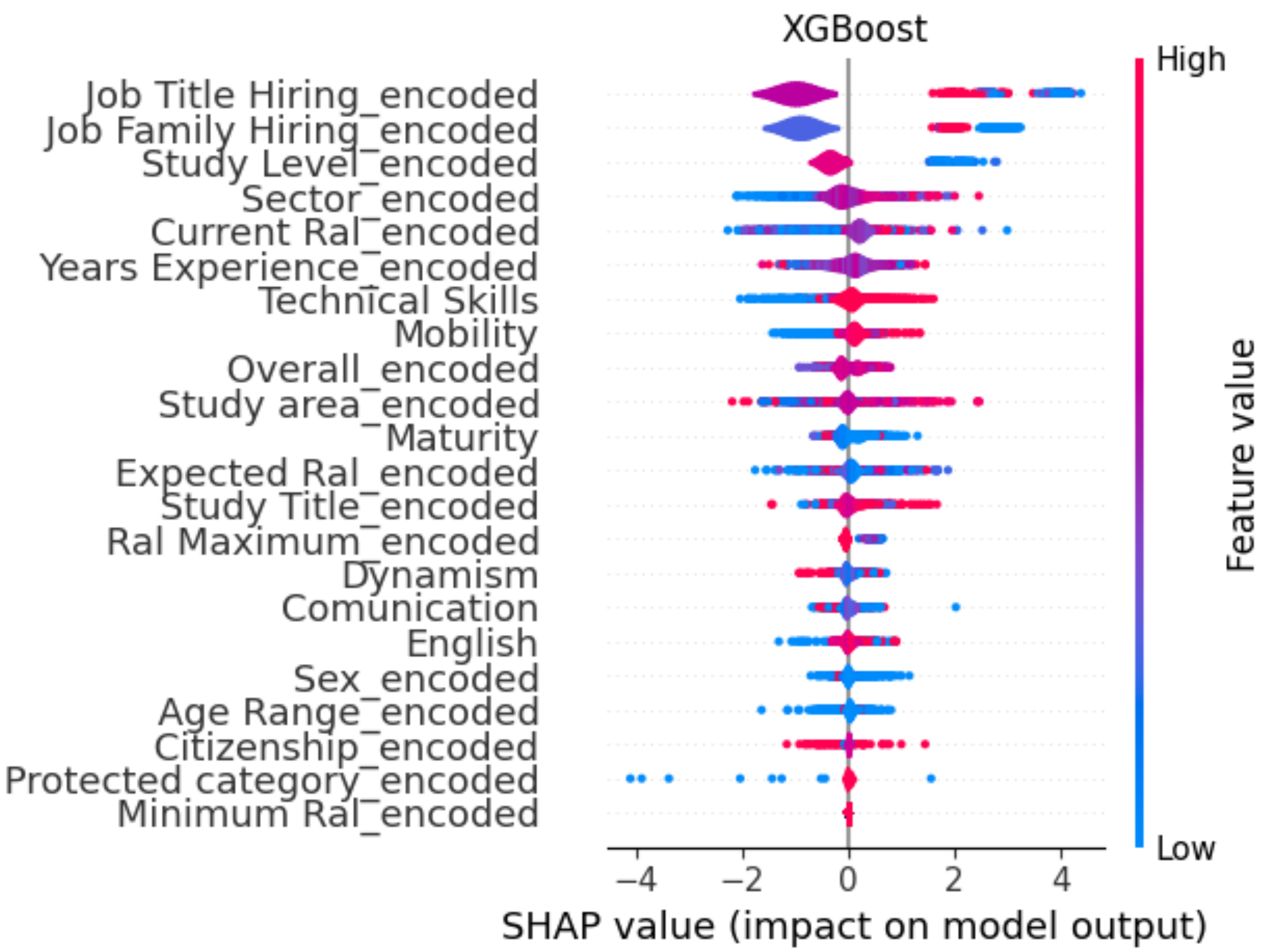
```
metrics = []
predictions = {}

# Fit models and evaluate
for name, model in models.items():
    model.fit(X_train_reweighted, y_train_reweighted)
    y_pred = model.predict(X_test)

    if name in ['Linear Regression', 'XGBoost']:
        y_pred = (y_pred > 0.5).astype(int)

    # Store predictions
    predictions[name] = y_pred

    accuracy = round(accuracy_score(y_test, y_pred), 3)
    precision = round(precision_score(y_test, y_pred), 3)
    recall = round(recall_score(y_test, y_pred), 3)
    f1 = round(f1_score(y_test, y_pred), 3)
    roc_auc = round(roc_auc_score(y_test, y_pred), 3)

    # Append metrics to the DataFrame
    metrics.append({
        'Model': name,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
```

```
        'F1-score': f1,
        'ROC AUC': roc_auc
    })

metrics = pd.DataFrame(metrics)
metrics
```

	Model	Accuracy	Precision	Recall	F1-score	ROC AUC
0	Linear Regression	0.678	0.762	0.468	0.580	0.668
1	Decision Tree	0.809	0.812	0.778	0.795	0.808
2	Naive Bayes	0.779	1.000	0.534	0.696	0.767
3	XGBoost	0.809	0.844	0.734	0.785	0.806
4	KNN	0.759	0.780	0.684	0.729	0.755

```
summaryPlot(models[models_list[0]], X_test, tot_columns, plot_type='violin', plot_r
summaryPlot(models[models_list[3]], X_test, tot_columns, plot_type='violin', plot_r
```





5.1.3 Re-fit and evaluate the Neural Network

```
neural_models = []

# Create and compile the 7 models
for seed in range(85,92):
    np.random.seed(seed)
    tf.random.set_seed(seed)
    model = create_model()
    neural_models.append(model)
```

```
# Fit the models
histories = []
for i, model in enumerate(neural_models):
    print(f"Fitting model {i+1}...")
    history = model.fit(X_train_reweighted, y_train_reweighted, epochs=15, batch_si
    histories.append(history)
    print(f"Model {i+1} fitted.\n")
```

```
Fitting model 1...
Epoch 1/15
124/124 [==============================] - 2s 5ms/step - loss: 0.3550 - accuracy:
0.8411 - val_loss: 0.4688 - val_accuracy: 0.7819
Epoch 2/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2681 - accuracy:
0.8855 - val_loss: 0.2945 - val_accuracy: 0.8803
Epoch 3/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2402 - accuracy:
```

```
0.8938 - val_loss: 0.2149 - val_accuracy: 0.9128
Epoch 4/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2097 - accuracy:
0.9136 - val_loss: 0.2706 - val_accuracy: 0.8793
Epoch 5/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2058 - accuracy:
0.9121 - val_loss: 0.3044 - val_accuracy: 0.8540
Epoch 6/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2030 - accuracy:
0.9091 - val_loss: 0.2229 - val_accuracy: 0.8955
Epoch 7/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1946 - accuracy:
0.9138 - val_loss: 0.4175 - val_accuracy: 0.8205
Epoch 8/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1781 - accuracy:
0.9219 - val_loss: 0.2332 - val_accuracy: 0.8849
Epoch 9/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1817 - accuracy:
0.9191 - val_loss: 0.1955 - val_accuracy: 0.9133
Epoch 10/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1770 - accuracy:
0.9238 - val_loss: 0.1890 - val_accuracy: 0.9173
Epoch 11/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1657 - accuracy:
0.9266 - val_loss: 0.1923 - val_accuracy: 0.9042
Epoch 12/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1613 - accuracy:
0.9250 - val_loss: 0.2222 - val_accuracy: 0.9082
Epoch 13/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1617 - accuracy:
0.9289 - val_loss: 0.2639 - val_accuracy: 0.8778
Epoch 14/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1540 - accuracy:
0.9306 - val_loss: 0.2215 - val_accuracy: 0.9092
Epoch 15/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1496 - accuracy:
0.9347 - val_loss: 0.2572 - val_accuracy: 0.8671
Model 1 fitted.

Fitting model 2...
Epoch 1/15
124/124 [==============================] - 2s 5ms/step - loss: 0.3375 - accuracy:
0.8478 - val_loss: 0.5204 - val_accuracy: 0.7338
Epoch 2/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2542 - accuracy:
0.8950 - val_loss: 0.3095 - val_accuracy: 0.8803
Epoch 3/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2297 - accuracy:
0.9011 - val_loss: 0.2164 - val_accuracy: 0.9163
Epoch 4/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2079 - accuracy:
0.9108 - val_loss: 0.2504 - val_accuracy: 0.8844
Epoch 5/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2044 - accuracy:
0.9106 - val_loss: 0.2393 - val_accuracy: 0.8732
Epoch 6/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1947 - accuracy:
```

```
0.9167 - val_loss: 0.1995 - val_accuracy: 0.9128
Epoch 7/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1925 - accuracy:
0.9145 - val_loss: 0.2118 - val_accuracy: 0.9204
Epoch 8/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1791 - accuracy:
0.9206 - val_loss: 0.2573 - val_accuracy: 0.8656
Epoch 9/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1702 - accuracy:
0.9228 - val_loss: 0.1999 - val_accuracy: 0.9163
Epoch 10/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1738 - accuracy:
0.9242 - val_loss: 0.2399 - val_accuracy: 0.8910
Epoch 11/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1638 - accuracy:
0.9272 - val_loss: 0.2492 - val_accuracy: 0.8727
Epoch 12/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1595 - accuracy:
0.9278 - val_loss: 0.1984 - val_accuracy: 0.9148
Epoch 13/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1536 - accuracy:
0.9311 - val_loss: 0.2108 - val_accuracy: 0.9037
Epoch 14/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1592 - accuracy:
0.9300 - val_loss: 0.1672 - val_accuracy: 0.9310
Epoch 15/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1571 - accuracy:
0.9310 - val_loss: 0.1636 - val_accuracy: 0.9224
Model 2 fitted.

Fitting model 3...
Epoch 1/15
124/124 [==============================] - 2s 5ms/step - loss: 0.3331 - accuracy:
0.8530 - val_loss: 0.5508 - val_accuracy: 0.7688
Epoch 2/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2572 - accuracy:
0.8889 - val_loss: 0.3201 - val_accuracy: 0.8519
Epoch 3/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2346 - accuracy:
0.9006 - val_loss: 0.2437 - val_accuracy: 0.9011
Epoch 4/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2098 - accuracy:
0.9135 - val_loss: 0.2342 - val_accuracy: 0.8859
Epoch 5/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2071 - accuracy:
0.9096 - val_loss: 0.2735 - val_accuracy: 0.8732
Epoch 6/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1902 - accuracy:
0.9186 - val_loss: 0.1960 - val_accuracy: 0.9184
Epoch 7/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1845 - accuracy:
0.9205 - val_loss: 0.2013 - val_accuracy: 0.9113
Epoch 8/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1827 - accuracy:
0.9209 - val_loss: 0.1867 - val_accuracy: 0.9006
Epoch 9/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1756 - accuracy:
```

```
0.9237 - val_loss: 0.2441 - val_accuracy: 0.8844
Epoch 10/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1779 - accuracy:
0.9230 - val_loss: 0.2186 - val_accuracy: 0.8991
Epoch 11/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1700 - accuracy:
0.9237 - val_loss: 0.1995 - val_accuracy: 0.9037
Epoch 12/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1670 - accuracy:
0.9244 - val_loss: 0.1718 - val_accuracy: 0.9300
Epoch 13/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1681 - accuracy:
0.9267 - val_loss: 0.1983 - val_accuracy: 0.8955
Epoch 14/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1580 - accuracy:
0.9289 - val_loss: 0.2190 - val_accuracy: 0.8834
Epoch 15/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1688 - accuracy:
0.9257 - val_loss: 0.1948 - val_accuracy: 0.9234
Model 3 fitted.

Fitting model 4...
Epoch 1/15
124/124 [==============================] - 3s 5ms/step - loss: 0.3584 - accuracy:
0.8380 - val_loss: 0.6069 - val_accuracy: 0.5928
Epoch 2/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2648 - accuracy:
0.8845 - val_loss: 0.3081 - val_accuracy: 0.8469
Epoch 3/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2378 - accuracy:
0.8950 - val_loss: 0.2308 - val_accuracy: 0.9042
Epoch 4/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2152 - accuracy:
0.9069 - val_loss: 0.2158 - val_accuracy: 0.9062
Epoch 5/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2038 - accuracy:
0.9116 - val_loss: 0.2103 - val_accuracy: 0.8960
Epoch 6/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1921 - accuracy:
0.9153 - val_loss: 0.2187 - val_accuracy: 0.9128
Epoch 7/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1825 - accuracy:
0.9212 - val_loss: 0.2413 - val_accuracy: 0.9108
Epoch 8/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1801 - accuracy:
0.9197 - val_loss: 0.2029 - val_accuracy: 0.9006
Epoch 9/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1783 - accuracy:
0.9234 - val_loss: 0.1899 - val_accuracy: 0.9260
Epoch 10/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1768 - accuracy:
0.9250 - val_loss: 0.1959 - val_accuracy: 0.9042
Epoch 11/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1631 - accuracy:
0.9285 - val_loss: 0.1926 - val_accuracy: 0.9016
Epoch 12/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1582 - accuracy:
```

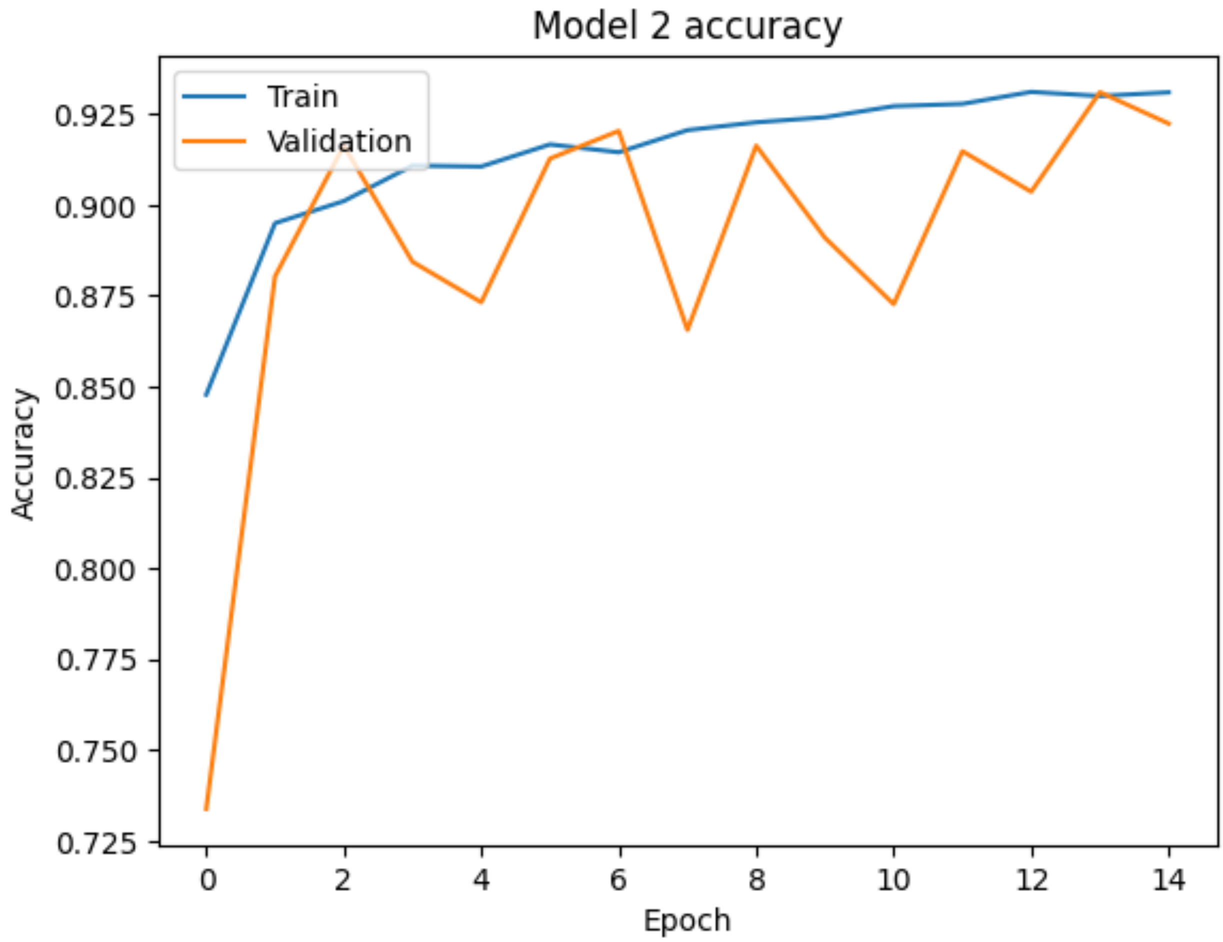
```
0.9299 - val_loss: 0.2293 - val_accuracy: 0.8712
Epoch 13/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1568 - accuracy:
0.9294 - val_loss: 0.1772 - val_accuracy: 0.9255
Epoch 14/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1672 - accuracy:
0.9269 - val_loss: 0.1658 - val_accuracy: 0.9280
Epoch 15/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1525 - accuracy:
0.9320 - val_loss: 0.1993 - val_accuracy: 0.8966
Model 4 fitted.

Fitting model 5...
Epoch 1/15
124/124 [==============================] - 2s 5ms/step - loss: 0.3470 - accuracy:
0.8457 - val_loss: 0.4632 - val_accuracy: 0.7779
Epoch 2/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2553 - accuracy:
0.8946 - val_loss: 0.2966 - val_accuracy: 0.8616
Epoch 3/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2307 - accuracy:
0.9006 - val_loss: 0.2382 - val_accuracy: 0.8895
Epoch 4/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2139 - accuracy:
0.9079 - val_loss: 0.2567 - val_accuracy: 0.8900
Epoch 5/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1996 - accuracy:
0.9148 - val_loss: 0.2348 - val_accuracy: 0.8991
Epoch 6/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1933 - accuracy:
0.9145 - val_loss: 0.2103 - val_accuracy: 0.9118
Epoch 7/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1912 - accuracy:
0.9150 - val_loss: 0.2121 - val_accuracy: 0.9052
Epoch 8/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1783 - accuracy:
0.9211 - val_loss: 0.2018 - val_accuracy: 0.9123
Epoch 9/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1705 - accuracy:
0.9247 - val_loss: 0.2351 - val_accuracy: 0.8788
Epoch 10/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1765 - accuracy:
0.9220 - val_loss: 0.2056 - val_accuracy: 0.9067
Epoch 11/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1641 - accuracy:
0.9280 - val_loss: 0.1637 - val_accuracy: 0.9300
Epoch 12/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1615 - accuracy:
0.9267 - val_loss: 0.2095 - val_accuracy: 0.9052
Epoch 13/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1517 - accuracy:
0.9328 - val_loss: 0.1897 - val_accuracy: 0.9270
Epoch 14/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1631 - accuracy:
0.9276 - val_loss: 0.1686 - val_accuracy: 0.9260
Epoch 15/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1530 - accuracy:
```

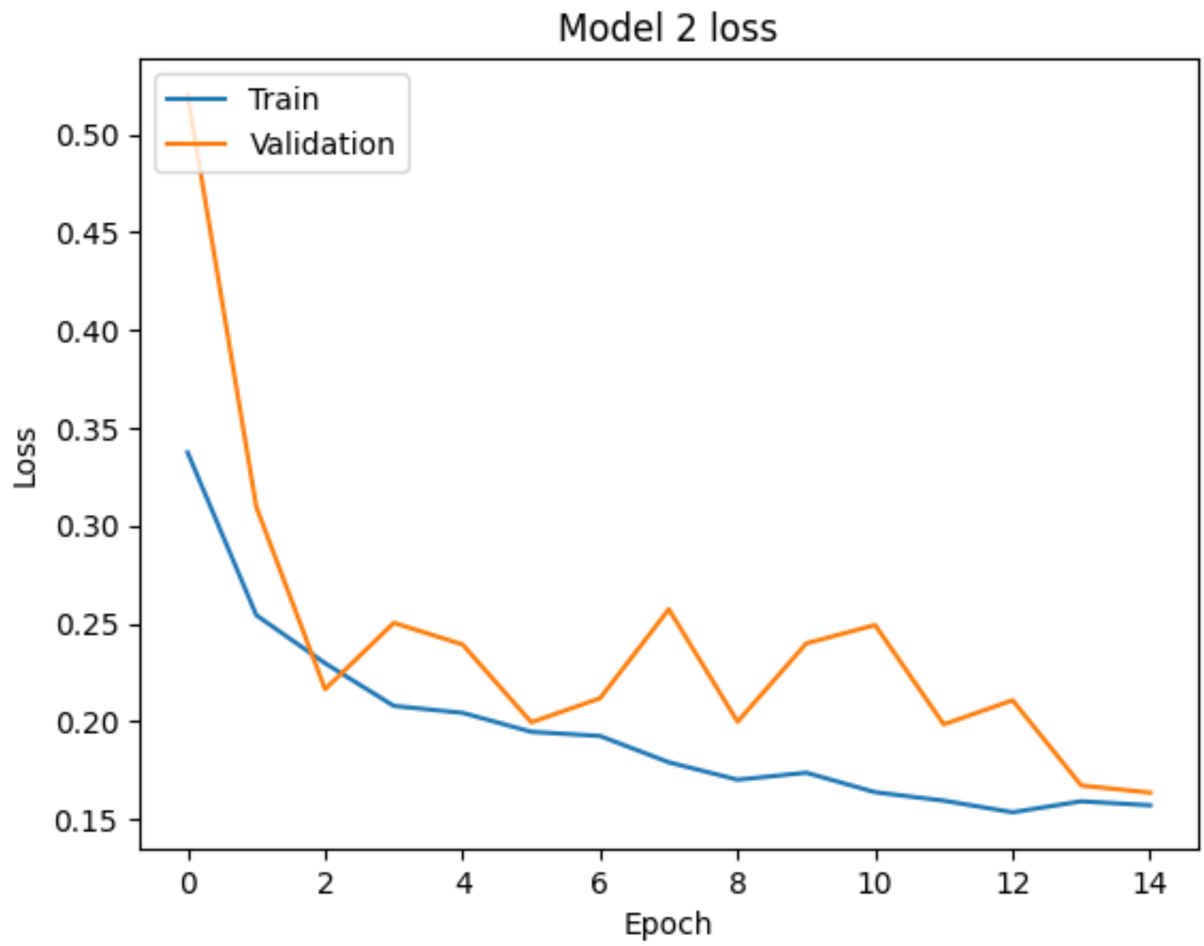
```
0.9320 - val_loss: 0.2311 - val_accuracy: 0.8915
Model 5 fitted.

Fitting model 6...
Epoch 1/15
124/124 [==============================] - 2s 5ms/step - loss: 0.3483 - accuracy:
0.8341 - val_loss: 0.5467 - val_accuracy: 0.6354
Epoch 2/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2658 - accuracy:
0.8867 - val_loss: 0.3564 - val_accuracy: 0.8266
Epoch 3/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2425 - accuracy:
0.8945 - val_loss: 0.2399 - val_accuracy: 0.8793
Epoch 4/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2146 - accuracy:
0.9077 - val_loss: 0.2155 - val_accuracy: 0.9031
Epoch 5/15
124/124 [==============================] - 0s 3ms/step - loss: 0.2093 - accuracy:
0.9098 - val_loss: 0.2421 - val_accuracy: 0.8793
Epoch 6/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1947 - accuracy:
0.9157 - val_loss: 0.2207 - val_accuracy: 0.8905
Epoch 7/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1977 - accuracy:
0.9138 - val_loss: 0.2040 - val_accuracy: 0.9244
Epoch 8/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1849 - accuracy:
0.9183 - val_loss: 0.2120 - val_accuracy: 0.8991
Epoch 9/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1820 - accuracy:
0.9205 - val_loss: 0.1766 - val_accuracy: 0.9244
Epoch 10/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1758 - accuracy:
0.9250 - val_loss: 0.1860 - val_accuracy: 0.9260
Epoch 11/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1663 - accuracy:
0.9262 - val_loss: 0.1779 - val_accuracy: 0.9255
Epoch 12/15
124/124 [==============================] - 0s 3ms/step - loss: 0.1617 - accuracy:
0.9267 - val_loss: 0.1733 - val_accuracy: 0.9234
Epoch 13/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1587 - accuracy:
0.9308 - val_loss: 0.1742 - val_accuracy: 0.9270
Epoch 14/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1544 - accuracy:
0.9299 - val_loss: 0.1685 - val_accuracy: 0.9229
Epoch 15/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1508 - accuracy:
0.9324 - val_loss: 0.2014 - val_accuracy: 0.9037
Model 6 fitted.

Fitting model 7...
Epoch 1/15
124/124 [==============================] - 2s 5ms/step - loss: 0.3504 - accuracy:
0.8441 - val_loss: 0.5671 - val_accuracy: 0.6846
Epoch 2/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2694 - accuracy:
```

```
0.8866 - val_loss: 0.3201 - val_accuracy: 0.8712
Epoch 3/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2357 - accuracy:
0.8970 - val_loss: 0.2296 - val_accuracy: 0.9037
Epoch 4/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2124 - accuracy:
0.9087 - val_loss: 0.2172 - val_accuracy: 0.9067
Epoch 5/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2056 - accuracy:
0.9127 - val_loss: 0.2092 - val_accuracy: 0.9118
Epoch 6/15
124/124 [==============================] - 0s 4ms/step - loss: 0.2006 - accuracy:
0.9130 - val_loss: 0.1870 - val_accuracy: 0.9189
Epoch 7/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1890 - accuracy:
0.9172 - val_loss: 0.1922 - val_accuracy: 0.9290
Epoch 8/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1872 - accuracy:
0.9193 - val_loss: 0.1848 - val_accuracy: 0.9239
Epoch 9/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1787 - accuracy:
0.9234 - val_loss: 0.2013 - val_accuracy: 0.9037
Epoch 10/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1841 - accuracy:
0.9209 - val_loss: 0.2192 - val_accuracy: 0.8966
Epoch 11/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1726 - accuracy:
0.9256 - val_loss: 0.1635 - val_accuracy: 0.9275
Epoch 12/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1619 - accuracy:
0.9249 - val_loss: 0.1834 - val_accuracy: 0.9189
Epoch 13/15
124/124 [==============================] - 1s 4ms/step - loss: 0.1567 - accuracy:
0.9313 - val_loss: 0.2183 - val_accuracy: 0.8874
Epoch 14/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1554 - accuracy:
0.9304 - val_loss: 0.1739 - val_accuracy: 0.9341
Epoch 15/15
124/124 [==============================] - 0s 4ms/step - loss: 0.1499 - accuracy:
0.9338 - val_loss: 0.1735 - val_accuracy: 0.9239
Model 7 fitted.
```

```python
# Check training procedure
plt.plot(histories[1].history['accuracy'])
plt.plot(histories[1].history['val_accuracy'])
plt.title('Model 2 accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(histories[1].history['loss'])
plt.plot(histories[1].history['val_loss'])
plt.title('Model 2 loss')
plt.ylabel('Loss')
```

```
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```





```python
neural_predictions = []

for i, model in enumerate(neural_models):
    print(f"Predicting with model {i+1}...")
    y_pred = (model.predict(X_test) > 0.5).astype("int32")
    neural_predictions.append(y_pred)
    print(f"Predictions from model {i+1} stored.\n")
```

```
Predicting with model 1...
62/62 [==============================] - 0s 2ms/step
Predictions from model 1 stored.

Predicting with model 2...
62/62 [==============================] - 0s 2ms/step
Predictions from model 2 stored.

Predicting with model 3...
62/62 [==============================] - 0s 1ms/step
Predictions from model 3 stored.

Predicting with model 4...
62/62 [==============================] - 0s 1ms/step
Predictions from model 4 stored.

Predicting with model 5...
62/62 [==============================] - 0s 1ms/step
Predictions from model 5 stored.

Predicting with model 6...
62/62 [==============================] - 0s 1ms/step
Predictions from model 6 stored.

Predicting with model 7...
62/62 [==============================] - 0s 2ms/step
Predictions from model 7 stored.
```

```python
nn_metrics = []

for i, y_pred in enumerate(neural_predictions):
    accuracy = round(accuracy_score(y_test, y_pred), 3)
    precision = round(precision_score(y_test, y_pred), 3)
    recall = round(recall_score(y_test, y_pred), 3)
    f1 = round(f1_score(y_test, y_pred), 3)
    roc_auc = round(roc_auc_score(y_test, y_pred), 3)

    nn_metrics.append({
        "Model": f"Neural Network {i+1}",
        "Accuracy": accuracy,
        "Precision": precision,
        "Recall": recall,
        "F1-score": f1,
        "ROC AUC": roc_auc
    })

# Save the 7 models performances
nn_metrics = pd.DataFrame(nn_metrics)
```

```python
combined_metrics = pd.concat([metrics, nn_metrics], ignore_index=True)
combined_metrics
```

	Model	Accuracy	Precision	Recall	F1-score	ROC AUC
0	Linear Regression	0.678	0.762	0.468	0.580	0.668
1	Decision Tree	0.809	0.812	0.778	0.795	0.808
2	Naive Bayes	0.779	1.000	0.534	0.696	0.767
3	XGBoost	0.809	0.844	0.734	0.785	0.806
4	KNN	0.759	0.780	0.684	0.729	0.755
5	Neural Network 1	0.760	0.802	0.657	0.722	0.755
6	Neural Network 2	0.767	0.770	0.725	0.747	0.765
7	Neural Network 3	0.766	0.846	0.618	0.714	0.758
8	Neural Network 4	0.722	0.687	0.759	0.722	0.724
9	Neural Network 5	0.758	0.805	0.646	0.717	0.753
10	Neural Network 6	0.755	0.768	0.691	0.727	0.751
11	Neural Network 7	0.756	0.757	0.714	0.735	0.754

```python
# Add the NN to the models and to the predicitons
for i, model in enumerate(neural_models):
    models[f"Neural Network {i+1}"] = model

for i, prediction_list in enumerate(neural_predictions):
    predictions_df[f'Neural Network {i+1}'] = prediction_list.flatten()
    predictions[f'Neural Network {i+1}'] = prediction_list.flatten()
```

5.1.4 Demographic Parity and Equalized Loss after the reweighing

```python
# Demographic Parity
table = []

for model in models:
    temp = []
    for i in range(len(sensitive_features)):
        Boolean_Output = calculate_demographic_parity(predictions[model], X_test[se
        temp.append(Boolean_Output)
    table.append(temp)

# DataFrame
dp_df = pd.DataFrame(table, index = list(models.keys()), columns=sensitive_features
dp_df
```

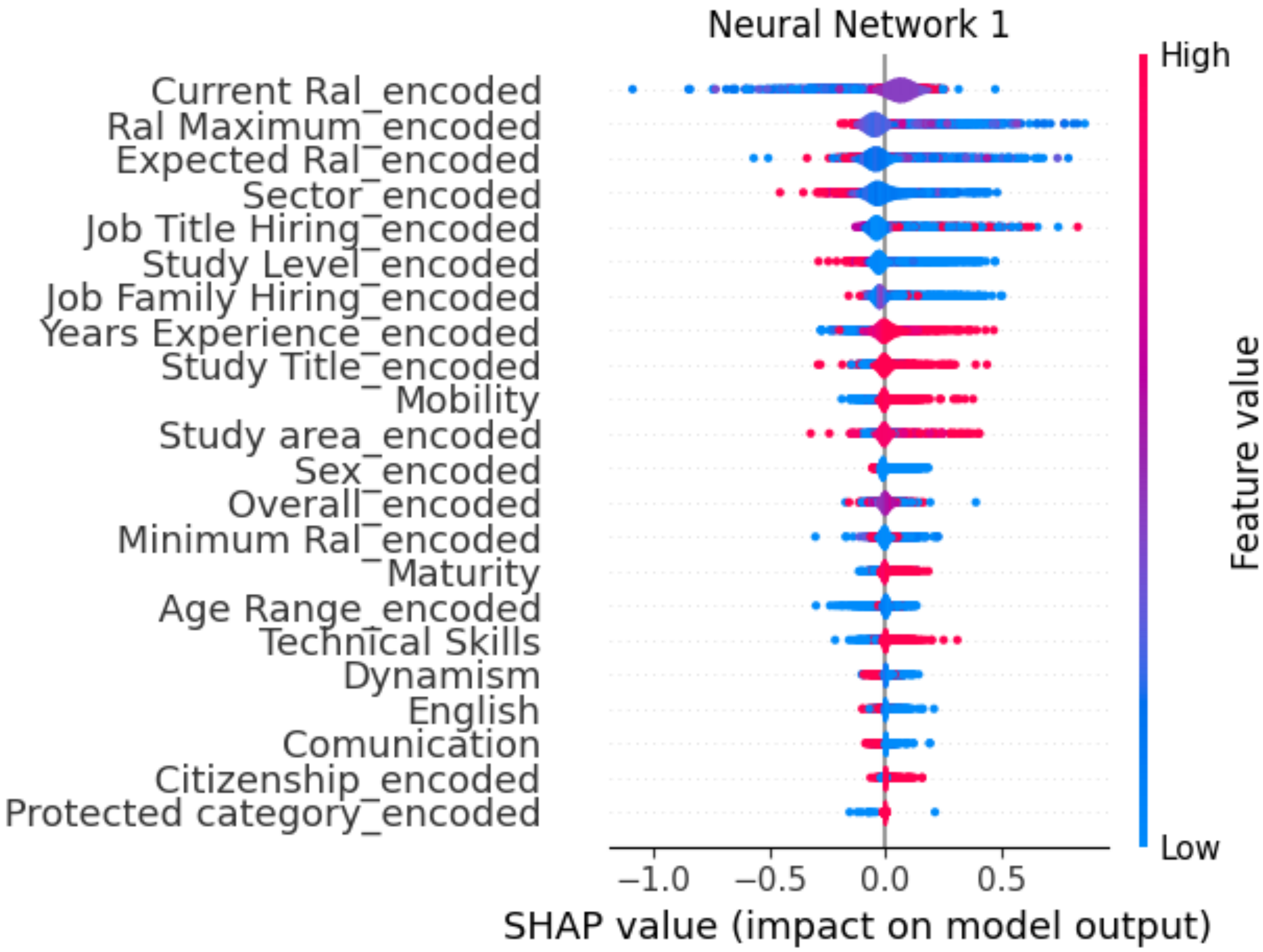
	Sex_encoded	Age Range_encoded	Citizenship_encoded	Protected category_encoded
Linear Regression	False	T	False	False
Decision Tree	T	T	T	T
Naive Bayes	False	T	T	False
XGBoost	T	False	T	T
KNN	False	False	T	T
Neural Network 1	False	T	T	False
Neural Network 2	False	T	T	T
Neural Network 3	False	T	T	False
Neural Network 4	T	T	T	T
Neural Network 5	False	T	T	False
Neural Network 6	T	T	T	T
Neural Network 7	T	False	T	T

```
# Equalized odds
table = []

for model in models:
    temp = []
    for i in range(len(sensitive_features)):
        Boolean_Output = calculate_equalized_odds(predictions[model], y_test, X_tes
        temp.append(Boolean_Output)
    table.append(temp)

# DataFrame
equalized_df = pd.DataFrame(table, index = list(models.keys()), columns=sensitive_f
equalized_df
```

	Sex_encoded	Age Range_encoded	Citizenship_encoded	Protected category_encoded
Linear Regression	False	T	T	False
Decision Tree	T	T	T	T
Naive Bayes	T	T	T	T
XGBoost	T	T	T	T
KNN	T	T	T	False
Neural Network 1	T	T	T	False
Neural Network 2	T	T	T	T
Neural Network 3	T	T	T	False
Neural Network 4	T	T	T	False
Neural Network 5	T	T	T	False
Neural Network 6	T	T	T	False
Neural Network 7	T	T	T	T

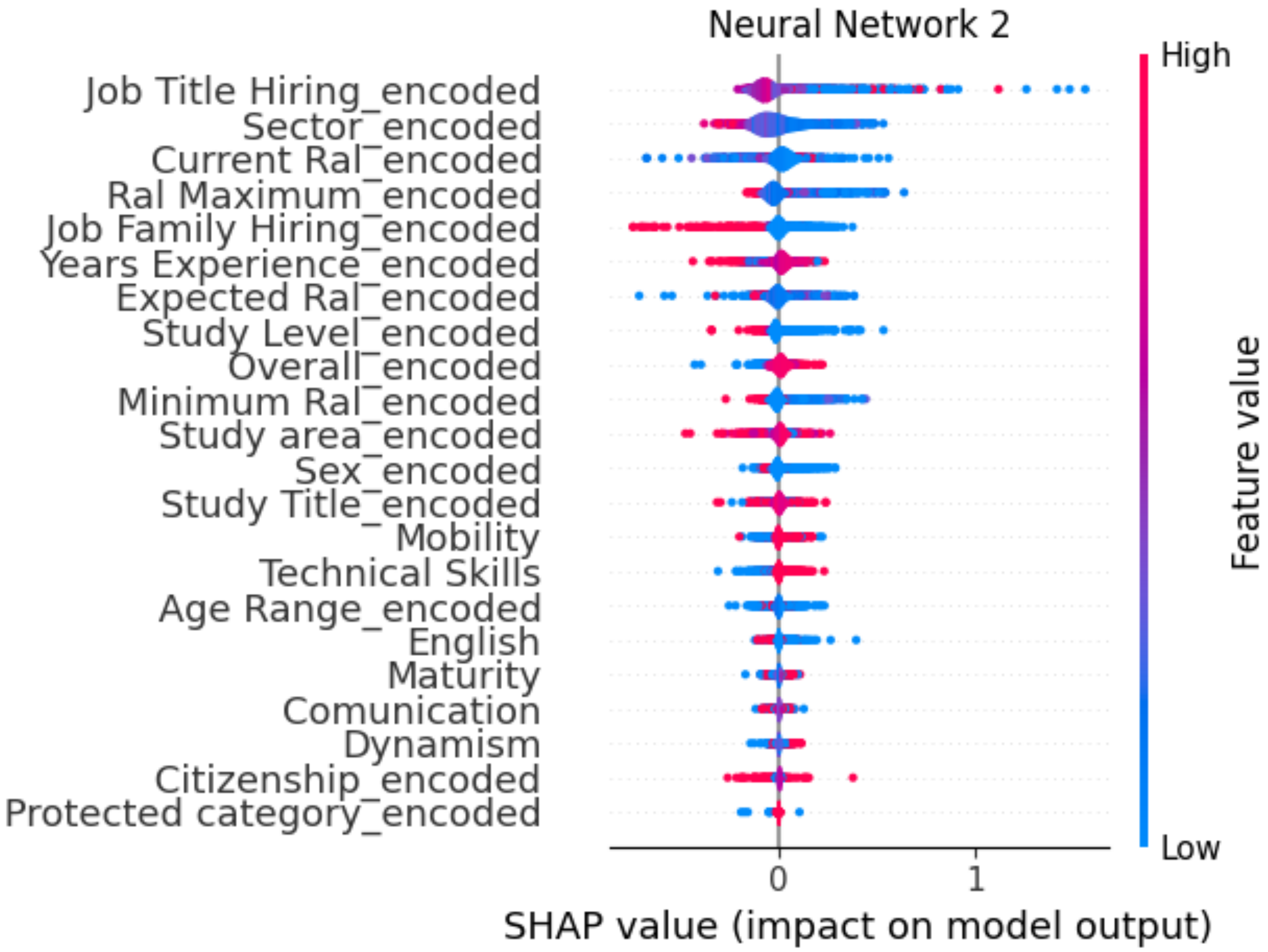
```
for i in range(5, 12):
    print(i)
    summaryPlot(models[models_list[i]], X_test, tot_columns, plot_type='violin', pl
```

```
5
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
(1972, 22)
```



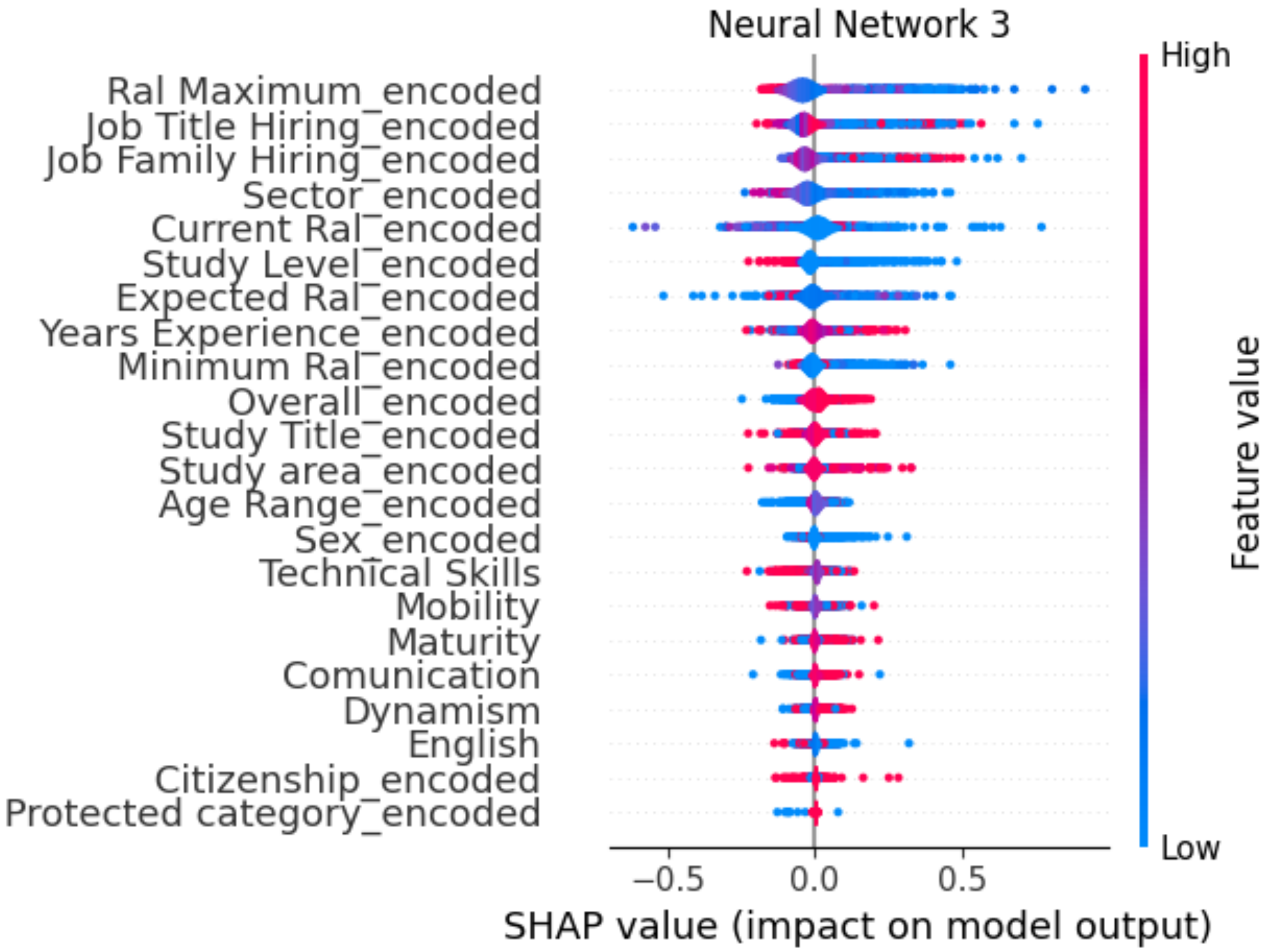
6

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



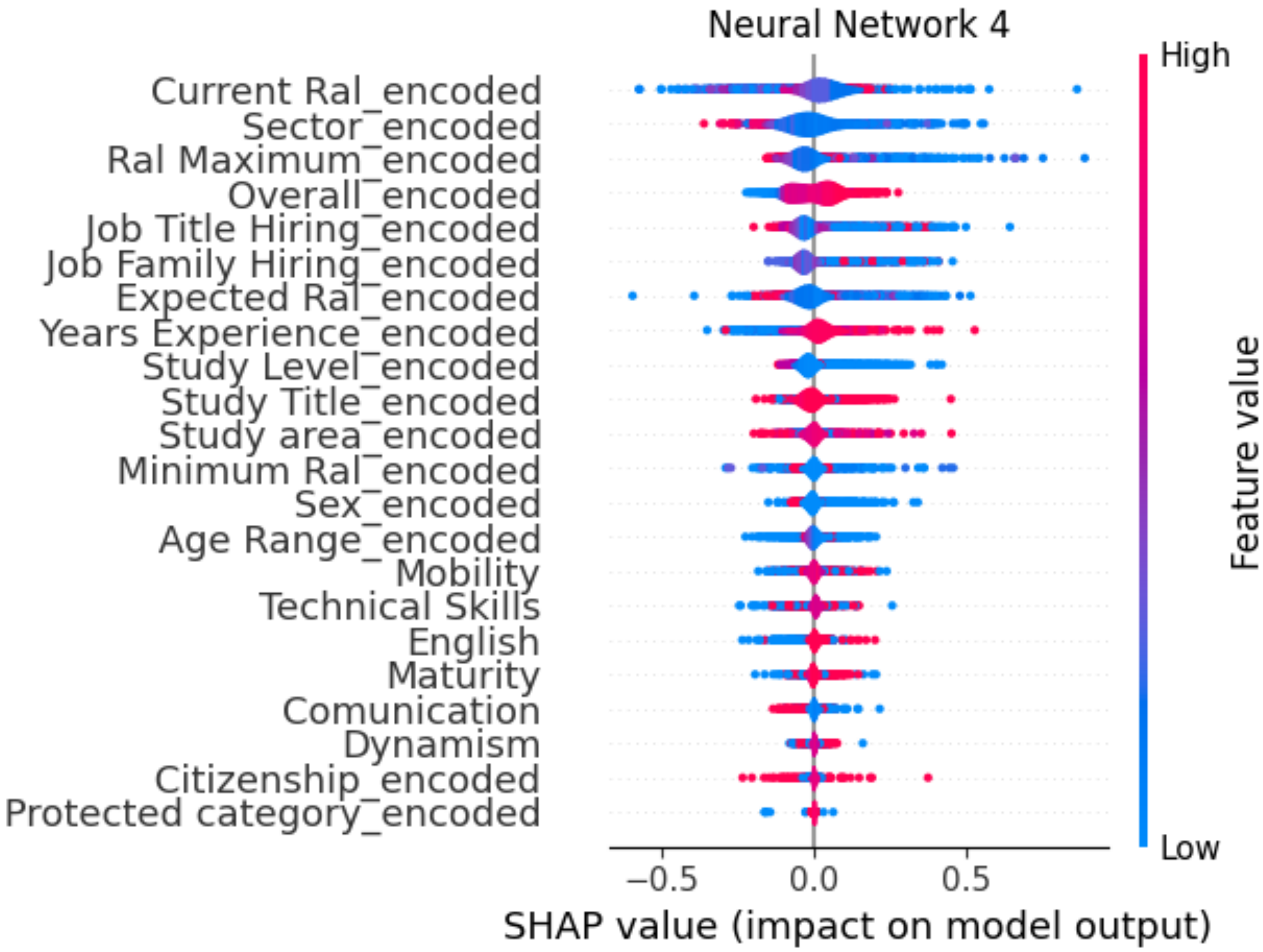
7

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



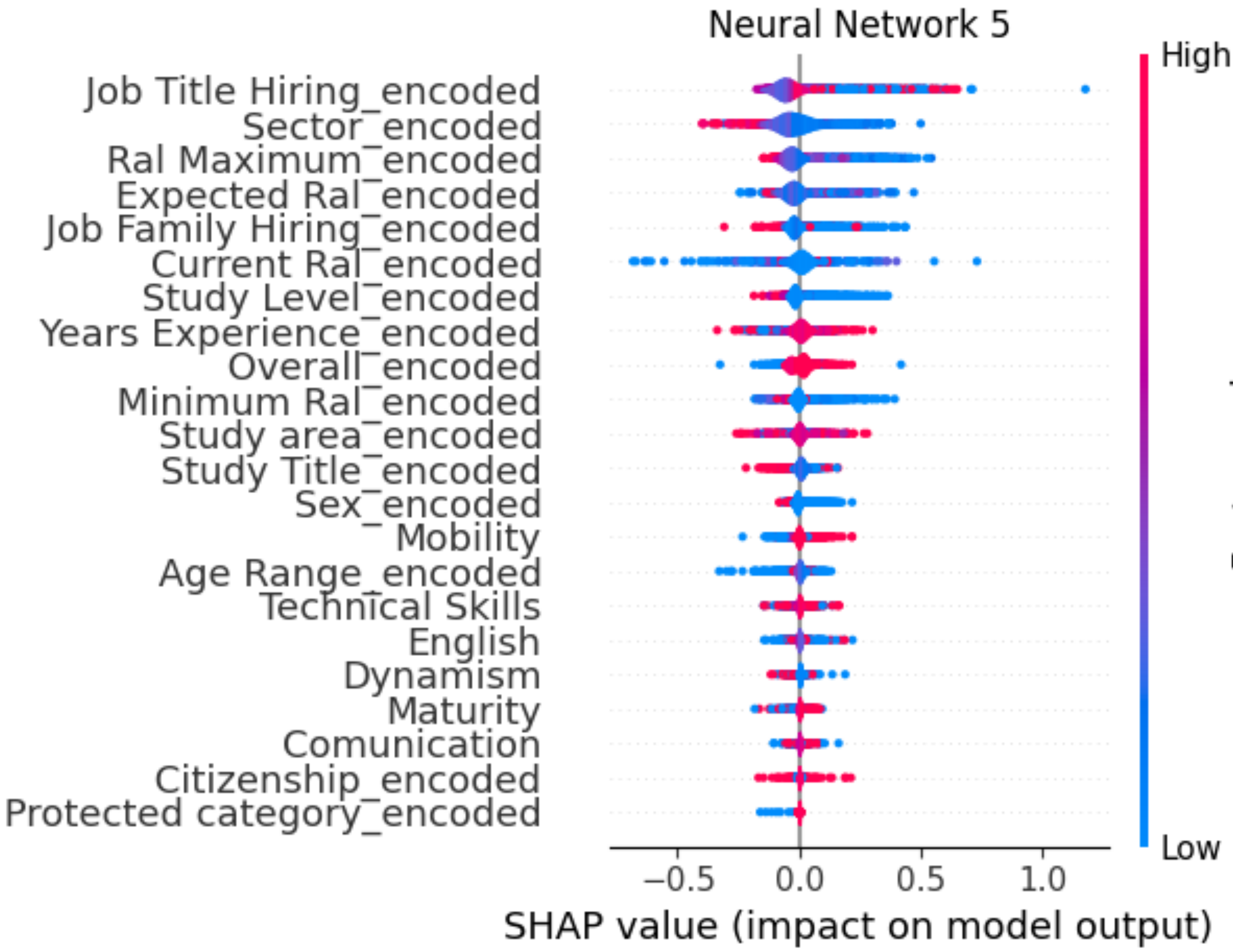
8

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



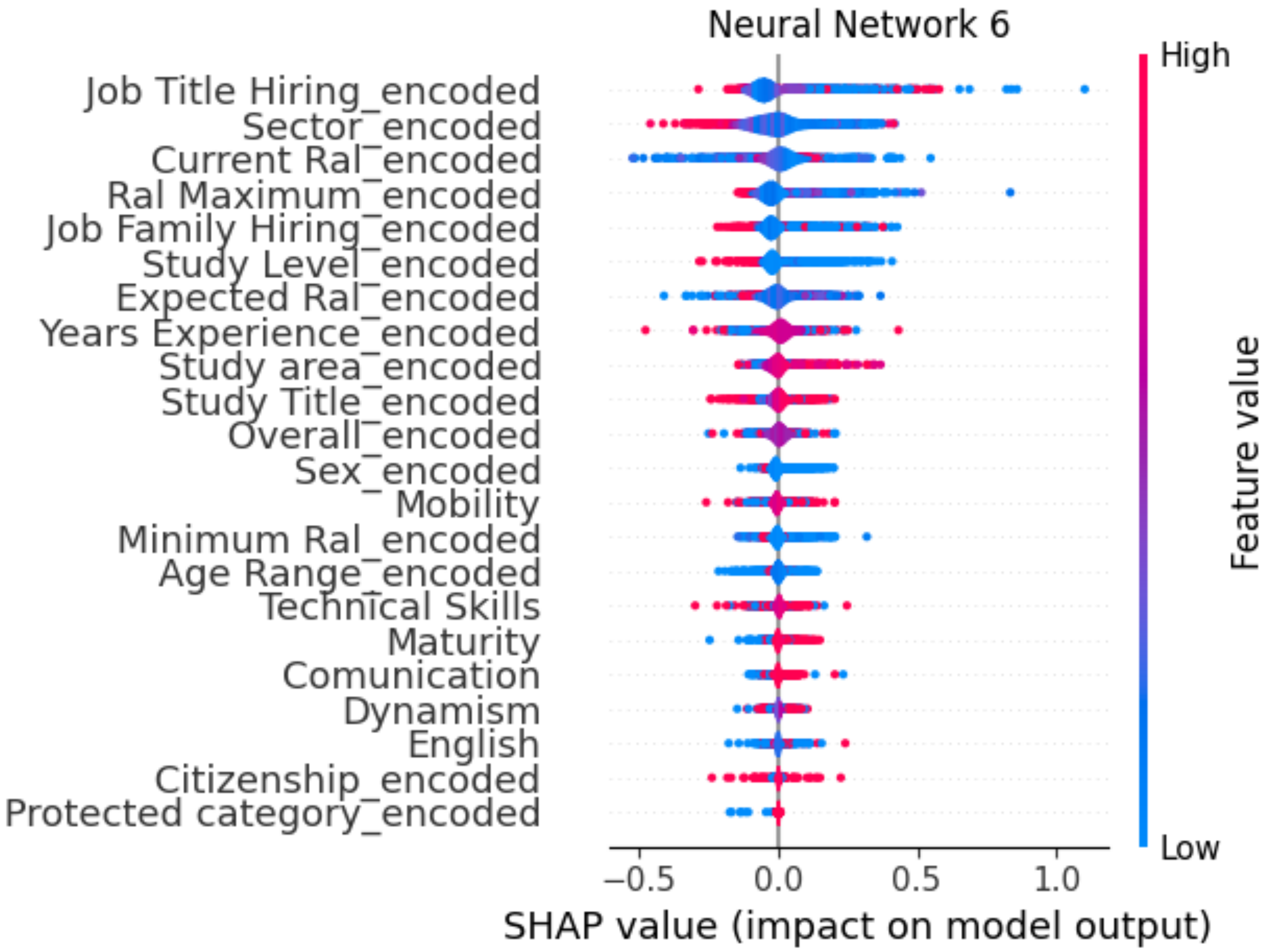
9

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



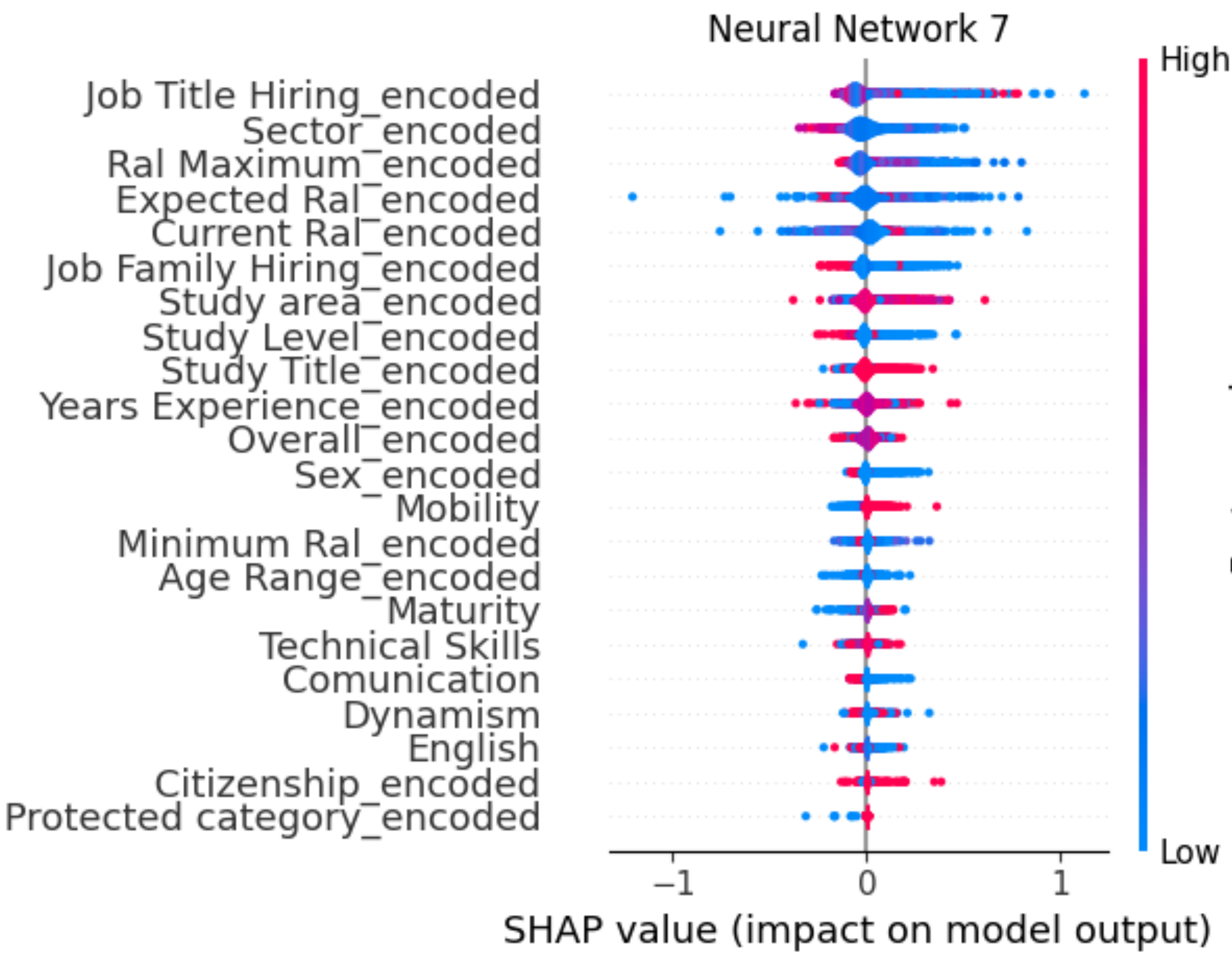

```
10
```

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



11

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



5.2 - Adversarial Debiasing

```
epochs = 15
batch_size = 64
threshold = 0.5
logits = True

names = [' Sex_encoded', ' Age Range_encoded', ' Citizenship_encoded', ' Protected
sensitive_f = [X_train[' Sex_encoded'], X_train[' Age Range_encoded'], X_train[' Ci
```

```
def create_Main():
    model = Sequential([
        Dense(128, input_dim=22, activation='relu'),
        BatchNormalization(),
        Dense(128, activation='relu'),
        BatchNormalization(),
        Dense(128, activation='relu'),
        BatchNormalization(),
        Dense(64, activation='relu'),
        BatchNormalization(),
        Dense(1, activation='relu'),
    ])

    return model

def create_adversary():
    adv = Sequential([
```

```python
        Dense(128, input_dim=1, activation='relu'),
        BatchNormalization(),
        Dense(64, activation='relu'),
        BatchNormalization(),
        Dense(1, activation='relu'),
    ])
    return adv
```

```python
# Main
loss_fcn_main = BinaryCrossentropy(from_logits=logits)

# Adversary
loss_fcn_sex = BinaryCrossentropy(from_logits=logits)
loss_fcn_age = BinaryCrossentropy(from_logits=logits)
loss_fcn_cit = BinaryCrossentropy(from_logits=logits)
loss_fcn_pro = BinaryCrossentropy(from_logits=logits)

loss_adv = [loss_fcn_sex, loss_fcn_age, loss_fcn_cit, loss_fcn_pro]

# Optimizers
optimizerm = Adam(learning_rate=0.001)
optimizers = Adam(learning_rate=0.0005)
optimizera = Adam(learning_rate=0.0005)
optimizerc = Adam(learning_rate=0.0005)
optimizerp = Adam(learning_rate=0.0005)
```

```python
# Redefine nn100 and the adversarial networks
neural_models = []

# Create and compile 7 models with different seeds
for seed in range(85, 92):
    np.random.seed(seed)
    tf.random.set_seed(seed)

    # Create neural network model
    model = create_Main()
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'
    neural_models.append(model)
```

```python
adversary_models = []

# Adversary for sex prediction
adversary_sex = create_adversary()
adversary_sex.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accur
adversary_models.append(adversary_sex)

# Adversary for age prediction
adversary_age = create_adversary()
adversary_age.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accur
adversary_models.append(adversary_age)

# Adversary for citizenship prediction
adversary_citizenship = create_adversary()
adversary_citizenship.compile(optimizer='adam', loss='binary_crossentropy', metrics
adversary_models.append(adversary_citizenship)
```

```
# Adversary for profession prediction
adversary_protected = create_adversary()
adversary_protected.compile(optimizer='adam', loss='binary_crossentropy', metrics=[
adversary_models.append(adversary_protected)
```

```
def build_optimizers(neural_models, adversary_models):
    main_vars = [var for model in neural_models for var in model.trainable_variable

    adv_vars = []
    for model in adversary_models:
        adv_vars.append([var for var in model.trainable_variables])

    optimizerm.build(main_vars)
    optimizers.build(adv_vars[0])
    optimizera.build(adv_vars[1])
    optimizerc.build(adv_vars[2])
    optimizerp.build(adv_vars[3])

    optimizer_list = [optimizers, optimizera, optimizerc, optimizerp]
    return optimizer_list

# Build optimizers
optimizer_list = build_optimizers(neural_models, adversary_models)
```

```
def build_bacthed_data(X, y, sensitive_features, batch_size, names):

    Xb = []
    yb = []
    X_adv_b = []
    sens_f = []

    for i in range(0, len(X), batch_size):
        Xb.append(X[i:i+batch_size])
        yb.append(y[i:i+batch_size])

    for i in range(len(sensitive_features)):

        temp1 = []
        temp2 = []
        X_adv = X.drop(names[i], axis=1)
        sensitive_feature = sensitive_features[i]

        for j in range(0, len(X), batch_size):
            temp1.append(X_adv[j:j+batch_size])
            temp2.append(sensitive_feature[j:j+batch_size])

        X_adv_b.append(temp1)
        sens_f.append(temp2)

    return Xb, yb, sens_f, X_adv_b

Xb, yb, sens_f, X_adv_b = build_bacthed_data(X_train, y_train, sensitive_f, batch_s
```

```
def train_nn(main, adv, X, y, sens_f, X_adv, epochs, adv_n, print_step=10):

    for epoch in range(epochs):
```

```
        print()
        for i in range(len(X)):

            X_batch = tf.convert_to_tensor(X[i].values, dtype=tf.float32)
            y_batch = tf.convert_to_tensor(y[i].values, dtype=tf.float32)
            sens = tf.convert_to_tensor(sens_f[adv_n][i], dtype=tf.float32)

            with tf.GradientTape(persistent=True) as tape:

                # Main loss
                main_preds = main(X_batch, training=True)
                loss_main = loss_fcn_main(y_batch, main_preds)

                # Adversary loss
                adv_preds = adv[adv_n](main_preds, training=True)
                loss_adversary = loss_adv[adv_n](sens, adv_preds)

                # Combined loss
                combined_loss = tf.subtract(loss_main, tf.multiply(0.25, loss_adver


            # Update main weights
            gradients_main = tape.gradient(combined_loss, main.trainable_variables)
            filtered_gradients_main_and_vars = [(grad, var) for grad, var in zip(gr
            filtered_gradients_main, filtered_vars_main = zip(*filtered_gradients_m
            optimizerm.apply_gradients(zip(filtered_gradients_main, filtered_vars_m

            # Update adversary weights
            model = adv[adv_n]

            optimizer = optimizer_list[adv_n]
            gradients_adv = tape.gradient(loss_adversary, model.trainable_variables
            filtered_gradients_adv_and_vars = [(grad, var) for grad, var in zip(gra
            filtered_gradients_adv, filtered_vars_adv = zip(*filtered_gradients_adv
            optimizer.apply_gradients(zip(filtered_gradients_adv, filtered_vars_adv

            del tape

            if i % print_step == 0:
                print(f"Epoch {epoch+1} Batch {i}/{len(X)}:")
                print(f"  Main Loss: {loss_main.numpy()}, Adversary Loss: {loss_adv
```

```
weight_sex = adversary_models[0].get_weights()
weight_age = adversary_models[1].get_weights()
weight_cit = adversary_models[2].get_weights()
weight_pro = adversary_models[3].get_weights()

for i,model in enumerate(neural_models):
  print()
  print("=================")
  print(f"Neural Network {i}")
  train_nn(model, adversary_models, Xb, yb, sens_f, X_adv_b, epochs, adv_n=0, print

  adversary_models[0].set_weights(weight_sex)
  adversary_models[1].set_weights(weight_age)
  adversary_models[2].set_weights(weight_cit)
  adversary_models[3].set_weights(weight_pro)
```

```
==================
Neural Network 0

Epoch 1 Batch 0/124:
  Main Loss: 0.7892932891845703, Adversary Loss: 0.6106728315353394, Combined Los
s: 0.6366250514984131
Epoch 1 Batch 60/124:
  Main Loss: 0.5489599704742432, Adversary Loss: 0.7076425552368164, Combined Los
s: 0.37204933166503906
Epoch 1 Batch 120/124:
  Main Loss: 0.5441529154777527, Adversary Loss: 0.6844561696052551, Combined Los
s: 0.3730388879776001

Epoch 2 Batch 0/124:
  Main Loss: 0.532342791557312, Adversary Loss: 0.65700364112854, Combined Loss:
0.368091881275177
Epoch 2 Batch 60/124:
  Main Loss: 0.5215277671813965, Adversary Loss: 0.7188718318939209, Combined Los
s: 0.34180980920791626
Epoch 2 Batch 120/124:
  Main Loss: 0.5238474011421204, Adversary Loss: 0.6713525056838989, Combined Los
s: 0.35600927472114563

Epoch 3 Batch 0/124:
  Main Loss: 0.5373138785362244, Adversary Loss: 0.659650444984436, Combined Loss:
0.37240126729011536
Epoch 3 Batch 60/124:
  Main Loss: 0.5145737528800964, Adversary Loss: 0.7213497161865234, Combined Los
s: 0.3342363238334656
Epoch 3 Batch 120/124:
  Main Loss: 0.5167654156684875, Adversary Loss: 0.6668011546134949, Combined Los
s: 0.35006511211395264

Epoch 4 Batch 0/124:
  Main Loss: 0.5325912237167358, Adversary Loss: 0.6532200574874878, Combined Los
s: 0.3692862093448639
Epoch 4 Batch 60/124:
  Main Loss: 0.5163037776947021, Adversary Loss: 0.7144297361373901, Combined Los
s: 0.3376963436603546
Epoch 4 Batch 120/124:
  Main Loss: 0.5027905702590942, Adversary Loss: 0.6640403270721436, Combined Los
s: 0.33678048849105835

Epoch 5 Batch 0/124:
  Main Loss: 0.5235905647277832, Adversary Loss: 0.6534041166305542, Combined Los
s: 0.36023953557014465
Epoch 5 Batch 60/124:
  Main Loss: 0.5059087872505188, Adversary Loss: 0.7177332043647766, Combined Los
s: 0.32647550106048584
Epoch 5 Batch 120/124:
  Main Loss: 0.4995185434818268, Adversary Loss: 0.664333701133728, Combined Loss:
0.3334351181983948

Epoch 6 Batch 0/124:
  Main Loss: 0.5244638323783875, Adversary Loss: 0.6539170742034912, Combined Los
s: 0.36098456382751465
Epoch 6 Batch 60/124:
```

```
  Main Loss: 0.5004552602767944, Adversary Loss: 0.7252731323242188, Combined Los
s: 0.31913697719573975
Epoch 6 Batch 120/124:
  Main Loss: 0.4985392093658447, Adversary Loss: 0.6670742034912109, Combined Los
s: 0.331770658493042

Epoch 7 Batch 0/124:
  Main Loss: 0.5190439224243164, Adversary Loss: 0.6530100107192993, Combined Los
s: 0.3557914197444916
Epoch 7 Batch 60/124:
  Main Loss: 0.4890604019165039, Adversary Loss: 0.7180585861206055, Combined Los
s: 0.30954575538635254
Epoch 7 Batch 120/124:
  Main Loss: 0.4985111355781555, Adversary Loss: 0.6351114511489868, Combined Los
s: 0.3397332727909088

Epoch 8 Batch 0/124:
  Main Loss: 0.520416259765625, Adversary Loss: 0.5947998762130737, Combined Loss:
0.37171629071235657
Epoch 8 Batch 60/124:
  Main Loss: 0.48990726470947266, Adversary Loss: 0.48158586025238037, Combined Lo
ss: 0.36951079964637756
Epoch 8 Batch 120/124:
  Main Loss: 0.49852219223976135, Adversary Loss: 0.5845062136650085, Combined Los
s: 0.3523956537246704

Epoch 9 Batch 0/124:
  Main Loss: 0.5211363434791565, Adversary Loss: 0.5100839138031006, Combined Los
s: 0.39361536502838135
Epoch 9 Batch 60/124:
  Main Loss: 0.48877787590026855, Adversary Loss: 0.48549118638038635, Combined Lo
ss: 0.36740508675575256
Epoch 9 Batch 120/124:
  Main Loss: 0.49831801652908325, Adversary Loss: 0.5857283473014832, Combined Los
s: 0.35188591480255127

Epoch 10 Batch 0/124:
  Main Loss: 0.5208358764648438, Adversary Loss: 0.5090606212615967, Combined Los
s: 0.3935707211494446
Epoch 10 Batch 60/124:
  Main Loss: 0.4969073534011841, Adversary Loss: 0.5035628080368042, Combined Los
s: 0.37101665139198303
Epoch 10 Batch 120/124:
  Main Loss: 0.521334707736969, Adversary Loss: 0.5590510964393616, Combined Loss:
0.3815719485282898

Epoch 11 Batch 0/124:
  Main Loss: 0.5207241773605347, Adversary Loss: 0.5055363178253174, Combined Los
s: 0.3943400979042053
Epoch 11 Batch 60/124:
  Main Loss: 0.4877723753452301, Adversary Loss: 0.4979054927825928, Combined Los
s: 0.3632960021495819
Epoch 11 Batch 120/124:
  Main Loss: 0.49874886870384216, Adversary Loss: 0.563538670539856, Combined Los
s: 0.3578642010688782

Epoch 12 Batch 0/124:
```

```
  Main Loss: 0.5203793048858643, Adversary Loss: 0.5058598518371582, Combined Los
s: 0.3939143419265747
Epoch 12 Batch 60/124:
  Main Loss: 0.489992618560791, Adversary Loss: 0.48598700761795044, Combined Los
s: 0.3684958815574646
Epoch 12 Batch 120/124:
  Main Loss: 0.49857860803604126, Adversary Loss: 0.5764341950416565, Combined Los
s: 0.35447007417678833

Epoch 13 Batch 0/124:
  Main Loss: 0.5210137963294983, Adversary Loss: 0.5043648481369019, Combined Los
s: 0.3949225842952728
Epoch 13 Batch 60/124:
  Main Loss: 0.48917996883392334, Adversary Loss: 0.5046913623809814, Combined Los
s: 0.363007128238678
Epoch 13 Batch 120/124:
  Main Loss: 0.49839311838150024, Adversary Loss: 0.5772051811218262, Combined Los
s: 0.3540918231010437

Epoch 14 Batch 0/124:
  Main Loss: 0.5203090906143188, Adversary Loss: 0.5070362687110901, Combined Los
s: 0.3935500383377075
Epoch 14 Batch 60/124:
  Main Loss: 0.4887163043022156, Adversary Loss: 0.49508610367774963, Combined Los
s: 0.36494478583335876
Epoch 14 Batch 120/124:
  Main Loss: 0.4989374876022339, Adversary Loss: 0.5851588249206543, Combined Los
s: 0.3526477813720703

Epoch 15 Batch 0/124:
  Main Loss: 0.5206577181816101, Adversary Loss: 0.5119969844818115, Combined Los
s: 0.3926584720611572
Epoch 15 Batch 60/124:
  Main Loss: 0.49062514305114746, Adversary Loss: 0.4838128983974457, Combined Los
s: 0.36967191100120544
Epoch 15 Batch 120/124:
  Main Loss: 0.49829599261283875, Adversary Loss: 0.5835626721382141, Combined Los
s: 0.352405309677124

==================
Neural Network 1

Epoch 1 Batch 0/124:
  Main Loss: 0.8465350270271301, Adversary Loss: 0.6305952072143555, Combined Los
s: 0.6888862252235413
Epoch 1 Batch 60/124:
  Main Loss: 0.5320091247558594, Adversary Loss: 0.4779662787914276, Combined Los
s: 0.4125175476074219
Epoch 1 Batch 120/124:
  Main Loss: 0.5216729044914246, Adversary Loss: 0.5679267644882202, Combined Los
s: 0.3796912133693695

Epoch 2 Batch 0/124:
  Main Loss: 0.551474928855896, Adversary Loss: 0.48740777373313904, Combined Los
s: 0.42962297797203064
Epoch 2 Batch 60/124:
  Main Loss: 0.511960506439209, Adversary Loss: 0.5018953680992126, Combined Loss:
```

```
0.38648664951324463
Epoch 2 Batch 120/124:
  Main Loss: 0.5201762914657593, Adversary Loss: 0.569921612739563, Combined Loss:
0.37769588828086853

Epoch 3 Batch 0/124:
  Main Loss: 0.5330288410186768, Adversary Loss: 0.5020105838775635, Combined Los
s: 0.4075261950492859
Epoch 3 Batch 60/124:
  Main Loss: 0.4989530146121979, Adversary Loss: 0.5014744997024536, Combined Los
s: 0.3735843896865845
Epoch 3 Batch 120/124:
  Main Loss: 0.5161185264587402, Adversary Loss: 0.5658630728721619, Combined Los
s: 0.3746527433395386

Epoch 4 Batch 0/124:
  Main Loss: 0.5330462455749512, Adversary Loss: 0.49861589074134827, Combined Los
s: 0.4083922803401947
Epoch 4 Batch 60/124:
  Main Loss: 0.49848148226737976, Adversary Loss: 0.501468300819397, Combined Los
s: 0.3731144070625305
Epoch 4 Batch 120/124:
  Main Loss: 0.503411054611206, Adversary Loss: 0.5669779181480408, Combined Loss:
0.36166656017303467

Epoch 5 Batch 0/124:
  Main Loss: 0.5448095798492432, Adversary Loss: 0.49527984857559204, Combined Los
s: 0.42098963260650635
Epoch 5 Batch 60/124:
  Main Loss: 0.49861282110214233, Adversary Loss: 0.5015443563461304, Combined Los
s: 0.37322673201560974
Epoch 5 Batch 120/124:
  Main Loss: 0.4985429644584656, Adversary Loss: 0.5830523371696472, Combined Los
s: 0.3527798652648926

Epoch 6 Batch 0/124:
  Main Loss: 0.5213834643363953, Adversary Loss: 0.5104068517684937, Combined Los
s: 0.39378175139427185
Epoch 6 Batch 60/124:
  Main Loss: 0.49847832322120667, Adversary Loss: 0.5016665458679199, Combined Los
s: 0.3730616867542267
Epoch 6 Batch 120/124:
  Main Loss: 0.4989999830722809, Adversary Loss: 0.5849436521530151, Combined Los
s: 0.3527640700340271

Epoch 7 Batch 0/124:
  Main Loss: 0.5200464725494385, Adversary Loss: 0.512787938117981, Combined Loss:
0.39184948801994324
Epoch 7 Batch 60/124:
  Main Loss: 0.49843621253967285, Adversary Loss: 0.5016899108886719, Combined Los
s: 0.3730137348175049
Epoch 7 Batch 120/124:
  Main Loss: 0.4984705150127411, Adversary Loss: 0.5844346284866333, Combined Los
s: 0.35236185789108276

Epoch 8 Batch 0/124:
  Main Loss: 0.5219999551773071, Adversary Loss: 0.5064467191696167, Combined Los
```

```
s: 0.39538827538490295
Epoch 8 Batch 60/124:
  Main Loss: 0.4915373623371124, Adversary Loss: 0.49695807695388794, Combined Los
s: 0.36729782819747925
Epoch 8 Batch 120/124:
  Main Loss: 0.4984217584133148, Adversary Loss: 0.5846089124679565, Combined Los
s: 0.3522695302963257

Epoch 9 Batch 0/124:
  Main Loss: 0.5208238363265991, Adversary Loss: 0.5087870955467224, Combined Los
s: 0.3936270475387573
Epoch 9 Batch 60/124:
  Main Loss: 0.493825227022171, Adversary Loss: 0.49536675214767456, Combined Los
s: 0.3699835538864136
Epoch 9 Batch 120/124:
  Main Loss: 0.4983580410480499, Adversary Loss: 0.5838930606842041, Combined Los
s: 0.3523847758769989

Epoch 10 Batch 0/124:
  Main Loss: 0.5203794240951538, Adversary Loss: 0.5116897821426392, Combined Los
s: 0.392456978559494
Epoch 10 Batch 60/124:
  Main Loss: 0.48800134658813477, Adversary Loss: 0.48429930210113525, Combined Lo
ss: 0.36692652106285095
Epoch 10 Batch 120/124:
  Main Loss: 0.5001207590103149, Adversary Loss: 0.5856074690818787, Combined Los
s: 0.3537188768386841

Epoch 11 Batch 0/124:
  Main Loss: 0.525795578956604, Adversary Loss: 0.5050066709518433, Combined Loss:
0.3995439112186432
Epoch 11 Batch 60/124:
  Main Loss: 0.49955910444259644, Adversary Loss: 0.5017838478088379, Combined Los
s: 0.37411314249038696
Epoch 11 Batch 120/124:
  Main Loss: 0.4986775517463684, Adversary Loss: 0.584649920463562, Combined Loss:
0.3525150716304779

Epoch 12 Batch 0/124:
  Main Loss: 0.5263721942901611, Adversary Loss: 0.5045329332351685, Combined Los
s: 0.400238960981369
Epoch 12 Batch 60/124:
  Main Loss: 0.4979877769947052, Adversary Loss: 0.5011205077171326, Combined Los
s: 0.37270766496658325
Epoch 12 Batch 120/124:
  Main Loss: 0.49833571910858154, Adversary Loss: 0.5840721130371094, Combined Los
s: 0.3523176908493042

Epoch 13 Batch 0/124:
  Main Loss: 0.5199722647666931, Adversary Loss: 0.5118722915649414, Combined Los
s: 0.39200419187545776
Epoch 13 Batch 60/124:
  Main Loss: 0.4874945878982544, Adversary Loss: 0.48292872309684753, Combined Los
s: 0.3667623996734619
Epoch 13 Batch 120/124:
  Main Loss: 0.4985561668872833, Adversary Loss: 0.5819171667098999, Combined Los
s: 0.35307687520980835
```

```
Epoch 14 Batch 0/124:
  Main Loss: 0.523624062538147, Adversary Loss: 0.5058115124702454, Combined Loss:
0.3971711993217468
Epoch 14 Batch 60/124:
  Main Loss: 0.48779967427253723, Adversary Loss: 0.48753976821899414, Combined Lo
ss: 0.3659147322177887
Epoch 14 Batch 120/124:
  Main Loss: 0.49831104278564453, Adversary Loss: 0.5828181505203247, Combined Los
s: 0.35260650515556335

Epoch 15 Batch 0/124:
  Main Loss: 0.520028829574585, Adversary Loss: 0.511690616607666, Combined Loss:
0.39210617542266846
Epoch 15 Batch 60/124:
  Main Loss: 0.4878919720649719, Adversary Loss: 0.4939717948436737, Combined Los
s: 0.3643990159034729
Epoch 15 Batch 120/124:
  Main Loss: 0.49831080436706543, Adversary Loss: 0.582082211971283, Combined Los
s: 0.3527902364730835

==================
Neural Network 2

Epoch 1 Batch 0/124:
  Main Loss: 0.8157639503479004, Adversary Loss: 0.6388908624649048, Combined Los
s: 0.6560412645339966
Epoch 1 Batch 60/124:
  Main Loss: 0.5654194355010986, Adversary Loss: 0.5043110847473145, Combined Los
s: 0.43934166431427
Epoch 1 Batch 120/124:
  Main Loss: 0.5371177196502686, Adversary Loss: 0.5472310781478882, Combined Los
s: 0.4003099501132965

Epoch 2 Batch 0/124:
  Main Loss: 0.5434926748275757, Adversary Loss: 0.49512094259262085, Combined Los
s: 0.4197124242782593
Epoch 2 Batch 60/124:
  Main Loss: 0.5531424283981323, Adversary Loss: 0.5124786496162415, Combined Los
s: 0.42502278089523315
Epoch 2 Batch 120/124:
  Main Loss: 0.5309850573539734, Adversary Loss: 0.5581384897232056, Combined Los
s: 0.391450434923172

Epoch 3 Batch 0/124:
  Main Loss: 0.5526469945907593, Adversary Loss: 0.4870122969150543, Combined Los
s: 0.4308939278125763
Epoch 3 Batch 60/124:
  Main Loss: 0.5439401865005493, Adversary Loss: 0.5203777551651001, Combined Los
s: 0.4138457477092743
Epoch 3 Batch 120/124:
  Main Loss: 0.5201330184936523, Adversary Loss: 0.5659371018409729, Combined Los
s: 0.3786487579345703

Epoch 4 Batch 0/124:
  Main Loss: 0.5480272769927979, Adversary Loss: 0.48712027072906494, Combined Los
s: 0.4262472093105316
```

```
Epoch 4 Batch 60/124:
  Main Loss: 0.5439159870147705, Adversary Loss: 0.5202325582504272, Combined Los
s: 0.4138578474521637
Epoch 4 Batch 120/124:
  Main Loss: 0.5308094024658203, Adversary Loss: 0.5638493895530701, Combined Los
s: 0.3898470401763916

Epoch 5 Batch 0/124:
  Main Loss: 0.5596630573272705, Adversary Loss: 0.4903164505958557, Combined Los
s: 0.4370839595794678
Epoch 5 Batch 60/124:
  Main Loss: 0.5417930483818054, Adversary Loss: 0.5200455188751221, Combined Los
s: 0.4117816686630249
Epoch 5 Batch 120/124:
  Main Loss: 0.5209841728210449, Adversary Loss: 0.5676910877227783, Combined Los
s: 0.37906140089035034

Epoch 6 Batch 0/124:
  Main Loss: 0.5359184741973877, Adversary Loss: 0.4955349564552307, Combined Los
s: 0.4120347499847412
Epoch 6 Batch 60/124:
  Main Loss: 0.5417696237564087, Adversary Loss: 0.5200246572494507, Combined Los
s: 0.411763459444046
Epoch 6 Batch 120/124:
  Main Loss: 0.5201817750930786, Adversary Loss: 0.5666821002960205, Combined Los
s: 0.3785112500190735

Epoch 7 Batch 0/124:
  Main Loss: 0.5232042074203491, Adversary Loss: 0.5045800805091858, Combined Los
s: 0.39705920219421387
Epoch 7 Batch 60/124:
  Main Loss: 0.5114164352416992, Adversary Loss: 0.5041005611419678, Combined Los
s: 0.3853912949562073
Epoch 7 Batch 120/124:
  Main Loss: 0.5061758756637573, Adversary Loss: 0.5894296169281006, Combined Los
s: 0.3588184714317322

Epoch 8 Batch 0/124:
  Main Loss: 0.5212750434875488, Adversary Loss: 0.5151575207710266, Combined Los
s: 0.39248567819595337
Epoch 8 Batch 60/124:
  Main Loss: 0.49106770753860474, Adversary Loss: 0.47777214646339417, Combined Lo
ss: 0.3716246783733368
Epoch 8 Batch 120/124:
  Main Loss: 0.49906036257743835, Adversary Loss: 0.5947654843330383, Combined Los
s: 0.3503689765930176

Epoch 9 Batch 0/124:
  Main Loss: 0.5214617848396301, Adversary Loss: 0.5140476822853088, Combined Los
s: 0.3929498791694641
Epoch 9 Batch 60/124:
  Main Loss: 0.48754313588142395, Adversary Loss: 0.4822971224784851, Combined Los
s: 0.36696887016296387
Epoch 9 Batch 120/124:
  Main Loss: 0.49831509590148926, Adversary Loss: 0.5924009084701538, Combined Los
s: 0.3502148687839508
```

```
Epoch 10 Batch 0/124:
  Main Loss: 0.520182728767395, Adversary Loss: 0.5120487213134766, Combined Loss:
0.3921705484390259
Epoch 10 Batch 60/124:
  Main Loss: 0.4886353015899658, Adversary Loss: 0.48331040143966675, Combined Los
s: 0.36780768632888794
Epoch 10 Batch 120/124:
  Main Loss: 0.5014442205429077, Adversary Loss: 0.5838059186935425, Combined Los
s: 0.3554927408695221

Epoch 11 Batch 0/124:
  Main Loss: 0.5213557481765747, Adversary Loss: 0.5125557780265808, Combined Los
s: 0.3932167887687683
Epoch 11 Batch 60/124:
  Main Loss: 0.48770931363105774, Adversary Loss: 0.4828784167766571, Combined Los
s: 0.36698970198631287
Epoch 11 Batch 120/124:
  Main Loss: 0.4985087811946869, Adversary Loss: 0.5933846235275269, Combined Los
s: 0.3501626253128052

Epoch 12 Batch 0/124:
  Main Loss: 0.5199600458145142, Adversary Loss: 0.5138083100318909, Combined Los
s: 0.39150798320770264
Epoch 12 Batch 60/124:
  Main Loss: 0.48756104707717896, Adversary Loss: 0.4830215275287628, Combined Los
s: 0.36680567264556885
Epoch 12 Batch 120/124:
  Main Loss: 0.49837517738342285, Adversary Loss: 0.5878919959068298, Combined Los
s: 0.3514021635055542

Epoch 13 Batch 0/124:
  Main Loss: 0.519919216632843, Adversary Loss: 0.5123509168624878, Combined Loss:
0.39183148741722107
Epoch 13 Batch 60/124:
  Main Loss: 0.48777827620506287, Adversary Loss: 0.48349449038505554, Combined Lo
ss: 0.3669046461582184
Epoch 13 Batch 120/124:
  Main Loss: 0.4983336925506592, Adversary Loss: 0.5922883749008179, Combined Los
s: 0.3502615988254547

Epoch 14 Batch 0/124:
  Main Loss: 0.5199014544487, Adversary Loss: 0.5131416916847229, Combined Loss:
0.3916160464286804
Epoch 14 Batch 60/124:
  Main Loss: 0.48765963315963745, Adversary Loss: 0.4839085340499878, Combined Los
s: 0.3666824996471405
Epoch 14 Batch 120/124:
  Main Loss: 0.4983653426170349, Adversary Loss: 0.5875522494316101, Combined Los
s: 0.3514772653579712

Epoch 15 Batch 0/124:
  Main Loss: 0.5198978185653687, Adversary Loss: 0.5121709108352661, Combined Los
s: 0.3918550908565521
Epoch 15 Batch 60/124:
  Main Loss: 0.48761263489723206, Adversary Loss: 0.48368775844573975, Combined Lo
ss: 0.3666906952857971
Epoch 15 Batch 120/124:
```

```
  Main Loss: 0.49849292635917664, Adversary Loss: 0.5864354372024536, Combined Los
s: 0.35188406705856323

==================
Neural Network 3

Epoch 1 Batch 0/124:
  Main Loss: 0.7943298816680908, Adversary Loss: 0.6081408262252808, Combined Los
s: 0.6422946453094482
Epoch 1 Batch 60/124:
  Main Loss: 0.5374588370323181, Adversary Loss: 0.5032445192337036, Combined Los
s: 0.4116477072238922
Epoch 1 Batch 120/124:
  Main Loss: 0.5142649412155151, Adversary Loss: 0.5738908052444458, Combined Los
s: 0.3707922399044037

Epoch 2 Batch 0/124:
  Main Loss: 0.5478485822677612, Adversary Loss: 0.4917649030685425, Combined Los
s: 0.4249073565006256
Epoch 2 Batch 60/124:
  Main Loss: 0.5053122043609619, Adversary Loss: 0.4990621507167816, Combined Los
s: 0.3805466592311859
Epoch 2 Batch 120/124:
  Main Loss: 0.49981486797332764, Adversary Loss: 0.5864666700363159, Combined Los
s: 0.35319820046424866

Epoch 3 Batch 0/124:
  Main Loss: 0.5428284406661987, Adversary Loss: 0.48791804909706116, Combined Los
s: 0.42084893584251404
Epoch 3 Batch 60/124:
  Main Loss: 0.5112817883491516, Adversary Loss: 0.4875548481941223, Combined Los
s: 0.3893930912017822
Epoch 3 Batch 120/124:
  Main Loss: 0.5030204057693481, Adversary Loss: 0.5795896053314209, Combined Los
s: 0.3581230044364929

Epoch 4 Batch 0/124:
  Main Loss: 0.523146390914917, Adversary Loss: 0.5045964121818542, Combined Loss:
0.39699727296829224
Epoch 4 Batch 60/124:
  Main Loss: 0.49940940737724304, Adversary Loss: 0.5015337467193604, Combined Los
s: 0.37402597069740295
Epoch 4 Batch 120/124:
  Main Loss: 0.4985153377056122, Adversary Loss: 0.5886589288711548, Combined Los
s: 0.3513506054878235

Epoch 5 Batch 0/124:
  Main Loss: 0.521225094795227, Adversary Loss: 0.5099194049835205, Combined Loss:
0.3937452435493469
Epoch 5 Batch 60/124:
  Main Loss: 0.49937373399734497, Adversary Loss: 0.5016137957572937, Combined Los
s: 0.37397027015686035
Epoch 5 Batch 120/124:
  Main Loss: 0.5055267810821533, Adversary Loss: 0.5777336955070496, Combined Los
s: 0.36109334230422974

Epoch 6 Batch 0/124:
```

```
  Main Loss: 0.5309302806854248, Adversary Loss: 0.5037379264831543, Combined Los
s: 0.40499579906463623
Epoch 6 Batch 60/124:
  Main Loss: 0.49682608246803284, Adversary Loss: 0.49352526664733887, Combined Lo
ss: 0.3734447658061981
Epoch 6 Batch 120/124:
  Main Loss: 0.4984877407550812, Adversary Loss: 0.5903316140174866, Combined Los
s: 0.35090482234954834

Epoch 7 Batch 0/124:
  Main Loss: 0.5221253633499146, Adversary Loss: 0.5083563923835754, Combined Los
s: 0.3950362801551819
Epoch 7 Batch 60/124:
  Main Loss: 0.48882347345352173, Adversary Loss: 0.48609811067581177, Combined Lo
ss: 0.36729896068573
Epoch 7 Batch 120/124:
  Main Loss: 0.5388333201408386, Adversary Loss: 0.5968573093414307, Combined Los
s: 0.38961899280548096

Epoch 8 Batch 0/124:
  Main Loss: 0.5212860703468323, Adversary Loss: 0.5105931162834167, Combined Los
s: 0.3936377763748169
Epoch 8 Batch 60/124:
  Main Loss: 0.48990076780319214, Adversary Loss: 0.4890381693840027, Combined Los
s: 0.3676412105560303
Epoch 8 Batch 120/124:
  Main Loss: 0.498432457447052, Adversary Loss: 0.5889585018157959, Combined Loss:
0.351192831993103

Epoch 9 Batch 0/124:
  Main Loss: 0.5209740996360779, Adversary Loss: 0.5114102959632874, Combined Los
s: 0.39312154054641724
Epoch 9 Batch 60/124:
  Main Loss: 0.48806196451187134, Adversary Loss: 0.4848347306251526, Combined Los
s: 0.3668532967567444
Epoch 9 Batch 120/124:
  Main Loss: 0.49831411242485046, Adversary Loss: 0.5881702303886414, Combined Los
s: 0.3512715697288513

Epoch 10 Batch 0/124:
  Main Loss: 0.5202499032020569, Adversary Loss: 0.5113940238952637, Combined Los
s: 0.39240139722824097
Epoch 10 Batch 60/124:
  Main Loss: 0.48783501982688904, Adversary Loss: 0.4839111566543579, Combined Los
s: 0.36685723066329956
Epoch 10 Batch 120/124:
  Main Loss: 0.4982934594154358, Adversary Loss: 0.5909801721572876, Combined Los
s: 0.3505484163761139

Epoch 11 Batch 0/124:
  Main Loss: 0.5200866460800171, Adversary Loss: 0.5117824673652649, Combined Los
s: 0.39214104413986206
Epoch 11 Batch 60/124:
  Main Loss: 0.4877745509147644, Adversary Loss: 0.4829798936843872, Combined Los
s: 0.3670295774936676
Epoch 11 Batch 120/124:
  Main Loss: 0.4984397292137146, Adversary Loss: 0.5859105587005615, Combined Los
```

```
s: 0.3519620895385742

Epoch 12 Batch 0/124:
  Main Loss: 0.5206410884857178, Adversary Loss: 0.5095914602279663, Combined Los
s: 0.3932432234287262
Epoch 12 Batch 60/124:
  Main Loss: 0.48830801248550415, Adversary Loss: 0.48954668641090393, Combined Lo
ss: 0.36592134833335876
Epoch 12 Batch 120/124:
  Main Loss: 0.4983235001564026, Adversary Loss: 0.5794976949691772, Combined Los
s: 0.3534490764141083

Epoch 13 Batch 0/124:
  Main Loss: 0.5205814838409424, Adversary Loss: 0.5064847469329834, Combined Los
s: 0.39396029710769653
Epoch 13 Batch 60/124:
  Main Loss: 0.4878460168838501, Adversary Loss: 0.48873937129974365, Combined Los
s: 0.3656611740589142
Epoch 13 Batch 120/124:
  Main Loss: 0.4983573853969574, Adversary Loss: 0.5841236710548401, Combined Los
s: 0.3523264527320862

Epoch 14 Batch 0/124:
  Main Loss: 0.5206040143966675, Adversary Loss: 0.5112504959106445, Combined Los
s: 0.39279139041900635
Epoch 14 Batch 60/124:
  Main Loss: 0.4989173114299774, Adversary Loss: 0.5001552700996399, Combined Los
s: 0.37387847900390625
Epoch 14 Batch 120/124:
  Main Loss: 0.4983938932418823, Adversary Loss: 0.585250735282898, Combined Loss:
0.35208120942115784

Epoch 15 Batch 0/124:
  Main Loss: 0.5209237337112427, Adversary Loss: 0.5108121633529663, Combined Los
s: 0.3932206928730011
Epoch 15 Batch 60/124:
  Main Loss: 0.4884575307369232, Adversary Loss: 0.4890903830528259, Combined Los
s: 0.36618494987487793
Epoch 15 Batch 120/124:
  Main Loss: 0.49840247631073, Adversary Loss: 0.588768720626831, Combined Loss:
0.3512102961540222

==================
Neural Network 4

Epoch 1 Batch 0/124:
  Main Loss: 0.7110051512718201, Adversary Loss: 0.6377692818641663, Combined Los
s: 0.5515628457069397
Epoch 1 Batch 60/124:
  Main Loss: 0.5342968702316284, Adversary Loss: 0.4979683458805084, Combined Los
s: 0.4098047912120819
Epoch 1 Batch 120/124:
  Main Loss: 0.5234884023666382, Adversary Loss: 0.563098669052124, Combined Loss:
0.3827137351036072

Epoch 2 Batch 0/124:
  Main Loss: 0.5358508825302124, Adversary Loss: 0.4979112148284912, Combined Los
```

```
s: 0.4113730788230896
Epoch 2 Batch 60/124:
  Main Loss: 0.5131560564041138, Adversary Loss: 0.47976821660995483, Combined Los
s: 0.39321398735046387
Epoch 2 Batch 120/124:
  Main Loss: 0.5204076170921326, Adversary Loss: 0.5734264254570007, Combined Los
s: 0.3770509958267212

Epoch 3 Batch 0/124:
  Main Loss: 0.5376477241516113, Adversary Loss: 0.49832695722579956, Combined Los
s: 0.41306596994400024
Epoch 3 Batch 60/124:
  Main Loss: 0.5135685205459595, Adversary Loss: 0.48804405331611633, Combined Los
s: 0.391557514667511
Epoch 3 Batch 120/124:
  Main Loss: 0.4994433522224426, Adversary Loss: 0.5855480432510376, Combined Los
s: 0.3530563414096832

Epoch 4 Batch 0/124:
  Main Loss: 0.5376941561698914, Adversary Loss: 0.4955439865589142, Combined Los
s: 0.4138081669807434
Epoch 4 Batch 60/124:
  Main Loss: 0.4884602427482605, Adversary Loss: 0.48384523391723633, Combined Los
s: 0.3674989342689514
Epoch 4 Batch 120/124:
  Main Loss: 0.49848321080207825, Adversary Loss: 0.5881505012512207, Combined Los
s: 0.35144558548927307

Epoch 5 Batch 0/124:
  Main Loss: 0.5308623313903809, Adversary Loss: 0.5035956501960754, Combined Los
s: 0.4049634337425232
Epoch 5 Batch 60/124:
  Main Loss: 0.4915006160736084, Adversary Loss: 0.4768223464488983, Combined Los
s: 0.3722950220108032
Epoch 5 Batch 120/124:
  Main Loss: 0.49840861558914185, Adversary Loss: 0.5876326560974121, Combined Los
s: 0.3515004515647888

Epoch 6 Batch 0/124:
  Main Loss: 0.5308687686920166, Adversary Loss: 0.5035192966461182, Combined Los
s: 0.40498894453048706
Epoch 6 Batch 60/124:
  Main Loss: 0.4895174205303192, Adversary Loss: 0.4923807978630066, Combined Los
s: 0.36642223596572876
Epoch 6 Batch 120/124:
  Main Loss: 0.49861299991607666, Adversary Loss: 0.580716073513031, Combined Los
s: 0.3534339666366577

Epoch 7 Batch 0/124:
  Main Loss: 0.5311378240585327, Adversary Loss: 0.5007044076919556, Combined Los
s: 0.4059617221355438
Epoch 7 Batch 60/124:
  Main Loss: 0.4893200099468231, Adversary Loss: 0.49245578050613403, Combined Los
s: 0.3662060499191284
Epoch 7 Batch 120/124:
  Main Loss: 0.4985809326171875, Adversary Loss: 0.5728976726531982, Combined Los
s: 0.35535651445388794
```

```
Epoch 8 Batch 0/124:
  Main Loss: 0.5321623086929321, Adversary Loss: 0.49710574746131897, Combined Los
s: 0.407885879278183
Epoch 8 Batch 60/124:
  Main Loss: 0.4984668791294098, Adversary Loss: 0.4781387746334076, Combined Los
s: 0.3789321780204773
Epoch 8 Batch 120/124:
  Main Loss: 0.4983653426170349, Adversary Loss: 0.5851179361343384, Combined Los
s: 0.3520858585834503

Epoch 9 Batch 0/124:
  Main Loss: 0.5206122398376465, Adversary Loss: 0.508995532989502, Combined Loss:
0.393363356590271
Epoch 9 Batch 60/124:
  Main Loss: 0.49393460154533386, Adversary Loss: 0.48451554775238037, Combined Lo
ss: 0.37280571460723877
Epoch 9 Batch 120/124:
  Main Loss: 0.5005797147750854, Adversary Loss: 0.5680049657821655, Combined Los
s: 0.35857847332954407

Epoch 10 Batch 0/124:
  Main Loss: 0.5236165523529053, Adversary Loss: 0.49861499667167664, Combined Los
s: 0.3989627957344055
Epoch 10 Batch 60/124:
  Main Loss: 0.4896453022956848, Adversary Loss: 0.5053821802139282, Combined Los
s: 0.36329975724220276
Epoch 10 Batch 120/124:
  Main Loss: 0.49847328662872314, Adversary Loss: 0.5793882608413696, Combined Los
s: 0.35362622141838074

Epoch 11 Batch 0/124:
  Main Loss: 0.5200340747833252, Adversary Loss: 0.5104215145111084, Combined Los
s: 0.3924286961555481
Epoch 11 Batch 60/124:
  Main Loss: 0.48785409331321716, Adversary Loss: 0.5009461045265198, Combined Los
s: 0.362617552280426
Epoch 11 Batch 120/124:
  Main Loss: 0.49829044938087463, Adversary Loss: 0.5833484530448914, Combined Los
s: 0.352453351020813

Epoch 12 Batch 0/124:
  Main Loss: 0.5199164152145386, Adversary Loss: 0.5118733644485474, Combined Los
s: 0.39194807410240173
Epoch 12 Batch 60/124:
  Main Loss: 0.488338828086853, Adversary Loss: 0.49285054206848145, Combined Los
s: 0.36512619256973267
Epoch 12 Batch 120/124:
  Main Loss: 0.49837514758110046, Adversary Loss: 0.587122019195557, Combined Los
s: 0.35159459710121155

Epoch 13 Batch 0/124:
  Main Loss: 0.522891640663147, Adversary Loss: 0.5068186521530151, Combined Loss:
0.3961869776248932
Epoch 13 Batch 60/124:
  Main Loss: 0.49916115403175354, Adversary Loss: 0.5131194591522217, Combined Los
s: 0.3708812892436981
```

```
Epoch 13 Batch 120/124:
  Main Loss: 0.49871665239334106, Adversary Loss: 0.5613816976547241, Combined Los
s: 0.35837122797966003

Epoch 14 Batch 0/124:
  Main Loss: 0.5240228176116943, Adversary Loss: 0.5016283392906189, Combined Los
s: 0.3986157178878784
Epoch 14 Batch 60/124:
  Main Loss: 0.4877896010875702, Adversary Loss: 0.49824637174606323, Combined Los
s: 0.3632280230522156
Epoch 14 Batch 120/124:
  Main Loss: 0.4983746409416199, Adversary Loss: 0.5811566114425659, Combined Los
s: 0.3530854880809784

Epoch 15 Batch 0/124:
  Main Loss: 0.5199885368347168, Adversary Loss: 0.5114409327507019, Combined Los
s: 0.3921282887458801
Epoch 15 Batch 60/124:
  Main Loss: 0.490161657333374, Adversary Loss: 0.49945369362831116, Combined Los
s: 0.36529824137687683
Epoch 15 Batch 120/124:
  Main Loss: 0.49827685952186584, Adversary Loss: 0.5725747346878052, Combined Los
s: 0.35513317584991455

==================
Neural Network 5

Epoch 1 Batch 0/124:
  Main Loss: 0.8593950867652893, Adversary Loss: 0.6277611255645752, Combined Los
s: 0.7024548053741455
Epoch 1 Batch 60/124:
  Main Loss: 0.5533556938171387, Adversary Loss: 0.5132721662521362, Combined Los
s: 0.4250376522541046
Epoch 1 Batch 120/124:
  Main Loss: 0.5384032130241394, Adversary Loss: 0.5483168363571167, Combined Los
s: 0.40132400393486023

Epoch 2 Batch 0/124:
  Main Loss: 0.5352683663368225, Adversary Loss: 0.4993920922279358, Combined Los
s: 0.41042035818099976
Epoch 2 Batch 60/124:
  Main Loss: 0.5115993618965149, Adversary Loss: 0.493307888507843, Combined Loss:
0.38827240467071533
Epoch 2 Batch 120/124:
  Main Loss: 0.5040310025215149, Adversary Loss: 0.5777381062507629, Combined Los
s: 0.35959649085998535

Epoch 3 Batch 0/124:
  Main Loss: 0.5314437747001648, Adversary Loss: 0.5035872459411621, Combined Los
s: 0.40554696321487427
Epoch 3 Batch 60/124:
  Main Loss: 0.4952414035797119, Adversary Loss: 0.502173662185669, Combined Loss:
0.3696979880332947
Epoch 3 Batch 120/124:
  Main Loss: 0.49894359707832336, Adversary Loss: 0.5881319642066956, Combined Los
s: 0.3519105911254883
```

```
Epoch 4 Batch 0/124:
  Main Loss: 0.5319451689720154, Adversary Loss: 0.5035428404808044, Combined Los
s: 0.4060594439506531
Epoch 4 Batch 60/124:
  Main Loss: 0.5058722496032715, Adversary Loss: 0.5173187255859375, Combined Los
s: 0.3765425682067871
Epoch 4 Batch 120/124:
  Main Loss: 0.4992836117744446, Adversary Loss: 0.5877692103385925, Combined Los
s: 0.35234129428863525

Epoch 5 Batch 0/124:
  Main Loss: 0.5307924151420593, Adversary Loss: 0.503642737865448, Combined Loss:
0.40488171577453613
Epoch 5 Batch 60/124:
  Main Loss: 0.49866777658462524, Adversary Loss: 0.501478910446167, Combined Los
s: 0.3732980489730835
Epoch 5 Batch 120/124:
  Main Loss: 0.49837881326675415, Adversary Loss: 0.5877197980880737, Combined Los
s: 0.3514488637447357

Epoch 6 Batch 0/124:
  Main Loss: 0.5203328132629395, Adversary Loss: 0.5119060277938843, Combined Los
s: 0.3923563063144684
Epoch 6 Batch 60/124:
  Main Loss: 0.49848073720932007, Adversary Loss: 0.5014734268188477, Combined Los
s: 0.37311238050460815
Epoch 6 Batch 120/124:
  Main Loss: 0.4989413321018219, Adversary Loss: 0.5913355350494385, Combined Los
s: 0.3511074483394623

Epoch 7 Batch 0/124:
  Main Loss: 0.5321925282478333, Adversary Loss: 0.5042344331741333, Combined Los
s: 0.4061339199542999
Epoch 7 Batch 60/124:
  Main Loss: 0.49832844734191895, Adversary Loss: 0.5014692544937134, Combined Los
s: 0.3729611337184906
Epoch 7 Batch 120/124:
  Main Loss: 0.49833741784095764, Adversary Loss: 0.5877580642700195, Combined Los
s: 0.35139790177345276

Epoch 8 Batch 0/124:
  Main Loss: 0.5241867303848267, Adversary Loss: 0.5066559314727783, Combined Los
s: 0.3975227475166321
Epoch 8 Batch 60/124:
  Main Loss: 0.49840858578681946, Adversary Loss: 0.5014764070510864, Combined Los
s: 0.37303948402404785
Epoch 8 Batch 120/124:
  Main Loss: 0.498323917388916, Adversary Loss: 0.5877901315689087, Combined Loss:
0.35137638449668884

Epoch 9 Batch 0/124:
  Main Loss: 0.5231325626373291, Adversary Loss: 0.5094404816627502, Combined Los
s: 0.39577245712280273
Epoch 9 Batch 60/124:
  Main Loss: 0.4987182319164276, Adversary Loss: 0.5016545057296753, Combined Los
s: 0.3733046054840088
Epoch 9 Batch 120/124:
```

```
  Main Loss: 0.49832382798194885, Adversary Loss: 0.591945469379425, Combined Los
s: 0.3503374457359314

Epoch 10 Batch 0/124:
  Main Loss: 0.5204225778579712, Adversary Loss: 0.5124243497848511, Combined Los
s: 0.3923164904117584
Epoch 10 Batch 60/124:
  Main Loss: 0.5004273653030396, Adversary Loss: 0.5016806125640869, Combined Los
s: 0.3750072121620178
Epoch 10 Batch 120/124:
  Main Loss: 0.4982663691043854, Adversary Loss: 0.5889471769332886, Combined Los
s: 0.35102957487106323

Epoch 11 Batch 0/124:
  Main Loss: 0.5205826163291931, Adversary Loss: 0.5118656158447266, Combined Los
s: 0.3926162123680115
Epoch 11 Batch 60/124:
  Main Loss: 0.5093377828598022, Adversary Loss: 0.4939819276332855, Combined Los
s: 0.38584229350090027
Epoch 11 Batch 120/124:
  Main Loss: 0.4985505938529968, Adversary Loss: 0.5905497074127197, Combined Los
s: 0.3509131669998169

Epoch 12 Batch 0/124:
  Main Loss: 0.5218337774276733, Adversary Loss: 0.5123996734619141, Combined Los
s: 0.3937338590621948
Epoch 12 Batch 60/124:
  Main Loss: 0.49840182065963745, Adversary Loss: 0.5015964508056641, Combined Los
s: 0.37300270795822144
Epoch 12 Batch 120/124:
  Main Loss: 0.49826520681381226, Adversary Loss: 0.593836784362793, Combined Los
s: 0.349806010723114

Epoch 13 Batch 0/124:
  Main Loss: 0.5201817750930786, Adversary Loss: 0.5151880383491516, Combined Los
s: 0.3913847804069519
Epoch 13 Batch 60/124:
  Main Loss: 0.4982762932777405, Adversary Loss: 0.5014777183532715, Combined Los
s: 0.3729068636894226
Epoch 13 Batch 120/124:
  Main Loss: 0.4982377588748932, Adversary Loss: 0.5857377052307129, Combined Los
s: 0.35180333256721497

Epoch 14 Batch 0/124:
  Main Loss: 0.5199256539344788, Adversary Loss: 0.5120159387588501, Combined Los
s: 0.39192166924476624
Epoch 14 Batch 60/124:
  Main Loss: 0.498329222202301, Adversary Loss: 0.5014957189559937, Combined Loss:
0.3729552924633026
Epoch 14 Batch 120/124:
  Main Loss: 0.4982805550098419, Adversary Loss: 0.582744836807251, Combined Loss:
0.3525943458080292

Epoch 15 Batch 0/124:
  Main Loss: 0.5426392555236816, Adversary Loss: 0.5166844725608826, Combined Los
s: 0.4134681224822998
Epoch 15 Batch 60/124:
```

```
  Main Loss: 0.49825096130371094, Adversary Loss: 0.5015551447868347, Combined Los
s: 0.37286216020584106
Epoch 15 Batch 120/124:
  Main Loss: 0.49823302030563354, Adversary Loss: 0.5901561975479126, Combined Los
s: 0.3506939709186554

==================
Neural Network 6

Epoch 1 Batch 0/124:
  Main Loss: 0.7228627800941467, Adversary Loss: 0.6272062063217163, Combined Los
s: 0.56606125831604
Epoch 1 Batch 60/124:
  Main Loss: 0.5504822731018066, Adversary Loss: 0.5174327492713928, Combined Los
s: 0.42112410068511963
Epoch 1 Batch 120/124:
  Main Loss: 0.5418523550033569, Adversary Loss: 0.5459290742874146, Combined Los
s: 0.4053700864315033

Epoch 2 Batch 0/124:
  Main Loss: 0.5428063273429871, Adversary Loss: 0.4951699376106262, Combined Los
s: 0.4190138578414917
Epoch 2 Batch 60/124:
  Main Loss: 0.5040337443351746, Adversary Loss: 0.4952715039253235, Combined Los
s: 0.3802158832550049
Epoch 2 Batch 120/124:
  Main Loss: 0.5024673342704773, Adversary Loss: 0.5831894874572754, Combined Los
s: 0.35666996240615845

Epoch 3 Batch 0/124:
  Main Loss: 0.5417467355728149, Adversary Loss: 0.49512794613838196, Combined Los
s: 0.41796475648880005
Epoch 3 Batch 60/124:
  Main Loss: 0.5001442432403564, Adversary Loss: 0.5010347962379456, Combined Los
s: 0.37488555908203125
Epoch 3 Batch 120/124:
  Main Loss: 0.5103387832641602, Adversary Loss: 0.5742058753967285, Combined Los
s: 0.366787314414978

Epoch 4 Batch 0/124:
  Main Loss: 0.5417506694793701, Adversary Loss: 0.4951217472553253, Combined Los
s: 0.4179702401161194
Epoch 4 Batch 60/124:
  Main Loss: 0.5067021250724792, Adversary Loss: 0.49135711789131165, Combined Los
s: 0.38386285305023193
Epoch 4 Batch 120/124:
  Main Loss: 0.5020797848701477, Adversary Loss: 0.5786578059196472, Combined Los
s: 0.3574153184890747

Epoch 5 Batch 0/124:
  Main Loss: 0.5315114259719849, Adversary Loss: 0.5036830902099609, Combined Los
s: 0.40559065341949463
Epoch 5 Batch 60/124:
  Main Loss: 0.49683046340942383, Adversary Loss: 0.5027450323104858, Combined Los
s: 0.37114420533180237
Epoch 5 Batch 120/124:
  Main Loss: 0.49896618723869324, Adversary Loss: 0.5885825157165527, Combined Los
```

```
s: 0.35182055830955505

Epoch 6 Batch 0/124:
  Main Loss: 0.5307181477546692, Adversary Loss: 0.5038533210754395, Combined Los
s: 0.4047548174858093
Epoch 6 Batch 60/124:
  Main Loss: 0.49951982498168945, Adversary Loss: 0.47592148184776306, Combined Lo
ss: 0.3805394470691681
Epoch 6 Batch 120/124:
  Main Loss: 0.498751163482666, Adversary Loss: 0.586587131023407, Combined Loss:
0.3521043658256531

Epoch 7 Batch 0/124:
  Main Loss: 0.5311392545700073, Adversary Loss: 0.5034368634223938, Combined Los
s: 0.40528005361557007
Epoch 7 Batch 60/124:
  Main Loss: 0.4954342842102051, Adversary Loss: 0.5044565796852112, Combined Los
s: 0.3693201541900635
Epoch 7 Batch 120/124:
  Main Loss: 0.4986691176891327, Adversary Loss: 0.5850037336349487, Combined Los
s: 0.3524181842803955

Epoch 8 Batch 0/124:
  Main Loss: 0.5298293828964233, Adversary Loss: 0.5032631158828735, Combined Los
s: 0.40401360392570496
Epoch 8 Batch 60/124:
  Main Loss: 0.48791542649269104, Adversary Loss: 0.4848599433898926, Combined Los
s: 0.3667004406452179
Epoch 8 Batch 120/124:
  Main Loss: 0.49859270453453064, Adversary Loss: 0.5841959118843079, Combined Los
s: 0.3525437116622925

Epoch 9 Batch 0/124:
  Main Loss: 0.5257471203804016, Adversary Loss: 0.5042082667350769, Combined Los
s: 0.3996950387954712
Epoch 9 Batch 60/124:
  Main Loss: 0.48774051666259766, Adversary Loss: 0.4876289665699005, Combined Los
s: 0.3658332824707031
Epoch 9 Batch 120/124:
  Main Loss: 0.4988847076892853, Adversary Loss: 0.5807932615280151, Combined Los
s: 0.3536863923072815

Epoch 10 Batch 0/124:
  Main Loss: 0.5201929807662964, Adversary Loss: 0.5116246938705444, Combined Los
s: 0.3922868072986603
Epoch 10 Batch 60/124:
  Main Loss: 0.48790448904037476, Adversary Loss: 0.48925548791885376, Combined Lo
ss: 0.3655906319618225
Epoch 10 Batch 120/124:
  Main Loss: 0.49840131402015686, Adversary Loss: 0.5836677551269531, Combined Los
s: 0.3524843752384186

Epoch 11 Batch 0/124:
  Main Loss: 0.5207776427268982, Adversary Loss: 0.5075111389160156, Combined Los
s: 0.3938998579978943
Epoch 11 Batch 60/124:
  Main Loss: 0.4876974821090698, Adversary Loss: 0.4950729310512543, Combined Los
```

```
s: 0.36392924189567566
Epoch 11 Batch 120/124:
  Main Loss: 0.4984506666660309, Adversary Loss: 0.5729845762252808, Combined Los
s: 0.3552045226097107

Epoch 12 Batch 0/124:
  Main Loss: 0.5200053453445435, Adversary Loss: 0.506267786026001, Combined Loss:
0.3934383988380432
Epoch 12 Batch 60/124:
  Main Loss: 0.48766613006591797, Adversary Loss: 0.49951431155204773, Combined Lo
ss: 0.36278754472732544
Epoch 12 Batch 120/124:
  Main Loss: 0.4984952211380005, Adversary Loss: 0.5652105808258057, Combined Los
s: 0.3571925759315491

Epoch 13 Batch 0/124:
  Main Loss: 0.5199968814849854, Adversary Loss: 0.5010631680488586, Combined Los
s: 0.3947311043739319
Epoch 13 Batch 60/124:
  Main Loss: 0.48782432079315186, Adversary Loss: 0.4969266951084137, Combined Los
s: 0.36359265446662903
Epoch 13 Batch 120/124:
  Main Loss: 0.49845802783966064, Adversary Loss: 0.5793356895446777, Combined Los
s: 0.3536241054534912

Epoch 14 Batch 0/124:
  Main Loss: 0.5199251174926758, Adversary Loss: 0.5116252899169922, Combined Los
s: 0.39201879501342773
Epoch 14 Batch 60/124:
  Main Loss: 0.48958995938301086, Adversary Loss: 0.49364060163497925, Combined Lo
ss: 0.36617982387542725
Epoch 14 Batch 120/124:
  Main Loss: 0.4984586238861084, Adversary Loss: 0.5827131271362305, Combined Los
s: 0.3527803421020508

Epoch 15 Batch 0/124:
  Main Loss: 0.5201034545898438, Adversary Loss: 0.5104336142539978, Combined Los
s: 0.3924950361251831
Epoch 15 Batch 60/124:
  Main Loss: 0.4887862205505371, Adversary Loss: 0.49524155259132385, Combined Los
s: 0.36497583985328674
Epoch 15 Batch 120/124:
  Main Loss: 0.4983420968055725, Adversary Loss: 0.5795924663543701, Combined Los
s: 0.35344398021698
```

```
main_predictions = []
for model in neural_models:
    y_pred = (model.predict(X_test) > 0.5).astype("int32").squeeze()
    main_predictions.append(y_pred)
```

```
62/62 [==============================] - 0s 2ms/step
62/62 [==============================] - 0s 2ms/step
62/62 [==============================] - 0s 2ms/step
62/62 [==============================] - 0s 2ms/step
62/62 [==============================] - 0s 2ms/step
62/62 [==============================] - 0s 2ms/step
62/62 [==============================] - 0s 2ms/step
```

```python
nn_metrics = []

for i, y_pred in enumerate(main_predictions):
    accuracy = round(accuracy_score(y_test, y_pred), 3)
    precision = round(precision_score(y_test, y_pred), 3)
    recall = round(recall_score(y_test, y_pred), 3)
    f1 = round(f1_score(y_test, y_pred), 3)
    roc_auc = round(roc_auc_score(y_test, y_pred), 3)

    nn_metrics.append({
        "Model": f"Neural Network {i+1}",
        "Accuracy": accuracy,
        "Precision": precision,
        "Recall": recall,
        "F1-score": f1,
        "ROC AUC": roc_auc
    })

# Display the 7 models performances
nn_metrics = pd.DataFrame(nn_metrics)
nn_metrics
```

	Model	Accuracy	Precision	Recall	F1-score	ROC AUC
0	Neural Network 1	0.778	1.000	0.533	0.695	0.766
1	Neural Network 2	0.779	1.000	0.534	0.696	0.767
2	Neural Network 3	0.773	0.990	0.527	0.688	0.761
3	Neural Network 4	0.777	0.994	0.533	0.694	0.765
4	Neural Network 5	0.780	0.992	0.541	0.700	0.769
5	Neural Network 6	0.779	1.000	0.534	0.696	0.767
6	Neural Network 7	0.779	1.000	0.534	0.696	0.767

```python
temp = []
row_names = [f'Neural Network {i+1}' for i in range(7)]

for j in range(len(main_predictions)):
        temp1 = []
        for i in range(len(sensitive_features)):
                Boolean_Output = calculate_demographic_parity(main_predictions[j],
                temp1.append(Boolean_Output)
        temp.append(temp1)
```

```
===
 Sex_encoded
sensitive_attribute
0    0.374026
1    0.223062
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.324121
```

```
1    0.234435
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.254871
1    0.191781
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.251656
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.374026
1    0.223693
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.324121
1    0.235070
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.255398
1    0.191781
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.252165
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.374026
1    0.223062
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.326633
1    0.233799
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.255398
1    0.178082
Name: predictions, dtype: float64
===
```

```
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.251656
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.371429
1    0.225583
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.319095
1    0.237611
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.255924
1    0.205479
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.253184
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.374026
1    0.230624
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.344221
1    0.236976
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.258557
1    0.260274
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.257769
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.374026
```

```
1    0.223693
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.324121
1    0.235070
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.255398
1    0.191781
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.252165
Name: predictions, dtype: float64
===
 Sex_encoded
sensitive_attribute
0    0.374026
1    0.223693
Name: predictions, dtype: float64
===
 Age Range_encoded
sensitive_attribute
0    0.324121
1    0.235070
Name: predictions, dtype: float64
===
 Citizenship_encoded
sensitive_attribute
0    0.255398
1    0.191781
Name: predictions, dtype: float64
===
 Protected category_encoded
sensitive_attribute
0    0.444444
1    0.252165
Name: predictions, dtype: float64
```

```python
AD_results = pd.DataFrame(temp, columns=sensitive_features, index=row_names)
AD_results
```

	Sex_encoded	Age Range_encoded	Citizenship_encoded	Protected category_encoded
Neural Network 1	False	T	T	False
Neural Network 2	False	T	T	False
Neural Network 3	False	T	T	False
Neural Network 4	T	T	T	False
Neural Network 5	T	T	T	False
Neural Network 6	False	T	T	False
Neural Network 7	False	T	T	False

```
temp = []
row_names = [f'Neural Network {i+1}' for i in range(7)]

for j in range(len(main_predictions)):
        temp1 = []
        for i in range(len(sensitive_features)):
                Boolean_Output = calculate_equalized_odds(main_predictions[j], y_te
                temp1.append(Boolean_Output)
        temp.append(temp1)
```

```
===
 Sex_encoded
0.0
0.20139190952588087
===
 Age Range_encoded
0.0
0.050345120261617415
===
 Citizenship_encoded
0.0
0.16059712273520732
===
 Protected category_encoded
0.0
0.26881720430107536
===
 Sex_encoded
0.0
0.2000144990575613
===
 Age Range_encoded
0.0
0.0489346830260744
```

```
===
 Citizenship_encoded
0.0
0.16171070848130975
===
 Protected category_encoded
0.0
0.26774193548387104
===
 Sex_encoded
0.005807200929152149
0.20827896186747863
===
 Age Range_encoded
0.0011896760317246938
0.057397306439332496
===
 Citizenship_encoded
0.004995004995004995
0.18316980677782463
===
 Protected category_encoded
0.00484027105517909
0.27419354838709686
===
 Sex_encoded
0.003484320557491289
0.19522981006234597
===
 Age Range_encoded
0.003468208092485549
0.038674688268407476
===
 Citizenship_encoded
0.002997002997002997
0.13245650996207786
===
 Protected category_encoded
0.002904162633107454
0.26881720430107536
===
 Sex_encoded
0.004645760743321719
0.1903726257793244
===
 Age Range_encoded
0.01628579110095443
0.06823770236029814
===
 Citizenship_encoded
0.05355755355755355
0.0850839703846385
===
 Protected category_encoded
0.003872216844143272
0.2602150537634409
===
```

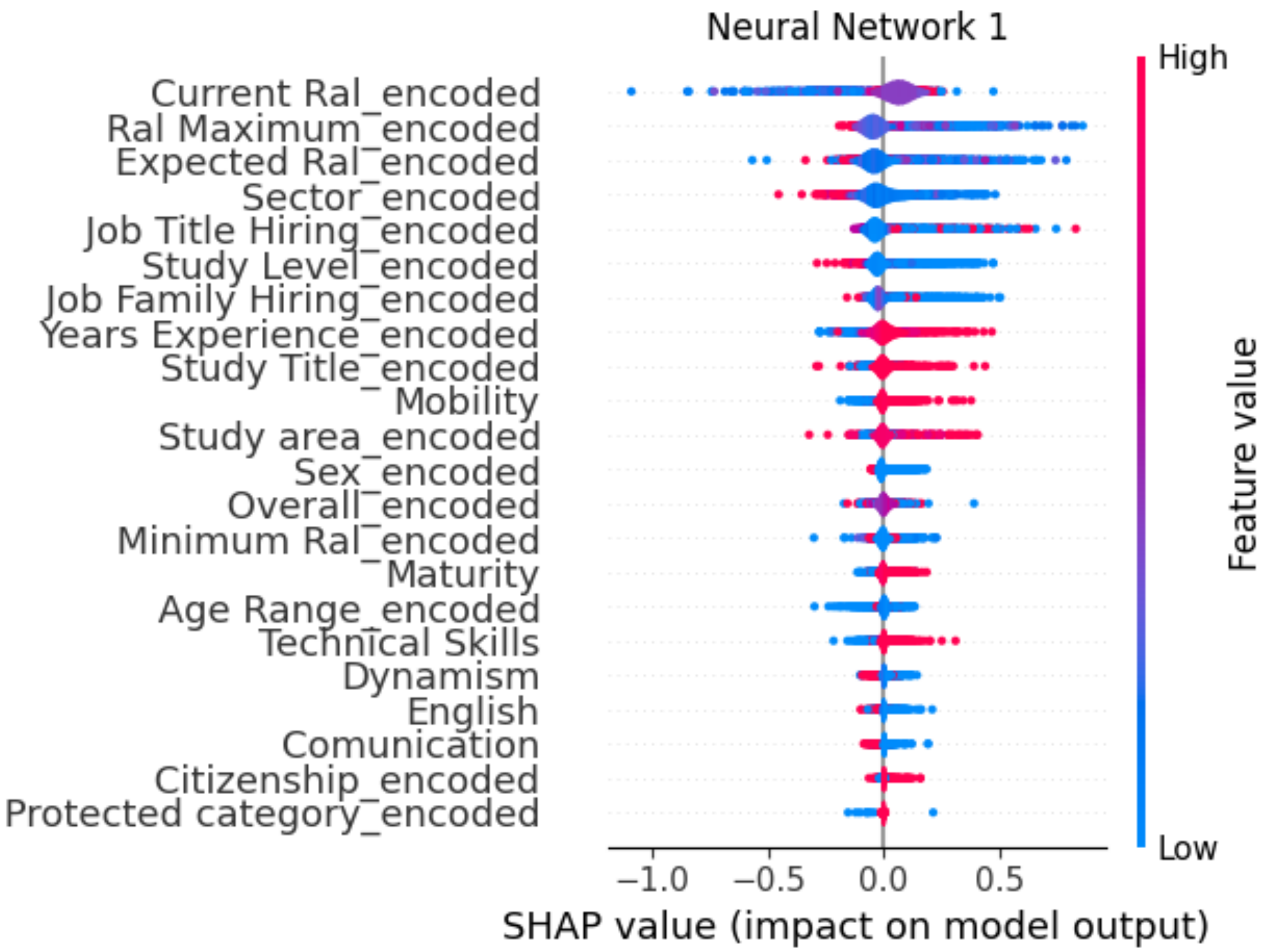
```
 Sex_encoded
0.0
0.2000144990575613
===
 Age Range_encoded
0.0
0.0489346830260744
===
 Citizenship_encoded
0.0
0.16171070848130975
===
 Protected category_encoded
0.0
0.26774193548387104
===
 Sex_encoded
0.0
0.2000144990575613
===
 Age Range_encoded
0.0
0.0489346830260744
===
 Citizenship_encoded
0.0
0.16171070848130975
===
 Protected category_encoded
0.0
0.26774193548387104
```

```
AD_results = pd.DataFrame(temp, columns=sensitive_features, index=row_names)
AD_results
```

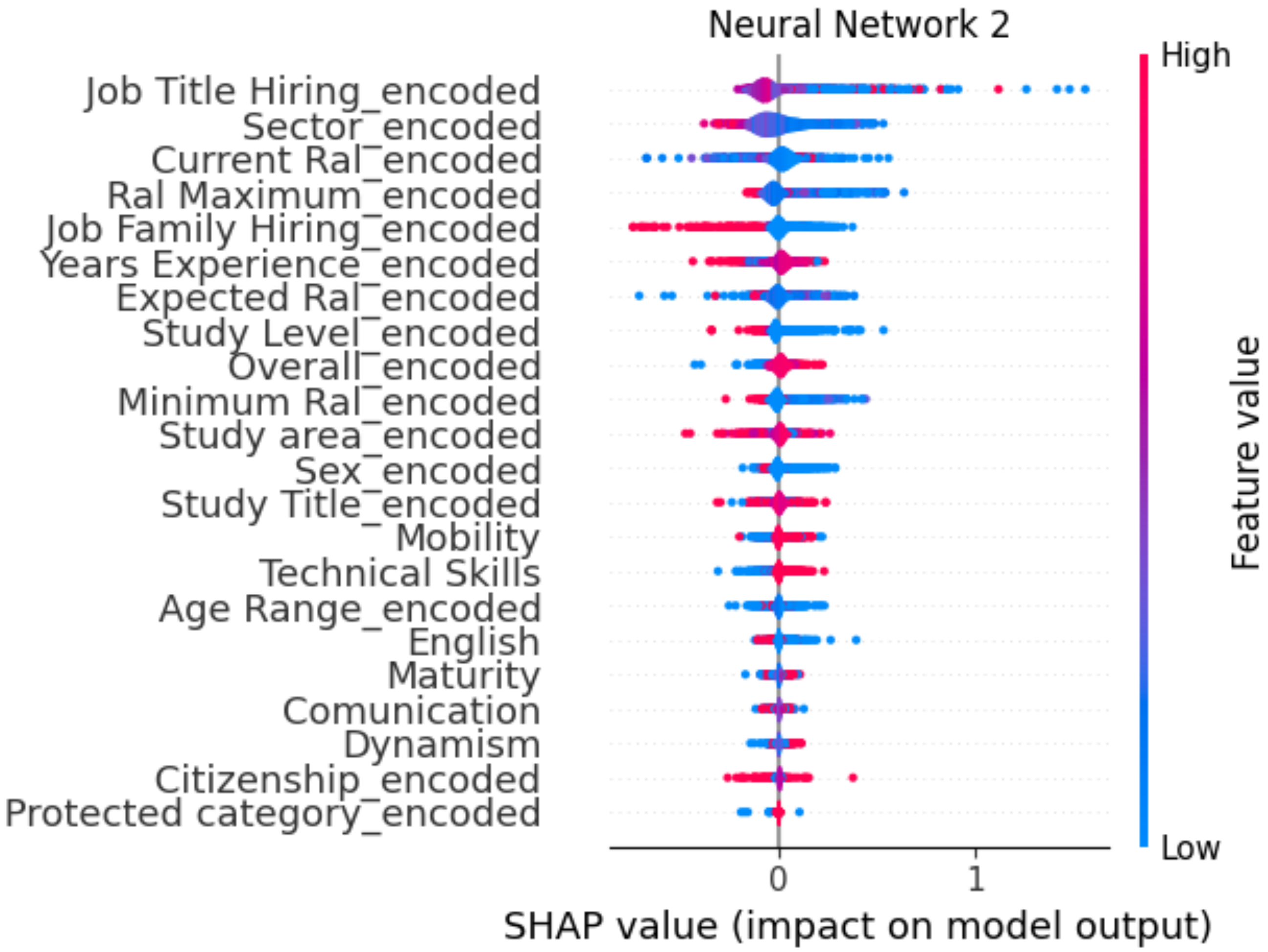
	Sex_encoded	Age Range_encoded	Citizenship_encoded	Protected category_encoded
Neural Network 1	T	T	T	T
Neural Network 2	T	T	T	T
Neural Network 3	T	T	T	T
Neural Network 4	T	T	T	T
Neural Network 5	T	T	T	T
Neural Network 6	T	T	T	T
Neural Network 7	T	T	T	T

```
for i in range(5, 12):
    print(i)
    summaryPlot(models[models_list[i]], X_test, tot_columns, plot_type='violin', pl
```

```
5
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
(1972, 22)
```

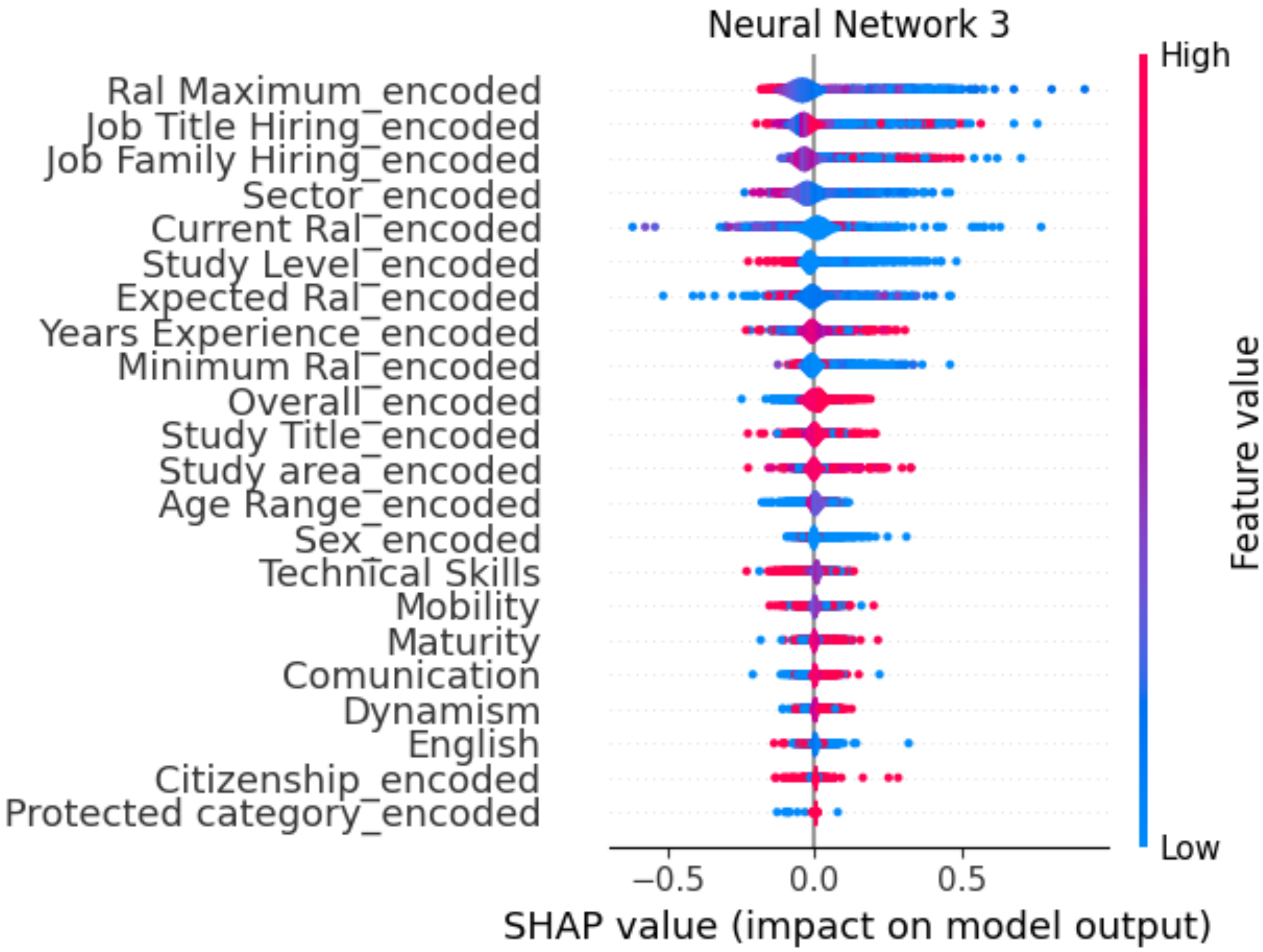



```
6
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



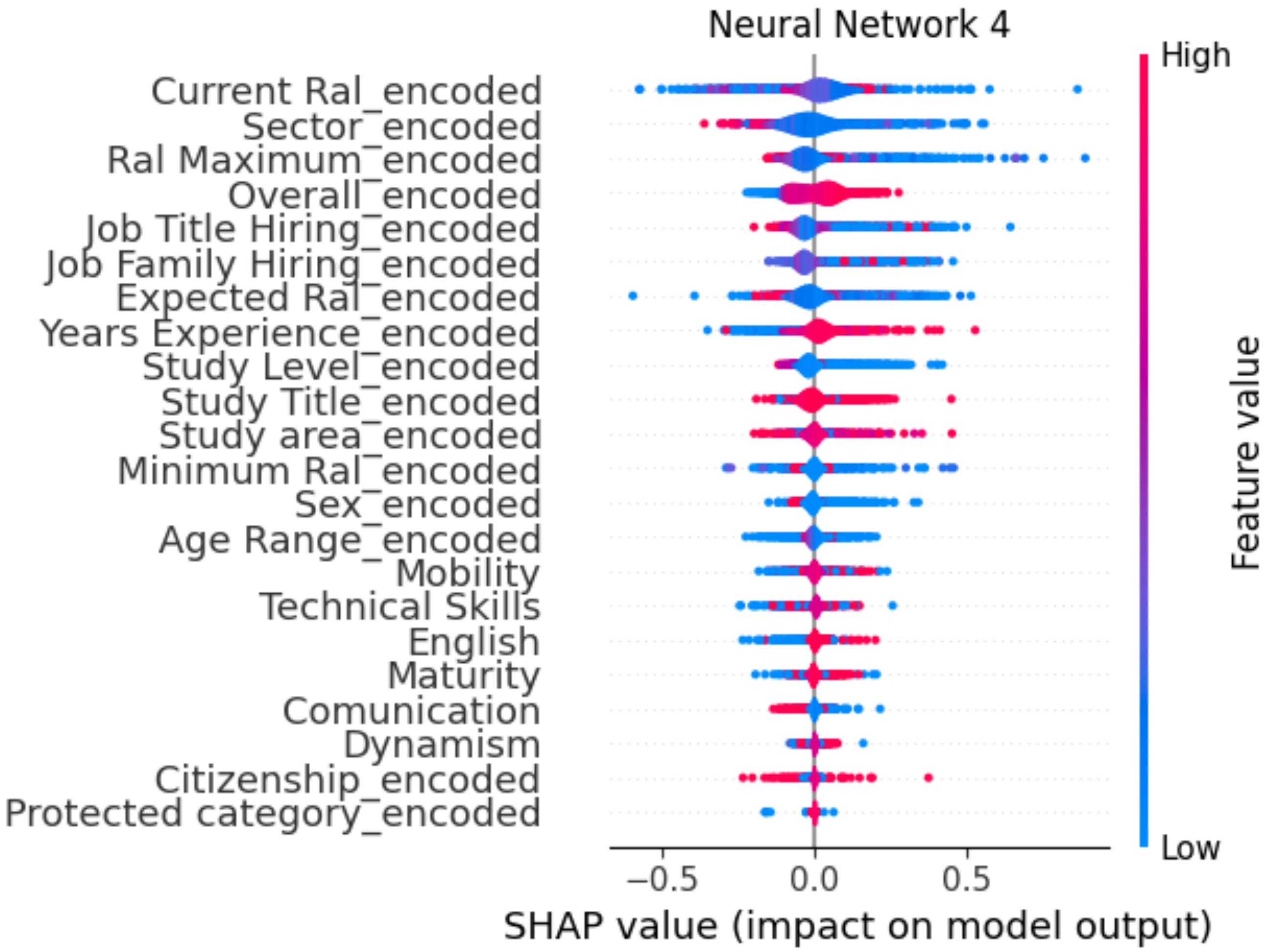
7

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



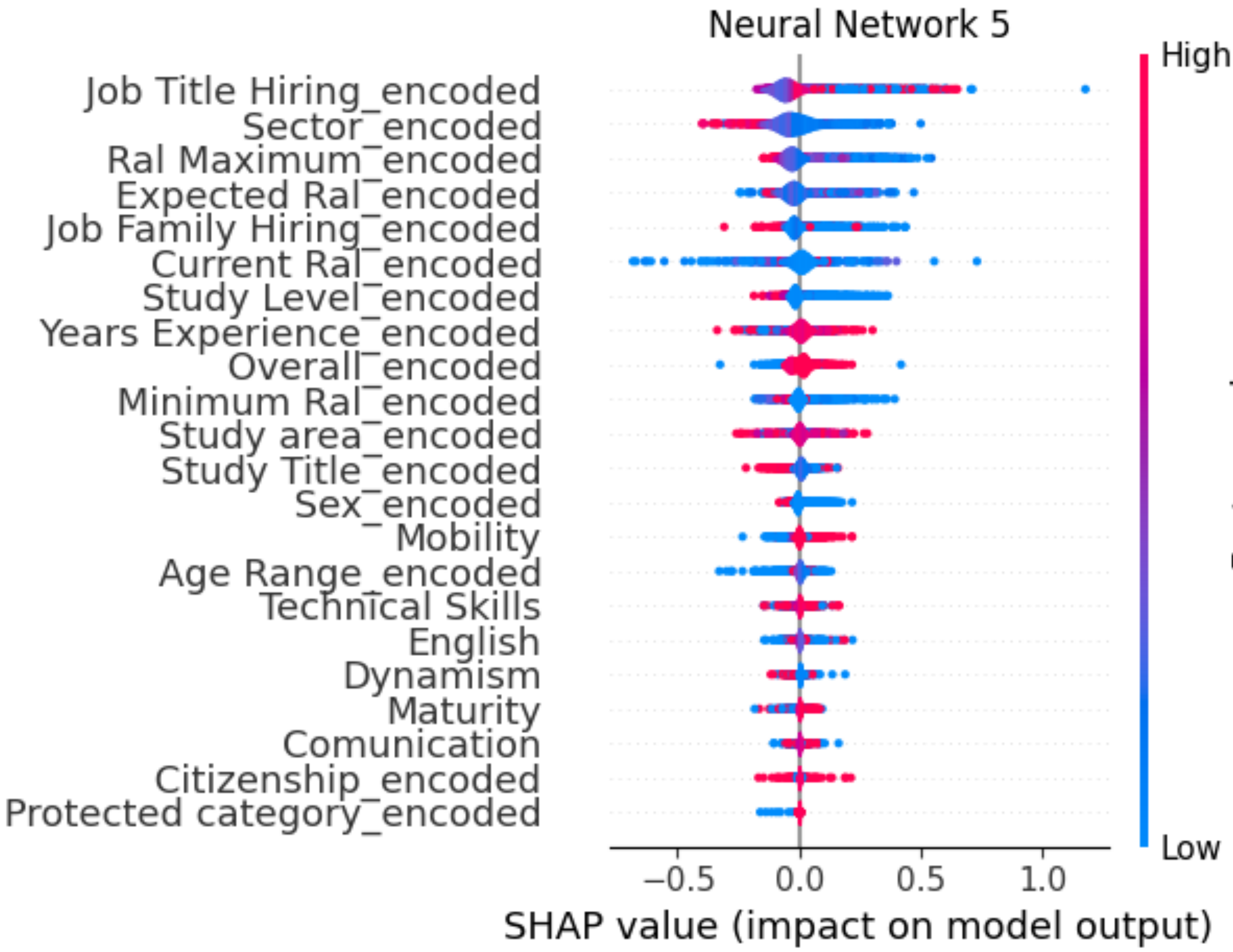
8

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



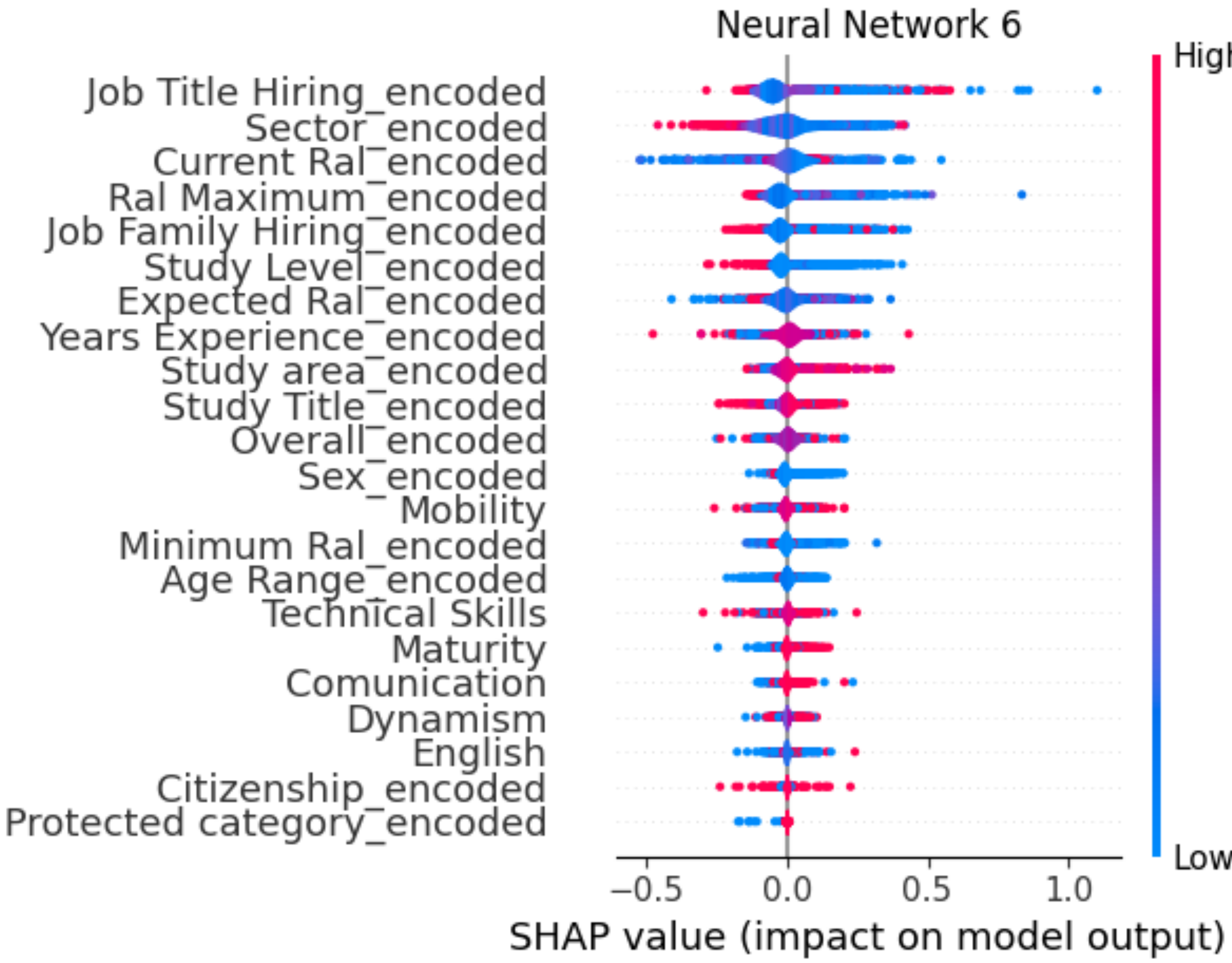
9

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```



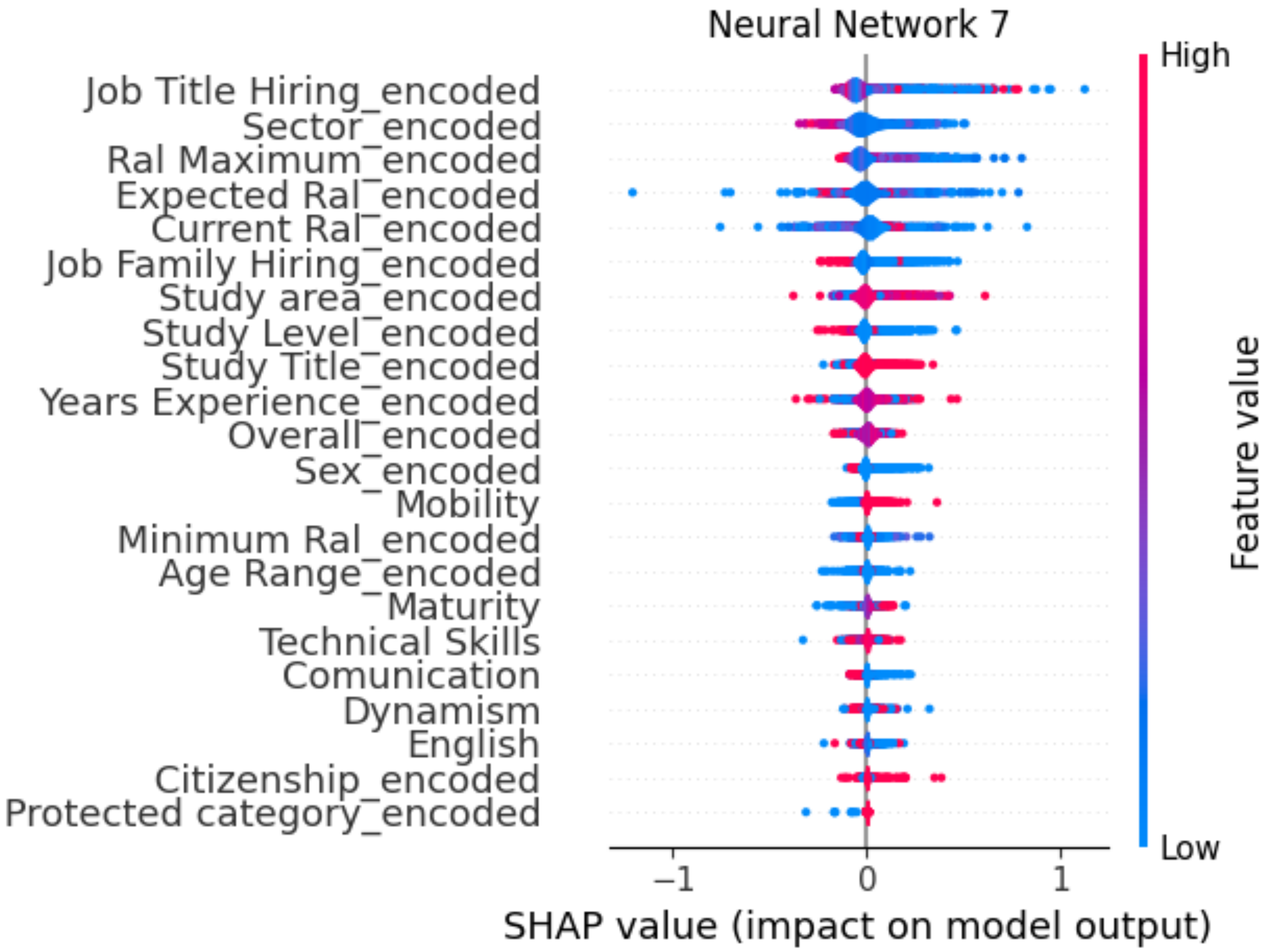

```
10
```

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020-10-11. To update it, simply pass a True/False value to the `training` argument of the `__call__` method of your layer or model.
```



11

```
`tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020
-10-11. To update it, simply pass a True/False value to the `training` argument of
the `__call__` method of your layer or model.
```


Neural Network 7
Job Title Hiring_encoded
Sector_encoded
Ral Maximum_encoded
Expected Ral_encoded
Current Ral_encoded
Job Family Hiring_encoded
Study area_encoded
Study Level_encoded
Study Title_encoded
Years Experience_encoded
Overall_encoded
Sex_encoded
Mobility
Minimum Ral_encoded
Age Range_encoded
Maturity
Technical Skills
Comunication
Dynamism
English
Citizenship_encoded
Protected category_encoded
−1
0
1
SHAP value (impact on model output)
High
Low
Feature value